Fakultät für Maschinenwesen
Lehrstuhl für Flugsystemdynamik

Utilization of Problem and Dynamic Characteristics for Solving Large Scale Optimal Control Problems

Matthias Bittner

Vollständiger Abdruck der von der Fakultät für Maschinenwesen der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs

genehmigten Dissertation.

Vorsitzender:	Prof. Carlo L. Bottasso, Ph.D.
Prüfer der Dissertation:	1. Prof. Dr.-Ing. Florian Holzapfel
	2. Prof. Dr. rer. nat. Matthias Gerdts Universität der Bundeswehr München
	3. Prof. Joseph Z. Ben-Asher, Ph.D. Technion – Israel Institute of Technology, Haifa, Israel

Die Dissertation wurde am 18.01.2017 bei der Technischen Universität München eingereicht und durch die Fakultät für Maschinenwesen am 06.04.2017 angenommen.

Abstract

This thesis presents several approaches, based on the characteristics of the problem formulation and the involved dynamic systems, for solving large scale, complex, and real world optimal control problems. Application-wise, the work focuses on aircraft trajectory optimization problems.

In many applications, the performance of a dynamic system needs to be increased without the possibility to change the system itself or any of its inherent properties. In these cases, optimal operation strategies are required. Optimal control theory is one means to calculate these. Currently, many optimal control problems resulting from real world engineering challenges are hard to handle, as the numerical optimization problems, one gets after the required discretization, cannot be solved efficiently.

The methods presented in this work target the bridging of the gap between the aforementioned real world applications and the available mathematical theory. On the one side, methods for improving the solution process, and on the other side, methods for the assessment of the solution quality of optimal control problems are presented. This includes the generation of enhanced initial guesses, the handling of multi criteria optimal control problems, a thorough analysis of model properties – suggesting improvements in the model formulation – and the exploitation of sparsity features. Besides, a novel, combined transcription method for models incorporating different time scales is introduced. Moreover, a previously published control grid refinement method is enhanced and adapted such that it can exploit physical properties of the dynamic models under consideration.

The ideas presented here are incorporated in an optimal control tool developed at the Institute of Flight System Dynamics of TU München in order to be able to easily reuse them. This tool is also used to solve illustrative examples from air traffic management and air race trajectory optimization that demonstrate the presented methods.

Zusammenfassung

Die vorliegende Dissertation stellt verschiedene Ansätze, die auf der Nutzung von Merkmalen der Problemformulierung und des beteiligten dynamischen Systems basieren, vor, die zur Lösung von großen, komplexen Optimalsteuerungsproblemen aus dem realen Leben verwendet werden können. In den Anwendungen konzentriert sich die Arbeit dabei vornehmlich auf Flugbahnoptimierungsprobleme.

In vielen Anwendungen soll die Leistungsfähigkeit eines dynamischen Systems erhöht werden, ohne dass das System oder seine Eigenschaften verändert werden können. In diesen Fällen sind optimale Nutzungsstrategien erforderlich, die unter anderem mit den Methoden der Optimalsteuerung bestimmt werden können. Derzeit sind viele der Optimalsteuerungsprobleme, die aus realen technologischen Herausforderungen resultieren, schwer zu lösen, da die numerischen Optimierungsprobleme, die nach der notwendigen Diskretisierung entstehen, nicht effizient gelöst werden können.

Die in dieser Arbeit vorgestellten Methoden zielen auf die Überbrückung der Kluft zwischen den vorgenannten realen Anwendungen und der verfügbaren mathematischen Theorie ab. Daher werden hier einerseits Methoden zur Verbesserung des Lösungsprozesses, und andererseits Ansätze zur Abschätzung der Lösungsqualität von Optimalsteuerungsproblemen vorgestellt. Dies umfasst die Erzeugung besserer Anfangslösungen, die Behandlung von multikriteriellen Optimalsteuerungsproblemen, eine detaillierte Analyse von Modelleigenschaften – die auch Vorschläge zur Verbesserung der Modellformulierung enthält – und die Ausnutzung der Besetztheit der Problemstruktur. Außerdem wird ein neuartiges, kombiniertes Transkriptionsverfahren für Modelle mit verschiedenen Zeitskalen eingeführt. Weiterhin wird ein zuvor veröffentlichtes Steuergitterverfeinerungsverfahren verbessert und so angepasst, dass physikalische Eigenschaften der betrachteten dynamischen Modelle ausgenutzt werden können.

Um eine einfache Wiederverwendung der hier präsentierten Ideen zu ermöglichen, sind diese in das am Lehrstuhl für Flugsystemdynamik der TU München entwickelte Optimalsteuerungstool integriert. Dieses wird auch verwendet um anschauliche Beispiele aus dem Flugverkehrsmanagement und der Flugbahnoptimierung für Luftrennen zu lösen und dabei die hier vorgestellten Methoden zu demonstrieren.

Danksagung

Diese Arbeit entstand während meiner Zeit als wissenschaftlicher Mitarbeiter am Lehrstuhl für Flugsystemdynamik der TU München, der von Prof. Florian Holzapfel geleitet wird.

Ebendiesem gilt zuallererst mein Dank dafür, dass er mir die Tätigkeit am Lehrstuhl ermöglicht hat und mich in allen Anliegen – seien sie fachlicher oder nicht-fachlicher Natur – stets bereitwillig unterstützt hat. Weiterhin möchte ich mich bei Prof. Matthias Gerdts bedanken, der meine Arbeit, genauso wie die gesamte Flugbahnoptimierungsgruppe, stets mit größter Hilfsbereitschaft gefördert hat. Besides, I really want to thank Prof. Joseph Z. Ben-Asher for the many enjoyable and revealing discussions and for his commitment as third examiner. I am also very grateful that Prof. Carlo L. Bottasso agreed to chair the exam.

Bei Michael Buhl möchte ich mich für die Übernahme des Mentorats im Rahmen der TUM Graduate School und die vielen interessanten Gespräche bedanken.

Zudem möchte ich mich ganz herzlich bei meinen Kollegen am Lehrstuhl bedanken, wobei vor allem Maximilian Richter hervorzuheben ist, der mich gerade in den besonders anstrengenden Phasen der Tätigkeit am Lehrstuhl immer unterstützt hat. Ich habe die Zusammenarbeit und die Diskussionen mit ihm stets sehr geschätzt. Jakob Lenz, Matthias Rieck, Benedikt Grüter, Johannes Diepolder und David Löbl gilt mein Dank für viele kritische Diskussionen und die überaus konstruktive Zusammenarbeit. Bei Christoph Göttlicher möchte ich mich für die hilfreichen Hinweise rund um Modellidentifikation und Filterung bedanken.

Mein besonderer Dank gilt meiner wundervollen Frau Clarissa, die mich trotz der notwendigen Entbehrungen während der gesamten Zeit bestärkt und unterstützt hat.

Garching, im Januar 2017 Matthias Bittner

Contents

List of Figures

List of Tables

Acronyms

ADOL-C	Automatic Differentiation by OverLoading in C++
AIAA	American Institute of Aeronautics and Astronautics
AMPL	A Mathematical Programming Language
API	Application Programming Interface
ATC	Air Traffic Control
ATM	Air Traffic Management
BADA	Base of Aircraft Data (published by EUROCONTROL)
BFGS	Broyden-Fletcher-Goldfarb-Shanno (update)
CARATS	Collaborative Action for Renovation of Air Traffic Systems (Japan)
CNS	Communication, Navigation, Surveillance
CPDLC	Controller-Pilot Data Link Communications
CPU	Central Processing Unit
DAE	Differential Algebraic Equation
DENMRA	Density Function Based Mesh Refinement Algorithm
DoF	Degree of Freedom
ECEF	Earth Centered Earth Fixed
ECI	Earth Centered Inertial
ENRI	Electronic Navigation Research Institute
FAA	Federal Aviation Administration
FALCON.m	FSD OptimAL CONtrol Tool for MATLAB
FEM	Finite Element Method
FRA	Frankfurt Airport
FSD	Institute of Flight System Dynamics (at TUM)
GESOP	Graphical Environment for Simulation and Optimization
GPOPS	General Purpose OPtimal Control Software
GRU	Guarulhos-Governador André Franco Montoro Int. Airport, São Paulo
GUI	Graphical User Interface
ICAO	International Civil Aviation Organization
IFR	Instrument Flight Rules
IP	Interior Point (Method)
IPOPT	Interior Point OPTimizer
ISA	International Standard Atmosphere

JHS	Sisimiut Airport
KKT	Karush-Kuhn-Tucker
LAX	Los Angeles International Airport
LICQ	Linear Independence Constraint Qualification
MILP	Mixed Integer Linear Programming (problem)
MINLP	Mixed Integer non-linear Programming (problem)
MIOCP	Mixed Integer Optimal Control Problem
MLIT	(Japanese) Ministry of Land, Infrastructure, Transport and Tourism
MPC	Model Predicitve Control
MSL	Mean Sea Level
MTOW	Maximum Take-Off Weight
MUC	Munich Airport
NED	North East Down
NLP	non-linear Programming (problem)
NUDOCCS	Numerical Discretisationmethod for Optimal Controlproblems with Constraints in Controls and States (sic)
OCP	Optimal Control Problem
OCPID-DAE1	Optimal Control and Parameter Identification with Differential-Algebraic Equations of Index 1
ODE	Ordinary Differential Equation
OWE	Operating Weight Empty
PSOPT	PseudoSpectral OPTimal control solver in C++
QP	Quadratic Programming
RAM	Random Access Memory
RCV	Row Column Value (format of sparse data)
RPAS	Remotely Piloted Aerial System
RTS	Rauch-Tung-Striebel (Smoother)
SAGE	System for Assessing Aviation's Global Emission
SESAR	Single European Sky ATM Research
SFO	San Francisco International Airport
SNOPT	Sparse non-linear OPTimizer
SOCS	Sparse Optimal Control Software
SQP	Sequential Quadratic Programming
TMA	Terminal Maneuvering Area
UAS	Unmanned Aerial System
UAV	Unmanned Aerial Vehicle
VIL	Dakhla Airport
WGS84	World Geodetic System 1984
WORHP	We Optimize Really Huge Problems
ZFW	Zero Fuel Weight

Symbols and Indices

Symbols

$\circ, \square, *$	General placeholders
$\mathbf{A}$	State matrix of the linearized system dynamic
$\mathbf{a}$	Acceleration vector
α	Angle of attack
$\boldsymbol{B}$	Base function for the control discretization
b	Wing span
β	Angle of sideslip
$\mathbf{C}$	State matrix of the linearized output equation
$\mathbf{C}$	Constraint vector
$\mathbf{c}_k$	k-th state defect
$C_\circ$	Aerodynamic coefficient for force or moment $\circ$
$C_{\circ\square}$	Aerodynamic coefficient for force or moment $\circ$ depending on quantity $\square$
$\bar{c}$	Mean aerodynamic chord
$\bar{\mathbf{C}}_{eq}$	Discretized equality path constraints
$\mathbf{C}_{eq}$	Equality path constraints
χ	Course angle
CI	Cost index
$\bar{\mathbf{C}}_{ineq}$	Discretized inequality path constraints
$\mathbf{C}_{ineq}$	Inequality path constraints
D	Drag force
d	Separation distance
deg	Degrees
$U_\delta(\circ)$	δ-Environment of $\circ$
δ_{SB}	Spoiler / speed brake position
δ_T	Thrust lever position
$\square'_\circ$	Derivative of $\square$ with respect to $\circ$
$\square''_{\circ *}$	Second derivative of $\square$ with respect to $\circ$ and $*$
$\nabla_\circ$	Gradient with respect to $\circ$
e	MAYER cost function

η	Elevator deflection
η	Weighing factor in the interior point algorithm
F	Cumulative distribution function for the grid refinement
$\mathbf{F}$	Forces acting on the aircraft
$\mathbf{f}$	System dynamics
$\bar{f}$	Normalized density function for the grid refinement
G	Center of gravity
$\mathbf{g}$	Gradient vector
$\mathbf{g}$	Gravitational acceleration vector
$\mathbf{\Gamma}$	Mapping matrix for the process noise
$\mathbf{M}$	State update matrix of the smoother (backward)
γ	Climb angle
$\mathbf{H}$	Angular momentum
$\mathbf{H}$	*Hessian* matrix
h	Altitude
H	HAMILTON function
$\tilde{H}$	Extended HAMILTON function
h_k	Numerical integration step size at time step k
$\mathbf{I}$	Inertia
J	Cost function
$\bar{J}$	Augmented cost function
$\mathbf{K}$	State update matrix of the smoother (forward)
k	Fading factor (in multi aircraft problem)
b	Base function for grid refinement
$\kappa_\circ$	Curvature of $\circ$
L	LAGRANGE cost function
L	Lift force
L	Rolling moment
l	Steepness of fading function for data interpolation
L	LAGRANGE function for constrained optimization problems
$\boldsymbol{\lambda}$	Costates
λ	Geodetic longitude
$\boldsymbol{\lambda}$	LAGRANGE multipliers for equality constraints
$\boldsymbol{\nu}$	LAGRANGE Multipliers for inequality constraints
$\mathbf{l}_h$	Local error of the numerical integration scheme
M	Mach number
$\mathbf{M}$	Moments
M	Pitching moment
$\mathbf{M}$	Transformation matrix between two coordinate frames
m	Mass

μ	Bank angle
$\boldsymbol{\mu}$	Multipliers for the constraint equations in the OCP
N	Yawing moment
n	Load factor
$\boldsymbol{\omega}$	Rotational vector
$\mathbf{P}$	State covariance matrix of the smoother
P	Arbitrary point on the aircraft
$\mathbf{p}$	Impulse acting on the aircraft
$\mathbf{p}$	Vector of additional parameters
p	Roll rate
Φ	Roll angle
φ	Geodetic latitude
Φ_k	Increment function of the numerical integration scheme at time step k
Ψ	Azimuth angle
$\boldsymbol{\Psi}_0$	Initial boundary conditions
$\boldsymbol{\Psi}_f$	Final boundary conditions
$\tilde{q}$	Dimensionless aerodynamic pitch rate
$\tilde{p}$	Dimensionless aerodynamic roll rate
$\tilde{r}$	Dimensionless aerodynamic yaw rate
Q	Cross-stream force
$\mathbf{Q}$	Power spectral density of discretized process noise
$\mathbf{Q}$	*Hessian* approximation
q	Pitch rate
q	Quaternion
$\bar{\mathbf{Q}}$	Power spectral density of process noise
$\bar{q}$	Dynamic pressure
$\mathbf{R}$	Power spectral density of discretized output noise
R	Reference point
R	Separation radius
$\mathbf{r}$	Interior point constraints
r	Yaw rate
$\bar{\mathbf{R}}$	Power spectral density of output noise
a_{ij}, b_j, c_j	Coefficients of the RUNGE-KUTTA integration scheme
$\mathbf{k}_j$	k-th Stage derivative of the RUNGE-KUTTA integration scheme
$\tilde{\mathbf{f}}_F$	Extended state dynamics, including zero dynamics for the controls in the smoothing
$\mathbb{G}_\circ$	Discretized grid for the values $\circ$
$\boldsymbol{S}$	Combined State and Control Constraint
$\mathbf{S}$	Sensitivity of the state / control with respect to a parameter
S	Wing reference area

s	Fading function for data interpolation
$\boldsymbol{\sigma}_0$	Multiplier for the initial boundary conditions
$\boldsymbol{\sigma}_f$	Multiplier for the final boundary conditions
σ_{SFC}	Specific fuel consumption
$\mathbf{s}$	Slack variables in the interior point algorithm
$\mathbf{T}_\circ$	Diagonal scaling matrix used to scale the value $\circ$
t	Independent variable, mostly time
Θ	Pitch angle
tol	Optimization tolerance
T_S	Dynamic time constant
$\mathbf{u}$	Control vector
u	Velocity component in x-direction
$\bar{u}$	Coefficient for control base function
$\mathbf{p}$	Optimization update step
V	Scalar velocity
V	Value function
$\mathbf{V}$	Velocity vector
$\mathbf{v}$	Discretized stochastic output noise
v	Velocity component in y-direction
$\bar{\mathbf{v}}$	Stochastic output noise
$\mathbf{w}$	Discretized stochastic process noise
w_i	Weight of the i-th cost function in the scalarized cost function
w	Velocity component in z-direction
$\bar{\mathbf{w}}$	Stochastic process noise
$\bar{\mathbf{R}}_\mathbf{u}$	Weighting matrix for penalty cost function
$\mathbf{x}_0^k$	k-th multiple shooting state
$\mathbf{x}$	Position of a point
$\mathbf{x}$	State vector
x	Position in x-direction
$\bar{\mathbf{x}}$	Augmented state vector
ξ	Aileron deflection
y	Position in y-direction
z	Optimization value
$\mathbf{z}$	Optimization parameter vector
z	Position in z-direction
ζ	Rudder deflection

Indices

A	With respect to Aerodynamic Frame

A	Aerodynamic quantity
$\bar{A}$	With respect to Rotated Aerodynamic Frame
B	With respect to Body Fixed Frame
b	Related to the backward run of the smoother
C	Constraint related (tolerance)
CAS	Calibrated air speed
CMD	Commanded value
E	With respect to Earth-Centered Earth Fixed Frame
eq	Equality
F	States of the fine simulation model, not appearing in the simplified model
f	Fast (states or dynamics)
f	Related to the forward run of the smoother
FB	Final boundary
G	Gravitational quantity
I	With respect to Earth-Centered Inertial Frame
IB	Initial boundary
$\mathbf{I}$	Identity matrix
$ineq$	Inequality
J	Cost related (tolerance)
K	With respect to Kinematic Frame
K	Kinematic quantity
$\bar{K}$	With respect to Rotated Kinematic Frame
MSL	Mean Sea Level
N	With respect to Navigational Frame
O	With respect to North-East-Down Frame
opt	Optimal value
P	Propulsive quantity
ref	reference
S	States appearing in the fine simulation model and in the simplified model
s	Slow (states or dynamics)
SFC	Specific fuel consumption
T	Target value in multi criteria optimization
0	Initial value
$\bar{\tau}$	Nodes on the fine grid of the combined discretization scheme
T_{Engine}	Time constant of the engine dynamics
f	Final value
u	Eastward wind component
upd	Updated value
v	Northward wind component
W	Wind quantity

Chapter 1

Introduction

Whenever the performance of a system should be increased without changing its design or any of its inherent properties, methods for optimizing its operation strategies are required. Different methods exist for such optimizations, ranging from empirical and statistical approaches to more theoretical methods based on modeling, simulation, and mathematical optimization. In this work, mathematical optimal control of dynamic systems – being a particular example for the aforementioned approaches – is considered. Here, the applications of the regarded technologies are all related to aircraft trajectory optimization and, hence, are using models of the underlying aircraft dynamics.

As always when using technologies that are based on the modeling of real systems, the quality of the achievable results strongly depends on the quality of the model used. Besides the determination of precise dynamic models itself, in optimal control the solution of such high fidelity problems poses its own challenges that will be described and approached in more detail in this work.

In the past decades, many applications of optimal control theory from different fields have been investigated in the literature, one of them being space sciences. In particular, this includes the calculation of optimal ascend trajectories for the so-called Goddard *Rocket*. Even though being one of the first optimal control problems ever solved, and initially coming up in 1919, it is still subject to investigations nowadays, like in [Ben10, ch. 9] and [BMT08]. [PP12]

Besides, many other space related applications have been investigated, like the determination of optimal orbit transfers by Wiesel and Alfano in the 1980s in [WA85]. Therein, the authors formulate a many-revolution orbit transfer as an unconstrained optimal control problem that can be solved analytically based on the indirect solution approach (which is described in more detail in section 3.2.3). With the evolution of digital computer technology, direct solution methods (see section 3.2.4) became more and more popular. They are e. g. used by Herman and Conway in [HC98], where optimal, low-thrust, earth-moon transfers are calculated. In 2003, Betts and Erb considered optimal low thrust trajectories to the moon in their work [BE03]. The cost functions minimized in their examples are the overall fuel consumption of the spacecraft powered by solar electric propulsion during the 201 day mission,

and the transfer time, then resulting in a 198 day mission. The large numerical optimization problem that results from the employed direct discretization can only be solved by exploiting its sparsity. Moreover, time minimal spin-up maneuvers of a flexible spacecraft are analyzed from an optimal control point of view in [Ben10, ch. 11], not only considering rigid body dynamics but also flexible modes of the structure.

In chemical engineering, optimal control is widely used to optimize processes and reactors. To mention only one, [Upr13] is a textbook that shows many basics of optimal control theory, as well as applications like batch distillation and chemotherapy. Moreover, production processes are, for example, analyzed in the works [CB07] and [BE12]. In the first, a production system is modeled in discrete time and the optimal controls – representing the work schedule – are determined using dynamic programming. In the latter, a production system with deteriorating items is regarded in continuous time. The respective optimality conditions are formulated based on an indirect solution approach, and solved numerically.

In the late 1940s and the 1950s, Hestenes, Bellmann and Isaacs researched a minimum time intercept problem for aircraft in the USA, while approximately at the same time, Pontryagin, Boltyanskii and Gamkrelidze tried to solve a similar problem in the Soviet Union. They realized that the commonly used theory for linear systems was not sufficient anymore, because of non-linear effects in the models that could not be neglected due to the increased speed of the aircraft. The respective problem is nowadays also known as the *minimum time to climb problem*, and may be seen as the *first aircraft trajectory optimization problem* ever solved. [PP12]

Since then, many researchers have been working on aircraft trajectory optimization problems, targeting different goals, using different models and different methods. Emissions and noise nuisance in departure and arrival scenarios for commercial airliners are investigated, for example, in [FB11, FB12, RBH13, Ric+13, Ric+14a, Ric+14b]. In [Tor+15], aircraft movement in terminal maneuvering areas is considered and the resulting optimization problems are solved by a combination of optimal control methods and *Mixed Integer non-linear Programming (MINLP)*. At the same conference, the papers [Mat+15, OWH15] regarding optimal sequencing and arrival management were presented. Therein, the problems were solved using dynamic programming and pseudospectral collocation.

Another challenge in the context of *air traffic management (ATM)* that has been researched during the last years (also using optimal control methods) is optimal conflict avoidance in air space sectors. In [Sol+12], such conflict avoidance problems, also considering hazardous weather obstacles, are modeled using a multiphase approach, resulting in a mixed-integer optimal control problem. The same author uses a hybrid optimal control problem formulation in the context of conflict avoidance in [Sol+16], solving problems for seven aircraft in one airspace sector. Wickramasinghe assesses the possible performance improvements in the ATM system by a comparison of the currently used trajectories with optimal trajectories based on dynamic programming in [Wic+15].

While the aforementioned examples mainly used point mass dynamics or even more simpli-

fied simulation models, others also published work on aircraft trajectory optimization problems incorporating more complex dynamics. In [DC08a], a full non-linear rigid body model of a reentry vehicle is used together with a two-timescale direct collocation method (see also chapter 8). Optimal collision avoidance for an *Ekranoplan* is considered in [CG09], again employing rigid body dynamics. Another full non-linear rigid body simulation model is used in [PKH11] for determining safe passage trajectories of a transport aircraft through microburst wind shear by solving an optimal control problem. Bottasso worked on optimization problems regarding complex rotorcraft models, including slow and fast states, and presents a direct solution approach combining single and multiple shooting in [BM09, BLM12]. In this thesis, air race trajectory optimization examples are considered whenever problems with high fidelity simulation models are investigated. Fisch used similar examples in his thesis [Fis11].

In [Len15], periodic optimal control problems are considered, illustrating the results by the optimization of trajectories for a solar powered aircraft, a powered glider with retractable engine, and a bird (the Siskin).

Another aerospace field in which optimal control problems have been solved during the last years is flight control law clearance, where optimal control methods are used for determining control inputs that push a flight control system outside its safety bounds. This way, deficiencies should be identified in order to finally improve the control laws. Two of the many publications in that field are [FP11] and [Die+16].

Moreover, aircraft trajectory optimization methods have been used to infer the intent of an aircraft from a trajectory by estimating the weights of the underlying cost functions in an inverse optimal control problem in [Yok16].

Nowadays, many textbooks on optimal control theory and respective applications exist. For this work, the books of Kirk [Kir70], Bryson and Ho [BH75], Betts [Bet09], Gerdts [Ger12], and Ben-Asher [Ben10] were especially helpful.

A more detailed literature review, concerning the particular aspects of optimal control problems investigated in this work, can be found in the introductions of the topics in the respective chapters.

1.1 Goals and Contributions

The main goal of the efforts taken in this research can be summarized as to solve large scale, complex, and especially *realistic* aircraft trajectory optimization problems. In order to be able to do so, the gap between the possibilities that are offered by mathematical optimal control theory on the one side, and the requirements that real life problems pose on the other side, needs to be bridged.

This high level goal can only be reached by considering both aforementioned sides of the gap and by dividing the challenges into several sub-aspects. Approaching these challenges, contributions to the state of the art are made by this thesis that are outlined and broken down to the respective sub-aspects within the next paragraphs.

After a revision of flight system dynamics in chapter 2, chapter 3 represents a comparably short but yet comprehensive summary of the theoretical fundamentals of optimal control and the required gradient based optimization methods for engineering purposes. Much emphasis was put on descriptive explanations and derivations supporting the understanding of the principles. The goal was clearly *not* to deliver another mathematical textbook including well-known proofs, but to highlight the relations between the many aspects of optimization and optimal control theory for engineering applications. This way, this chapter contributes a very condensed guide to optimal control theory for engineering applications.

Besides, in section 3.2.4 the general sensitivity equations are reviewed, before the numerical integration of the sensitivity equations based on a RUNGE-KUTTA scheme is compared to the differentiation of the numerical integration scheme for the dynamics. The derivation shows that both calculations lead to the same results, if the gradients of the dynamic system are evaluated at the same intermediate stages as the model dynamics itself in both approaches. This section delivers rigorous conditions for the equality of the two approaches.

Additionally, in section 3.2.5, the relations between the direct and the indirect solution approach are discussed in detail. The derivation is based on the book [Ger12, Sec. 5.4] but is extended to general RUNGE-KUTTA integration schemes. Consequently, relationships for the approximation of the costates from LAGRANGE multipliers for many integration schemes can be established using the findings, also enabling the estimation of the *Hamiltonian* of a problem and allowing for an assessment of the achieved optimality. This analysis is conducted exemplarily in the application in section 11.2–11.3, showing that it can be done for real life problems.

In order to be able to solve many different, large, and complex optimal control problems without the need to re-implement the discretization code over and over again, the optimal control tool *FALCON.m* was developed during the time of this research. Of course, as described in chapter 4, the tool has been developed by a team of engineers at the Institute. However, the structure of the software, including the implemented class hierarchy, has been mainly developed by this author. This architecture is described in chapter 4 in more detail.

Furthermore, the generation of best possible initial guesses is considered in chapter 5. Therein, the widely used *homotopy* approach is extended, on the one side, to the special class of optimal control problems containing multiple systems (like multi aircraft trajectory optimization problems), and, on the other side, by the use of sensitivity information for an improved homotopy update. Both of these extensions contribute to the generation of initial guesses for complex optimal control problems. Moreover, a novel method to create initial trajectories for a high fidelity problem from the solution of a simplified problem is presented in section 5.2 that employs a RAUCH-TUNG-STRIEBEL (*RTS*) smoother.

In chapter 6, an analysis of the properties of the point mass aircraft simulation model and the rigid body simulation model for the use in optimal control problems is presented. This analysis reveals that under certain circumstances more complex and thus more nonlinear optimization models improve the performance of the solution process of the optimal

control problem, compared to more linear and simplified simulation models. In section 6.4 an approach presented in [Bet09, p. 172] is applied to the point mass simulation model, resulting in a reformulation of the respective optimal control problem that clearly improves solution performance. In section 6.5, a novel method for smoothly approximating empirical data is introduced. Examples for these approaches are presented in chapter 11.

Chapter 8 introduces a novel discretization method that uses direct collocation on a relatively coarse grid for slow varying states, while multiple shooting is used to represent the fast varying states on a finer grid. This way, the size and the complexity of the discretized optimal control problem can be reduced without any deterioration in solution accuracy. A corresponding illustrative example is discussed in section 11.4.

The last novelty in the theoretical part of this work is the physically motivated control grid refinement that is presented in chapter 9. It improves the previously published *DENMRA* algorithm ([ZT11]) for real world applications by exploiting underlying physical properties. Besides, the algorithm is extended by a small optimization process that can determine the ideal number of grid points based on a given minimum and maximum allowed grid spacing.

To the knowledge of this author, the application presented in section 10.2 is the largest ever published aircraft trajectory optimization problem by means of number of participants in the scenario. Therein the simultaneous approach of 18 aircraft to *Tokyo International Airport* is optimized with respect to a combination of arrival time and fuel burn. In the next section, civil aerial refueling scenarios regarding a reduction in fuel burn for airliners are optimized. The analysis reveals the potential savings of the concept when using current aircraft types as well as especially tailored redesigns. In section 10.4, multi criteria optimal control methods are employed to calculate trajectories of several aircraft passing through a single airspace sector. Therein, not only the minimum overall cost, but also a fair distribution of the costs is considered. To the knowledge of the author, such a multi aircraft and multi criteria analysis has not yet been published elsewhere.

The applications in chapter 11 need to be seen as illustrative examples for the increased performance and robustness of the solution process for complex optimal control problems. Although improvements can be clearly seen in most of the examples, none of the tools presented here is the *perfect* tool that helps in every case. In optimization and optimal control, still a lot of problem specific tuning is required. Here, the best a user can get is a large tool set with different tools for a great variety of problems. This thesis contributes to the algorithmic and methodological tools available in this toolbox.

1.2 Structure of the Thesis

After this introduction, the thesis starts with a description of the required aircraft simulation models in chapter 2. After introducing the nomenclature and coordinate frames in sections 2.1 and 2.2, section 2.3 summarizes all assumptions and requirements for the following model derivations. Next, the equations of motion for a point mass simulation model are derived from

kinematic relations and NEWTON's second law. Afterwards, the dynamic system is extended by rotational dynamics, resulting in a description of the movement of a rigid body aircraft in section 2.5. Section 2.6 reviews the *International Standard Atmosphere (ISA)* and the basics for modeling wind influences. In section 2.7 the external forces and moments acting on the aircraft are described, before in the last section of this chapter some commonly used model extensions are summarized.

In chapter 3, the mathematical background for the remainder of the work is recapitulated. In the first section, an extensive overview over numerical optimization is given, starting with unconstrained optimization, before heading over to constrained problems. Next, two commonly used types of algorithms for solving constrained optimization problems are briefly described in sections 3.1.3 and 3.1.4. The next section emphasizes the importance of sparsity in numerical optimization problems, which will be discussed for discretized optimal control problems in more detail in chapter 7 and in section 10.2.9. The discussion of parametric optimization problems presented in section 3.1.6 will later be required for the derivation of the relationship between an optimal control problem and the numerical optimization problem that results from the discretization. Section 3.2 gives an extensive overview of the mathematical class of optimal control problems and the existing solution strategies. After a definition of the problem class under consideration, transformation techniques are shown that are commonly used to map the wide class of general optimal control problems to a more specific class of optimal control problems. In section 3.2.3, the indirect solution approach, in which optimality conditions for an optimal control problem are derived analytically before it is discretized, is presented. An overview of the direct solution approach is given in section 3.2.4, where the optimal control problem is first discretized and then solved using one of the numerical optimization algorithms from section 3.1. Next, the direct and the indirect solution approach are put into relation, resulting in a connection between the LAGRANGE multipliers of the discretized problem and the *costates* of the continuous problem. In the last subsection considering optimal control problems, issues are discussed that appear when controls enter the dynamic model linearly. Methods for handling these issues are later presented in chapter 6. The last section of this chapter focuses on multi criteria optimization and optimal control problems. In there, different scalarization techniques are repeated that can be used to transform a multi criteria problem into multiple regular optimization problems. An application for these methods is presented in section 10.4.

In the fourth chapter of this thesis, the implementation of a discretization scheme in the optimal control tool *FALCON.m* is presented. After a short introduction of the core features of the tool, an overview of other existing optimal control software is given. The problem formulation that is used within *FALCON.m* is presented in section 4.2, before the process for setting up such a problem with the tool is described. The structure that is internally used in *FALCON.m* to represent an optimal control problem within *MATLAB* is presented in section 4.4. Next, sections 4.5, 4.6, and 4.7 present some details of the implementation for the calculation of derivative information, code generation, sorting, indexing, and numerical scaling.

The last section of this chapter roughly describes the software components of *FALCON.m*, their functionalities, and their interaction.

Methods for enhancing the robustness of the solution process of optimal control problems based on improved initial guesses are presented in chapter 5. First, the general and well-known idea of *homotopy* is reviewed. Second, an extended homotopy scheme for problems containing several dynamic systems – like multi aircraft problems – is introduced. Finally, an enhanced homotopy principle for regular optimal control problems is presented that uses sensitivity information to improve the initial guesses between the homotopy steps. In section 5.2, a method for the generation of initial guesses for high fidelity optimal control problems based on the solution of simplified problems is introduced. In order to determine any potentially missing state and/or control histories from the available information, a *RTS* smoother based on the high fidelity simulation model is suggested. The ideas of this chapter are illustrated by examples in sections 10.2, 11.6, and 11.7.

Chapter 6 focuses on different properties of the dynamic models and the constraints appearing in an optimal control problem as well as their particular influence on the solution performance. The aspects regarded here are model fidelity and linear appearing controls. After generally introducing the aspects of a model analysis for optimal control in the first section, this analysis is performed for the point mass simulation model and the rigid body simulation model. The results are used in the examples in chapter 11, in sections 6.2, and 6.3. The respective issues are approached by a model reformulation based on the idea from [Bet09, p. 172]. In the last section of the chapter, a method for representing tabulated data using different base functions that are combined by *hyperbolic tangent* functions is presented. The resulting models feature beneficial smoothness properties.

In the next chapter, several aspects of problem sparsity – meaning the sparsity patterns of the *Jacobian* and the *Hessian* of discretized optimal control problems – are discussed. After a brief introduction of the different elements influencing the sparsity in section 7.1, section 7.2 focuses on the *local sparsity*, which is the sparsity of the model, the constraints and the control discretization. Next, the sparsity resulting from the discretization scheme is combined with the local one, resulting in the overall sparsity pattern. Finally, the special sparsity properties of problems with multiple participants, like multi aircraft problems, are briefly introduced. The main sparsity considerations for this particular class of problems can be found with the respective example, in section 10.2.

One aspect that makes the solution of high fidelity optimal control problems so challenging, is that normally high fidelity simulation models feature multiple time scales. In order to properly resolve the dynamic effects of these systems, many discretization nodes for the state and control histories are required, posing a numerical challenge for solving the discretized problem. First, section 8.1 gives an overview of already existing methods for handling this issue. In section 8.2 a new approach is presented that combines a multiple shooting scheme for the fast part of a dynamic system with a collocation scheme for the slow part of a dynamic system. This way, the number of discretization points – and consequently also the number of

optimization parameters – is reduced without deteriorating the solution accuracy. The benefit that can be achieved by this method is presented in the example in section 11.4.

In the last theoretical chapter, a novel control grid refinement method is presented that uses physical properties of the system under consideration. After an overview of the different grid refinement methods in section 9.1, the idea of control grid refinement based on density functions is recapitulated. In section 9.3, a physically motivated density function for point mass aircraft models is presented. An enhanced strategy for selecting an appropriate number of grid points and an appropriate scaling for the density function is introduced in section 9.4. This strategy formulates another small optimization problem for determining the relevant quantities. An illustrative example for the method is presented in section 11.8.

In chapter 10 the exemplary aircraft trajectory optimization problems that are related to *ATM* are collected. After a short introduction of the challenges to be approached with the methods presented here, a brief overview of other work is given. Next, the optimization of approach scenarios including up to 18 aircraft, configuration changes, and disturbances is presented in section 10.2. In the first subsections of this section, the models used and the constraints to be respected are presented. Then, the combination of the models for the individual participants to one overall simulation model and the modeling of the resulting separation constraints are discussed. Reconsidering the thoughts on problem sparsity from chapter 7, a detailed analysis of the sparsity in the particular case is given. Besides, the homotopy process for multi aircraft problems from section 5.1.2 is employed. Finally, the results for the scenario are presented and discussed. In section 10.3 another problem involving multiple aircraft is described. Therein, airliners are refueled by tanker aircraft in order to improve the overall fuel efficiency. The models and the constraints are introduced, before the redesign of a long haul aircraft is considered, as aerial refueling during the flight changes the general performance requirements for such an aircraft. Similar to the previously mentioned example, all individual aircraft are combined to one large optimal control problem in the next step. This time another approach is used for this purpose that is based on multiple phases. Before moving over to the next application, the results for two exemplary scenarios are presented and discussed. The first scenario includes one aircraft being refueled once, while in the second two aircraft are refueled by the same tanker aircraft. In the last example related to *ATM*, the fairness within airspace sectors is investigated based on multi criteria optimal control. Once more, the models, the constraints, and the construction of the overall problem are shown in the first parts of the section. Before presenting the results for two example scenarios and discussing them, the different scalarization techniques shown in section 3.3 are adopted to the particular problem to be solved.

All illustrative examples related to air races can be found in chapter 11. After a short introduction of a race track, that is used for the first couple of examples, the respective simulation models are described in detail in section 11.1. The results and the achievable performance for a low fidelity optimization based on a point mass simulation model can be found in section 11.2, while those for a high fidelity rigid body model are in section 11.3.

The analysis of the results uses the derivations from section 3.2.5 for an estimation of the *Hamiltonian* and the optimality of the underlying continuous optimal control problem. In section 11.4 the results for the same problem, this time solved using the combined collocation and shooting approach from section 8.2, are presented. Afterwards, section 11.5 compares the outcomes and the performance for the different model fidelities. The extended homotopy idea from section 5.1.3 is applied to another air race optimal control problem in section 11.6. In section 11.7, a further application demonstrates the method based on the *RTS* smoother that was introduced in section 5.2. Finally, in the last section of the chapter, the control grid refinement introduced in sections 9.3 and 9.4 is exemplarily demonstrated.

The work conducted in this research and the respective results that have been achieved are summarized in chapter 12, before the conclusions drawn throughout the thesis are recalled and a perspective on work yet to be done is given.

This thesis contains two appendices. In appendix A the coordinate frames used, and partially introduced in chapter 2, are listed in more detail. The scientific work that has been published by this author during the time at the Institute of Flight System Dynamics at TUM is listed in appendix B.

Chapter 2

Aircraft Modeling

This chapter focuses on the mathematical modeling of the dynamic behavior of a flight system. The goal of the modeling is the derivation of a system of (non-linear) differential equations that can be used to simulate and consequently also optimize trajectories for these systems. The dynamics are modeled in two different levels of fidelity, being motivated by the principles of the flight physics.

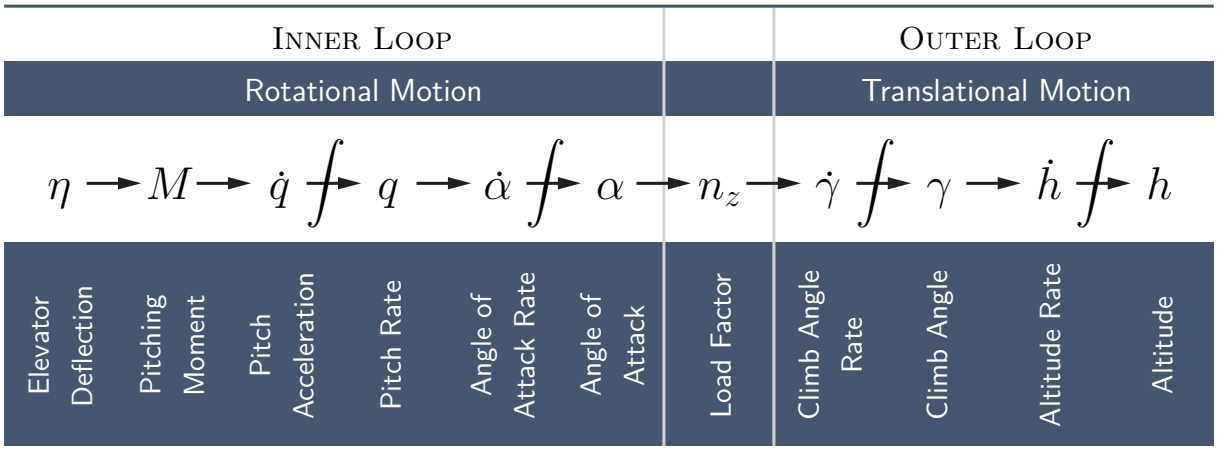

Figure 2.1: *Causal chain from an elevator deflection to an altitude change. [Fis11]*

Figure 2.1 depicts the exemplary physical causal chain of the vertical movement of an aircraft, that is flying with zero bank angle, between an elevator deflection η and the resulting change in altitude h. Whenever the pilot deflects the elevator of the aircraft, the lift at the elevator changes, which for a negative deflection leads to a positive pitching moment M. This moment produces a pitch acceleration $\dot{q}$ finally leading to a positive pitch rate q after one time integration. As a consequence to the aircraft rotating its nose up, an angle of attack rate $\dot{\alpha}$ is created. This rate – after another time integration – leads to an increase in the angle of attack α. Now, the aerodynamic situation for the whole aircraft changes, inducing an additional lift force and consequently also an additional load factor n_z. A load factor that is oriented perpendicular to the aircraft's velocity – as it is the case here – bends the flight

trajectory, leading to a climb angle rate $\dot{\gamma}$ which is integrated to a change in the climb angle γ. As a change in the climb angle directly correlates with a changed altitude rate $\dot{h}$, the aircraft finally moves upwards and reaches its new altitude h. The first part of this chain, ranging from the elevator deflection to the load factor incorporates two time integrations. It covers the rotational motion of the aircraft dynamics and is also called *inner loop*. Typically, it shows much faster dynamic behavior than the *outer loop*, representing the translational movement of an aircraft and ranging from the load factor to the change in altitude with another two time integrations.

When using the knowledge that the rotational dynamics are much faster than the translational dynamics, for trajectory analysis, the aircraft may be approximated as a point mass traveling through the atmosphere. This assumption appears to be valid as e.g. for an airliner pitching the aircraft up or down several degrees is only a matter of seconds, whereas climbing several hundred meters may be a matter of minutes. For high performance aircraft the timescales are scaled down accordingly, but the difference in the timescales remains. In the simplified point mass approach the aircraft model only features a total of three degrees of freedom, being the *translational* motion. Consequently, this aircraft model is often also called *3-degree-of-freedom* or *3-DoF* model. As all degrees of freedom are of second order, a basic point mass model features six states.

On the contrary, in some applications the approximation of the aircraft as a point mass is not sufficient, requiring the model to comprise the full causal chain as described above. In this case, one speaks of the *6-degree-of-freedom* or *6-DoF* model, also called rigid body model. Additionally to the degrees of freedom specified above, the rotational motion of the aircraft, being the *roll*, *bank*, and *yaw* motion, is considered now. Again, all degrees of freedom are of second order, now leading to a baseline of twelve states. When using conventional simulation or optimization techniques the whole simulation of these (something like) *stiff* dynamics needs to be done on a comparably fine grid as otherwise the fast rotational dynamics cannot be resolved accurately (section 3.2.4 gives more details on numerical simulation in general, while chapter 8 suggests an approach for handling optimal control problems with dynamics with different time scales).

As flight system dynamics have been a field of research and analysis for several decades now, there is plenty of literature available on the topic. More details can e.g. be found in [BAL11, Hol04, Hol15, Hol+15, Fis11, Höc14, Len15, Ros79, ER96, Zip07].

The remainder of this chapter is structured as follows: First the nomenclature to be used is briefly introduced before afterwards the required coordinate frames and transformation matrices are recalled. More details on the relevant coordinate frames can be found in appendix A. In section 2.3 the modeling requirements are stated. Afterwards, the equations of motion for the point mass simulation model and those needed additionally to build up the full rigid body dynamics are recapitulated. Furthermore, the influence of wind on the dynamics of each of the models is shortly recapitulated. At this point the pure dynamics and kinematics of the motion of an aircraft are fully stated. However, the calculation of external forces and moments

is still an open point. Hence, these effects are discussed in section 2.7. A basic model for the aerodynamic behavior of an aircraft is given in 2.7.1, while 2.7.2 repeats some propulsion models. In 2.7.3 the modeling of the gravitational force is discussed, before this section is closed by models for the fuel consumption in 2.8.1 and some for actuator dynamics in 2.8.2. The chapter is concluded by a short revision of the standard atmosphere model.

2.1 Nomenclature and Relevant Quantities

In order to be able to uniquely identify all quantities required to model the motion of an aircraft, a set of nomenclature principles and indices is used throughout this work. When considering physical quantities, vectors are printed using bold characters, while for scalars regular characters are used. Angles and angular rates use mainly Greek letters, while quantities that are related to linear motion use Latin letters.

For scalar quantities, the following index notation is used

$$\text{quantity}_{\text{type of quantity}}^{\text{reference point(s)}}, \tag{2.1}$$

where the type of the quantity may e.g. be aerodynamic or kinematic. A missing index for the type indicates an overall force or moment, or a kinematic quantity, respectively.

Vector valued quantities in *Euclidean* space feature additional properties as they may be written with respect to different coordinate frames. Moreover, they may express a change over time relative to a coordinate frame. Consequently, vectors are denoted as

$$\left(\text{vector}_{\text{type of quantity}}^{\text{reference point(s)}}\right)_{\text{notational frame}}^{\text{derivative frame(s)}} \tag{2.2}$$

The vector notation is also used in combination with scalars in case they are single components of a vector requiring the specification of reference frames. For plain scalar quantities, like e.g. the kinematic velocity V_K or the angles χ_K and γ_K, that are uniquely defined, the vector notation is omitted. The notation scheme is based on the one used at the *Institute of Flight System Dynamics at TUM* and can e.g. be found in [Hol15].

In the remainder of this work, the index P denotes an arbitrary point on the body of the aircraft, while R designates the *Aircraft Reference Point* and G the center of gravity of the aircraft.

Table 2.1 lists selected quantities that can be used as states within the dynamic model of the aircraft. They are grouped by their physical meaning where in every group one subgroup may be selected for modeling the respective physical effects. Special care has to be taken as not every combination of subgroups is possible.

2.2 Coordinate Frames and Transformations

In order to model the flight system dynamics, a set of coordinate frames and transformations between them is required. Figure 2.2 lists the main coordinate frames used in the derivation.

Table 2.1: *Relevant quantities for modeling aircraft dynamics with their default units.*

Position

Symbol	Description	Unit
λ	Geodetic longitude	rad
φ	Geodetic latitude	rad
h	*WGS84* altitude	m
$(x)_N$	x-position in N Frame	m
$(y)_N$	y-position in N Frame	m
$(z)_N$	Downward position	m

(Kinematic) Velocity

Symbol	Description	Unit
$(u_K)_O^E$	Northward velocity	m/s
$(v_K)_O^E$	Eastward velocity	m/s
$(w_K)_O^E$	Downward velocity	m/s
V_K	Absolute velocity	m/s
χ_K	Course angle	rad
γ_K	Climb angle	rad
V_K	Absolute velocity	m/s
α_K	Kin. angle of attack	rad
β_K	Kin. angle of sideslip	rad

Attitude

Symbol	Description	Unit
Ψ	Azimuth angle	rad
Θ	Pitch angle	rad
Φ	Roll angle	rad
α_K	Kin. angle of attack	rad
β_K	Kin. angle of sideslip	rad
μ_K	Kin. bank angle	rad
q_0	First quaternion	-
q_1	Second quaternion	-
q_2	Third quaternion	-
q_3	Fourth quaternion	-

(Kinematic) Angular Rate

Symbol	Description	Unit
p_K	Roll rate	rad/s
q_K	Pitch rate	rad/s
r_K	Yaw rate	rad/s
$\left(\omega_x^{IB}\right)_B$	Inertial ang. rate in x_B	rad/s
$\left(\omega_y^{IB}\right)_B$	Inertial ang. rate in y_B	rad/s
$\left(\omega_z^{IB}\right)_B$	Inertial ang. rate in z_B	rad/s

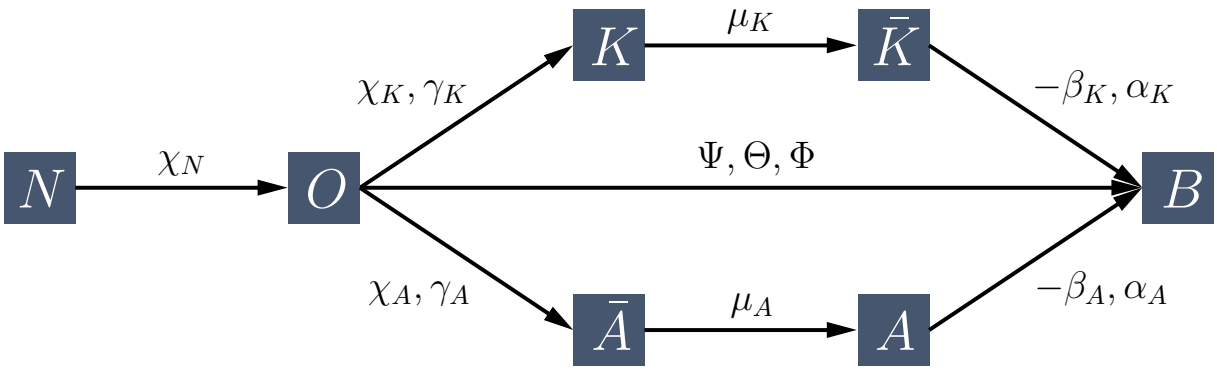

Figure 2.2: *Main coordinate frames required to model the dynamics of a flight system and the angles between them. [Hol15]*

Starting from the left it shows the *Navigational Frame* N which is still fixed to the surface of the earth, whereas the next frame, the *North-East-Down Frame* O is fixed to the aircraft and marks the starting point for several ways of describing the aircrafts attitude. These ways are via the *Kinematic Frame* K and the *Rotated Kinematic Frame* $\bar{K}$ on the top, via the *Rotated Aerodynamic Frame* $\bar{A}$ and the *Aerodynamic Frame* A on the bottom, or directly to the *Body Fixed Frame* B using the EULER angles (or more precise the TAIT-BRYAN angles) Ψ, Θ, Φ in the center. More details on the coordinate frames can be found in appendix A.

During the formulation of the aircraft dynamics in the next sections, several Cartesian vectors need to be transformed between different coordinate frames. Using the predefined sequences of rotation for each of these transformations, the following transformation matrices and rotation vectors can be derived. The coordinate frames and transformations used here can similarly be found e.g. in [Hol15, Hol+15, Höc14, Fis11].

For the transformation from the *North-East-Down Frame* to the *Navigational Frame*

$$\mathbf{M}_{NO} = \begin{bmatrix} \cos\chi_N & \sin\chi_N & 0 \\ -\sin\chi_N & \cos\chi_N & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad \text{and} \qquad (\boldsymbol{\omega}^{NO})_O = \begin{pmatrix} 0 \\ 0 \\ \dot{\chi}_N \end{pmatrix}_O \tag{2.3}$$

are used. The rotational rate between the frames can generally be derived using the *strapdown equation*.

In order to transform the coordinate frame O into the K-frame, first a rotation about χ_K is required and afterwards the resulting frame needs to be rotated around γ_K. The overall coordinate transformation can be achieved by the transformation matrix

$$\mathbf{M}_{OK} = \begin{bmatrix} \cos\chi_K\cos\gamma_K & -\sin\chi_K & \cos\chi_K\sin\gamma_K \\ \sin\chi_K\cos\gamma_K & \cos\chi_K & \sin\chi_K\sin\gamma_K \\ -\sin\gamma_K & 0 & \cos\gamma_K \end{bmatrix} \tag{2.4}$$

and the rotational rate:

$$\left(\boldsymbol{\omega}^{OK}\right)_K = \begin{pmatrix} -\dot{\chi}_K\sin\gamma_K \\ \dot{\gamma}_K \\ \dot{\chi}_K\cos\gamma_K \end{pmatrix}_K \tag{2.5}$$

For performing the transformation to the *Body Fixed Frame* B from the K frame, three more rotations are necessary that can be combined into the matrix:

$$\mathbf{M}_{KB} = \begin{bmatrix} \cos\alpha_K\cos\beta_K & \sin\beta_K & \sin\alpha_K\cos\beta_K \\ -\cos\alpha_K\sin\beta_K\cos\mu_K + \sin\alpha_K\sin\mu_K & \cos\beta_K\cos\mu_K & -\sin\alpha_K\sin\beta_K\cos\mu_K - \cos\alpha_K\sin\mu_K \\ -\cos\alpha_K\sin\beta_K\sin\mu_K + \sin\alpha_K\cos\mu_K & \cos\beta_K\sin\mu_K & -\sin\alpha_K\sin\beta_K\sin\mu_K + \cos\alpha_K\cos\mu_K \end{bmatrix} \tag{2.6}$$

The rotational rate between the two frames consequently is:

$$\left(\boldsymbol{\omega}^{KB}\right)_B = \begin{pmatrix} \dot{\mu}_K \cos\alpha_K \cos\beta_K + \dot{\beta}_K \sin\alpha_K \\ \dot{\alpha}_K + \dot{\mu}_K \sin\beta_K \\ -\dot{\beta}_K \cos\alpha_K + \dot{\mu}_K \sin\alpha_K \cos\beta_K \end{pmatrix}_B \tag{2.7}$$

Similarly to the path along the kinematic transformations, the *Body Fixed Frame* can also be reached starting from the *NED Frame* and using the aerodynamic angles. As the z-axis of the *Aerodynamic Frame* is already rotated into the symmetry plane of the aircraft, the transformation matrix

$$\mathbf{M}_{OA} = \begin{bmatrix} \cos\chi_A \cos\gamma_A & \cos\chi_A \sin\gamma_A \sin\mu_A - \sin\chi_A \cos\mu_A & \cos\chi_A \sin\gamma_A \cos\mu_A + \sin\chi_A \sin\mu_A \\ \sin\chi_A \cos\gamma_A & \sin\chi_A \sin\gamma_A \sin\mu_A + \cos\chi_A \cos\mu_A & \sin\chi_A \sin\gamma_A \cos\mu_A - \cos\chi_A \sin\mu_A \\ -\sin\gamma_A & \cos\gamma_A \sin\mu_A & \cos\gamma_A \cos\mu_A \end{bmatrix} \tag{2.8}$$

results. Again, the rotational rate between the two coordinate frames can be derived from this matrix as:

$$\left(\boldsymbol{\omega}^{OA}\right)_O = \begin{pmatrix} \dot{\mu}_A \cos\chi_A \cos\gamma_A - \dot{\gamma}_A \sin\chi_A \\ \dot{\mu}_A \sin\chi_A \cos\gamma_A + \dot{\gamma}_A \cos\chi_A \\ \dot{\chi}_A - \dot{\mu}_A \sin\gamma_A \end{pmatrix}_O \tag{2.9}$$

Starting from the *Aerodynamic Frame*, the *Body Fixed Frame* can be reached by first rotating around $-\beta_A$ and afterwards around α_A. The transformation matrix that results from this sequence is:

$$\mathbf{M}_{AB} = \begin{bmatrix} \cos\alpha_A \cos\beta_A & \sin\beta_A & \sin\alpha_A \cos\beta_A \\ -\cos\alpha_A \sin\beta_A & \cos\beta_A & -\sin\alpha_A \sin\beta_A \\ -\sin\alpha_A & 0 & \cos\alpha_A \end{bmatrix} \tag{2.10}$$

The rotational rate between the two frames is:

$$\left(\boldsymbol{\omega}^{AB}\right)_B = \begin{pmatrix} \dot{\beta}_A \sin\alpha_A \\ \dot{\alpha}_A \\ -\dot{\beta}_A \cos\alpha_A \end{pmatrix}_B \tag{2.11}$$

When using the EULER angles to directly transform the *North-East-Down Frame* into the *Body Fixed Frame*, three rotations need to be coupled together, resulting in the transformation matrix:

$$\mathbf{M}_{BO} = \begin{bmatrix} \cos\Psi \cos\Theta & \sin\Psi \cos\Theta & -\sin\Theta \\ \cos\Psi \sin\Theta \sin\Phi - \sin\Psi \cos\Phi & \sin\Psi \sin\Theta \sin\Phi - \cos\Psi \cos\Phi & \cos\Theta \sin\Phi \\ \cos\Psi \sin\Theta \cos\Phi - \sin\Psi \sin\Phi & \sin\Psi \sin\Theta \cos\Phi - \cos\Psi \sin\Phi & \cos\Theta \cos\Phi \end{bmatrix} \tag{2.12}$$

The rotational rate of the *Body Fixed Frame* with respect to the *North-East-Down Frame* can be derived as:

$$\left(\boldsymbol{\omega}^{OB}\right)_B = \begin{pmatrix} \dot{\Phi} - \dot{\Psi} \cdot \sin\Theta \\ \dot{\Theta}\cos\Phi + \dot{\Psi}\sin\Phi\cos\Theta \\ -\dot{\Theta}\sin\Phi + \dot{\Psi}\cos\Phi\cos\Theta \end{pmatrix}_B = \begin{pmatrix} p_K \\ q_K \\ r_K \end{pmatrix} \tag{2.13}$$

The kinematic body rates p_K, q_K and r_K may also be used as states.

Besides the transformations presented above, the one between the *Earth-Centered Earth Fixed Frame* and the *North-East-Down Frame* is required for the derivation of the equations of motion of an aircraft. The transformation matrix is found to be

$$\mathbf{M}_{OE} = \begin{bmatrix} -\sin\varphi\cos\lambda & -\sin\varphi\sin\lambda & \cos\varphi \\ -\sin\lambda & \cos\lambda & 0 \\ -\cos\varphi\cos\lambda & -\cos\varphi\sin\lambda & -\sin\varphi \end{bmatrix} \tag{2.14}$$

while the *transport rate*, that keeps the *North-East-Down Frame* always aligned to the North, the East and downwards, is:

$$\left(\boldsymbol{\omega}^{EO}\right)_E = \begin{pmatrix} \dot{\varphi}\sin\lambda \\ -\dot{\varphi}\cos\lambda \\ \dot{\lambda} \end{pmatrix}_E \tag{2.15}$$

2.3 Modeling Requirements

When building a mathematical model to represent the behavior of real systems, some assumptions have to be made in order to keep complexity at a reasonable level. The general principle one needs to follow is *"as complex as necessary, as simple as possible"*. Here, the following assumptions are made in order to achieve a reasonable model complexity (similar assumptions are also made in [Fis11, Höc14, Hol15]).

- The mass of the flight system is considered to be quasi-steady. This means that even if fuel consumption is considered, the influence of any exhaust mass on the translation equations of motion is neglected.
- Aircraft are considered to be rigid bodies, meaning that no shifting of mass elements inside the body is allowed. Consequently, the mass distribution is assumed to be static.
- The reference point R is selected to be the center of gravity G:

$$R = G \tag{2.16}$$

- The *Earth Centered Inertial Frame* I is considered to be an *Euclidean* frame in which NEWTON's second law may be applied.
- The rotation of the earth $\boldsymbol{\omega}^{IE}$, modeled between the *ECEF*-frame and the *ECI*-frame, is assumed to be constant. Consequently, the respective rotational acceleration is zero:

$$\left(\dot{\boldsymbol{\omega}}^{IE}\right)^E = \left(\dot{\boldsymbol{\omega}}^{IE}\right)^I = 0 \tag{2.17}$$

More details on the errors resulting from some of these assumptions can be found in [Len15].

2.4 Point Mass Simulation Model

As stated above, for most real aircraft configurations, the rotational dynamics are much faster than the translational dynamics, allowing for the simplified modeling as a point mass moving in three dimensional space. In this model, the rotational dynamics are always seen to be decayed. All equations used in the next sections are intentionally not fully simplified according to the point mass assumptions as they are intended to stay valid for the rigid body case.

In order to be able to model the point mass dynamics, three degrees of freedom – all being second order – are necessary resulting in a model comprising six states. As several modeling alternatives exist, depending on the applications and any inherent additional assumptions, the states need to be selected from table 2.1 appropriately. The following sections give a short overview of the equations of motion resulting for the different formulations. The model equations presented here are mostly taken from [Hol15, Hol+15, Fis11, Höc14].

2.4.1 Position Propagation Equations of Motion

In a general aircraft simulation model, the earth has to be considered round and rotating. Then, the position equations of motion are normally given in *WGS84* coordinates [Nat00], using the *ECEF Frame* as a reference frame and using the geodetic longitude λ, the geodetic latitude φ and the geodetic altitude h as coordinates and consequently also as states in the model. In the *WGS84* model, the earth is approximated by a rotational ellipsoid that features the following properties: The length of the *semi major axis* is

$$a = 6378137.0m, \tag{2.18}$$

the *flattening* is

$$f = \frac{1}{298.257223563}, \tag{2.19}$$

while the *semi minor axis* is

$$b = a \cdot (1 - f) \tag{2.20}$$

and the *eccentricity* is:

$$e^2 = 2f - f^2 \tag{2.21}$$

Hence, the differential equation for the position (of the center of gravity of the aircraft) can be formulated as

$$\begin{pmatrix} \dot{\lambda} \\ \dot{\varphi} \\ \dot{h} \end{pmatrix}^E = \begin{pmatrix} \frac{\left(v_K^G\right)_O^E}{(N_\varphi + h) \cdot \cos\varphi} \\ \frac{\left(u_K^G\right)_O^E}{M_\varphi + h} \\ -\left(w_K^G\right)_O^E \end{pmatrix} = \begin{pmatrix} \frac{V_K \cdot \sin\chi_K \cdot \cos\gamma_K}{(N_\varphi + h) \cdot \cos\varphi} \\ \frac{V_K \cdot \cos\chi_K \cdot \cos\gamma_K}{M_\varphi + h} \\ V_K \cdot \sin\gamma_K \end{pmatrix}, \tag{2.22}$$

with

$$N_\varphi = \frac{a}{\sqrt{1 - e^2 \sin^2 \varphi}} \tag{2.23}$$

being the *Radius of Curvature in the Prime Vertical* and

$$M_\varphi = N_\varphi \cdot \frac{1 - e^2}{1 - e^2 \sin^2 \varphi} = \frac{a \cdot (1 - e^2)}{(1 - e^2 \sin^2 \varphi)^{\frac{3}{2}}} \tag{2.24}$$

being the *Meridian Radius of Curvature*.

In case flight trajectories of a small spatial extent are considered (Brockhaus suggests a range of less than 100 NM in [BAL11, Sec. 2.4]), the earth may also be considered as flat. If the kinematic velocity of the considered system is also low enough (Lenz presents some numerical studies on error estimations for different velocities in Appendix A of [Len15]), the earth may besides be considered as non-rotating. In this case the position equations of motion for the aircraft can be given with respect to the *Navigational Frame* fixed at a reference point on the surface of the earth. The resulting dynamics are

$$\begin{aligned}
\begin{pmatrix} \dot{x}^G \\ \dot{y}^G \\ \dot{z}^G \end{pmatrix}_N^E &= \left(\mathbf{V}_K^G\right)_N^E = \mathbf{M}_{NO} \cdot \mathbf{M}_{OK} \cdot \begin{pmatrix} V_K \\ 0 \\ 0 \end{pmatrix}_K^E \\
&= \mathbf{M}_{NO} \cdot \begin{pmatrix} V_K \cdot \cos\chi_K \cdot \cos\gamma_K \\ V_K \cdot \sin\chi_K \cdot \cos\gamma_K \\ -V_K \cdot \sin\gamma_K \end{pmatrix}_O^E \\
&= \begin{pmatrix} V_K \cdot \cos(\chi_K - \chi_N) \cdot \cos\gamma_K \\ V_K \cdot \sin(\chi_K - \chi_N) \cdot \cos\gamma_K \\ -V_K \cdot \sin\gamma_K \end{pmatrix}_N^E ,
\end{aligned} \tag{2.25}$$

where $\chi_N = 0$ and consequently $(\mathbf{V}_K^G)_O^E = (\mathbf{V}_K^G)_N^E$ in case the *Navigational Frame* is oriented like the *North-East-Down Frame*.

More details on these models can besides many others be found in [Fis11, Hol15, Hol+15, Len15].

2.4.2 Translation Equations of Motion

The translation equations of motion lead to the time derivative of the Cartesian velocity vector for the center of gravity G of the aircraft (also being the reference point R). The derivation presented here can similarly be found in e.g. [Hol15, Höc14, BAL11, Len15].

The derivation is started with the change of the velocity of the aircraft with respect to the *NED* frame O, where the velocity itself represents the change of the position of the aircraft with respect to the *Earth-centered Earth Fixed Frame* E:

$$\left(\frac{d}{dt}\right)^O \left(\mathbf{V}_K^G\right)^E = \left(\dot{\mathbf{V}}_K^G\right)^{EO} = \left(\dot{\mathbf{V}}_K^R\right)^{EO} \tag{2.26}$$

In general, the translation equations of motion are derived using NEWTON's second law applied in an inertial frame, which is the *ECI Frame* I in this case:

$$\sum \mathbf{F}^G = \left(\frac{d}{dt}\right)^I \mathbf{p}^G = \left(\frac{d}{dt}\right)^I \int_m \left(\mathbf{V}_K^P\right)^I \, dm \tag{2.27}$$

with $\mathbf{p}^G$ being the *linear momentum* acting in the center of gravity of the aircraft. Using the vector chain

$$\left(\mathbf{x}^P\right) = \left(\mathbf{x}^G\right) + \left(\mathbf{x}^{GP}\right) \tag{2.28}$$

and EULER's derivative rule, the velocity of an arbitrary point on the aircraft can be calculated as:

$$\left(\mathbf{V}_K^P\right)^I = \left(\dot{\mathbf{x}}^G\right)^I + \left(\dot{\mathbf{x}}^{GP}\right)^I = \left(\mathbf{V}_K^G\right)^I + \left(\dot{\mathbf{x}}^{GP}\right)^B + \left(\boldsymbol{\omega}^{IB}\right) \times \left(\mathbf{x}^{GP}\right) \tag{2.29}$$

The rigid body assumption leads to

$$\left(\frac{d}{dt}\right)^B \left(\mathbf{x}^{GP}\right) = \left(\dot{\mathbf{x}}^{GP}\right)^B = 0, \tag{2.30}$$

and consequently, NEWTON's second law becomes

$$\begin{aligned} \sum \mathbf{F}^G = \left(\frac{d}{dt}\right)^I \mathbf{p}^G &= \left(\frac{d}{dt}\right)^I \int_m \left(\mathbf{V}_K^G\right)^I + \left(\boldsymbol{\omega}^{IB}\right) \times \left(\mathbf{x}^{GP}\right) \, dm \\ &= \left(\frac{d}{dt}\right)^I \left(\int_m \left(\mathbf{V}_K^G\right)^I \, dm + \left(\boldsymbol{\omega}^{IB}\right) \times \int_m \left(\mathbf{x}^{GP}\right) \, dm \right) \end{aligned} \tag{2.31}$$

with the definition of the center of gravity

$$\int_m \left(\mathbf{x}^{GP}\right) \, dm = 0, \tag{2.32}$$

and the fact that $\left(\mathbf{V}_K^G\right)^I$ is independent of the mass, the equations of motion result in:

$$\left(\dot{\mathbf{V}}_K^G\right)^{II} = \frac{1}{m} \cdot \sum \mathbf{F}^G \tag{2.33}$$

Now, the expression on the left needs further investigation, as we are looking for a relationship to calculate the time derivative $\left(\dot{\mathbf{V}}_K^G\right)^{EO}$ of the velocity. The total acceleration $\left(\dot{\mathbf{V}}_K^G\right)^{II}$ of

the aircraft's center of gravity can be written as:

$$\begin{aligned}\left(\dot{\mathbf{V}}_K^G\right)^{II} &= \left(\frac{d}{dt}\right)^I \left(\mathbf{V}_K^G\right)^I \\ &= \left(\frac{d}{dt}\right)^I \left(\left(\mathbf{V}_K^G\right)^E + \left(\boldsymbol{\omega}^{IE}\right) \times \left(\mathbf{x}^G\right)\right) \\ &= \underbrace{\left(\dot{\mathbf{V}}_K^G\right)^{EE} + \left(\boldsymbol{\omega}^{IE}\right) \times \left(\mathbf{V}_K^G\right)^E}_{\left(\dot{\mathbf{v}}_K^G\right)^{EI}} + \underbrace{\left(\dot{\boldsymbol{\omega}}^{IE}\right)^I}_{=0} \times \left(\mathbf{x}^G\right) \\ &\qquad + \left(\boldsymbol{\omega}^{IE}\right) \times \underbrace{\left(\left(\mathbf{V}_K^G\right)^E + \left(\boldsymbol{\omega}^{IE}\right) \times \left(\mathbf{x}^G\right)\right)}_{\left(\mathbf{v}_K^G\right)^I} \\ &= \left(\dot{\mathbf{V}}_K^G\right)^{EE} + 2 \cdot \left(\boldsymbol{\omega}^{IE}\right) \times \left(\mathbf{V}_K^G\right)^E + \left(\boldsymbol{\omega}^{IE}\right) \times \left(\left(\boldsymbol{\omega}^{IE}\right) \times \left(\mathbf{x}^G\right)\right) \end{aligned} \tag{2.34}$$

Using the EULER derivative rule for the first element

$$\left(\dot{\mathbf{V}}_K^G\right)^{EE} = \left(\dot{\mathbf{V}}_K^G\right)^{EO} + \left(\boldsymbol{\omega}^{EO}\right) \times \left(\mathbf{V}_K^G\right)^E, \tag{2.35}$$

the equations of motion can further be reformulated to:

$$\begin{aligned}\left(\dot{\mathbf{V}}_K^G\right)^{II} = \underbrace{\left(\dot{\mathbf{V}}_K^G\right)^{EO}}_{A} + \underbrace{\left(\boldsymbol{\omega}^{EO}\right) \times \left(\mathbf{V}_K^G\right)^E}_{B} + \underbrace{2 \cdot \left(\boldsymbol{\omega}^{IE}\right) \times \left(\mathbf{V}_K^G\right)^E}_{C} \\ + \underbrace{\left(\boldsymbol{\omega}^{IE}\right) \times \left(\left(\boldsymbol{\omega}^{IE}\right) \times \left(\mathbf{x}^G\right)\right)}_{D}\end{aligned} \tag{2.36}$$

Physically, the expression

- A represents the kinematic acceleration of the aircraft with respect to the *NED Frame*,
- B represents the kinematic acceleration due to the *transport rate*,
- C represents the CORIOLIS acceleration due to the rotation of the earth and
- D represents the *centripetal* acceleration due to the rotation of the earth.

Using NEWTON's second law from equation (2.31), the acceleration of the aircraft with respect to the *NED Frame* becomes:

$$\begin{aligned}\begin{pmatrix}\dot{u}_K \\ \dot{v}_K \\ \dot{w}_K\end{pmatrix}^{EO} = \left(\dot{\mathbf{V}}_K^G\right)^{EO} = \frac{1}{m}\sum \mathbf{F}^G - \left(\boldsymbol{\omega}^{EO}\right) \times \left(\mathbf{V}_K^G\right)^E - 2 \cdot \left(\boldsymbol{\omega}^{IE}\right) \times \left(\mathbf{V}_K^G\right)^E \\ - \left(\boldsymbol{\omega}^{IE}\right) \times \left(\left(\boldsymbol{\omega}^{IE}\right) \times \left(\mathbf{x}^G\right)\right)\end{aligned} \tag{2.37}$$

In case the effects of a round earth are neglected, $\left(\boldsymbol{\omega}^{EO}\right) = 0$ and consequently expression B vanishes. Whenever the rotation of the earth is neglected, $\left(\boldsymbol{\omega}^{IE}\right) = 0$ and expressions C and D vanish. For the most simple case of a flat and non-rotating earth

$$\left(\dot{\mathbf{V}}_K^G\right)^{EO} = \frac{1}{m}\sum \mathbf{F}^G \tag{2.38}$$

results. Furthermore, in case the equations are written down with respect to the K-Frame, the relation

$$\begin{aligned}\left(\dot{\mathbf{V}}_K^G\right)_K^{EO} &= \left(\dot{\mathbf{V}}_K^G\right)_K^{EK} + \left(\boldsymbol{\omega}^{OK}\right)_K \times \left(\mathbf{V}_K^G\right)_K^E \\ &= \begin{pmatrix} \dot{V}_K^G \\ 0 \\ 0 \end{pmatrix}_K^{EK} + \begin{pmatrix} -\dot{\chi}_K \cdot \sin\gamma_K \\ \dot{\gamma}_K \\ \dot{\chi}_K \cos\gamma_K \end{pmatrix}_K \times \begin{pmatrix} V_K \\ 0 \\ 0 \end{pmatrix}_K^E \\ &= \begin{pmatrix} \dot{V}_K \\ V_K \cdot \dot{\chi}_K \cdot \cos\gamma_K \\ -V_K \cdot \dot{\gamma}_K \end{pmatrix}_K^{EO} \end{aligned} \tag{2.39}$$

can be derived. Using equations (2.38) and (2.39), the well-known translation equations of motion for the absolute kinematic velocity and the flight path angles result as:

$$\dot{V}_K = \frac{1}{m}\left(F_x^G\right)_K \tag{2.40}$$

$$\dot{\chi}_K = \frac{1}{m \cdot V_K \cdot \cos\gamma_K}\left(F_y^G\right)_K \tag{2.41}$$

$$\dot{\gamma}_K = -\frac{1}{m \cdot V_K}\left(F_z^G\right)_K \tag{2.42}$$

When using an approach that is more focused on the *Body Fixed Frame* and the angles between the $\bar{K}$-Frame and the B-Frame, for a flat and non-rotating earth, V_K, α_K and β_K may also be used to describe the aircraft's translational state. Starting with equation 2.39 denoted in the rotated kinematic frame $\bar{K}$:

$$\begin{aligned}\left(\dot{\mathbf{V}}_K^G\right)_{\bar{K}}^{EO} &= \left(\dot{\mathbf{V}}_K^G\right)_{\bar{K}}^{E\bar{K}} + \left(\boldsymbol{\omega}^{O\bar{K}}\right)_{\bar{K}} \times \left(\mathbf{V}_K^G\right)_{\bar{K}}^E \\ &= \left(\dot{\mathbf{V}}_K^G\right)_{\bar{K}}^{E\bar{K}} + \left(\left(\boldsymbol{\omega}^{OB}\right)_{\bar{K}} + \left(\boldsymbol{\omega}^{B\bar{K}}\right)_{\bar{K}}\right) \times \left(\mathbf{V}_K^G\right)_{\bar{K}}^E \\ &= \left(\dot{\mathbf{V}}_K^G\right)_{\bar{K}}^{E\bar{K}} + \left(\mathbf{M}_{\bar{K}B}\left(\boldsymbol{\omega}^{OB}\right)_B - \left(\boldsymbol{\omega}^{\bar{K}B}\right)_K\right) \times \left(\mathbf{V}_K^G\right)_{\bar{K}}^E \\ &= \begin{pmatrix} \dot{V}_K^G \\ 0 \\ 0 \end{pmatrix}_{\bar{K}}^{E\bar{K}} \\ &\quad + \begin{pmatrix} \sin\beta_K \cdot (q_K - \dot{\alpha}_K) + \cos\beta_K \cdot (r_K \cdot \sin\alpha_K + p_K \cdot \cos\alpha_K) - \dot{\mu}_K \\ \cos\beta_K \cdot (q_K - \dot{\alpha}_K) - \sin\beta_K \cdot (r_K \cdot \sin\alpha_K + p_K \cdot \cos\alpha_K) \\ \dot{\beta}_K + r_K \cdot \cos\alpha_K - p_K \cdot \sin\alpha_K \end{pmatrix}_{\bar{K}} \times \begin{pmatrix} V_K \\ 0 \\ 0 \end{pmatrix}_{\bar{K}}^E \\ &= \begin{pmatrix} \dot{V}_K \\ V_K \cdot (\dot{\beta}_K + r_K \cdot \cos\alpha_K - p_K \cdot \sin\alpha_K) \\ V_K \cdot (\cos\beta_K \cdot (\dot{\alpha}_K - q_K) + \sin\beta_K \cdot (p_K \cdot \cos\alpha_K + r_K \cdot \sin\alpha_K)) \end{pmatrix}_{\bar{K}}^{EO} \end{aligned} \tag{2.43}$$

equations of motion similar to those in (2.40)–(2.42) can be derived:

$$\dot{V}_K = \frac{1}{m}\left(F_x^G\right)_{\bar{K}} = \frac{1}{m}\left(F_x^G\right)_K \tag{2.44}$$

$$\dot{\beta}_K = \frac{1}{m \cdot V_K}\left(F_y^G\right)_{\bar{K}} + (p_K \cdot \sin\alpha_K - r_K \cdot \cos\alpha_K) \tag{2.45}$$

$$\dot{\alpha}_K = \frac{1}{m \cdot V_K \cdot \cos\beta_K}\left(F_z^G\right)_{\bar{K}} + q_K - \tan\beta_K \cdot (p_K \cdot \cos\alpha_K + r_K \cdot \sin\alpha_K) \tag{2.46}$$

In this case, special care has to be taken in the selection of the rotational and attitude states of the aircraft. The different formulations of the equations of motion have been taken from [Hol15, Fis11, Len15, Hol+15].

2.5 Rigid Body Simulation Model

In case the rotational dynamics cannot be neglected, the aircraft may be considered as a rigid body. With the same assumptions as above, the attitude and rotation differential equations – that add three more degrees of freedom and six more states to the system (seven in case Quaternions are used) – can now be derived. Again, in general, several possibilities exist for the choice of states as well as reference frames. In the following, the most common representations will be recapitulated.

As mentioned before, the rotational dynamics are normally much faster than the translational dynamics resulting in a system of differential equations that is hard to solve (it is something like *stiff*, even though the word should be used carefully). The consequence is that for accurate simulation results, all dynamic equations need to be simulated with either a numerical scheme of high order or a very small step size, which results in a big computational burden. This problem is tackled in chapter 8.

More details on the rigid body simulation model can besides others be found in [BAL11, Fis11, Len15, Hol15, Hol+15, Hol04].

2.5.1 Attitude Propagation

The attitude dynamics represent the kinematics of the aircraft's rotation, calculating the attitude derivatives from the respective rotational rates. In the conventional approach, the attitude of the aircraft is given with respect to the *NED Frame*, using the EULER angles as states. In contrary to this, FISCH proposes a new modeling approach in [Fis11] that states the attitude dynamics with respect to the *Kinematic Frame* or the kinematic flight path of the aircraft. He uses α_K, β_K and μ_K as the attitude states of the aircraft. The benefit FISCH sees in this new approach is the fact that the rotational dynamics of the aircraft can easily be changed to more simple linear models without affecting any other part of the dynamic model.

Both of the above mentioned representations feature a singularity for either a climb angle or a pitch angle of $\pm\frac{\pi}{2}$. The solution to overcome this issue is to use *Quaternions* for the

representation of the respective attitude. Here, *Quaternions* are only presented for replacing the EULER angles, while FISCH uses them to replace χ_K, γ_K and μ_K.

Equation (2.13) can be solved for the time derivatives of the EULER angles, resulting in

$$\begin{pmatrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{pmatrix} = \begin{bmatrix} 1 & \sin\Phi \cdot \tan\Theta & \cos\Phi \cdot \tan\Theta \\ 0 & \cos\Phi & -\sin\Phi \\ 0 & \frac{\sin\Phi}{\cos\Theta} & \frac{\cos\Phi}{\cos\Theta} \end{bmatrix} \cdot \underbrace{\begin{pmatrix} p_K \\ q_K \\ r_K \end{pmatrix}}_{=(\boldsymbol{\omega}^{OB})_B} \tag{2.47}$$

As can clearly be seen, the matrix in equation (2.47) shows the aforementioned singularity for $\Theta = \pm\frac{\pi}{2}$ (as $\cos\pm\frac{\pi}{2} = 0$). Consequently, trajectories featuring vertical climb or descent segments cannot be modeled this way.

The four *Quaternions* q_0, q_1, q_2 and q_3 can be seen as a three dimensional rotational axis and an angle around this axis. The attitude equation in *Quaternions* can be written as:

$$\begin{pmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{pmatrix} = \frac{1}{2} \begin{bmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & -q_3 & q_2 \\ q_3 & q_0 & -q_1 \\ -q_2 & q_1 & q_0 \end{bmatrix} \cdot \underbrace{\begin{pmatrix} p_K \\ q_K \\ r_K \end{pmatrix}}_{=(\boldsymbol{\omega}^{OB})_B} \tag{2.48}$$

The *Quaternion* vector is not uniquely defined as it features four degrees of freedom for only three physical degrees of freedom. In order to make it locally unique again, it is normalized to unit length:

$$\left\| \begin{pmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{pmatrix} \right\|_2 = 1 \tag{2.49}$$

The *Quaternion* representation of the attitude can be calculated from the EULER angles according to [SL03, p. 52]:

$$q_0 = \cos\frac{\Psi}{2}\cos\frac{\Theta}{2}\cos\frac{\Phi}{2} + \sin\frac{\Psi}{2}\sin\frac{\Theta}{2}\sin\frac{\Phi}{2} \tag{2.50}$$

$$q_1 = \cos\frac{\Psi}{2}\cos\frac{\Theta}{2}\sin\frac{\Phi}{2} - \sin\frac{\Psi}{2}\sin\frac{\Theta}{2}\cos\frac{\Phi}{2} \tag{2.51}$$

$$q_2 = \cos\frac{\Psi}{2}\sin\frac{\Theta}{2}\cos\frac{\Phi}{2} + \sin\frac{\Psi}{2}\cos\frac{\Theta}{2}\sin\frac{\Phi}{2} \tag{2.52}$$

$$q_3 = \sin\frac{\Psi}{2}\cos\frac{\Theta}{2}\cos\frac{\Phi}{2} - \cos\frac{\Psi}{2}\sin\frac{\Theta}{2}\sin\frac{\Phi}{2} \tag{2.53}$$

Conversely, the EULER angles can be calculated from:

$$\Phi = \tan^{-1}\frac{2(q_2q_3 + q_0q_1)}{q_0^2 - q_1^2 - q_2^2 + q_3^2} \tag{2.54}$$

$$\Theta = \sin^{-1}\left(-2(q_1q_3 - q_0q_2)\right) \tag{2.55}$$

$$\Psi = \tan^{-1}\frac{2(q_1q_2 + q_0q_3)}{q_0^2 + q_1^2 - q_2^2 - q_3^2} \tag{2.56}$$

The transformation matrix from the *NED Frame* to the *Body Fixed Frame* may also be written using *Quaternions*:

$$\mathbf{M}_{BO} = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1 q_2 + q_0 q_3) & 2(q_1 q_3 - q_0 q_2) \\ 2(q_1 q_2 - q_0 q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2 q_3 + q_0 q_1) \\ 2(q_0 q_2 + q_1 q_3) & 2(q_2 q_3 - q_0 q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \tag{2.57}$$

Alternatively to the *Quaternions* or the EULER angles, FISCH uses the definition of the rotational rate between the *Body Fixed Frame* and the *Kinematic Frame*

$$\left(\boldsymbol{\omega}^{KB}\right)_K = \begin{pmatrix} \omega_x^{KB} \\ \omega_y^{KB} \\ \omega_z^{KB} \end{pmatrix}_K = \begin{pmatrix} \dot{\mu}_K + \dot{\alpha}_K \cdot \sin\beta_K \\ \dot{\alpha}_K \cdot \cos\beta_K \cdot \cos\mu_K + \dot{\beta}_K \cdot \sin\mu_K \\ \dot{\alpha}_K \cdot \cos\beta_K \cdot \sin\mu_K - \dot{\beta}_K \cdot \cos\mu_K \end{pmatrix}_K \tag{2.58}$$

for deriving relations for the time derivatives of the kinematic angle of attack α_K, the kinematic angle of sideslip β_K and the kinematic bank angle μ_K and introduces the three of them as states. The resulting differential equation reads

$$\begin{pmatrix} \dot{\alpha}_K \\ \dot{\beta}_K \\ \dot{\mu}_K \end{pmatrix} = \begin{pmatrix} \frac{1}{\cos\beta_K} \cdot \left(\left(\omega_y^{KB}\right)_K \cdot \cos\mu_K + \left(\omega_z^{KB}\right)_K \cdot \sin\mu_K\right) \\ \left(\omega_y^{KB}\right)_K \cdot \sin\mu_K - \left(\omega_z^{KB}\right)_K \cdot \cos\mu_K \\ \left(\omega_x^{KB}\right)_K - \tan\beta_K \cdot \left(\left(\omega_y^{KB}\right)_K \cdot \cos\mu_K + \left(\omega_z^{KB}\right)_K \cdot \sin\mu_K\right) \end{pmatrix}, \tag{2.59}$$

where the required rotational rate $\left(\boldsymbol{\omega}^{KB}\right)_K$ can be calculated from:

$$\left(\boldsymbol{\omega}^{KB}\right)_K = \mathbf{M}_{KB} \cdot \left(\boldsymbol{\omega}^{OB}\right)_B - \left(\boldsymbol{\omega}^{OK}\right)_K \tag{2.60}$$

When using this representation of the attitude, special care has to be taken concerning the selection of the translation states as otherwise some states may be "used twice" resulting in unsolvable dynamic equations.

2.5.2 Rotational Equations of Motion

The rotational dynamics can be seen as the equivalent to the translational equations of motion, but affecting the rotational dynamics of the aircraft. They are also based on NEWTON's second law and use the moments acting on the system as their input. Again, the reference point R is selected to be the aircraft's center of gravity G which is assumed to be fixed. The *angular momentum* of a body with respect to the inertial center of the earth O is given as

$$\mathbf{H}^O = \int_m \left(\mathbf{x}^P\right) \times \left(\mathbf{V}^P\right)^I \, dm \tag{2.61}$$

using equations (2.28), (2.29) and (2.32) it results that:

$$\begin{aligned}\mathbf{H}^O &= \int_m \left(\left(\mathbf{x}^G\right) + \left(\mathbf{x}^{GP}\right)\right) \times \left(\left(\mathbf{V}^G\right)^I + \left(\boldsymbol{\omega}^{IB}\right) \times \left(\mathbf{x}^{GP}\right)\right) dm \\ &= m \cdot \left(\left(\mathbf{x}^G\right) \times \left(\mathbf{V}^G\right)^I\right) + \left(\mathbf{x}^G\right) \times \left(\boldsymbol{\omega}^{IB}\right) \times \underbrace{\int_m \left(\mathbf{x}^{GP}\right) dm}_{=0} \\ &\quad + \underbrace{\int_m \left(\mathbf{x}^{GP}\right) dm}_{=0} \times \left(\mathbf{V}^G\right)^I + \int_m \left(\mathbf{x}^{GP}\right) \times \left(\left(\boldsymbol{\omega}^{IB}\right) \times \left(\mathbf{x}^{GP}\right)\right) dm \\ &= m \cdot \left(\left(\mathbf{x}^G\right) \times \left(\mathbf{V}^G\right)^I\right) + \int_m \left(\mathbf{x}^{GP}\right) \times \left(\left(\boldsymbol{\omega}^{IB}\right) \times \left(\mathbf{x}^{GP}\right)\right) dm \end{aligned} \tag{2.62}$$

Using the anti commutativity property of the cross product and the fact that the cross product can be written as a matrix vector product involving a skew-symmetric matrix, the last term of equation (2.62) can be written as

$$\begin{aligned}\int_m \left(\mathbf{x}^{GP}\right) \times \left(\left(\boldsymbol{\omega}^{IB}\right) \times \left(\mathbf{x}^{GP}\right)\right) dm &= -\int_m \left(\mathbf{x}^{GP}\right) \times \left(\mathbf{x}^{GP}\right) \times \left(\boldsymbol{\omega}^{IB}\right) dm \\ &= -\int_m \tilde{\mathbf{R}}^{GP} \cdot \tilde{\mathbf{R}}^{GP}\, dm \cdot \left(\boldsymbol{\omega}^{IB}\right), \end{aligned} \tag{2.63}$$

where $\tilde{\mathbf{R}}^{GP}$ denotes the skew-symmetric representation of the cross product $\left(\mathbf{x}^{GP}\right) \times \circ$. The element-wise multiplication and integration of $\int_m \tilde{\mathbf{R}}^{GP} \cdot \tilde{\mathbf{R}}^{GP}\, dm$ leads to the widely used symmetric *moment of inertia* $\mathbf{I}^G$ with respect to the point G. Now, using the definition for the *linear momentum* $\mathbf{p}^G$, equation (2.62) can be written as:

$$\mathbf{H}^O = \left(\mathbf{x}^G\right) \times \left(\mathbf{p}^G\right) + \mathbf{I}^G \cdot \left(\boldsymbol{\omega}^{IB}\right) \tag{2.64}$$

NEWTON's second law applied to the rotational dynamics results in

$$\left(\frac{d}{dt}\right)^I \mathbf{H}^O = \sum \mathbf{M}^O \tag{2.65}$$

Consequently, equation (2.64) has to be differentiated with respect to time and the *Earth Centered Inertial Frame*:

$$\begin{aligned}\left(\frac{d}{dt}\right)^I \mathbf{H}^O &= \left(\mathbf{V}_K^G\right)^I \times \left(\mathbf{p}^G\right) + \left(\mathbf{x}^G\right) \times \left(\dot{\mathbf{p}}^G\right)^I \\ &\quad + \left(\frac{d}{dt}\right)^B \left(\mathbf{I}^G \cdot \left(\boldsymbol{\omega}^{IB}\right)\right) + \left(\boldsymbol{\omega}^{IB}\right) \times \left(\mathbf{I}^G \cdot \left(\boldsymbol{\omega}^{IB}\right)\right) \\ &= \underbrace{\left(\mathbf{V}_K^G\right)^I \times \left(\mathbf{V}_K^G\right)^I}_{=0} \cdot m + \left(\mathbf{x}^G\right) \times \left(\dot{\mathbf{p}}^G\right)^I \\ &\quad + \mathbf{I}^G \cdot \left(\dot{\boldsymbol{\omega}}^{IB}\right)^B + \left(\boldsymbol{\omega}^{IB}\right) \times \left(\mathbf{I}^G \cdot \left(\boldsymbol{\omega}^{IB}\right)\right) \end{aligned} \tag{2.66}$$

Here, the fact that the *moment of inertia* does not change over time – as the mass distribution in the body was assumed to be stationary – was used. On the other side, the moments acting on the body can be shifted into the center of gravity of the body:

$$\sum \mathbf{M}^O = \sum \mathbf{M}^G + \left(\mathbf{x}^G\right) \times \sum \mathbf{F}^G \tag{2.67}$$

Using NEWTON's second law for the *linear momentum* (2.27) and combining equations (2.66) and (2.67), the commonly known equation

$$\sum \mathbf{M}^G = \mathbf{I}^G \cdot \left(\dot{\boldsymbol{\omega}}^{IB}\right)^B + \left(\boldsymbol{\omega}^{IB}\right) \times \mathbf{I}^G \cdot \left(\boldsymbol{\omega}^{IB}\right) \tag{2.68}$$

results. This equation can be solved for the rotational acceleration of the aircraft as:

$$\left(\dot{\boldsymbol{\omega}}^{IB}\right)^B = \left(\mathbf{I}^G\right)^{-1} \cdot \left(\sum \mathbf{M}^G - \left(\boldsymbol{\omega}^{IB}\right) \times \mathbf{I}^G \cdot \left(\boldsymbol{\omega}^{IB}\right)\right) \tag{2.69}$$

In case a round and rotating earth is considered, $\left(\boldsymbol{\omega}^{IB}\right)_B$ is mostly used as the aircraft's rotational state, as then equation (2.69) can directly be used to simulate the rotational dynamics. The body rates p_K, q_K and r_K can then be calculated from

$$\left(\boldsymbol{\omega}^{IB}\right) = \left(\boldsymbol{\omega}^{IE}\right) + \left(\boldsymbol{\omega}^{EO}\right) + \left(\boldsymbol{\omega}^{OB}\right) \tag{2.70}$$

and consequently:

$$\begin{pmatrix} p_K \\ q_K \\ r_K \end{pmatrix} = \left(\boldsymbol{\omega}^{OB}\right)_B = \left(\boldsymbol{\omega}^{IB}\right)_B - \mathbf{M}_{BO} \cdot \left(\left(\boldsymbol{\omega}^{IE}\right)_O + \left(\boldsymbol{\omega}^{EO}\right)_O\right) \tag{2.71}$$

This requires the formulation of the rotational dynamics in the *Body Fixed Frame* B.

In case, a flat and non-rotating earth is considered,

$$\left(\boldsymbol{\omega}^{OB}\right)_B = \left(\boldsymbol{\omega}^{IB}\right)_B \tag{2.72}$$

holds. Then, the body rates p_K, q_K and r_K can directly be used as states and the resulting dynamic equation reads:

$$\begin{pmatrix} \dot{p}_K \\ \dot{q}_K \\ \dot{r}_K \end{pmatrix} = \left(\dot{\boldsymbol{\omega}}^{OB}\right)_B^B = \left(\mathbf{I}^G\right)_{BB}^{-1} \cdot \left(\sum \left(\mathbf{M}^G\right)_B - \left(\boldsymbol{\omega}^{OB}\right)_B \times \left(\mathbf{I}^G\right)_{BB} \cdot \left(\boldsymbol{\omega}^{OB}\right)_B\right) \tag{2.73}$$

Further details on the errors that result from modeling the earth as flat and non-rotating in different cases can be found in [Len15].

2.6 Atmosphere Model

As atmospheric flights are considered within this work, a description of the air mass surrounding the vehicle is required. The following sections first describe a standard model for a static atmosphere, before the influences of wind within a dynamic atmosphere are discussed. The effects within the second part are further divided into those influencing the translational motion of the aircraft and those influencing the rotational motion.

2.6.1 Static Atmosphere

The *Standard Atmosphere* model is used to describe the physics of the air surrounding the aircraft. It is based on DIN ISO 2533:1979-12 [Deu79]. Table 2.2 lists the most important parameters of the model. More details on the *Standard Atmosphere* model can besides others be found in [ICA02, Len15, Hol+15, Hol15].

To remove the influence of the decrease in the gravitational acceleration with increasing altitude, the *Standard Atmosphere* introduces the geopotential altitude

$$H_G = \frac{r_E \cdot h}{r_E + h}, \tag{2.74}$$

where r_E is the mean radius of the earth as listed in table 2.2.

Table 2.2: *Constants for the* Standard Atmosphere *model. [Len15]*

Symbol	Value	Unit	Description
r_E	$6.356766 \cdot 10^6$	m	Mean radius of the earth
$H_{G,\text{ref, Trop}}$	0	m	Reference altitude for *Troposphere*
g_S	9.80665	m/s^2	Acceleration of gravity at sea level
$p_{\text{ref, MSL}}$	$101.325 \cdot 10^3$	N/m^2	Air pressure at sea level
$T_{\text{ref, MSL}}$	288.15	K	Temperature at sea level
$\rho_{\text{ref, MSL}}$	1.225	kg/m^3	Air density at sea level
R	287.05287	$J/(kg \cdot K)$	Gas constant of air
κ	1.4	$-$	Isentropic exponent of air
$\gamma_{\text{ref, Trop}}$	$-6.5 \cdot 10^{-3}$	K/m	Temperature gradient in *Troposphere*
$\gamma_{\text{ref, U-Strat}}$	$1 \cdot 10^{-3}$	K/m	Temperature gradient in *Upper Stratosphere*

Furthermore, in the model, the lower atmosphere is divided into three parts, being the *Troposphere* ($-2000 \leq H_G < 11000$), the *Lower Stratosphere* ($11000 \leq H_G < 20000$) and the *Upper Stratosphere* ($20000 \leq H_G < 32000$). For the last two, the reference altitude is their respective minimum altitude, whereas the *Troposphere* uses the mean sea level as reference altitude.

Within the polytropic *Troposphere* and *Upper Stratosphere*, the temperature can be calculated using the temperature gradient from table 2.2 and the equation:

$$T_{\text{Layer}}(H_G) = T_{\text{ref, Layer}} \cdot \left(1 + \frac{\gamma_{\text{ref, Layer}}}{T_{\text{ref, Layer}}} \cdot (H_G - H_{G,\text{ref, Layer}})\right) \tag{2.75}$$

while the correlated pressure can be calculated from:

$$p_{\text{Layer}}(H_G) = p_{\text{ref, Layer}} \cdot \left(1 + \frac{\gamma_{\text{ref, Layer}}}{T_{\text{ref, Layer}}} \cdot (H_G - H_{\text{ref, Layer}})\right)^{-\frac{g_S}{\gamma_{\text{ref, Layer}} \cdot R}} \tag{2.76}$$

The air density can be calculated from its profile:

$$\rho_{\text{Layer}}(H_G) = \rho_{\text{ref, Layer}} \cdot \left(1 + \frac{\gamma_{\text{ref, Layer}}}{T_{\text{ref, Layer}}} \cdot (H_G - H_{\text{ref, Layer}})\right)^{-\left(\frac{g_S}{\gamma_{\text{ref, Layer}} \cdot R}+1\right)} \tag{2.77}$$

In the isothermal *Lower Stratosphere*, the temperature is constant

$$T_{\text{L-Strat}}(H_G) = T_{\text{ref, L-Strat}}, \tag{2.78}$$

while the pressure and the density change exponentially with the geopotential altitude:

$$p_{\text{L-Strat}}(H_G) = p_{\text{ref, Layer}} \cdot \exp\left(-\frac{g_S}{R} \cdot \frac{H_G - H_{\text{ref, Layer}}}{T_{\text{ref, Layer}}}\right) \tag{2.79}$$

$$\rho_{\text{L-Strat}}(H_G) = \rho_{\text{ref, Layer}} \cdot \exp\left(-\frac{g_S}{R} \cdot \frac{H_G - H_{\text{ref, Layer}}}{T_{\text{ref, Layer}}}\right) \tag{2.80}$$

Any one of the three states T, p, rho of the ideal gas may also be calculated using the ideal gas law, whenever two of the quantities are known:

$$\rho = \frac{p}{R \cdot T} \tag{2.81}$$

The speed of sound can be calculated from

$$a = \sqrt{\kappa \cdot R \cdot T}, \tag{2.82}$$

where κ is the isentropic exponent of air and can also be found in table 2.2. Using the current speed of sound, any aerodynamic velocity can be transformed into a MACH number by:

$$M = \frac{V_A}{a} \tag{2.83}$$

The MACH number is important for analyzing the fluid dynamic effects around the wings or the body of the aircraft. In optimal control problems, the MACH number may appear in some of the constraints.

Deviations from the standard conditions in the atmosphere may be implemented in the *Standard Atmosphere* model by changing the temperature and the pressure at mean sea level:

$$T_{\text{ref Trop}} = T_{\text{ref, MSL}} + \Delta T \tag{2.84}$$

$$p_{\text{ref Trop}} = p_{\text{ref, MSL}} + \Delta p \tag{2.85}$$

As the *Standard Atmosphere* model is continuous at the boundaries of the different layers, it is smooth enough to be used within optimal control problems. Whenever problems are solved that take place directly along one of the boundaries, it is recommended to make the problem continuously differentiable by smoothing the atmosphere model in the derivatives at these points. The smooth table interpolation presented in 6.5 may be used for this purpose by introducing a fading between the equations for the different layers.

2.6.2 Dynamic Atmosphere

Whenever wind should be considered within dynamic simulations of aircraft movement, a distinction between *aerodynamic* and *kinematic* quantities is required. The next subsections go through the effects of wind on the translational and the rotational motion of a flight system again.

Wind Influences on the Translational Dynamics

In this section, the influence of wind on the point mass simulation model is revisited. In case no wind is considered, all aerodynamic values coincide with their kinematic counterparts and the *Rotated Kinematic Frame* $\bar{K}$ is equivalent to the *Aerodynamic Frame* A and vice versa. This is obviously not true in case of wind – then the aerodynamic velocity required for the calculation of the aerodynamic forces can be derived from the kinematic velocity using the wind equation:

$$\left(\mathbf{V}_A^G\right)^E = \left(\mathbf{V}_K^G\right)^E - \left(\mathbf{V}_W^G\right)^E \tag{2.86}$$

Therein, $\left(\mathbf{V}_W^G\right)^E$ denotes the wind velocity at the center of gravity of the aircraft with respect to the *Earth-Centered Earth Fixed Frame* E. It has to be noted here that in the nomenclature used, $\left(V^G\right)^E$ in the first place is uniquely defined as the velocity of the point G with respect to an inertial point. Anyway, the index A indicates that only the part of the velocity with respect to the surrounding airflow is considered while the index W marks the velocity of the air with respect to the initial point. Both components together result in the overall velocity $\left(V_K^G\right)^E = \left(V^G\right)^E$. More precisely, the velocity $\left(V_W^G\right)^E$ does not indicate the velocity of the point G with respect to an inertial point at all, but it marks the velocity of the surrounding air with respect to an inertial point at the position G. The restoration of the aerodynamic flight path quantities can be performed using

$$V_A = \sqrt{\left[(u_A)_O^E\right]^2 + \left[(v_A)_O^E\right]^2 + \left[(w_A)_O^E\right]^2} \tag{2.87}$$

$$\chi_A = \arctan_2\left(\frac{(v_A)_O^E}{(u_A)_O^E}\right) \tag{2.88}$$

$$\gamma_A = \arctan_2\left(\frac{(w_A)_O^E}{\sqrt{\left[(u_A)_O^E\right]^2 + \left[(v_A)_O^E\right]^2}}\right), \tag{2.89}$$

where $\arctan_2$ denotes the extended version of the arcus tangens also considering the signs in the components of the fraction and selecting the appropriate quadrant for the resulting angle. Depending on the chosen state set, the aerodynamic angle of attack and the aerodynamic angle of sideslip can be recovered from the aerodynamic velocity vector given in the *Body*

Fixed Frame by:

$$\alpha_A = \arctan_2\left(\frac{(w_A)_B^E}{(u_A)_B^E}\right) \tag{2.90}$$

$$\beta_A = \arctan_2\left(\frac{(v_A)_B^E}{\sqrt{\left[(u_A)_B^E\right]^2 + \left[(w_A)_B^E\right]^2}}\right) \tag{2.91}$$

By definition of the aerodynamic frame, the aerodynamic velocity of the aircraft can be written as:

$$\left(\mathbf{V}_A^G\right)_A^E = \begin{pmatrix} V_A^G \\ 0 \\ 0 \end{pmatrix}_A^E \tag{2.92}$$

Besides, the wind equation can be written with respect to the *NED Frame* resulting in

$$\begin{pmatrix} V_K \cdot \cos\chi_K \cdot \cos\gamma_K \\ V_K \cdot \sin\chi_K \cdot \cos\gamma_K \\ -V_K \cdot \sin\gamma_K \end{pmatrix}_O^E = \begin{pmatrix} V_A \cdot \cos\chi_A \cdot \cos\gamma_A \\ V_A \cdot \sin\chi_A \cdot \cos\gamma_A \\ -V_A \cdot \sin\gamma_A \end{pmatrix}_O^E + \begin{pmatrix} V_W \cdot \cos\chi_W \cdot \cos\gamma_W \\ V_W \cdot \sin\chi_W \cdot \cos\gamma_W \\ -V_W \cdot \sin\gamma_W \end{pmatrix}_O^E \tag{2.93}$$

The derivative of the wind equation with respect to time is:

$$\left(\dot{\mathbf{V}}_A^G\right)^{EO} = \left(\dot{\mathbf{V}}_K^G\right)^{EO} - \left(\dot{\mathbf{V}}_W^G\right)^{EO} \tag{2.94}$$

In case the wind field is not depending on the position of the aircraft, the total time derivative of the wind field can be replaced by the partial one. Otherwise, the total derivative comprises three components:

$$\begin{aligned} \left(\dot{\mathbf{V}}_W^G\right)^{EO} &= \left(\frac{d}{dt}\right)^O \left(\mathbf{V}_W^G\right)^E \\ &= \left(\frac{\partial}{\partial t}\right)^E \left(\mathbf{V}_W^G\right)^E + \left(\boldsymbol{\omega}^{OE}\right) \times \left(\mathbf{V}_W^G\right)^E + \left(\nabla^O \left(\mathbf{V}_W^G\right)^E\right)^\mathsf{T} \cdot \left(\mathbf{V}_K^G\right)^E, \end{aligned} \tag{2.95}$$

where ∇^O represents the spatial *Jacobian* of the wind velocity with respect to the O-Frame. Mostly, equation (2.95) is evaluated in the *NED Frame*. In case of a time-invariant wind, it simplifies to:

$$\left(\dot{\mathbf{V}}_W^G\right)^{EO} = \left(\nabla^O \left(\mathbf{V}_W^G\right)^E\right)^\mathsf{T} \cdot \left(\mathbf{V}_K^G\right)^E \tag{2.96}$$

For a rigid body aircraft of a certain spatial extent, more facets of wind influence need to be considered as will be shown in section 2.6.2. More details on wind modeling and its influence on the aircraft dynamics can besides others be found in [BAL11, Hol15].

Wind Influence on the Rotational Dynamics

In contrast to the wind considerations from section 2.6.2, here the influence of wind on an aircraft of finite spatial extent is considered. As one can imagine, if the wind field changes

along the body of the aircraft, not only are wind forces affecting the aircraft but also wind moments. These influences are especially relevant in case of turbulent or strong local winds. In general, not all effects of a turbulent, time and spatially dependent wind field can be taken into account [BAL11, Ch. 4]. Hence, only the effects of the mathematical rotation of the wind field – that can directly be modeled using aerodynamic derivatives – is considered in the model presented here. More details on the rotational effects of wind can, besides many others, be found in [BAL11].

Similarly to equation (2.86) for the wind velocities, the rotation of the aircraft can be divided into the rotation with respect to the surrounding air and the rotation of the air itself, resulting from wind. In order to calculate the aerodynamic rotation of the aircraft,

$$\boldsymbol{\omega}_A = \boldsymbol{\omega}_K - \boldsymbol{\omega}_W \tag{2.97}$$

can be used. Using a notation similar to the one used in section 2.6.2, equation (2.97) can be written as:

$$\left(\boldsymbol{\omega}_A^{OB}\right) = \left(\boldsymbol{\omega}_K^{OB}\right) - \left(\boldsymbol{\omega}_W^{OB}\right) \tag{2.98}$$

Again, special care has to be taken considering the nomenclature, as now the aerodynamic rotation $\left(\boldsymbol{\omega}_A^{OB}\right)$ denotes only the part of the rotation between the coordinate frames O and B that can be "seen" with respect to the surrounding air. $\left(\boldsymbol{\omega}_W^{OB}\right)$ consequently marks the rotation of the air with respect to the O-Frame such that the overall rotation of the aircraft $\left(\boldsymbol{\omega}_K^{OB}\right) = \left(\boldsymbol{\omega}^{OB}\right)$ results. Technically, in a simulation, $\left(\boldsymbol{\omega}^{OB}\right)$ can be calculated from the equations of motion, while $\left(\boldsymbol{\omega}_W^{OB}\right)$ needs to be calculated from the wind field. If equation (2.98) is written with respect to the *NED Frame*, the rotation of the wind field can be calculated from:

$$\left(\boldsymbol{\omega}_W^{OB}\right)_O = \frac{1}{2}\mathrm{rot}_O\left(\mathbf{V}_W^G\right)_O^E = \frac{1}{2}\left[\nabla^O \times \left(\mathbf{V}_W^G\right)_O^E\right] = \frac{1}{2}\cdot\begin{pmatrix} \frac{\partial\left(w_W^G\right)_O^E}{\partial(y)_O} - \frac{\partial\left(v_W^G\right)_O^E}{\partial(z)_O} \\ \frac{\partial\left(u_W^G\right)_O^E}{\partial(z)_O} - \frac{\partial\left(w_W^G\right)_O^E}{\partial(x)_O} \\ \frac{\partial\left(v_W^G\right)_O^E}{\partial(x)_O} - \frac{\partial\left(u_W^G\right)_O^E}{\partial(y)_O} \end{pmatrix}_O \tag{2.99}$$

Then, the aerodynamic rotation results as:

$$\begin{aligned} \left(\boldsymbol{\omega}_A^{OB}\right)_O &= \mathbf{M}_{OB}\cdot\left(\boldsymbol{\omega}_A^{OB}\right)_B \\ &= \mathbf{M}_{OB}\cdot\begin{pmatrix} p_A \\ q_A \\ r_A \end{pmatrix} \\ &= \left(\boldsymbol{\omega}_K^{OB}\right)_O - \left(\boldsymbol{\omega}_W^{OB}\right)_O \end{aligned} \tag{2.100}$$

Equation (2.99) shows that the rotation of the wind field may not be modeled independently of the velocities. More details on this model and more complex wind models can e.g. be found in [BAL11].

2.7 External Forces and Moments

The equations of motion above require the calculation of input forces and moments. For an aircraft, these cannot directly be controlled but result from external effects and the current configuration of the aircraft. In general, the forces and moments comprise aerodynamic effects (index A), the system's propulsion (index P) and the gravitational effects (index G). The total forces and moments can consequently be written as

$$\sum \left(\mathbf{F}^G\right) = \left(\mathbf{F}_A^G\right) + \left(\mathbf{F}_P^G\right) + \left(\mathbf{F}_G^G\right) \tag{2.101}$$

$$\sum \left(\mathbf{M}^G\right) = \left(\mathbf{M}_A^G\right) + \left(\mathbf{M}_P^G\right) + \underbrace{\left(\mathbf{M}_G^G\right)}_{=0}, \tag{2.102}$$

where gravity does not create moments in case the center of gravity is chosen to be the reference point. More information on modeling the external forces and moments can e.g. be found in [Fis11, Höc14, Len15, BAL11, And11, Hol15, Hol+15].

For the air traffic application presented in section 10.2, the aerodynamic data as well as the propulsion and fuel consumption models are based on the *Base of Aircraft Data Family 4 (BADA4)* published by EUROCONTROL [EUR12].

2.7.1 Aerodynamics

The most important forces and moments acting on the aircraft's body are the aerodynamic forces and moments. Hereby, lift and drag have the most significant impact on the motion of the aircraft. In general, the three components of the aerodynamic force (with respect to the *Aerodynamic Frame*) and the moment (with respect to the *Body Fixed Frame*) are written as:

$$\left(\mathbf{F}_A^G\right)_A = \begin{pmatrix} -D \\ Q \\ -L \end{pmatrix} = \bar{q} \cdot S \cdot \begin{pmatrix} -C_D \\ C_Q \\ -C_L \end{pmatrix}_A \tag{2.103}$$

$$\left(\mathbf{M}_A^G\right)_B = \begin{pmatrix} L \\ M \\ N \end{pmatrix} = \bar{q} \cdot S \cdot \begin{pmatrix} \frac{b}{2} \cdot C_l \\ \bar{c} \cdot C_m \\ \frac{b}{2} \cdot C_n \end{pmatrix}_B \tag{2.104}$$

In these equations, the following quantities are important:

- D, Q and L: The drag, cross-stream and lift force
- L, M and N: The aerodynamic roll, pitch, and yaw moments
- S, b and $\bar{c}$: The wing reference area, the wing span and the mean aerodynamic chord
- $\bar{q} = \frac{1}{2}\rho V_A^2$: The dynamic pressure
- $C_\circ$: The aerodynamic coefficients

The aerodynamic coefficients of the models considered here mainly depend on the aerodynamic angle of attack, α_A, the aerodynamic angle of sideslip, β_A, the control surface deflections ξ, η and ζ as well as the dimensionless aerodynamic angular rates:

$$\tilde{p} = \frac{p_A \cdot b}{2 \cdot V_A} \tag{2.105}$$

$$\tilde{q} = \frac{q_A \cdot \bar{c}}{2 \cdot V_A} \tag{2.106}$$

$$\tilde{r} = \frac{r_A \cdot b}{2 \cdot V_A} \tag{2.107}$$

The aerodynamic rates are defined as the rotational motion of the aircraft with respect to the surrounding air and can be calculated according to equation (2.100). In case a point mass model is considered, the control surface deflections cannot be used as controls as they hardly affect the linear motion of the aircraft but mainly the rotational dynamics.

The particular models used for calculating the aerodynamic derivatives depend on the particular application. In the air traffic problems considered in section 10.2, the aerodynamic forces are calculated based on the *Base of Aircraft Data* Family 4 *(BADA 4)* [EUR12]. There, the equations for the forces depend on the Mach number M, the speed of sound a, the current air temperature T and other atmospheric quantities. These values are calculated according to the *ISA Standard Atmosphere* which is briefly reviewed in section 2.6.1.

In the air race examples within this work the aerodynamic models as described in the work by FISCH, [Fis11] have been used, with slightly different parameter values and an updated lift model. The model is based on a set of basic aerodynamic derivatives as they are e.g. also described in [Hol15]. More details can be found in section 11.1.

2.7.2 Propulsion Forces and Moments

The goal of the propulsion system on board an aircraft is the creation of thrust force (and in some cases also moments) that are used to accelerate and control the aircraft. In general, the thrust reference point T is not identical to the center of gravity of the aircraft, resulting in additional moments. Besides, the thrust force does neither necessarily act in the direction of the x-axis of the *Body Fixed Frame* nor in the direction of the aerodynamic or the kinematic velocity. Consequently, a new coordinate frame is required, called the *Propulsion Frame*. It is defined by the *engine mounting angle* κ in the *Body Fixed* x-y plane and the *thrust elevation angle* σ, resulting in the transformation matrix:

$$\mathbf{M}_{BP} = \begin{bmatrix} \cos\kappa \cdot \cos\sigma & -\sin\kappa & \cos\kappa \cdot \sin\sigma \\ \sin\kappa \cdot \cos\sigma & \cos\kappa & \sin\kappa \cdot \sin\sigma \\ -\sin\sigma & 0 & \cos\sigma \end{bmatrix} \tag{2.108}$$

Thus, the propulsion forces in the *Body Fixed Frame* can be calculated from:

$$\left(\mathbf{F}_P^G\right)_B = \mathbf{M}_{BP} \cdot \left(\mathbf{F}_P^G\right)_P \tag{2.109}$$

In some simplified approaches the *Propulsion Frame* may be chosen similar to the *Body Fixed Frame* or the *Aerodynamic Frame*. Physically, the thrust has to be modeled as the vectorial difference between the impulse of the air flowing through the outlet of the aircraft's engine and the one flowing into its inlet. This model may generally also be used if no clear geometrical inlet and outlet of an engine can be defined (as e. g. for propeller engines). As the inlet and outlet forces do not necessarily affect the aircraft at the same point, additional moments may also result. Here, the resulting thrust force is directly modeled, constituting a simplification.

The moment resulting from an offset between the propulsion reference point and the center of gravity of the system can be calculated from:

$$\left(\mathbf{M}_P^G\right) = \left(\mathbf{x}^{GT}\right) \times \left(\mathbf{F}_P^T\right) \tag{2.110}$$

Additionally, the propulsion system may create gyroscopic moments $\left(\mathbf{M}_{\text{Gyro}}^T\right)$ that need to be added to the overall propulsion moments $\left(\mathbf{M}_P^G\right)$.

As in real systems, the thrust commanded by the pilot and the one created by the aircraft's engine feature a time delay, the relation between the two may be modeled using a *PT1* dynamic system. In the model the dynamics are directly coupled to the thrust lever position δ_T:

$$\dot{\delta}_T = \frac{1}{T_S}\left(\delta_{T,CMD} - \delta_T\right) \tag{2.111}$$

The thrust lever position is controlled by the value of $\delta_{T,CMD} \in [0, 1]$, while the resulting engine thrust is a function of δ_T. Considering the problem of singular arcs, described in section 3.2.6, this model may not be favorable in optimal control applications. This is why it is also considered in chapter 6.4. More details on the model can be found in [Fis11, Hol+15].

2.7.3 Gravitational Force

In [Len15], LENZ gives a detailed overview of the different fidelities of gravitational modeling available for aircraft trajectory optimization. Besides, he explains the difference between gravitation and gravity, where the latter is the combination of the gravitational and the centrifugal components of force or acceleration. LENZ finally shows, that for aircraft flying below $30km$, a series expansion for modeling the gravitational force is far sufficient as it features an absolute accuracy of $3.4 \cdot 10^{-4} m/s^2$. Moreover, he shows that the horizontal component of the acceleration created by gravity is far below $1‰$ and can also be ignored in atmospheric flight optimization problems.

In the examples presented in the remainder of this work, the spatial extent of the trajectories is small and the flights are performed far below an altitude of $30km$, allowing for the use of a constant gravitational model. Additionally, the reference point is selected to be the center of gravity, resulting in no additional gravitational moments. The resulting gravitational force can be transformed into the *Kinematic Frame*, K:

$$\left(\mathbf{F}_G^G\right)_K = \mathbf{M}_{KO} \cdot \left(\mathbf{F}_G^G\right)_O = \mathbf{M}_{KO} \cdot \begin{pmatrix} 0 \\ 0 \\ m \cdot g \end{pmatrix}_O \tag{2.112}$$

In case the influence of the latitude φ on gravity should be taken into account, the formula of SOMIGLIANA may be used to calculate a *local gravity constant* g_φ replacing g in equation (2.112):

$$g_\varphi = \frac{a \cdot g_e \cdot \cos^2\varphi + b \cdot g_p \cdot \sin^2\varphi}{\sqrt{a^2 \cdot \cos^2\varphi + b^2 \cdot \sin^2\varphi}} \tag{2.113}$$

with

- $g_e = 9.78032533590 m/s^2$ being the normal gravity at the equator,
- $g_p = 9.83218493786 m/s^2$ being the normal gravity at the pole and
- a and b being the *semi major axis* from equation (2.18) and the *semi minor axis* from equation (2.20).

2.8 Model Extensions

Besides the pure mechanics presented before, in the applications shown below, auxiliary effects influence the dynamics and consequently the solutions to the problems. The most important effects to be incorporated are a change in the aircraft mass due to fuel consumption and actuator dynamics representing the movement of the control surfaces of an aircraft.

2.8.1 Fuel Consumption

Especially when considering air traffic applications, fuel consumption may be one of the optimization criteria. In order to accurately model the influence of fuel consumption on the aircraft dynamics, the resulting change in mass needs to be taken into account in these applications. Even when the fuel consumption is not part of the cost function, but long trajectories are considered, the fuel consumption needs to be taken into account as the mass of the aircraft may change dramatically. In [Hol+15] the example of an *Airbus A380* is given, that features an approximate maximum takeoff weight of 569 tons, where 256 tons thereof are fuel. Consequently, the change in mass may not be neglected.

The general equation for modeling the fuel consumption is given as

$$\dot{m} = -\dot{m}_{\text{fuel}} \tag{2.114}$$

where several different models for the calculation of the current fuel flow $\dot{m}_{\text{fuel}}$ exist. The most simple approach suggested in [Hol+15] is to use a constant *specific fuel consumption* σ_{SFC} that is multiplied by the current thrust force resulting in the current fuel flow:

$$\dot{m}_{\text{fuel}} = \sigma_{SFC} \cdot \left\| \left(\mathbf{F}_P^G \right) \right\| \tag{2.115}$$

More elaborate models for the approximation of fuel consumption exist, where the following list gives a short overview.

- **Base of Aircraft Data, Family 3 (BADA 3)** *Eurocontrol's* Base of Aircraft Data in Family 3 contains relationships and parameters to approximate the specific fuel consumption during flight depending on the aircraft's velocity as well as a model for estimating the fuel flow for engine idle. [EUR14]
- **Base of Aircraft Data, Family 4 (BADA 4)** In Family 4 of *Eurocontrol's* Base of Aircraft data much more elaborate fuel flow models are included, now taking the aircraft weight, the pressure, the temperature and the MACH number into account, modeling idle and non-idle fuel flow values. [EUR12]
- **System for Assessing Aviation's Global Emission (SAGE)** This model developed by the American *Federal Aviation Administration (FAA)* Is based on the fuel consumption model in BADA 3, introducing additional correction terms. [Kim+07, Lee+07]
- **ICAO Aircraft Engine Emissions Databank** The Aircraft Engine Emissions Databank developed by the *International Civil Aviation Organization (ICAO)* gives approximate fuel flow values for takeoff, cruise, approach, and idle settings at mean sea level. [ICA95, ICA16]
- **Fuel Flow Method 2** The Fuel Flow Method 2 from Boeing is based on the data from the Aircraft Engine Emissions Databank but introduces additional correction terms for air pressure and temperature. [DP06]

This listing of fuel consumption models has mainly been taken from [Hol+15]. The air traffic examples presented in this work use the fuel flow models from *BADA 4* and *BADA 3*, as stated.

2.8.2 Actuator Dynamics

Similarly to the engine dynamics, the control surfaces of an aircraft may not be able to instantaneously follow a given command. In this case, additional dynamics need to be introduced modeling the behavior of the control surface deflections. These dynamics are called actuator dynamics.

In [Fis11], FISCH uses a second order linear system to model the actuator dynamics. This approach is also suggested in [Hol15, Hol+15]. The control surface deflection u can be calculated from the input control u_{CMD} using the dynamic equation

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega_0^2 & -2\zeta\omega_0 \end{bmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ \omega_0^2 \end{pmatrix} \cdot u_{CMD} \tag{2.116}$$

and the output equation:

$$u = \begin{pmatrix} 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \tag{2.117}$$

Considering the problem described in section 3.2.6, this model extension is problematic as now the control of the dynamic system appears linearly even though the original control

surface deflections may not have appeared linearly in the flight dynamic model. In section 6.4 approaches to overcome this issue are discussed.

Chapter 3

Mathematical Preliminaries

This chapter presents the mathematical preliminaries required for solving optimal control problems. However, no full mathematical proof and derivation is given here (they can be found e.g. in [Ger12, Ger08, Ger09, Ben10, BH75, Lei81, Bie10]), but the purpose of this chapter is more to be a *"quick start guide"* in optimization and optimal control for engineers. Thus, the statements made are mostly not of general nature but more application driven and limited to the respective cases.

The chapter is organized as follows: First, an introduction to numerical optimization is given, starting with the simplest problem of numerical optimization, which is the minimization of an unconstrained, scalar function. After focusing on multidimensional optimization problems, constrained problems are treated, leading to the well-known KARUSH-KUHN-TUCKER conditions (KKT conditions). Besides those theoretical considerations, sections 3.1.3 and 3.1.4 focus on the sequential quadratic programming algorithm and the interior point algorithm which help solving numerical optimization problems in real life applications. The last section on numerical optimization focuses on parametric optimization problems, which on the one hand help to understand some aspects of regular optimization problems and on the other hand will later also be used in the context of optimal control problems.

The second part of this chapter focuses on the solution of optimal control problems. After stating the problem to be treated, some theoretical aspects are discussed. These include transformation techniques required to be able to reduce a wide set of optimal control problems to a basic problem type, as well as the indirect solution approach, introducing the *Hamiltonian* and finally leading to the respective optimality conditions. In section 3.2.4 the direct solution methods are presented that – similarly to before – can be used to solve complicated real life optimal control problems. Afterwards, the relations between the indirect approach and the direct approach, including the transformation of the LAGRANGE multipliers to the costates, are recalled.

The third part of the chapter gives a short introduction and an overview of commonly used methods for multi criteria optimization and multi criteria optimal control problems. The findings from this chapter will later be used to determine solutions for multi criteria aircraft trajectory optimization problems (see section 10.4).

An overview of a general implementation of a direct optimal control algorithm, that has been developed at the Institute of Flight System Dynamics at TUM, and is implemented in *MATLAB*, is given in chapter 4 of this work.

3.1 Numerical Optimization

From a historical point of view, the problem of finding extremal values of functions was solved long before optimal control problems came up. When considering continuous values for the variables, these so-called finite dimensional optimization problems can mainly be divided into two groups: Those not having additional equality or inequality constraints and those being constrained. However, when using direct discretization methods (see 3.2.4 for more details) for solving optimal control problems, they are transformed to constrained numerical optimization problems.

Before focusing on constrained numerical maximization and minimization problems, this section gives a short overview of the solution techniques for unconstrained optimization problems. A more detailed discussion of the problem classes and the solution techniques can besides others be found in [Bet09, Ben10, Ger08].

Throughout the remainder of this chapter the following wide spread terms will be used:

- $\mathbf{z}_{opt}$ is called global minimum if $J(\mathbf{z}_{opt}) \leq J(\mathbf{z}) \quad \forall \mathbf{z}$ in the admissible set.
- $\mathbf{z}_{opt}$ is called strict global minimum if $J(\mathbf{z}_{opt}) < J(\mathbf{z}) \quad \forall \mathbf{z} \neq \mathbf{z}_{opt}$ in the admissible set.
- $\mathbf{z}_{opt}$ is called local minimum if $J(\mathbf{z}_{opt}) \leq J(\mathbf{z}) \quad \forall \mathbf{z}$ in a bounded area around $\mathbf{z}_{opt}$.
- $\mathbf{z}_{opt}$ is called strict local minimum if $J(\mathbf{z}_{opt}) < J(\mathbf{z}) \quad \forall \mathbf{z} \neq \mathbf{z}_{opt}$ in a bounded area around $\mathbf{z}_{opt}$.

This section will focus on gradient based approaches as they are normally used for solving the large scale numerical optimization problems resulting from discretized optimal control problems. For general problems, gradient based approaches can only find *local* minima.

3.1.1 Unconstrained Optimization Problems

The basic problem of optimization is finding a minimum of a sufficiently smooth, vector dependent function $J : \mathbb{R}^{n_\mathbf{z}} \to \mathbb{R}$ not being constrained:

$$\min_{\mathbf{z}} J(\mathbf{z}), \qquad \mathbf{z} = \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_{n_z} \end{pmatrix} \quad \in \mathbb{R}^{n_\mathbf{z}} \tag{3.1}$$

Any minimization problem can be turned into an equivalent maximization problem by multiplying J by -1.

The derivative of the function J is written as the gradient vector

$$(J'_{\mathbf{z}})^{\mathsf{T}} = \nabla_{\mathbf{z}} J = \begin{pmatrix} \frac{\partial J}{\partial z_1} \\ \frac{\partial J}{\partial z_2} \\ \vdots \\ \frac{\partial J}{\partial z_{n_z}} \end{pmatrix}, \tag{3.2}$$

and the second derivative is represented by the symmetric *Hessian* matrix:

$$\mathbf{H}(\mathbf{z}) = J''_{\mathbf{zz}} = (J''_{\mathbf{zz}})^{\mathsf{T}} = \begin{bmatrix} \frac{\partial^2 J}{\partial z_1^2} & \frac{\partial^2 J}{\partial z_1 \partial z_2} & \cdots & \frac{\partial^2 J}{\partial z_1 \partial z_{n_z}} \\ \frac{\partial^2 J}{\partial z_2 \partial z_1} & \frac{\partial^2 J}{\partial z_2^2} & \cdots & \frac{\partial^2 J}{\partial z_2 \partial z_{n_z}} \\ \vdots & & \ddots & \vdots \\ \frac{\partial^2 J}{\partial z_{n_z} \partial z_1} & \frac{\partial^2 J}{\partial z_{n_z} \partial z_2} & \cdots & \frac{\partial^2 J}{\partial z_{n_z}^2} \end{bmatrix} \tag{3.3}$$

The conditions for a local minimum point $\mathbf{z}_{opt}$ are [Bet09, Ger08, Ben10, Bie10]:
Necessary conditions:

$$\nabla_{\mathbf{z}} J(\mathbf{z}_{opt}) = \mathbf{0} \tag{3.4}$$

$$(\mathbf{z}_{opt} - \mathbf{z})^{\mathsf{T}} \cdot \mathbf{H}(\mathbf{z}_{opt}) \cdot (\mathbf{z}_{opt} - \mathbf{z}) \geq 0 \quad \forall \mathbf{z} \in U_\delta(\mathbf{z}_{opt}) \tag{3.5}$$

$$\text{with} \quad U_\delta(\mathbf{z}_{opt}) = \{\mathbf{z} \in \mathbb{R}^{n_{\mathbf{z}}} | \, \|\mathbf{z} - \mathbf{z}_{opt}\| < \delta\}$$

Sufficient conditions:

$$\nabla_{\mathbf{z}} J(\mathbf{z}_{opt}) = \mathbf{0} \tag{3.6}$$

$$(\mathbf{z}_{opt} - \mathbf{z})^{\mathsf{T}} \cdot \mathbf{H}(\mathbf{z}_{opt}) \cdot (\mathbf{z}_{opt} - \mathbf{z}) > 0 \quad \forall \mathbf{z} \in U_\delta(\mathbf{z}_{opt}), \quad \mathbf{z} \neq \mathbf{z}_{opt} \tag{3.7}$$

$$\text{with} \quad U_\delta(\mathbf{z}_{opt}) = \{\mathbf{z} \in \mathbb{R}^{n_{\mathbf{z}}} | \, \|\mathbf{z} - \mathbf{z}_{opt}\| < \delta\}$$

Equations (3.5) and (3.7) are equivalent to the *Hessian* being positive semi-definite or positive definite, respectively.

In the special case of $n_{\mathbf{z}} = 1$, the well-known conditions for a *local* minimum of J result:

1. First order condition: $J'_{\mathbf{z}}(z_{opt}) = 0$

2. Second order condition: $J''_{\mathbf{zz}}(z_{opt}) \geq 0$

with the second one being not sufficient. Whenever the second order derivative $J''(z_{opt}) = 0$ one needs to evaluate higher oder derivatives until one of them evaluates to a value different from zero: $J^{(p)}(z_{opt}) \neq 0$. If p is an even number and $J^{(p)}(z_{opt}) > 0$ then z_{opt} is a minimum, too. In case no derivative evaluates to a value other than zero or p is an odd number, z_{opt} is neither a minimum nor a maximum. This can easily be verified using a TAYLOR series approximation.

Newton Method

When trying to numerically determine the minimum of a function, one approach is to use NEWTON's method in order to find candidates fulfilling the first order condition from above. The main problem there is that no distinction between maximum values, minimum values, and saddle points is possible as the first order condition is true for every stationary point. One approach when using NEWTON's method is to approximate the first derivative of the function to be minimized by a linear function and to determine the zero crossing of it iteratively. For sufficiently smooth functions, this approach leads to the same result as the approach used here and also presented in [Ger08, sec. 3.9]. This approach is to iteratively approximate the function itself by a quadratic function (instead of using an approximation for the derivative) and then determine the minimum of this approximation. It yields that (again, for sufficiently smooth functions) the new point $\mathbf{z}_{k+1}$ can be found as [Bet09, Ger08, Ben10, Bie10]:

$$\mathbf{z}_{k+1} = \mathbf{z}_k + \mathbf{p}_k \tag{3.8}$$

with $\mathbf{p}_k$ being the update step:

$$\mathbf{p}_k = -[J''_{\mathbf{zz}}(\mathbf{z}_k)]^{-1} \nabla_{\mathbf{z}} J(\mathbf{z}_k) = -\mathbf{H}_k^{-1} (\nabla_{\mathbf{z}} J)_k \tag{3.9}$$

In order to ensure that the update step improves the cost function, the projection of the update step on the gradient should be negative:

$$(\nabla_{\mathbf{z}} J)_k^{\mathrm{T}} \mathbf{p}_k = -(\nabla_{\mathbf{z}} J)_k^{\mathrm{T}} \mathbf{H}_k^{-1} (\nabla_{\mathbf{z}} J)_k < 0 \tag{3.10}$$

If (3.10) holds, then $\mathbf{p}_k$ is a direction of descent. Furthermore, (3.10) holds if $\mathbf{H}_k$ is positive definite and $(\nabla_{\mathbf{z}} J)_k \neq 0$. Normally this is not the case in all points passed during the solution of a problem. The next section gives some approaches to overcome this issue. [Bet09, sec. 1.8.1]

Hessian Free Methods

Besides the problem stated before, in a lot of applications – particularly in optimal control problems – the calculation of the gradient and especially the *Hessian* of a problem may be quite costly. One approach to tackle these issues is to (at least) use an approximation of the *Hessian* matrix. The most simple idea is to replace $\mathbf{H}_k$ by the identity matrix leading to the gradient method with the update step:

$$\mathbf{p}_k = -(\nabla_{\mathbf{z}} J)_k \tag{3.11}$$

The benefits of this method are quickly visible as it does not require any information about the *Hessian*, nor is it complicated to implement. The main drawback is its poor convergence behavior.

Another method that does not require information about the *Hessian* is the so-called Quasi-NEWTON method. There, the *Hessian* $\mathbf{H}_k$ is replaced by the symmetric and positive definite

matrix $\mathbf{Q}_k$ that is updated in each iteration according to an appropriate update scheme. Here, the *BFGS* (Broyden, Fletcher, Goldfarb, Shanno) update rule is considered:

$$\bar{\mathbf{Q}}_k = \mathbf{Q}_k + \frac{\Delta(\nabla_{\mathbf{z}} J)(\Delta(\nabla_{\mathbf{z}} J))^\intercal}{(\Delta(\nabla_{\mathbf{z}} J))^\intercal \Delta\mathbf{z}} - \frac{\mathbf{Q}_k \Delta\mathbf{z}(\Delta\mathbf{z})^\intercal \mathbf{Q}_k}{(\Delta\mathbf{z})^\intercal \mathbf{Q}_k \Delta\mathbf{z}} \tag{3.12}$$

$$\text{with} \quad \Delta\mathbf{z} = \mathbf{z}_{k+1} - \mathbf{z}_k \text{ and } \Delta(\nabla_{\mathbf{z}} J) = (\nabla_{\mathbf{z}} J)_{k+1} - (\nabla_{\mathbf{z}} J)_k$$

The *BFGS* method can e.g. be initialized using the identity matrix. The theoretical motivation of the *BFGS* update rule is based on a minimization of the Frobenius matrix norm of the matrix equation resulting from the Taylor expansion of the gradient. [HG16]

Of course, other update rules for the *Hessian* exist and may also be used here. [Ger08, Bet09, Bie10]

Line Search

In order to achieve convergence from any starting point and not just from points in the vicinity of the solution, a line search procedure can be utilized. The goal of this is to determine the step size for each iteration individually. Instead of doing a full step

$$\mathbf{z}_{k+1} = \mathbf{z}_k + \mathbf{p}_k \tag{3.13}$$

the scalar step size $\alpha \in [0, 1]$ is introduced:

$$\mathbf{z}_{k+1} = \mathbf{z}_k + \alpha\mathbf{p}_k \tag{3.14}$$

The step size is determined such that

$$J(\mathbf{z}_{k+1}) \leq J(\mathbf{z}_k) + \alpha_k \cdot \sigma \cdot J'_{\mathbf{z}} \cdot \mathbf{p}_k, \qquad \sigma \in [0, 1] \tag{3.15}$$

is ensured. To achieve this, most of the currently used algorithms first perform a full iteration step using $\alpha = 1$, before reducing α until a step length is found that satisfies a sufficient decrease condition, like e.g. the Armijo rule or the Wolfe-Powell rule [Bet09, sec. 1.11.2], [Arm66, Wol69, Wol71]. Figure 3.1 visualizes the general idea. The point on the left is the current point and the cost function may be sufficiently improved for $\alpha < 1$, but may not be improved for a full step with $\alpha = 1$.

3.1.2 Constrained Optimization Problems

In real world applications, optimization problems are hardly ever unconstrained. In these cases, instead of only having a cost function to be minimized, additional constraints have to be taken into account. The resulting problem formulation for the multi dimensional case is:

$$\min_{\mathbf{z}} J(\mathbf{z}) \tag{3.16}$$

$$\text{s. t.} \quad \mathbf{h}(\mathbf{z}) = 0 \tag{3.17}$$

$$\mathbf{g}(\mathbf{z}) \leq 0 \tag{3.18}$$

$$\mathbf{z} \in \mathbb{R}^{n_{\mathbf{z}}}$$

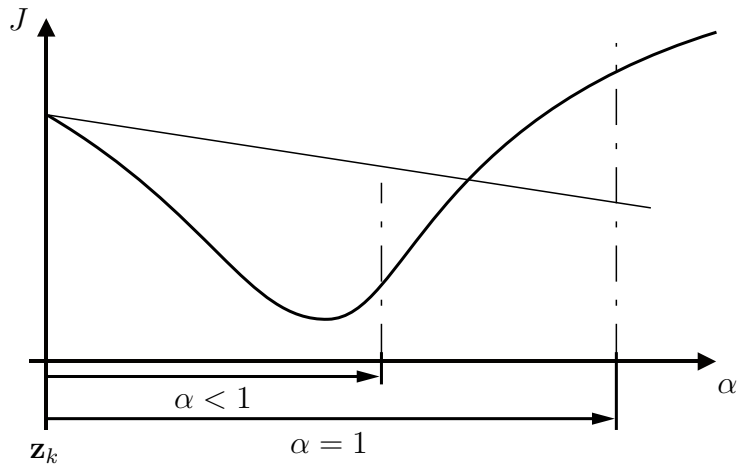

Figure 3.1: *Principle of a line search algorithm for determining the step size. In this example,* $\alpha = 1$ *is not a valid step size, whereas the shortened step is accepted.*

with

$$J : \mathbb{R}^{n_\mathbf{z}} \to \mathbb{R}, \tag{3.19}$$

$$\mathbf{h} : \mathbb{R}^{n_\mathbf{z}} \to \mathbb{R}^{n_h} \quad \text{and} \tag{3.20}$$

$$\mathbf{g} : \mathbb{R}^{n_\mathbf{z}} \to \mathbb{R}^{n_g} \tag{3.21}$$

being sufficiently smooth functions.

An inequality constraint $g_i(\mathbf{z}) \leq 0$ is called *active* at $\mathbf{z}$ if $g_i(\mathbf{z}) = 0$ holds. On the other hand, it is called *inactive* at $\mathbf{z}$ if $g_i(\mathbf{z}) < 0$ holds. The index i describes the i-th element of the vector $\mathbf{g}$.

Lagrange Function

For solving constrained optimization problems the LAGRANGE function is introduced:

$$L(\mathbf{z}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = J(\mathbf{z}) + \boldsymbol{\lambda}^\mathsf{T}\mathbf{h}(\mathbf{z}) + \boldsymbol{\nu}^\mathsf{T}\mathbf{g}(\mathbf{z}) \tag{3.22}$$

with $\boldsymbol{\lambda} \in \mathbb{R}^{n_h}$ and $\boldsymbol{\nu} \in \mathbb{R}^{n_g}$ being the LAGRANGE multipliers for the equality and inequality constraints, respectively. At this point the LAGRANGE function must be seen as an auxiliary function simplifying the results derived below. A more descriptive explanation of the results, also explaining the meaning of the LAGRANGE function in more detail, will be given below and in 3.1.6. [Ger08, Ben10, Bie10]

Linear Independence Constraint Qualification (LICQ)

In order to be able to mathematically derive the optimality conditions listed in the next section, the problem formulation has to fulfill certain regularity conditions. One commonly used set of regularity conditions for constrained optimization problems is the so-called *Linear Independence*

Constraint Qualification (LICQ). It is fulfilled in $\mathbf{z}_{opt}$ if the gradients for the active set of the inequality constraints and the gradients of the equality constraints are linearly independent.

The outer lines in figure 3.2 – illustrating the LICQ – represent some inequality constraints. If the optimal solution is located in the interior, none of the constraints is active and the LICQ is fulfilled. For the points in the second and the third case, the gradients of all active constraints are independent, eventually also fulfilling the LICQ. In the last two cases the gradients of the active constraints are dependent, resulting in the LICQ not being fulfilled.

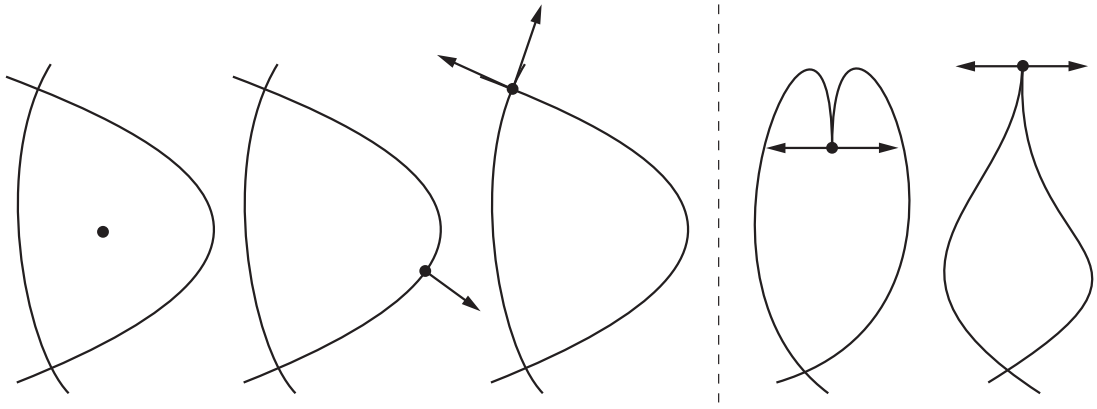

Figure 3.2: *Examples for the LICQ. In the first three points the LICQ are fulfilled, in the last two they are not. cf. [Ger12]*

The LICQ and alternative regularity conditions for optimization problems can e.g. be found in [BH75, Ger08, Bie10]. The regularity conditions are required for the KARUSH-KUHN-TUCKER conditions and the sufficient conditions of the next sections.

Karush-Kuhn-Tucker Conditions (KKT Conditions)

We assume $\mathbf{z}_{opt}$ to be a local minimum of the problem stated in (3.16)–(3.21). Furthermore, the conditions stated in the *LICQ* are fulfilled in $\mathbf{z}_{opt}$. Then, there exist unique LAGRANGE multipliers $\boldsymbol{\lambda}_{opt}$ and $\boldsymbol{\nu}_{opt}$ such that the following KARUSH-KUHN-TUCKER (KKT) conditions hold [KT51], [Ger08, p. 146], [Bie10, ch. 4], [BH75, Ben10]:

1. **Sign conditions**

$$\boldsymbol{\nu}_{opt} \geq 0 \tag{3.23}$$

2. **Optimality conditions**

$$\nabla_{\mathbf{z}} L(\mathbf{z}_{opt}, \boldsymbol{\lambda}_{opt}, \boldsymbol{\nu}_{opt}) = \nabla_{\mathbf{z}} J(\mathbf{z}_{opt}) + \nabla_{\mathbf{z}} \mathbf{h}(\mathbf{z}_{opt}) \boldsymbol{\lambda}_{opt} + \nabla_{\mathbf{z}} \mathbf{g}(\mathbf{z}_{opt}) \boldsymbol{\nu}_{opt} = 0 \tag{3.24}$$

3. **Complementary conditions**

$$g_i(\mathbf{z}_{opt}) \cdot \nu_{i,opt} = 0, \qquad \forall i = 1, \dots n_g \tag{3.25}$$

4. **Constraints**

$$\nabla_{\boldsymbol{\lambda}} L(\mathbf{z}_{opt}, \boldsymbol{\lambda}_{opt}, \boldsymbol{\nu}_{opt}) = \mathbf{h}(\mathbf{z}_{opt}) = 0 \tag{3.26}$$

$$\nabla_{\boldsymbol{\nu}} L(\mathbf{z}_{opt}, \boldsymbol{\lambda}_{opt}, \boldsymbol{\nu}_{opt}) = \mathbf{g}(\mathbf{z}_{opt}) \leq 0 \tag{3.27}$$

The complementary conditions in 3. can element-wise be further resolved to two cases:

- If the constraint $g_i(\mathbf{z}_{opt})$ is active, $g_i(\mathbf{z}_{opt}) = 0$ and $\nu_{i,opt} \geq 0$ (with sign conditions 1.).
- If the constraint $g_i(\mathbf{z}_{opt})$ is not active, $g_i(\mathbf{z}_{opt}) < 0$ and $\nu_{i,opt} = 0$.

Geometric Interpretation

The optimality conditions from (3.24) can be reformulated to:

$$-\nabla_{\mathbf{z}} J(\mathbf{z}_{opt}) = \nabla_{\mathbf{z}} \mathbf{h}(\mathbf{z}_{opt}) \boldsymbol{\lambda}_{opt} + \nabla_{\mathbf{z}} \mathbf{g}(\mathbf{z}_{opt}) \boldsymbol{\nu}_{opt} \tag{3.28}$$

This means that, in the optimal solution, the negative gradient of the cost function J can be written as a linear combination of the gradients of the active inequality constraints and the equality constraints. Special care has to be taken considering the sign of $\boldsymbol{\nu}_{opt} \geq 0$, not allowing all possible linear combinations. In general, only two cases can appear:

1. **Unconstrained in the current solution $\mathbf{z}_{opt}$:** Then the KKT point is located in the interior of the constrained set and the *Jacobian* of the cost function is equal to zero.

$$\nabla_{\mathbf{z}} J(\mathbf{z}_{opt}) = 0 \tag{3.29}$$

2. **Some or all constraints are active:** Then the KKT point is located on the border of the constrained set and the *Jacobian* of the cost function can be represented as a combination of the gradients of the active constraints (regular case as stated above).

Figure 3.3 shows two exemplary points illustrating the KKT conditions. In the left graph, the negative gradient of the cost function can be represented as a linear combination of the gradients of the (active) constraints with positive LAGRANGE multipliers. The considered cost function is

$$J(\mathbf{z}) = (z_1 - 1)^2 + (z_2 - 1)^2 \tag{3.30}$$

while the constraints are:

$$\mathbf{g} = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} \leq \mathbf{0} \tag{3.31}$$

In this case the gradients evaluate to:

$$\nabla_{\mathbf{z}} J(\mathbf{z}) = \begin{bmatrix} 2 \cdot (z_1 - 1) \\ 2 \cdot (z_2 - 1) \end{bmatrix}, \qquad \nabla_{\mathbf{z}} \mathbf{g}(\mathbf{z}) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.32}$$

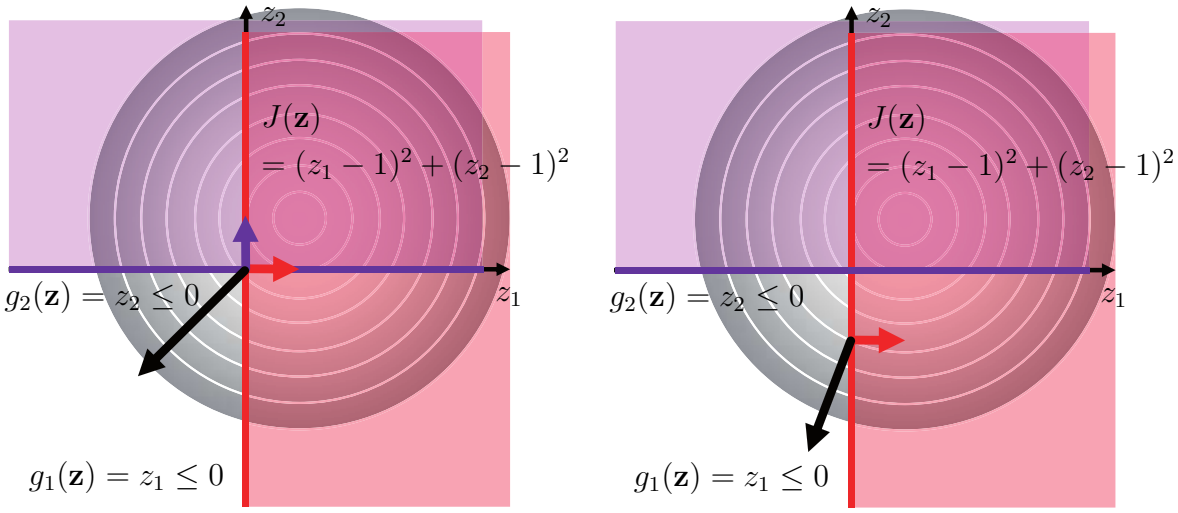

Figure 3.3: *Examples illustrating the KKT conditions. The point on the left is a KKT point, the one on the right not cf. [Hol+14].*

The optimal solution is

$$\mathbf{z}_{opt} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{3.33}$$

and consequently the LAGRANGE multipliers evaluate to:

$$-\nabla_{\mathbf{z}} J(\mathbf{z}_{opt}) = -\begin{pmatrix} -2 \\ -2 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \nu_1 + \begin{pmatrix} 0 \\ 1 \end{pmatrix} \nu_2 \tag{3.34}$$

$$\nu_1 = 2 \qquad \nu_2 = 2$$

The KKT conditions are necessary conditions which means that they are fulfilled in every optimal point as long as the LICQ are fulfilled and we remain in the framework discussed here (sufficiently smooth functions, etc.). On the other hand, in this framework, no optimal point can exist that violates the KKT conditions. In the right graph of figure 3.3, the considered point is not a valid KKT point as the gradient of the cost function (black) cannot be represented by the gradient of the active constraint (red). Furthermore, it can be seen that the cost can further be improved (until the optimal solution is reached) as there remains a component of the negative gradient that is not aligned perpendicular to the constraint. Or, in other words, if the KKT are fulfilled, a decrease in the cost function can only be achieved by violating the constraints (the improvement that can be achieved by a violation is related to the LAGRANGE multipliers, see section 3.1.6). Moreover, if the LICQ are not fulfilled the determination of the multipliers may have no unique solution anymore and no unique direction of improvement may be found. [BH75, Bie10, Hol+15]

Sufficient conditions

Using the necessary conditions (3.23)–(3.27) (KKT conditions), one can only determine candidates for optimal points. In order to validate if a given candidate is really an optimal solution

to the constrained optimization problem, sufficient conditions are required. For stating a sufficient condition [Ger08, sec. 5.3] introduces the critical cone T_K:

$$\begin{aligned} T_K(\mathbf{z}_{opt}) = \Big\{ d \in \mathbb{R}^{n_\mathbf{z}} \mid & \nabla \bar{g}_i(\mathbf{z}_{opt})^\intercal d \leq 0, \nu_{i,opt} = 0, \\ & \nabla \bar{g}_i(\mathbf{z}_{opt})^\intercal d = 0, \nu_{i,opt} > 0, \\ & \nabla h_j(\mathbf{z}_{opt})^\intercal d = 0 \} \end{aligned} \tag{3.35}$$

with $\bar{g}$ being the active subset of the inequality constraints.

Now, the sufficient conditions are:
If $(\mathbf{z}_{opt}, \boldsymbol{\lambda}_{opt}, \boldsymbol{\nu}_{opt})$ is a KKT point with

$$d^\intercal L''_{\mathbf{zz}}(\mathbf{z}_{opt}, \boldsymbol{\lambda}_{opt}, \boldsymbol{\nu}_{opt}) d > 0 \qquad \forall d \in T_K(\mathbf{z}_{opt}), d \neq 0 \tag{3.36}$$

of the problem stated above, then there exists a neighborhood $U_\delta(\mathbf{z}_{opt}) = \{\mathbf{z} \in \mathbb{R}^{n_\mathbf{z}} | \|\mathbf{z} - \mathbf{z}_{opt}\| \leq \delta\}$ with $\alpha > 0$ such that:

$$J(\mathbf{z}) \geq J(\mathbf{z}_{opt}) + \alpha \|\mathbf{z} - \mathbf{z}_{opt}\|^2 \qquad \forall \mathbf{z} \in U_\delta(\mathbf{z}_{opt}) \tag{3.37}$$

This means that $\mathbf{z}_{opt}$ is a local minimum of the problem and the cost function is growing at least quadratic in a neighborhood of $\mathbf{z}_{opt}$. Visually speaking, the *Hessian* of the *Lagrangian* must be positive definite in all directions of the critical cone, which means that its curvature must be positive in all directions of the critical cone.

3.1.3 Sequential Quadratic Programming (SQP)

As the conditions from 3.1.2 can – in most cases – not directly be used to calculate optimal points, a numerical approach is needed that returns (at least) candidates for optimal points. In section 3.1.1, the NEWTON method for unconstrained problems was presented, also introducing two ideas for motivating it. Here, again, the approach of approximating the function to be optimized by quadratic functions will be used. Now, the method will be applied to a problem with equality and inequality constraints. The considered auxiliary *Quadratic Programming (QP)* subproblem for the point $(\mathbf{z}, \boldsymbol{\lambda}, \boldsymbol{\nu})$ is

$$\begin{aligned} \min_{\mathbf{d} \in \mathbb{R}^{n_\mathbf{z}}} \quad & \frac{1}{2} \mathbf{d}^\intercal L''_{\mathbf{zz}}(\mathbf{z}, \boldsymbol{\lambda}, \boldsymbol{\nu}) \mathbf{d} + \nabla_\mathbf{z} J(\mathbf{z})^\intercal \mathbf{d} \\ \text{s. t.} \quad & g_i(\mathbf{z}) + \nabla_\mathbf{z} g_i(\mathbf{z})^\intercal \mathbf{d} \leq 0, \quad i = 1, \ldots, n_\mathbf{g} \\ & h_j(\mathbf{z}) + \nabla_\mathbf{z} h_j(\mathbf{z})^\intercal \mathbf{d} = 0, \quad j = 1, \ldots, n_\mathbf{h} \end{aligned} \tag{3.38}$$

Using the aforementioned stepwise quadratic approximation, the local *Sequential Quadratic Programming (SQP)* method can be derived.

The SQP algorithm requires the following steps:

1. Choose initial values $\left(\mathbf{z}^{[0]}, \boldsymbol{\lambda}^{[0]}, \boldsymbol{\nu}^{[0]}\right)$.

2. If the current point $\left(\mathbf{z}^{[k]}, \boldsymbol{\lambda}^{[k]}, \boldsymbol{\nu}^{[k]}\right)$ is a KKT point of the constrained optimization problem then STOP.

3. Otherwise, calculate a new KKT step $\left(\mathbf{d}^{[k]}, \boldsymbol{\lambda}^{[k+1]}, \boldsymbol{\nu}^{[k+1]}\right)$ by solving the QP problem (3.38).

4. Perform the step $\mathbf{z}^{[k+1]} = \mathbf{z}^{[k]} + \mathbf{d}^{[k]}$, increase the step counter $k = k + 1$ and continue with 2.

It can be shown that under certain conditions, the SQP method shows quadratic convergence to the real solution of the constrained problem.

Similar to the *Hessian* free methods presented in 3.1.1, the *Quasi-Newton-Method* can also be applied to constrained optimization problems. In this case, the BFGS update formula needs slight modifications and the resulting algorithm shows superlinear convergence properties. Details can e.g. be found in [Ger08, sec. 5.8], [Bet09, sec. 1.13], [Bie10].

There exists a variety of numerical optimization algorithms based on SQP methods that can be used to solve real life numerical optimization problems. All of them are far more sophisticated than the basic approach presented here, as one can imagine that there are many pitfalls in real applications that need to be overcome. This includes numerical problems as well as implementational details that were left out here for the sake of simplicity. Currently available implementations include e.g. *MATLAB*'s *fmincon* function (depending on the settings) [Mat], *SNOPT* [GMS02], *KNITRO* [Art], *SOCS* [Bet09], *WORHP* [BW13] and many others.

3.1.4 Interior Point Algorithm

The two basic ideas of an interior point algorithm are

- to introduce slack variables $\mathbf{s} > 0$ for the inequality conditions and
- to couple these to the cost function using logarithmic expressions with a weighting factor $\eta > 0$.

When doing so the following barrier problem results:
Determine $\mathbf{z} \in \mathbb{R}^{n_\mathbf{z}}$ and $\mathbf{s} \in \mathbb{R}^{n_\mathbf{g}}$, s. t.

$$\begin{aligned} & \min_{\mathbf{z},\mathbf{s}} J(\mathbf{z}) - \eta \sum_{i=1}^{n_\mathbf{g}} \log(s_i) \\ \text{under the constraints} \quad & g_i(\mathbf{z}) + s_i = 0, \qquad i = 1, \ldots, n_\mathbf{g} \\ & h_j(\mathbf{z}) = 0, \qquad j = 1, \ldots, n_\mathbf{h} \end{aligned} \tag{3.39}$$

Hence, the LAGRANGE function of the barrier problem becomes:

$$L(\mathbf{z}, \mathbf{s}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = J(\mathbf{z}) - \eta \sum_{i=1}^{n_\mathbf{g}} \log(s_i) + \boldsymbol{\nu}^\mathsf{T} \left(\mathbf{g}(\mathbf{z}) - \mathbf{s}\right) + \boldsymbol{\lambda}^\mathsf{T} \mathbf{h}(\mathbf{z}) \tag{3.40}$$

The KKT conditions for this equality constrained problem evaluate to

$$\nabla_{\mathbf{z}} L(\mathbf{z}, \mathbf{s}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = 0 \tag{3.41}$$
$$\nabla_{\mathbf{s}} L(\mathbf{z}, \mathbf{s}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = 0 \tag{3.42}$$
$$g_i(\mathbf{z}) + s_i = 0 \qquad i = 1, \ldots, n_{\mathbf{g}} \tag{3.43}$$
$$h_j(\mathbf{z}) = 0 \qquad j = 1, \ldots, n_{\mathbf{h}}. \tag{3.44}$$

Now, the optimality condition (3.41) is used to determine candidates for the optimal solution, introducing the diagonal Matrix $\mathbf{S} = diag(s_i)$:

$$\nabla_{\mathbf{z}} L(\mathbf{z}, \mathbf{s}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = \nabla_{\mathbf{z}} J(\mathbf{z}) + \nabla_{\mathbf{z}} \mathbf{g}(\mathbf{z}) \boldsymbol{\nu} + \nabla_{\mathbf{z}} \mathbf{h}(\mathbf{z}) \boldsymbol{\lambda} \tag{3.45}$$
$$\nabla_{\mathbf{s}} L(\mathbf{z}, \mathbf{s}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = -\eta \mathbf{S}^{-1} \mathbf{e} + \boldsymbol{\nu} \qquad \Rightarrow \eta = s_i \nu_i \tag{3.46}$$

with $\mathbf{e}$ being a vector of ones of appropriate size. The conclusion from equation (3.46) is used to determine s_i.

Again, the NEWTON method may be applied for solving the resulting problem. While doing so, the barrier parameter is stepwise decreased. As a large barrier parameter results in smoothed inequality constraints, the first solutions are forced further away from the border of the feasible set. Consequently, in this simplified version of the method, the sequence of iterates tends to approach the barrier from the inside and therefore "orthogonal" to the border instead of "tangential" along the constraints. An example for the augmented cost function

$$J(\mathbf{z}) - \eta \sum_{i=1}^{n_{\mathbf{g}}} \log(s_i) \tag{3.47}$$

for different values of η, also showing the described behavior, can be seen in figure 3.4. More details on the interior point method can be e.g. be found in [Ger08, sec. 5.11], [Bet09, p. 1.14], [Hol+14].

Similar to the SQP methods, there are algorithms available that are based on interior point methods. These include e.g. *MATLAB*'s *fmincon* function (depending on the settings) [Mat], *KNITRO* [Art] and *IPOPT* [WB06].

3.1.5 Sparse Optimization Problems

Before considering sparse optimization problems, a definition of the word "sparse" is required. In [Bet09, sec. 2.1], BETTS calls matrices sparse if *many of the elements are zero*. Furthermore, he states that for most problems the number of non-zero elements remains below 1%. When tackling sparse optimization problems, the *Jacobian* and the *Hessian* matrix of the problem are the matrices that have to be sparse in order to have a sparse problem. In this work, we mainly focus on the *Jacobian* of the problem, as the *Hessian* is mostly only approximated anyway. Moreover, in general, a structurally sparse *Jacobian* results in a mostly sparse *Hessian*, too. When considering discretized optimal control problems, sparse optimization problems mostly have a band structure, meaning that most of the non-zero entries are

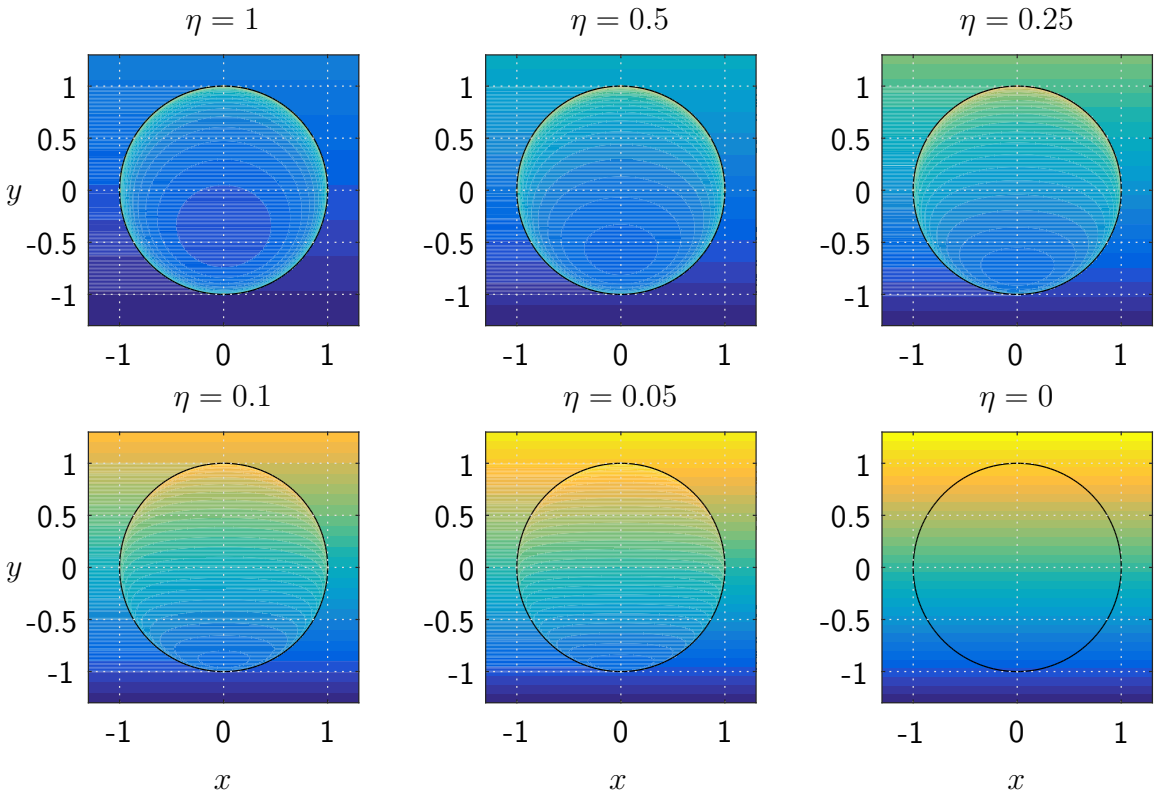

Figure 3.4: *Augmented cost function of the interior point method for different weighting factors η. The example shown is $\min_{(x,y)} J(x,y) = y$, under the conditions that $g(x,y) = y^2 + x^2 \leq 1$. The augmented cost function can only be evaluated inside the constrained area. [Bet09]*

grouped around the diagonal of the matrix. This structure can be created in discretized optimal control problems by sorting all values with respect to their time of appearance. More details on the use of problem sparsity for discretized optimal control problems can be found in chapter 7.

A lot of technologies exist for efficiently solving sparse numerical optimization problems. These include special approaches for the efficient calculation of the *Jacobian* or *Hessian* using finite differences if the sparsity pattern of a problem is known. Furthermore, special techniques for solving sparse QP problems as well as sparse formulations for merit functions exist. Using the aforementioned techniques, mostly sparse SQP and interior point algorithms can be built up. Examples for sparse *Non-Linear Programming (NLP)* problem implementations include *IPOPT* [WB06], *SNOPT* [GMS02] or *WORHP* [BW13]. In [Bet09], Betts gives an overview of the different techniques for sparse optimization problems.

3.1.6 Parametric Optimization Problems

In some cases, an optimization problem does not only depend on the parameters that are subject to optimization in order to minimize the given cost function, but also on additional parameters that influence the problem without being part of the solution. Problems of this

class are called parametric optimization problems and can generally be stated as

$$\begin{aligned} &\min_{\mathbf{z}} J(\mathbf{z}, \mathbf{p}) \\ \text{s. t.} \quad &\mathbf{g}(\mathbf{z}, \mathbf{p}) \leq 0 \\ &\mathbf{h}(\mathbf{z}, \mathbf{p}) = 0 \end{aligned} \tag{3.48}$$

for given parameters $\mathbf{p} \in \mathbb{R}^{n_\mathrm{P}}$.

Value Function

When analyzing this class of problems, the main question often is how different values for the parameter vector $\mathbf{p}$ influence the solution of the optimization problem. The function to be considered for this analysis is called *value function*:

$$V(\mathbf{p}) = \inf \{J(\mathbf{z}, \mathbf{p}) \,|\, \mathbf{g}(\mathbf{z}, \mathbf{p}) \leq 0, \mathbf{h}(\mathbf{z}, \mathbf{p}) = 0\} \tag{3.49}$$

It represents the greatest lower bound for the cost function of the parametric optimization problem. The investigation of this value function on the one hand leads to another interpretation of the LAGRANGE multipliers, which will later be used to establish a relationship between continuous optimal control problems and their discretized counterparts. Besides, the thoughts enable the fast calculation of suboptimal solutions for perturbed problems that may be used to perform on-line updates of optimal trajectories or that may be used to further improve the initial guess generation. Chapter 5 gives more details on the generation of initial guesses for optimal control problems. [Ger08]

Perturbed Optimization Problem

The perturbed optimization problem

$$\begin{aligned} &\min_{\mathbf{z}} J(\mathbf{z}) \\ \text{s. t.} \quad &\mathbf{g}(\mathbf{z}) + \epsilon_\mathbf{g} \leq \mathbf{0} \\ &\mathbf{h}(\mathbf{z}) + \epsilon_\mathbf{h} = \mathbf{0} \end{aligned} \tag{3.50}$$

represents a special case of a parametric optimization problem with $\epsilon_\mathbf{g}$ and $\epsilon_\mathbf{h}$ being arbitrary disturbance vectors of appropriate dimensions. In this case, $\epsilon_\mathbf{g}$ and $\epsilon_\mathbf{h}$ can be seen as the external parameters $\mathbf{p} = [\epsilon_\mathbf{g}, \epsilon_\mathbf{h}]^\mathsf{T}$ from (3.48) with nominal values $\epsilon_\mathbf{g} = \mathbf{0}$ and $\epsilon_\mathbf{h} = \mathbf{0}$.

Now, the value function of the optimization problem is defined as

$$V(\epsilon_\mathbf{g}, \epsilon_\mathbf{h}) = \inf_{\mathbf{z}} \{J(\mathbf{z}) \,|\, \mathbf{g}(\mathbf{z}) + \epsilon_\mathbf{g} \leq \mathbf{0}, \mathbf{h}(\mathbf{z}) + \epsilon_\mathbf{h} = \mathbf{0}\} \tag{3.51}$$

Analyzing the value function, the following relations between V and the LAGRANGE multipliers of the nominal optimization problem can be derived [BH75], [Ger08, p. 180ff.], [Ben10, p. 17]:

$$\frac{\partial V(0,0)}{\partial(\epsilon_\mathbf{h})} = \boldsymbol{\lambda}_{opt}^\mathsf{T} \tag{3.52}$$

$$\frac{\partial V(0,0)}{\partial(\epsilon_\mathbf{g})} = \boldsymbol{\nu}_{opt}^\mathsf{T} \tag{3.53}$$

This delivers another interpretation of the LAGRANGE multipliers: They are the local gradient of the value function of the optimization problem with respect to disturbances in the constraints. In section 3.2.4 discretized optimal control problems will be analyzed, where some of the equality constraints are a discretized representation of the differential equations governing the respective dynamic system. There, the LAGRANGE multipliers can be seen as a sensitivity of the optimal solution with respect to the dynamic equations (more details can be found in section 3.2.5).

Besides the analysis of parametric optimization problems, the value function can be used to formulate a dual problem which leads to other optimality criteria that can be used in numerical optimization algorithms. Further details can e.g. be found in [Ger08, GL11].

Sensitivities

Looking at the value function $V(\mathbf{p})$, one question to be answered is how the optimal solution of the underlying optimization problem changes, when the parameters $\mathbf{p}$ are perturbed. It is important to note that the question is **not** how the cost function value $J(\mathbf{z}, \mathbf{p})$ or the constraints change in the current point when altering $\mathbf{p}$ but one wants to know how the **optimal** point $(J_{opt}, \mathbf{z}_{opt}, \boldsymbol{\lambda}_{opt}, \boldsymbol{\nu}_{opt})$ moves with perturbations in $\mathbf{p}$. The value function can deliver information on how the cost function value J changes, but here the question is how the solution to the problem $\mathbf{z}_{opt}$ changes. Mathematically, the following derivatives in the nominal point $\hat{\mathbf{p}}$ are sought:

$$\frac{d\mathbf{z}_{opt}}{d\mathbf{p}}(\hat{\mathbf{p}}), \quad \frac{d\boldsymbol{\lambda}_{opt}}{d\mathbf{p}}(\hat{\mathbf{p}}), \quad \frac{d\boldsymbol{\nu}_{opt}}{d\mathbf{p}}(\hat{\mathbf{p}}) \tag{3.54}$$

Using the theorem of FIACCO [Fia76] – also referred to as the sensitivity theorem (e.g. by [Ger08, sec. 5.4], [Ger12, sec. 6.1.1], [Büs98, ch. 9]) – these derivatives can be calculated. The theorem requires the functions J, $\mathbf{g}$ and $\mathbf{h}$ to be twice continuous differentiable. Let

$$(\mathbf{z}_{opt}, \boldsymbol{\lambda}_{opt}, \boldsymbol{\nu}_{opt}) \tag{3.55}$$

be a unique and strictly regular local minimum. Then there exist neighborhoods

$$V_\epsilon(\hat{\mathbf{p}}) \quad \text{and} \quad U_\delta(\mathbf{z}_{opt}, \boldsymbol{\lambda}_{opt}, \boldsymbol{\nu}_{opt}) \tag{3.56}$$

in which the local minimum is pertained and the set of active constraints does not change. The sought derivatives can be calculated from

$$\begin{pmatrix} \frac{d\mathbf{z}}{d\mathbf{p}} \\ \frac{d\boldsymbol{\lambda}}{d\mathbf{p}} \\ \frac{d\boldsymbol{\nu}}{d\mathbf{p}} \end{pmatrix} = - \begin{bmatrix} \nabla^2_{\mathbf{zz}} L & \nabla_{\mathbf{z}}\mathbf{g} & \nabla_{\mathbf{z}}\mathbf{h} \\ \Lambda \cdot (\nabla_{\mathbf{z}}\mathbf{g})^\intercal & \Gamma & 0 \\ (\nabla_{\mathbf{z}}\mathbf{h})^\intercal & 0 & 0 \end{bmatrix}^{-1} \cdot \begin{pmatrix} \nabla^2_{\mathbf{zp}} L \\ \Lambda \cdot \nabla_{\mathbf{p}}\mathbf{g} \\ \nabla_{\mathbf{p}}\mathbf{h} \end{pmatrix} \tag{3.57}$$

with $\Lambda = diag(\boldsymbol{\lambda}_{opt})$ and $\Gamma = diag(\mathbf{g})$. All functions need to be evaluated in $(\mathbf{z}_{opt}, \boldsymbol{\lambda}_{opt}, \boldsymbol{\nu}_{opt}, \hat{\mathbf{p}})$. The theorem of FIACCO can be derived by applying the *implicit function theorem* to the *KKT* conditions. It is helpful to regard $\mathbf{z}$, $\boldsymbol{\lambda}$ and $\boldsymbol{\nu}$ as one vector of unknowns in this derivation.

3.2 Applied Optimal Control

This section gives a brief overview of the class of optimal control problems to which aircraft trajectory optimization problems belong to. More details on optimal control problems, the theory behind them and their application can e.g. be found in [Bet09, Büs98, Kir70, Ger09, Ger12, Ben10, Alb13, BH75].

This section is structured as follows: First, the general optimal control problem to be solved is stated. Next, transformation techniques are revisited that enable the formulation of one standard optimal control problem for several different problem formulations. In the next section these transformation techniques are used to reformulate the optimal control problem such that it can be tackled using the indirect solution approach. Therein, first, optimality conditions for unconstrained optimal control problems are derived and the general idea of costates is introduced and interpreted. Afterwards, the basic ideas required to extend the indirect approach to constrained problems is described. Subsection 3.2.4 introduces the basic principles required to solve optimal control problems based on the direct approach. Some facts about the required numerical simulation techniques, on sensitivity equations and the control parametrization are recalled, before the shooting and collocation approaches are discussed, both forming the basis for the solution of the problems presented in chapters 10 and 11. Moreover, the numerical scaling of optimal control problems is briefly discussed as it is crucial for a fast and stable solution process. Subsections 3.2.5 and 3.2.6 set the direct and indirect approaches for solving optimal control problems into relation and reveal some additional properties that are helpful for understanding optimal control problems better.

3.2.1 Optimal Control Problems

The main goal in optimal control is to determine time dependent state histories $\mathbf{x}(t)$, control histories $\mathbf{u}(t)$ and possibly additional parameters $\mathbf{p}$ minimizing a given cost function J, without violating any of the algebraic or dynamic constraints. The general optimal control problem arising in aircraft trajectory optimization, and thus also treated here, can be stated as follows: Find the optimal control history

$$\mathbf{u}_{opt}(t) \in \mathbb{R}^{n_\mathbf{u}} \tag{3.58}$$

and the corresponding optimal state trajectory

$$\mathbf{x}_{opt}(t) \in \mathbb{R}^{n_\mathbf{x}} \tag{3.59}$$

as well as the optimal parameters

$$\mathbf{p}_{opt} \in \mathbb{R}^{n_\mathbf{p}} \tag{3.60}$$

that minimize the BOLZA cost functional

$$J = e(\mathbf{x}(t_0), \mathbf{x}(t_f), \mathbf{p}, t_0, t_f) + \int_{t_0}^{t_f} L(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}, t)\, dt \tag{3.61}$$

subject to the dynamics:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}, t) \qquad \mathbf{f} : \mathbb{R}^{n_\mathbf{x}} \times \mathbb{R}^{n_\mathbf{u}} \times \mathbb{R}^{n_\mathbf{p}} \times \mathbb{R} \to \mathbb{R}^{n_\mathbf{x}} \tag{3.62}$$

the initial and final boundary conditions:

$$\mathbf{\Psi}_0(\mathbf{x}(t_0), \mathbf{p}, t_0) = 0 \qquad \mathbf{\Psi}_0 : \mathbb{R}^{n_\mathbf{x}} \times \mathbb{R}^{n_\mathbf{p}} \times \mathbb{R} \to \mathbb{R}^{n_{\mathbf{\Psi}_0}} \tag{3.63}$$

$$\mathbf{\Psi}_f(\mathbf{x}(t_f), \mathbf{p}, t_f) = 0 \qquad \mathbf{\Psi}_f : \mathbb{R}^{n_\mathbf{x}} \times \mathbb{R}^{n_\mathbf{p}} \times \mathbb{R} \to \mathbb{R}^{n_{\mathbf{\Psi}_f}} \tag{3.64}$$

the interior point constraints:

$$\mathbf{r}_i(\mathbf{x}(t_i), \mathbf{p}, t_i) = 0 \qquad \mathbf{r}_i : \mathbb{R}^{n_\mathbf{x}} \times \mathbb{R}^{n_\mathbf{p}} \times \mathbb{R} \to \mathbb{R}^{n_\mathbf{r}}, \quad \forall i = 1, \ldots, N_r \tag{3.65}$$

and the equality and inequality path constraints:

$$\mathbf{C}_{eq}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}, t) = 0 \qquad \mathbf{C}_{eq} : \mathbb{R}^{n_\mathbf{x}} \times \mathbb{R}^{n_\mathbf{u}} \times \mathbb{R}^{n_\mathbf{p}} \times \mathbb{R} \to \mathbb{R}^{n_{\mathbf{C}_{eq}}} \tag{3.66}$$

$$\mathbf{C}_{ineq}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}, t) \leq 0 \qquad \mathbf{C}_{ineq} : \mathbb{R}^{n_\mathbf{x}} \times \mathbb{R}^{n_\mathbf{u}} \times \mathbb{R}^{n_\mathbf{p}} \times \mathbb{R} \to \mathbb{R}^{n_{\mathbf{C}_{ineq}}} \tag{3.67}$$

In this formulation the final time t_f may be fixed or free. In aircraft trajectory optimization problems the final time is often subject to optimization and thus needs to be regarded as free.

Technically, optimal control problems may also be stated as multi phase problems, meaning that several sub-problems of the form presented above are combined into one big optimal control problem. The sub-problems may have their own time axis, their own constraints, parameters, and models and may be connected by additional constraints as necessary. The only restriction is, that all of them have to contribute (directly or indirectly) to one common cost function. This multi phase formulation allows the use of different models and different constraints in the different phases. [Hol15]

3.2.2 Transformation Techniques for Optimal Control Problems

The optimal control problem formulated in (3.58)–(3.67) is mathematically equivalent to several other formulations that may arise from similar applications. This section gives a short overview of the most important transformation techniques for optimal control problems. Again, it shall only give a glimpse on the relevant ideas, as more information on the transformations can, e. g., be found in [Ger09, Ger12, Büs98]. At the end of this section a problem formulation will be stated that is used for the derivation of the theoretical results in section 3.2.3.

Transformation of Problems with Free Initial or Final Time

Problems with free final (and) or initial time can be transformed into problems with fixed final (and) or initial time by introducing one additional parameter per free time that is used to perform a linear time transformation. In the following equations, $\tau \in [0, 1]$ represents the normalized time and t_0 and t_f are the initial and final time parameters. The real simulation time t is mapped to the normalized fixed time τ using the linear transformation:

$$t(\tau) = t_0 + (t_f - t_0) \cdot \tau \tag{3.68}$$

Consequently, the derivatives with respect to t have to be transformed, too. The transformed state dynamics resulting from

$$\frac{dt}{d\tau} = (t_f - t_0) \tag{3.69}$$

are:

$$\frac{d\mathbf{x}(t(\tau))}{d\tau} = \frac{d\mathbf{x}(t)}{dt} \cdot \frac{dt}{d\tau} = \mathbf{f}\left(\mathbf{x}(t(\tau)), \mathbf{u}(t(\tau)), \mathbf{p}, t(\tau)\right) \cdot (t_f - t_0) \tag{3.70}$$

Similarly, the LAGRANGE cost function becomes:

$$\int_{t_0}^{t_f} L(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}, t)\, dt = \int_{0}^{1} L(\mathbf{x}(t(\tau)), \mathbf{u}(t(\tau)), \mathbf{p}, t(\tau)) \cdot (t_f - t_0)\, d\tau \tag{3.71}$$

With the transformation technique shown below, the newly introduced parameters t_0 and t_f can also be further transformed to constant states with free initial conditions if one wants to remove the parameters from the problem formulation. [Ger12, Büs98, Hol15]

Transformation of Cost Function Formulations

Different cost function formulations for optimal control problems exist, also determining the name of the problem type. An optimal control problem is called LAGRANGE problem if the cost function is of the form:

$$J = \int_{t_0}^{t_f} L(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}, t)\, dt \tag{3.72}$$

In case the cost function is formulated as

$$J = e(\mathbf{x}(t_0), \mathbf{x}(t_f), \mathbf{p}, t_0, t_f) \tag{3.73}$$

the problem is called MAYER problem. A TSCHEBYSCHEFF problem results if the goal is to minimize a function of the form:

$$J = \max_{t \in [t_0, t_f]} f_0(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}, t) \tag{3.74}$$

From a mathematical point of view, all those cost function formulations are equivalent and also similar to the one given in (3.61). In order to transform a LAGRANGE problem into a MAYER problem an additional state keeping the integrated LAGRANGE cost needs to be added to the problem. Here, $\bar{\mathbf{x}}$ is the new, augmented state vector:

$$\begin{aligned} \dot{l}(t) &= L(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}, t), \qquad l(t_0) = 0 \\ \bar{\mathbf{x}} &= [\mathbf{x}^\intercal, l]^\intercal \\ J &= l(t_f) \end{aligned} \tag{3.75}$$

The inverse transformation in general uses the same idea but is far more complicated and may even be impossible, depending on the problem formulation.

When looking at the TSCHEBYSCHEFF formulation from (3.74) the introduction of an additional parameter p_T is required that is used as a limit within a new path constraint:

$$\begin{aligned} & p_T, \text{free} \\ & f_0(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}, t) \leq p_T \\ & J = p_T \end{aligned} \tag{3.76}$$

More details on these and other transformations can e.g. be found in [Bet09, Ger12, Büs98, Hol15].

Transformation of Problems with Interior Point Constraints

Problems with interior point constraints of the form

$$\mathbf{r}(\mathbf{x}(t_i), \mathbf{p}, t_i) = 0 \qquad \mathbf{r} \in \mathbb{R}^{n_\mathbf{r}} \tag{3.77}$$

can be transformed into problems without these constraints by dividing the overall time interval in multiple phases at the respective points t_i in time. Afterwards, constraints have to be added ensuring continuity of the relevant values and ensuring that the interior point conditions are met [Ger12, Büs98].

Transformation of Problems with Parameters

Problems with additional optimization parameters can be reformulated to optimal control problems without parameter dependencies by augmenting the state vector with constant states with free initial values representing the parameters:

$$\dot{\bar{\mathbf{x}}} = \begin{bmatrix} \dot{\mathbf{x}}(t) \\ \dot{\mathbf{p}}(t) \end{bmatrix} = \begin{bmatrix} \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}(t)) \\ \mathbf{0} \end{bmatrix} = \bar{\mathbf{f}}(\bar{\mathbf{x}}, \mathbf{u}) \tag{3.78}$$

$$\mathbf{p}(t_0) = \mathbf{p}_0, \qquad \text{free} \tag{3.79}$$

Transformation of Non-Autonomous Problems

Without loss of generality any non-autonomous problem can be transformed into an autonomous one, where neither $\mathbf{f}$ nor J nor the path constraints $\mathbf{C}_{eq}$ and $\mathbf{C}_{ineq}$ are dependent on t explicitly, by adding an additional state representing the simulation time [Ger09, Ger12, Büs98, Hol15]:

$$\dot{T}(t) = 1, \qquad T(0) = T_0 \tag{3.80}$$

$$\bar{\mathbf{x}}(t) = \begin{pmatrix} \mathbf{x}(t) \\ T(t) \end{pmatrix} \tag{3.81}$$

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}, T(t)) \tag{3.82}$$

$$\dot{\bar{\mathbf{x}}}(t) = \bar{\mathbf{f}}(\bar{\mathbf{x}}(t), \mathbf{u}(t), \mathbf{p}) \tag{3.83}$$

Normally, flight systems can be represented by an autonomous differential equation anyway.

Standard Optimal Control Problem

When applying the aforementioned transformations as necessary, the optimal control problem can always be stated in the form:
Determine the optimal states and controls as stated in (3.58)–(3.59) that minimize the cost function

$$J = e(\mathbf{x}(t_0), \mathbf{x}(t_f)) + \int_{t_0}^{t_f} L(\mathbf{x}(t), \mathbf{u}(t))\, dt \tag{3.84}$$

respecting the dynamic constraints

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \tag{3.85}$$

as well as the algebraic constraints:

$$\boldsymbol{\Psi}_0(\mathbf{x}(t_0)) = 0 \qquad \boldsymbol{\Psi}_0 : \mathbb{R}^{n_\mathbf{x}} \to \mathbb{R}^{n_{\boldsymbol{\Psi}_0}} \tag{3.86}$$

$$\boldsymbol{\Psi}_f(\mathbf{x}(t_f)) = 0 \qquad \boldsymbol{\Psi}_f : \mathbb{R}^{n_\mathbf{x}} \to \mathbb{R}^{n_{\boldsymbol{\Psi}_f}} \tag{3.87}$$

$$\mathbf{C}_{eq}(\mathbf{x}(t), \mathbf{u}(t)) = 0 \qquad \mathbf{C}_{eq} : \mathbb{R}^{n_\mathbf{x}} \times \mathbb{R}^{n_\mathbf{u}} \to \mathbb{R}^{n_{\mathbf{C}_{eq}}} \tag{3.88}$$

$$\mathbf{C}_{ineq}(\mathbf{x}(t), \mathbf{u}(t)) \leq 0 \qquad \mathbf{C}_{ineq} : \mathbb{R}^{n_\mathbf{x}} \times \mathbb{R}^{n_\mathbf{u}} \to \mathbb{R}^{n_{\mathbf{C}_{ineq}}} \tag{3.89}$$

with fixed initial and final time t_0 and t_f.

Consequently, this problem formulation will be used for the derivation of the theoretical results in the next section.

3.2.3 Indirect Solution Approach

The general idea of the indirect solution approach is to first derive optimality conditions for the problem stated above. Afterwards, these conditions are evaluated and thus a solution for the problem can be determined. From a historical point of view, the indirect solution approach for optimal control problems is the older one, emerging from the field of *calculus of variations*. The basic problem in this mathematical domain is to determine a continuous function that minimizes another functional. The basic principles of calculus of variations have been laid by Johann Bernoulli. In 1696 he – besides Jacob Bernoulli, Sir Isaac Newton, Guillaume Francois Antoine Marquis de L'Hospital, Ehrenfried Walter von Tschirnhaus and Gottfried Wilhelm Leibnitz – found a solution to the famous *Brachistochrone* problem. After those first steps, Euler and Lagrange made significant progress in that field deriving the famous *Euler-Lagrange Equation*. In the twentieth century Hestenes, Bellmann, and Isaacs on the one hand and Pontryagin, Boltyanskii, and Gamkrelidze on the other hand independently developed the *Minimum Principle* (sometimes also called *Maximum Principle*), laying the foundation for the optimal control theory as we know it today [Kir70, Ben10, PP12, Ger12, Ger09].

Most of the real world applications arising in the field of aircraft trajectory optimization are too complex to be solved using indirect methods. Consequently, this approach may be seen

as a pure "mathematical" approach helping a lot in understanding the properties of optimal control problems but not so much in determining solutions for real world applications.

Unconstrained Optimal Control Problems

First of all, problems without any path constraints and only having equality conditions for the initial and final boundaries are considered. Using the transformations from above, the resulting problem formulation is:
Find the control history $\mathbf{u}_{opt}(t) \in \mathbb{R}^{n_\mathbf{u}}$ and the state history $\mathbf{x}_{opt}(t) \in \mathbb{R}^{n_\mathbf{x}}$ that minimize the cost functional

$$J = e(\mathbf{x}(t_0), \mathbf{x}(t_f)) + \int_{t_0}^{t_f} L(\mathbf{x}(t), \mathbf{u}(t))\, dt \tag{3.90}$$

subject to the dynamics

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \tag{3.91}$$

and the boundary conditions

$$\boldsymbol{\Psi}_0(\mathbf{x}(t_0)) = 0 \tag{3.92}$$

$$\boldsymbol{\Psi}_f(\mathbf{x}(t_f)) = 0 \tag{3.93}$$

Similarly to the idea of LAGRANGE in section 3.1, for sufficiently smooth functions, an extended cost function $\bar{J}$ can be constructed by coupling the dynamic constraints to the cost function using multipliers. Opposite to the pure optimization in 3.1, now the multipliers $\boldsymbol{\lambda}(t) \in \mathbb{R}^n$ for the state equations are no single values any more, but continuous functions over time that are called costates. This idea has been taken from the calculus of variations approach. Anyway, the multipliers l_0 for the cost function components that are appearing there, have already been set to $l_0 = 1$ as the required conditions (e.g. MANGASARIAN-FROMOWITZ constraint qualification) for a regular problem are assumed to be fulfilled here (see [Ger09], [Ger12, sec. 2.3] for details):

$$\begin{aligned}\bar{J} = e(\mathbf{x}(t_0), \mathbf{x}(t_f)) + \boldsymbol{\sigma}_0^\mathsf{T}\boldsymbol{\Psi}_0(\mathbf{x}(t_0)) + \boldsymbol{\sigma}_f^\mathsf{T}\boldsymbol{\Psi}_f(\mathbf{x}(t_f)) + \\ \int_{t_0}^{t_f} L(\mathbf{x}(t), \mathbf{u}(t)) + \boldsymbol{\lambda}(t)^\mathsf{T}\left(\mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) - \dot{\mathbf{x}}(t)\right)\, dt\end{aligned} \tag{3.94}$$

In the following, the argument t will be omitted for better readability. For the sake of simplicity, the *Hamiltonian* for an optimal control problem in BOLZA form is defined as [Büs98, Ger09, Ger12, Ben10]:

$$H(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{u}) = L(\mathbf{x}, \mathbf{u}) + \boldsymbol{\lambda}^\mathsf{T}\mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{3.95}$$

For problems in MAYER formulation the *Hamiltonian* simplifies to:

$$H(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{u}) = \boldsymbol{\lambda}^\mathsf{T}\mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{3.96}$$

Applying the minimum principle to the problem stated above using the extended cost function (3.94), the following optimality conditions can be derived. A proof and more details can besides others be found in [Ger09, Ger12, Büs98].

If the MANGASARIAN-FROMOWITZ constraint qualification (or another set of appropriate conditions) is fulfilled, for the optimal solution $(\mathbf{x}_{opt}, \mathbf{u}_{opt})$ it holds that

$$\begin{aligned} &\exists \quad \boldsymbol{\lambda}_{opt} : [t_0, t_f] \to \mathbb{R}^{n_\mathbf{x}} \qquad \text{and} \\ &\exists \quad \boldsymbol{\sigma}_0 \in \mathbb{R}^{n_{\boldsymbol{\Psi}_0}} \\ &\exists \quad \boldsymbol{\sigma}_f \in \mathbb{R}^{n_{\boldsymbol{\Psi}_f}} \end{aligned} \tag{3.97}$$

such that the following conditions are fulfilled:

1. **Minimum condition**
   For all t where $\mathbf{u}_{opt}$ is continuous it holds

   $$H(\mathbf{x}_{opt}, \boldsymbol{\lambda}_{opt}, \mathbf{u}_{opt}) = \min_{\mathbf{u}} H(\mathbf{x}_{opt}, \boldsymbol{\lambda}_{opt}, \mathbf{u}) \tag{3.98}$$

   where U is the admissible set for the controls.

   Remark. *For sufficiently smooth functions, the minimum condition can be weakened to*

   $$H'_{\mathbf{u}} = 0 \tag{3.99}$$

   In order to achieve strong minimum conditions either (3.98) *has to hold, or* (3.99) *and the so-called* LEGENDRE-CLEBSH *condition*

   $$H''_{\mathbf{uu}} \geq 0 \tag{3.100}$$

   both have to hold.

2. **Adjoint differential equation**
   For all t where $\mathbf{u}_{opt}$ is continuous it holds

   $$\dot{\boldsymbol{\lambda}} = -\nabla_{\mathbf{x}} H(\mathbf{x}_{opt}, \boldsymbol{\lambda}, \mathbf{u}_{opt}) = -\nabla_{\mathbf{x}} \mathbf{f}(\mathbf{x}_{opt}, \mathbf{u}_{opt}) \cdot \boldsymbol{\lambda} - \nabla_{\mathbf{x}} L(\mathbf{x}_{opt}, \mathbf{u}_{opt}) \tag{3.101}$$

3. **Transversality conditions**

   $$\begin{aligned} \boldsymbol{\lambda}(t_0) &= -\ \nabla_{\mathbf{x}(t_0)} \left(e(\mathbf{x}_{opt}(t_0), \mathbf{x}_{opt}(t_f)) + \boldsymbol{\sigma}_0^\mathsf{T} \boldsymbol{\Psi}_0(\mathbf{x}_{opt}(t_0)) \right) \\ &= -\ \left(\nabla_{\mathbf{x}(t_0)} e + \nabla_{\mathbf{x}(t_0)} \boldsymbol{\Psi}_0 \cdot \boldsymbol{\sigma}_0 \right) \\ \boldsymbol{\lambda}(t_f) &= \quad \nabla_{\mathbf{x}(t_f)} \left(e(\mathbf{x}_{opt}(t_0), \mathbf{x}_{opt}(t_f)) + \boldsymbol{\sigma}_f^\mathsf{T} \boldsymbol{\Psi}_f(\mathbf{x}_{opt}(t_f)) \right) \\ &= \quad \left(\nabla_{\mathbf{x}(t_f)} e + \nabla_{\mathbf{x}(t_f)} \boldsymbol{\Psi}_f \cdot \boldsymbol{\sigma}_f \right) \end{aligned} \tag{3.102}$$

4. In case of free final time

   $$H(\mathbf{x}_{opt}(t_{f,opt}), \boldsymbol{\lambda}(t_{f,opt}), \mathbf{u}_{opt}(t_{f,opt})) = 0 \tag{3.103}$$

5. Furthermore, for autonomous problems as discussed here, it holds that

$$\frac{d}{dt}H(\mathbf{x}_{opt}, \boldsymbol{\lambda}, \mathbf{u}_{opt}) = 0 \tag{3.104}$$

and consequently H is constant over time

$$H(\mathbf{x}_{opt}, \boldsymbol{\lambda}, \mathbf{u}_{opt}) = L(\mathbf{x}_{opt}, \mathbf{u}_{opt}) + \boldsymbol{\lambda}^{\mathsf{T}}\mathbf{f}(\mathbf{x}_{opt}, \mathbf{u}_{opt}) = \text{const.}, \quad \forall t \in [t_0, t_f] \tag{3.105}$$

For autonomous problems with free final time:

$$H(\mathbf{x}_{opt}(t), \boldsymbol{\lambda}(t), \mathbf{u}_{opt}(t)) = 0, \qquad \text{for almost all } t \in [t_0, t_f] \tag{3.106}$$

The *Hamiltonian* is called regular if the condition (3.98) determines the control uniquely in some neighborhood of $\mathbf{x}_{opt}$ and $\boldsymbol{\lambda}_{opt}$ for all $t \in [t_0, t_f]$. In this case, the optimal control $\mathbf{u}_{opt}$ is a continuous function in t [Ben10].

In cases where the controls only appear linearly in the dynamic constraints and the cost function, the solution has a special bang-bang structure. In these cases, the optimal solution for the control takes values at the upper or lower limit of the given control range and possibly has singular arcs in between. In some cases these singular arcs cannot be determined exactly. As this is an inherent structural problem it also appears if the direct approach from 3.2.4 is used. See section 3.2.6 for more details concerning this issue. [Büs98, Ger09]

Solution Process Using Indirect Methods

In case of a regular *Hamiltonian* (controls appearing non-linearly) the procedure for solving an optimal control problem is as follows:

1. Solve equation (3.98) (or its weakened version) for $\mathbf{u}_{opt}(t)$.
2. Eliminate $\mathbf{u}_{opt}(t)$ in equations (3.91) and (3.101) leading to the following system of dynamic equations:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}_{opt}) = \mathbf{h}_1(\mathbf{x}, \boldsymbol{\lambda}) \tag{3.107}$$

$$\dot{\boldsymbol{\lambda}} = -\nabla_x H(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{u}_{opt}) = \mathbf{h}_2(\mathbf{x}, \boldsymbol{\lambda}) \tag{3.108}$$

3. Solve this resulting two-point boundary value problem for $\mathbf{x}(t)$ and $\boldsymbol{\lambda}(t)$ using the initial and final conditions from (3.92), (3.93) and (3.102).
4. Evaluate the solution of step 1. using the solution of step 3. to get the control history $\mathbf{u}_{opt}(t)$.

The main difficulty when applying this solution scheme lies in finding a solution for the resulting boundary value problem. Anyway, as long as no inequality constraints are present it is possible to numerically determine a solution for this problem by e.g. solving it using a multiple shooting approach similar to the one shown in 3.2.4. Compared to the direct solution approach presented in 3.2.4, where the minimization of the cost still needs to be performed, here a pure boundary value problem has to be solved. In the latter case, the conditions for obtaining a minimum have already been evaluated in the optimality conditions before.

Interpretation of the Costates

Recalling the findings from the geometric interpretation in section 3.1.2 as well as the findings on the LAGRANGE multipliers from 3.1.6, one may find an interpretation for the costates $\boldsymbol{\lambda}$, accordingly. A value function for continuous optimal control problems may be defined as:

$$V(\hat{t}, \hat{\mathbf{x}}) = \inf_{\mathbf{u}} J(\mathbf{x}, \mathbf{u}) \quad \text{s. t. } \mathbf{x}_{opt}(\hat{t}) = \hat{\mathbf{x}} \tag{3.109}$$

In the optimal solution, $V(\hat{t}, \mathbf{x}_{opt}) = J(\mathbf{x}_{opt}, \mathbf{u}_{opt})$. Using the HAMILTON-JACOBI-BELLMAN equation, (assuming that V is differentiable) it can be shown that:

$$\frac{\partial V(t, \mathbf{x})}{\partial \mathbf{x}} = \boldsymbol{\lambda}^{\intercal}(t) \tag{3.110}$$

Details and references for further reading can e.g. be found in [Ger09, sec. 7.1], [Ben10].

It can be seen from the *Lagrangian* of the optimal control problem in equation (3.94) that the *Hamiltonian* represents one part of it. In this setting, the costates can be interpreted equivalently to the LAGRANGE multipliers from section 3.1.2, applied to the dynamic constraints posed by the differential equation (3.91). Above, the LAGRANGE multipliers were constant vectors, whereas now, the costate vectors are functions over time. Using the value function (and assuming sufficient smoothness), the costates can be seen as the sensitivity of the optimal value of the cost function with respect to the states:

$$\delta J_{opt} = \boldsymbol{\lambda}^{\intercal}(t) \cdot \delta \mathbf{x}(t) \tag{3.111}$$

Moreover, it has to be emphasized that the costates give the local sensitivity of the *optimal* cost value with respect to the states, meaning that they represent the first order approximation of the optimal solution of the optimal control problem with respect to the state histories. As the states cannot be chosen freely but have to fulfill the dynamic equation of the system, the costates – similarly to the LAGRANGE multipliers in 3.1.2 – are a measure for how the cost function could be improved in case the dynamic constraints were changed.

Constrained Optimal Control Problems

When using the indirect solution approach one has to distinguish between problems with combined state and control constraints and problems with pure state constraints. As here only the basic idea should be presented, this work focuses on problems with combined state and control constraints. These constraints can be stated as

$$\boldsymbol{S}(\mathbf{x}(t), \mathbf{u}(t)) \leq 0 \qquad \forall t \in [t_0, t_f] \tag{3.112}$$

In order to tackle this problem, the extended HAMILTON function is defined:

$$\begin{aligned} \tilde{H}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}, \mathbf{u}) &= L(\mathbf{x}, \mathbf{u}) + \boldsymbol{\lambda}^{\intercal}\mathbf{f}(\mathbf{x}, \mathbf{u}) + \boldsymbol{\mu}^{\intercal}\boldsymbol{S}(\mathbf{x}, \mathbf{u}) \\ &= H(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{u}) + \boldsymbol{\mu}^{\intercal}\boldsymbol{S}(\mathbf{x}, \mathbf{u}) \end{aligned} \tag{3.113}$$

For problems

- without pure control constraints,
- if all relevant functions are sufficiently smooth,
- the MANGASARIAN-FROMOWITZ constraint qualification (or another appropriate qualification) is fulfilled,
- and the solution is feasible,

then for a solution $(\mathbf{u}_{opt}, \mathbf{x}_{opt})$, there exist multipliers $\boldsymbol{\lambda}_{opt}$, $\boldsymbol{\mu}_{opt}$, $\boldsymbol{\sigma}_0$, and $\boldsymbol{\sigma}_f$ such that the following conditions hold:

1. **Minimum condition**
   For almost all t it holds that $\mathbf{u}_{opt}$ may be determined from
   $$\mathbf{u}_{opt} = \arg\min_{\mathbf{u}} \tilde{H}(\mathbf{x}_{opt}, \boldsymbol{\lambda}_{opt}, \boldsymbol{\mu}_{opt}, \mathbf{u}_{opt}) \tag{3.114}$$
   which is equivalent to
   $$\tilde{H}'_{\mathbf{u}}(\mathbf{x}_{opt}, \boldsymbol{\lambda}_{opt}, \boldsymbol{\mu}_{opt}, \mathbf{u}_{opt}) = 0 \tag{3.115}$$
   as no pure control constraints are considered here.

2. **Adjoint differential equation**
   $$\dot{\boldsymbol{\lambda}} = -\nabla_{\mathbf{x}} \tilde{H}(\mathbf{x}_{opt}, \boldsymbol{\lambda}, \boldsymbol{\mu}_{opt}, \mathbf{u}_{opt}) \tag{3.116}$$

3. **Transversality conditions**
   $$\begin{aligned} \boldsymbol{\lambda}(t_0) &= -\ \nabla_{\mathbf{x}(t_0)} \left(e(\mathbf{x}_{opt}(t_0), \mathbf{x}_{opt}(t_f)) + \boldsymbol{\sigma}_0^{\mathsf{T}} \boldsymbol{\Psi}_0(\mathbf{x}_{opt}(t_0))\right) \\ &= -\ \left(\nabla_{\mathbf{x}(t_0)} e + \nabla_{\mathbf{x}(t_0)} \boldsymbol{\Psi}_0 \cdot \boldsymbol{\sigma}_0\right) \\ \boldsymbol{\lambda}(t_f) &= \quad \nabla_{\mathbf{x}(t_f)} \left(e(\mathbf{x}_{opt}(t_0), \mathbf{x}_{opt}(t_f)) + \boldsymbol{\sigma}_f^{\mathsf{T}} \boldsymbol{\Psi}_f(\mathbf{x}_{opt}(t_f))\right) \\ &= \quad \left(\nabla_{\mathbf{x}(t_f)} e + \nabla_{\mathbf{x}(t_f)} \boldsymbol{\Psi}_f \cdot \boldsymbol{\sigma}_f\right) \end{aligned} \tag{3.117}$$

4. **Complementary conditions**
   For almost all t it holds:
   $$\boldsymbol{\mu}_{opt}(t)^{\mathsf{T}} \boldsymbol{S}(\mathbf{x}_{opt}(t), \mathbf{u}_{opt}(t)) = 0, \qquad \boldsymbol{\mu}_{opt}(t) \geq 0 \tag{3.118}$$

In case no combined control and state constraints appear in the problem the conditions get more complicated as the adjoint variables may be discontinuous then. More details can e.g. be found in [Büs98, Ger09, Ger12, BH75, Bet09].

The procedure for solving optimal control problems with inequality constraints using the indirect approach is similar to the procedure for the unconstrained case that was presented above but with one critical change. Now, the complementary conditions from (3.118) need to be taken into account, too. The problem is now, that one needs to know in which segment of the solution which set of constraints is active in order to determine the solution. Consequently, information about the structure of the solution is necessary in order to determine the solution of the problem. In most cases this information is not available.

An Engineers View on Indirect Methods

When using indirect methods for solving optimal control problems, BETTS identifies three major drawbacks in [Bet09, sec. 4.3]:

1. The computation of the *Hamiltonian* and the required derivatives may be complex and furthermore needs to be redone each time a new problem should be solved.
2. Problems with path inequality conditions require an a priori estimate of the sequence of constrained arcs.
3. The method lacks robustness as good initial guesses for the costates $\boldsymbol{\lambda}$ are required which are hard to estimate because they normally do not have a physical meaning.

Due to these difficulties that arise when using indirect optimization methods, they will not be considered in more detail in the remainder of this work. The following section outlines the most important ideas of direct discretization methods and the required numerical optimization procedures. However, understanding the general principles of optimal control problems will help understand the mechanisms presented here to improve the solution methods.

3.2.4 Direct Solution Approach

When using the direct solution approach, the optimal control problem is first discretized and then optimized whereas in the indirect approach it was done the other way round. Instead of deriving optimality conditions for the optimal control problem, this section focuses on techniques to numerically discretize the problems in order to prepare them for a numerical optimization. The discretized problem represents a regular (and mostly sparse) numerical optimization problem that can be solved using the ideas from section 3.1. Here, problems explicitly depending on additional parameters $\mathbf{p}$ are considered.

In [Ger09, sec. 6.1], GERDTS lists the three main components for solving a discretized optimal control problem as:

- **The control parametrization.** The control history $\mathbf{u}(t)$ needs to be represented by a finite set of parameters $\mathbf{u}_k$.
- **The discretization scheme for the differential equation.** The differential equation $\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p})$ needs to be discretized using e. g. a numerical integration or collocation scheme.
- **The numerical optimization.** The problem resulting from the discretization is a non-linear, constrained (mostly sparse) optimization problem that needs to be solved numerically.

These main components and their relationships will be described in more detail in the next sections of this work.

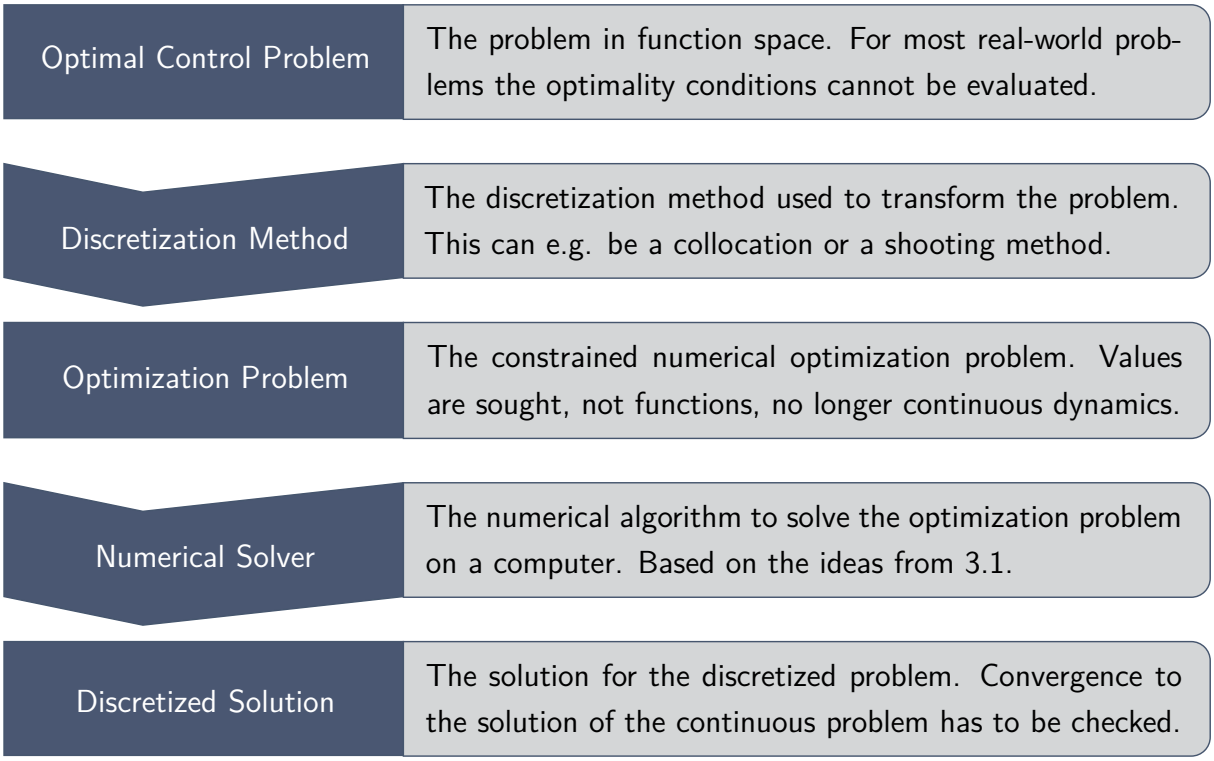

Figure 3.5: *Solution scheme used in direct discretization methods.*

Figure 3.5 illustrates the basic steps of a direct solution scheme. First, the continuous optimal control problem is stated. Afterwards, the problem is discretized over time (or the respective independent variable of the problem) using an appropriate discretization method. After doing so, a constrained numerical optimization problem of the form presented in 3.1.2 results. This problem needs to be solved numerically where the ideas from section 3.1.3, 3.1.4 and 3.1.5 are helpful. As the discretized problems may become relatively large, even though they are sparse, this step normally requires considerable computational power. This is also the reason why the direct solution approach became popular during the last couple of years, with the wide availability of increasingly powerful computer systems. The result of the numeric optimization is a discretized representation of the solution to the optimal control problem. As this solution does not necessarily need to converge to the solution of the original problem it is generally recommended to evaluate the optimality conditions from section 3.2.3 for validation.

The most intuitive way of motivating the direct solution methods is from a discretization of the state and control histories. Instead of using the continuous dynamics as stated in the problem definition in (3.62) and a continuous function $\mathbf{u}(t)$ the functions are transformed into sequences $\{\mathbf{x}_k = \mathbf{x}(t_k)\}$ and $\{\mathbf{u}_k = \mathbf{u}(t_k)\}$ on discrete points $t_k \in \{t_0, \ldots, t_f\}$ with $k = 0, \ldots, N$ in time. After transforming all relevant constraints to this grid, a numerical optimization problem results. Besides, the direct discretization approach may also be derived using the idea of dynamic programming (see [Kir70, Ben10]).

There exists a variety of methods for transforming optimal control problems into optimization problems. Like the widely used *Pseudospectral Methods* that use global approximation

polynomials for the state and the control histories. In order to avoid oscillations in the global representation, the grid points may not be chosen freely. One possible set of grid points is determined by the roots of so-called LEGENDRE-GAUSS-LOBATTO polynomials. The main benefit of the pseudospectral methods is their good convergence behavior, while the main drawback is that only smooth time histories can be represented accurately by the global approximation. To overcome this issue, most pseudospectral codes (like e.g. GPOPS-II [PR14]) come with algorithms that determine necessary stitching points and afterwards combine the initial problem out of several smaller subproblems – the idea used is similar to the multi-phase idea described in section 3.2.1. More details on pseudospectral methods can besides other be found in [Ben10].

In general, the solution of multi-phase optimal control problems using direct discretization techniques is possible by first discretizing the problem for every phase and then combining them together in one numerical optimization problem. Additionally, the cost functions of all phases need to be summed up and the constraint vectors need to be "stacked" together.

After presenting required techniques for numerical simulation and sensitivity calculation, the shooting approach as well as the collocation approach will be revisited based on the optimal control problem formulation stated in equations (3.58)–(3.67).

Numerical Simulation

In order to be able to discretize the differential equation of the model dynamics, several methods exist. Here, mainly one-step methods are discussed as they are widely used in optimal control.

For sufficiently smooth functions, the continuous dynamics

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{p}), \quad \mathbf{x}(t_0) = \mathbf{x}_0 \tag{3.119}$$

may be approximated by a discrete system of the form:

$$\begin{aligned} \mathbf{x}(t_{k+1}) &= \mathbf{x}(t_k) + h_k \cdot \Phi(\mathbf{x}(t_k), \mathbf{x}(t_{k+1}), \bar{\mathbf{u}}, \mathbf{p}; h_k) \\ \mathbf{x}(t_0) &= \mathbf{x}_0 \end{aligned} \tag{3.120}$$

on a grid

$$\mathbb{G}_N = \{t_0 < t_1 < t_2 < \cdots < t_{N-1} < t_N = t_f\} \tag{3.121}$$

with the time step

$$h_k = t_{k+1} - t_k, \tag{3.122}$$

using numerical integration techniques. The function Φ is called increment function. In the following, the nomenclature $\mathbf{x}(t_k) = \mathbf{x}_k$ will be used. Additionally, the input h_k to Φ will be omitted for notational convenience. Besides, the control parametrization from equation (3.148) is used with the parameters introduced there being combined in $\bar{\mathbf{u}} = [\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_N]$. Consequently, $\mathbf{u}(t)$ can be determined at any point in time.

If Φ does only depend on values at time t_k the method is called explicit and the determination of the value $\mathbf{x}_{k+1}$ from the value $\mathbf{x}_k$ is possible without further efforts:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + h_k \cdot \Phi(\mathbf{x}_k, \bar{\mathbf{u}}, \mathbf{p}) = \mathbf{x}_k + h_k \cdot \Phi[t_k] \tag{3.123}$$

Otherwise, if Φ depends on the state value $\mathbf{x}_{k+1}$, the method is called implicit and the calculation of $\mathbf{x}_{k+1}$ requires the solution of the (mostly non-linear) system of equations (3.120).

The most simple approach for solving this problem is the explicit forward EULER method, where $\Phi = \mathbf{f}(\mathbf{x}_k, \mathbf{u}(t_k), \mathbf{p})$ and consequently:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + h_k \cdot \mathbf{f}(\mathbf{x}_k, \mathbf{u}(t_k), \mathbf{p}) \tag{3.124}$$

Similarly, the most simple implicit integration method is the implicit backward EULER method with:

$$\Phi[t_k] = \mathbf{f}(\mathbf{x}_{k+1}, \mathbf{u}(t_{k+1}), \mathbf{p}) \tag{3.125}$$

The benefits of implicit methods are better stability properties, in some cases justifying the computational burden required to solve the non-linear system of equations.

Generalizing the idea of the EULER method to a multi stage approach (multi stage methods should not be confused with multi step methods), one ends at the general RUNGE-KUTTA methods. The general s-stage RUNGE-KUTTA method is defined by the coefficients b_j, c_j, a_{ij} with $i, j = 1, \ldots, s$ and the increment equation

$$\mathbf{x}_{k+1} = \mathbf{x}_k + h_k \sum_{j=1}^{s} b_j \mathbf{k}_j \left(\mathbf{x}_k, \bar{\mathbf{u}}, \mathbf{p}\right), \tag{3.126}$$

with the stage derivatives:

$$\mathbf{k}_j \left(\mathbf{x}_k, \bar{\mathbf{u}}, \mathbf{p}\right) = \mathbf{f}\left(\mathbf{x}_k + h_k \sum_{l=1}^{s} a_{jl} \mathbf{k}_l \left(\mathbf{x}_k, \mathbf{u}(t_k)\right), \mathbf{u}(t_k + c_j h_k), \mathbf{p}\right) \tag{3.127}$$

Therein, using the control parametrization and the respective parameters $\bar{\mathbf{u}}$, the values $\mathbf{u}(t_k + c_j h_k)$ can be calculated. The coefficients b_j, c_j, and a_{ij} are often written into a condensed table, called the BUTCHER tableau, uniquely defining a RUNGE-KUTTA scheme. Table 3.1 shows the BUTCHER tableaus for some numerical integration schemes, with subtable 3.1a giving the general scheme and subtable 3.1b showing the widely used short notation. In an explicit RUNGE-KUTTA scheme, the upper right triangular part of the matrix $\mathbf{A}$ needs to contain only zeros that are normally omitted in the tableaus.

The local discretization error $\mathbf{l}_h$ of a numerical integration scheme is defined as the difference between the real solution $\tilde{\mathbf{x}}$ of the respective differential equation $\mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{p})$ and the solution $\Phi(\mathbf{x}(t_k), \mathbf{x}(t_{k+1}), \bar{\mathbf{u}}, \mathbf{p})$ calculated in one integration step using a numerical integration method:

$$\mathbf{l}_h(t_k) = \frac{\tilde{\mathbf{x}}(t_{k+1}) - \mathbf{x}(t_k)}{h} - \Phi\left(\mathbf{x}(t_k), \mathbf{x}(t_{k+1}), \bar{\mathbf{u}}, \mathbf{p}\right) \tag{3.128}$$

The order of consistency p of a numerical integration scheme is defined by

$$\max_{t_k} \|\mathbf{l}_h(t_k)\| \leq C h^p \qquad \forall \quad 0 < h < h_0 \tag{3.129}$$

Table 3.1: BUTCHER *tableaus for numerical integration methods*

(a) *General*

c_1	a_{11}	a_{12}	$\dots$	a_{1s}
c_2	a_{21}	a_{22}	$\dots$	a_{2s}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
c_s	a_{s1}	a_{s2}	$\dots$	a_{ss}
	b_1	b_2	$\dots$	b_s

(b) *General short*

$\mathbf{c}$	$\mathbf{A}$
	$\mathbf{b}^\mathsf{T}$

(c) EULER *forward method*

0	
	1

(d) EULER *backward method*

1	1
	1

(e) *Trapezoidal rule*

0		
1	$\frac{1}{2}$	$\frac{1}{2}$
	$\frac{1}{2}$	$\frac{1}{2}$

(f) *Classical* RUNGE-KUTTA *method*

0				
$\frac{1}{2}$	$\frac{1}{2}$			
$\frac{1}{2}$	0	$\frac{1}{2}$		
1	0	0	1	
	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{6}$

with the constants $C > 0$ and $h_0 > 0$ independent of h. The consistency order is widely used to characterize the accuracy of a numerical integration scheme. Table 3.2 lists some commonly used methods as well as their order and type. Special care has to be taken as the conditions for determining the order of a RUNGE-KUTTA scheme from the coefficients are more strict for optimal control problems than for pure forward integration (see [Hag00, Ger09, Ger12] for more details). However, the methods listed in table 3.2 fulfill all the requirements for optimal control problems and feature the same order for forward integration and for integration in optimal control problems.

There exist two ways of improving the accuracy of a numerical simulation, as can clearly be seen from (3.129): Either the order p is increased or the step size h is decreased. The problem when lowering the step size too far, is that in parts of a trajectory that do not require a small step size, additional (and superfluous) computations are done. To overcome this issue, automatic step size selection algorithms exist. More details can e.g. be found in [Ger12].

Besides the selection of one step methods shortly listed here, a variety of other methods like

Table 3.2: *Selected numerical integration methods*

Method	Type	Order
EULER Forward	Explicit	1
EULER Backward	Implicit	1
Method of HEUN	Explicit	2
Trapezoidal Rule	Implicit	2
Classical RUNGE-KUTTA Method	Explicit	4
Fourth Order LOBATTO IIIA	Implicit	4

other one step approaches or multi step methods exist. More details on numerical integration methods can be found in many sources, like e.g. [Ger12, Bet09].

Sensitivity Equations

As gradient based optimization algorithms will be used to solve the discretized optimal control problem, it is necessary to calculate the gradient of the discretized optimal control problem with respect to all relevant optimization parameters. When using a shooting method (see the section below for details) a forward integration is performed and the gradients of the integrated states need to be calculated. Besides numerical finite differences (which is quite costly in this case), for sufficiently smooth functions, these gradients can be calculated using sensitivity equations [Ger12, Hol15].

Starting from the general parametric dynamic system

$$\dot{\mathbf{x}}(t, \mathbf{p}) = \mathbf{f}(\mathbf{x}(t, \mathbf{p}), \mathbf{p}), \qquad \mathbf{x}(t_0, \mathbf{p}) = \mathbf{x}_0(\mathbf{p}) \tag{3.130}$$

the derivative of (3.130) with respect to the parameter $\mathbf{p}$ evaluates to:

$$\frac{d\dot{\mathbf{x}}(t, \mathbf{p})}{d\mathbf{p}} = \frac{d\mathbf{f}(\mathbf{x}(t, \mathbf{p}), \mathbf{p})}{d\mathbf{p}} = \frac{\partial \mathbf{f}(\mathbf{x}(t, \mathbf{p}), \mathbf{p})}{\partial \mathbf{x}(t, \mathbf{p})} \cdot \frac{d\mathbf{x}(t, \mathbf{p})}{d\mathbf{p}} + \frac{\partial \mathbf{f}(\mathbf{x}(t, \mathbf{p}), \mathbf{p})}{\partial \mathbf{p}} \tag{3.131}$$

$$\frac{d\mathbf{x}(t_0, \mathbf{p})}{d\mathbf{p}} = \frac{d\mathbf{x}_0(\mathbf{p})}{d\mathbf{p}} \tag{3.132}$$

In this derivation, a control parametrization like the one presented in equation (3.148) in the next section is assumed. The respective parameters can be seen to be part of the parameter vector $\mathbf{p}$. With

$$\frac{d\dot{\mathbf{x}}(t, \mathbf{p})}{d\mathbf{p}} = \frac{d}{d\mathbf{p}} \left(\frac{d\mathbf{x}(t, \mathbf{p})}{dt} \right) = \frac{d}{dt} \left(\frac{d\mathbf{x}(t, \mathbf{p})}{d\mathbf{p}} \right), \tag{3.133}$$

the sensitivity matrix

$$\mathbf{S}(t) = \frac{d\mathbf{x}(t, \mathbf{p})}{d\mathbf{p}}, \tag{3.134}$$

and the short form of the partial derivatives

$$\frac{\partial \mathbf{f}(\mathbf{x}(t,\mathbf{p}),\mathbf{p})}{\partial \mathbf{x}(t,\mathbf{p})} = \mathbf{f}'_{\mathbf{x}}, \qquad \frac{\partial \mathbf{f}(\mathbf{x}(t,\mathbf{p}),\mathbf{p})}{\partial \mathbf{p}} = \mathbf{f}'_{\mathbf{p}}, \tag{3.135}$$

equation (3.131) can be reformulated as:

$$\dot{\mathbf{S}}(t) = \mathbf{f}'_{\mathbf{x}} \cdot \mathbf{S}(t) + \mathbf{f}'_{\mathbf{p}} = \mathbf{G}(\mathbf{S},\mathbf{x},\mathbf{p}) \tag{3.136}$$

$$\mathbf{S}(t_0) = \mathbf{S}_0 \tag{3.137}$$

It is worth mentioning that equation (3.136) contains one column for every parameter appearing in the problem (as will later be shown, this may be model parameters, initial states, or the discretized controls) and integrated along with the dynamic equations of the system.

In the next paragraphs the numerical integration of the sensitivity equations will be compared to the differentiation of the numerical integration scheme for the states. When using the sensitivity equations together with a numerical integration scheme, they must be integrated numerically on the same grid as the dynamics themselves (using the same integration method) in order to achieve accurate results. Applying the general RUNGE-KUTTA integration scheme from equation (3.126) to equation (3.136), one integration step of the sensitivity equation results as

$$\mathbf{S}_{k+1} = \mathbf{S}_k + h_k \sum_{j=1}^{s} b_j \mathbf{K}_j(\mathbf{S}_k, \bar{\mathbf{x}}_{kj}, \mathbf{p}) \tag{3.138}$$

with the stage derivatives from equation (3.127)

$$\mathbf{K}_j(\mathbf{S}_k, \bar{\mathbf{x}}_{kj}, \mathbf{p}) = \mathbf{G}\left(\mathbf{S}_k + h_k \sum_{l=1}^{s} a_{jl} \mathbf{K}_l(\mathbf{S}_k, \bar{\mathbf{x}}_{kl}, \mathbf{p}), \bar{\mathbf{x}}_{kj}, \mathbf{p}\right), \tag{3.139}$$

using the intermediate stages $\bar{\mathbf{x}}_{kj}$ and $\bar{\mathbf{x}}_{kl}$ from the integration of the state dynamics (see equation (3.143)).

On the other side, one integration step of the state dynamics needs to be considered. When deriving the equation for one numerical integration step with respect to $\mathbf{p}$, the following equation can be derived from equations (3.126):

$$\frac{d\mathbf{x}_{k+1}}{d\mathbf{p}} = \frac{d}{d\mathbf{p}}\left(\mathbf{x}_k + h_k \sum_{j=1}^{s} b_j \mathbf{k}_j\left(\mathbf{x}_k, \mathbf{p}\right)\right) \tag{3.140}$$

$$= \frac{d\mathbf{x}_k}{d\mathbf{p}} + \frac{d}{d\mathbf{p}}\left(h_k \sum_{j=1}^{s} b_j \mathbf{k}_j(\mathbf{x}_k, \mathbf{p})\right)$$

$$= \frac{d\mathbf{x}_k}{d\mathbf{p}} + h_k \sum_{j=1}^{s} b_j \frac{d}{d\mathbf{p}}\left(\mathbf{k}_j(\mathbf{x}_k, \mathbf{p})\right) \tag{3.141}$$

The same can be done for the stage derivatives (equation (3.127)):

$$\frac{d}{d\mathbf{p}}\left(\mathbf{k}_j(\mathbf{x}_k, \mathbf{p})\right) = \frac{d}{d\mathbf{p}}\left(\mathbf{f}(\mathbf{x}_k + h_k \sum_{l=1}^{s} a_{jl} \mathbf{k}_l(\mathbf{x}_k, \mathbf{p}), \mathbf{p})\right) \tag{3.142}$$

Using the abbreviation

$$\bar{\mathbf{x}}_{kj} = \mathbf{x}_k + h_k \sum_{l=1}^{s} a_{jl} \mathbf{k}_l(\mathbf{x}_k, \mathbf{p}) \tag{3.143}$$

for the intermediate stages, equation (3.142) can be written as:

$$\begin{aligned}
\frac{d}{d\mathbf{p}} \left(\mathbf{k}_j(\mathbf{x}_k, \mathbf{p}) \right) &= \frac{d}{d\mathbf{p}} \left(\mathbf{f}(\bar{\mathbf{x}}_{kj}, \mathbf{p}) \right) \\
&= \frac{\partial \mathbf{f}(\bar{\mathbf{x}}_{kj}, \mathbf{p})}{\partial \bar{\mathbf{x}}_{kj}} \cdot \frac{d\bar{\mathbf{x}}_{kj}}{d\mathbf{p}} + \frac{\partial \mathbf{f}(\bar{\mathbf{x}}_{kj}, \mathbf{p})}{\partial \mathbf{p}} \\
&= \frac{\partial \mathbf{f}(\bar{\mathbf{x}}_{kj}, \mathbf{p})}{\partial \bar{\mathbf{x}}_{kj}} \cdot \left(\frac{d\mathbf{x}_k}{d\mathbf{p}} + h_k \sum_{l=1}^{s} a_{jl} \frac{d}{d\mathbf{p}} \mathbf{k}_l(\mathbf{x}_k, \mathbf{p}) \right) + \frac{\partial \mathbf{f}(\bar{\mathbf{x}}_{kj}, \mathbf{p})}{\partial \mathbf{p}}
\end{aligned} \tag{3.144}$$

Now, for comparison, taking equation (3.139) and replacing $\mathbf{S}_k$ and $\mathbf{G}$ by their definitions,

$$\mathbf{K}_j(\mathbf{S}_k, \bar{\mathbf{x}}_{kj}, \mathbf{p}) = \frac{\partial \mathbf{f}(\bar{\mathbf{x}}_{kj}, \mathbf{p})}{\partial \bar{\mathbf{x}}_{kj}} \cdot \left(\frac{d\mathbf{x}_k}{d\mathbf{p}} + h_k \sum_{l=1}^{s} a_{jl} \mathbf{K}_l(\mathbf{S}_k, \bar{\mathbf{x}}_{kl}, \mathbf{p}) \right) + \frac{\partial \mathbf{f}(\bar{\mathbf{x}}_{kj}, \mathbf{p})}{\partial \mathbf{p}} \tag{3.145}$$

results. Finally, equations (3.144) and (3.145) can be compared, showing that for a general RUNGE-KUTTA scheme the numerical integration of the sensitivity dynamic equation and the differentiation of the integration scheme lead to the same results *if* the gradients of $\mathbf{f}$ are evaluated at the intermediate stages from the integration of the state dynamics. Then

$$\frac{d}{d\mathbf{p}} \left(\mathbf{k}_j(\mathbf{x}_k, \mathbf{p}) \right) = \mathbf{K}_j(\mathbf{S}_k, \bar{\mathbf{x}}_{kj}, \mathbf{p}) \tag{3.146}$$

and consequently:

$$\frac{d}{d\mathbf{p}} \left(\mathbf{k}_l(\mathbf{x}_k, \mathbf{p}) \right) = \mathbf{K}_l(\mathbf{S}_k, \bar{\mathbf{x}}_{kl}, \mathbf{p}) \tag{3.147}$$

Control Parametrization

As stated above, one component required for solving optimal control problems using a discretization scheme is the parameterization of the control history $\mathbf{u}(t)$. In general, almost any control parameterization

$$\mathbf{u}(t) = \mathbf{u}(t, \mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_N), \tag{3.148}$$

mapping a set of parameter values to the continuous control history, can be used. Anyway, as highlighted in section 3.1.5, it is desirable to obtain sparse numerical optimization problems. Hence, control parameterizations having local support are favorable as there one control value only has an influence on a limited time range of the control function, resulting in more sparse optimization problems. Chapter 7 gives more details on problem sparsity considerations. In [Ger12, sec. 5.1.3], GERDTS recommends the use of B-splines as basis functions for the control discretization. In order to use B-splines an arbitrarily chosen (but not necessarily equidistant) grid

$$\mathbb{G}_{\mathbf{u}} = \{ t_{\mathbf{u},0} < t_{\mathbf{u},1} < \cdots < t_{\mathbf{u},N-1} < t_{\mathbf{u},N} = t_f \} \tag{3.149}$$

is required. Afterwards, the control history can be reconstructed based on that grid.

The easiest continuous B-splines are second order B-splines that result in a linear interpolation scheme:

$$\mathbf{u}(t) = \mathbf{u}_k + (\mathbf{u}_{k+1} - \mathbf{u}_k) \cdot \frac{t - t_{\mathbf{u},k}}{t_{\mathbf{u},k+1} - t_{\mathbf{u},k}}, \qquad t_{\mathbf{u},k} \leq t < t_{\mathbf{u},k+1} \tag{3.150}$$

It can clearly be seen, that the linear interpolation scheme features local support as changing a control value $\mathbf{u}_k$ only influences the grid segments $t_{k-1} < t < t_k$ and $t_k < t < t_{k+1}$. When later calculating the gradients of the control parametrization

$$\frac{\partial \mathbf{u}(t)}{\partial \mathbf{z}} = \frac{\partial \mathbf{u}(t)}{\partial \mathbf{u}_k} \cdot \frac{\partial \mathbf{u}_k}{\partial \mathbf{z}} \tag{3.151}$$

it becomes clear that the local support leads to more sparse gradient matrices for the discretized problem (see equation (3.155) for details on the vector $\mathbf{z}$, and section 7.2 for details on sparsity).

The first and second order approximation have the nice feature of being interpolating approximations, meaning that:

$$\mathbf{u}(t_k) = \mathbf{u}_k \quad \forall k = 0, \ldots, N \tag{3.152}$$

In many real world application, limits for the control values can directly be specified as $\mathbf{u}_{min} \leq \mathbf{u}(t) \leq \mathbf{u}_{max}$. Even though the higher order splines are not interpolating, they are smoothing, meaning that choosing

$$\mathbf{u}_{min} \leq \mathbf{u}_k \leq \mathbf{u}_{max} \qquad \forall k = 0, \ldots, N \tag{3.153}$$

directly results in:

$$\mathbf{u}_{min} \leq \mathbf{u}(t) \leq \mathbf{u}_{max} \qquad \forall t \in [t_0, t_f] \tag{3.154}$$

Besides the classical B-spline representation, cardinal B-splines may be used for approximating the controls [MU10]. They are more simple to calculate as the same basis function (which is just shifted) is used for all sampling points. The main drawback is that cardinal B-splines with an order higher than two are neither passing through the interior points nor through the boundary points, where regular B-splines meet the first and the last point of the approximation.

Single Shooting Method

The probably most intuitive approach to directly solve optimal control problems is via direct single shooting. Therein, the control history is parametrized using the control discretization from above. Besides, the system dynamics are simulated using the numerical simulation methods from above. Overall, apart from the control values $\mathbf{u}_k$, the initial state $\mathbf{x}_0$ and the additional problem parameters $\mathbf{p}$ that are subject to optimization need to be combined to the numerical optimization vector:

$$\mathbf{z} = \begin{pmatrix} \mathbf{x}_0 \\ \mathbf{u}_0 \\ \vdots \\ \mathbf{u}_N \\ \mathbf{p} \end{pmatrix} \tag{3.155}$$

The values for this vector need to be determined by the numerical optimization algorithm during the optimization process.

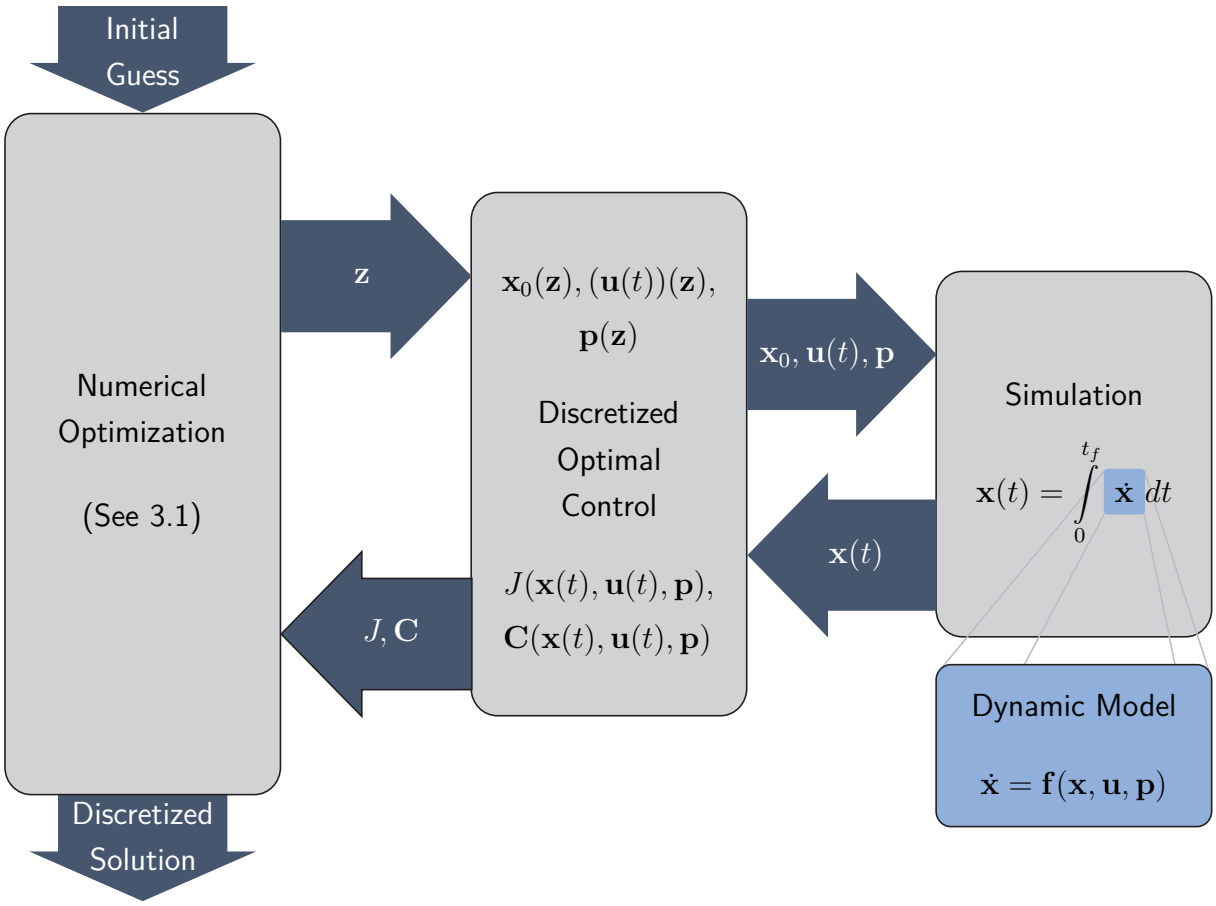

Figure 3.6: *General scheme of the shooting approach. Gradient information is omitted for better readability. [Hol15]*

Having the discretized controls and the numerical integration techniques at hand, the process for solving an optimal control problem is straight forward. Figure 3.6 gives an overview of the main components involved in the solution of an optimal control problem using shooting methods as well as their interaction.

- The block on the very left represents the whole numerical optimization from section 3.1, which needs to be initialized with an initial guess for the problem. The initial guess may be specified as a continuous function together with the continuous optimal control problem before also being discretized or it may directly be specified for the discretized problem.
- In the first iteration, the initial guess is used as the optimization parameter vector $\mathbf{z}$ and is fed into the function comprising the discretized optimal control problem. This function extracts the initial state, the discretized control histories and the additional required parameters from the parameter vector. It is represented by the block in the middle of figure 3.6.

- After reconstructing the control history using the selected control parameterization, the simulation of the dynamic model can be performed (see the blocks on the right of figure 3.6). This simulation may either be done on a fixed grid, which does not have to be equal to the control grid, or even on a variable grid determined by an automatic step size selection algorithm during the simulation. Anyway, if choosing a grid based control representation, the grid points of the control grid should also be part of the integration grid as otherwise changes in the control values cannot be resolved correctly.
- Afterwards, the resulting state trajectory is used to calculate the cost function value J as well as all relevant constraints $\mathbf{C}$ within the function taking care of the discretization of the optimal control problem in the block in the middle. These values and their derivatives, namely the cost and its gradient, and the constraints and their *Jacobian* are then required to solve the numerical optimization problem.
- In order to calculate these aforementioned derivatives, on the one hand, the chain rule is very helpful as it enables the step-wise calculation of the derivatives not requiring a full derivation of the mapping from the parameters of the optimal control problem to the cost and constraint values. FISCH gives a detailed explanation on how to use the chain rule in discretized optimal control problems in [Fis11]. On the other hand, the sensitivity equations from section 3.2.4 can be used to calculate the derivatives of the states along the simulation.
- Using J and $\mathbf{C}$ and their derivatives, the numerical optimization algorithm in the block on the left can determine the next iterate of the parameter vector $\mathbf{z}$, starting the next optimization iteration.

In order to be able to solve problems with variable initial and final time, the linear time transformation from section 3.2.2 needs to be applied, resulting in the extended state dynamics

$$\mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{p}) = \bar{\mathbf{f}}\,(\mathbf{x}, \mathbf{u}, \mathbf{p}) \cdot (t_f(\mathbf{p}) - t_0(\mathbf{p})), \tag{3.156}$$

with $\bar{\mathbf{f}}$ being the original state dynamics and $\mathbf{f}$ being the new transformed state dynamics. Furthermore, the cost function (3.61) can be transformed to its discretized representation

$$\begin{aligned} J(\mathbf{z}) = e(\mathbf{x}_0(\mathbf{z}), \mathbf{x}_f(\mathbf{z}), \mathbf{p}(\mathbf{z}), t_0(\mathbf{z}), t_f(\mathbf{z})) + \\ \int_0^1 \bar{L}(\mathbf{x}(\mathbf{z}), \mathbf{u}(\mathbf{z}), \mathbf{p}(\mathbf{z}), t(\mathbf{z})) \cdot (t_f(\mathbf{p}) - t_0(\mathbf{p}))\, d\tau \end{aligned} \tag{3.157}$$

with

$$L(\mathbf{x}(\mathbf{z}), \mathbf{u}(\mathbf{z}), \mathbf{p}(\mathbf{z}), t(\mathbf{z})) = \bar{L}(\mathbf{x}(\mathbf{z}), \mathbf{u}(\mathbf{z}), \mathbf{p}(\mathbf{z}), t(\mathbf{z})) \cdot (t_f(\mathbf{p}) - t_0(\mathbf{p})) \tag{3.158}$$

where the continuous integral is also replaced by a numerical integration algorithm. By col-

lecting all discretized constraints of the problem, the constraint vector

$$\mathbf{C}(\mathbf{z}) = \begin{pmatrix} \boldsymbol{\Psi}_0(\mathbf{x}_0(\mathbf{z}), \mathbf{p}(\mathbf{z}), t_0(\mathbf{z})) = 0 \\ \boldsymbol{\Psi}_f(\mathbf{x}_f(\mathbf{z}), \mathbf{p}(\mathbf{z}), t_f(\mathbf{z})) = 0 \\ \mathbf{r}(\mathbf{x}_i(\mathbf{z}), \mathbf{p}(\mathbf{z}), t_i(\mathbf{z})) = 0 \\ \bar{\mathbf{C}}_{eq}(\mathbf{x}(\mathbf{z}), \mathbf{u}(\mathbf{z}), \mathbf{p}(\mathbf{z}), t(\mathbf{z})) = 0 \\ \bar{\mathbf{C}}_{ineq}(\mathbf{x}(\mathbf{z}), \mathbf{u}(\mathbf{z}), \mathbf{p}(\mathbf{z}), t(\mathbf{z})) \leq 0 \end{pmatrix} = \begin{pmatrix} \boldsymbol{\Psi}_0(\mathbf{z}) = 0 \\ \boldsymbol{\Psi}_f(\mathbf{z}) = 0 \\ \mathbf{r}(\mathbf{z}) = 0 \\ \bar{\mathbf{C}}_{eq}(\mathbf{z}) = 0 \\ \bar{\mathbf{C}}_{ineq}(\mathbf{z}) \leq 0 \end{pmatrix} \tag{3.159}$$

results. The discretized constraints $\bar{\mathbf{C}}_{eq}$ and $\bar{\mathbf{C}}_{ineq}$ contain the values of the continuous path constraints $\mathbf{C}_{eq}$ and $\mathbf{C}_{ineq}$ evaluated on an appropriate user defined grid.

If the gradients are evaluated using the sensitivity equations from 3.2.4, the following relationships hold:

$$\frac{d\dot{\mathbf{x}}(t)}{d\mathbf{x}_0} = \frac{d}{dt}\left(\frac{d\mathbf{x}(t)}{d\mathbf{x}_0}\right) = \frac{\partial \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p})}{\partial \mathbf{x}(t)} \cdot \frac{d\mathbf{x}(t)}{d\mathbf{x}_0} \tag{3.160}$$

$$\frac{d\dot{\mathbf{x}}(t)}{d\mathbf{u}_k} = \frac{d}{dt}\left(\frac{d\mathbf{x}(t)}{d\mathbf{u}_k}\right) = \frac{\partial \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p})}{\partial \mathbf{x}(t)} \cdot \frac{d\mathbf{x}(t)}{d\mathbf{u}_k} + \frac{\partial \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p})}{\partial \mathbf{u}(t)} \cdot \frac{d\mathbf{u}(t)}{d\mathbf{u}_k} \tag{3.161}$$

$$\frac{d\dot{\mathbf{x}}(t)}{d\mathbf{p}} = \frac{d}{dt}\left(\frac{d\mathbf{x}(t)}{d\mathbf{p}}\right) = \frac{\partial \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p})}{\partial \mathbf{x}(t)} \cdot \frac{d\mathbf{x}(t)}{d\mathbf{p}} + \frac{\partial \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p})}{\partial \mathbf{p}} \tag{3.162}$$

Using similar abbreviations as above,

$$\mathbf{S}_{\mathbf{x}\mathbf{x}_0} = \frac{d\mathbf{x}(t)}{d\mathbf{x}_0}, \quad \mathbf{S}_{\mathbf{x}\mathbf{u}_k} = \frac{d\mathbf{x}(t)}{d\mathbf{u}_k}, \quad \mathbf{S}_{\mathbf{x}\mathbf{p}} = \frac{d\mathbf{x}(t)}{d\mathbf{p}}, \quad \mathbf{S}_{\mathbf{u}\mathbf{u}_k} = \frac{d\mathbf{u}(t)}{d\mathbf{u}_k},$$
$$\mathbf{f}'_{\mathbf{x}} = \frac{\partial \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p})}{\partial \mathbf{x}(t)}, \quad \mathbf{f}'_{\mathbf{u}} = \frac{\partial \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p})}{\partial \mathbf{u}(t)}, \quad \mathbf{f}'_{\mathbf{p}} = \frac{\partial \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p})}{\partial \mathbf{p}},$$

the equations can be written as:

$$\dot{\mathbf{S}}_{\mathbf{x}\mathbf{x}_0} = \mathbf{f}'_{\mathbf{x}} \cdot \mathbf{S}_{\mathbf{x}\mathbf{x}_0} \tag{3.163}$$

$$\dot{\mathbf{S}}_{\mathbf{x}\mathbf{u}_k} = \mathbf{f}'_{\mathbf{x}} \cdot \mathbf{S}_{\mathbf{x}\mathbf{u}_k} + \mathbf{f}'_{\mathbf{u}} \cdot \mathbf{S}_{\mathbf{u}\mathbf{u}_k} \tag{3.164}$$

$$\dot{\mathbf{S}}_{\mathbf{x}\mathbf{p}} = \mathbf{f}'_{\mathbf{x}} \cdot \mathbf{S}_{\mathbf{x}\mathbf{p}} + \mathbf{f}'_{\mathbf{p}} \tag{3.165}$$

The overall state *Jacobian* can then be evaluated using:

$$\begin{aligned} \frac{d\mathbf{x}(t)}{d\mathbf{z}} &= \frac{d\mathbf{x}(t)}{d\mathbf{x}_0} \cdot \frac{d\mathbf{x}_0}{d\mathbf{z}} + \sum_k \frac{d\mathbf{x}(t)}{d\mathbf{u}_k} \cdot \frac{d\mathbf{u}_k}{d\mathbf{z}} + \frac{d\mathbf{x}(t)}{d\mathbf{p}} \cdot \frac{d\mathbf{p}}{d\mathbf{z}} \\ &= \mathbf{S}_{\mathbf{x}\mathbf{x}_0} \cdot \frac{d\mathbf{x}_0}{d\mathbf{z}} + \sum_k \mathbf{S}_{\mathbf{x}\mathbf{u}_k} \cdot \frac{d\mathbf{u}_k}{d\mathbf{z}} + \mathbf{S}_{\mathbf{x}\mathbf{p}} \cdot \frac{d\mathbf{p}}{d\mathbf{z}} \end{aligned} \tag{3.166}$$

The evaluation of the local derivatives $\frac{d\mathbf{x}_0}{d\mathbf{z}}$, $\frac{d\mathbf{u}_k}{d\mathbf{z}}$ and $\frac{d\mathbf{p}}{d\mathbf{z}}$ is trivial as all those values explicitly appear in $\mathbf{z}$ resulting in constant matrices.

Multiple Shooting Method

As one can imagine, the single shooting approach may, in some cases, be very sensitive to the initial state values or control values at the beginning of the trajectory, resulting in large

sensitivities and in some cases even in instabilities in the simulation. To overcome these issues and to additionally increase the sparsity of the problem (see chapter 7 for more details), the idea of multiple shooting came up. Therein, the simulation interval is partitioned into several subintervals, called multiple shooting segments. Instead of simulating the whole time range of the dynamics in one sweep, the integration is stopped at the boundaries of the multiple shooting segments and afterwards restarted using a set of intermediate states $\mathbf{x}_0^m$ – where $m = 1, \ldots, M$ is the index of the current segment and M is the total number of segments – that are also determined by the optimization algorithm. The multiple shooting segments are defined on a grid:

$$\mathbb{G}_{\mathbf{x}_0} = \{t_{\mathbf{x}_0,0} < t_{\mathbf{x}_0,1} < \cdots < t_{\mathbf{x}_0,M}\}, \qquad \text{with } t_{\mathbf{x}_0,M} < t_f \tag{3.167}$$

Again, it is recommended to have a control grid point located at every multiple shooting node as otherwise a change in multiple control values may affect multiple segments of the discretized problem which has a negative influence on the sparsity of the problem. Figure 3.7 shows the general idea of multiple shooting compared to single shooting. Summing up, the optimization parameter vector becomes:

$$\mathbf{z} = \begin{pmatrix} \mathbf{x}_0^1 & \mathbf{x}_0^2 & \ldots & \mathbf{x}_0^M & \mathbf{u}_0 & \ldots & \mathbf{u}_N & \mathbf{p} \end{pmatrix}^\intercal \tag{3.168}$$

In order to ensure the state history forming a valid solution to the problem, the last state of each segment has to be equal to the first one of the subsequent segment. This is ensured using additional equality constraints that might not be fulfilled as long as the optimization is in progress but that have to be fulfilled as soon as a valid solution is found. The overall constraint vector consequently becomes

$$\mathbf{C}(\mathbf{z}) = \begin{pmatrix} \boldsymbol{\Psi}_0(\mathbf{x}_0(\mathbf{z}), \mathbf{p}(\mathbf{z}), t_0(\mathbf{z})) = 0 \\ \boldsymbol{\Psi}_f(\mathbf{x}_f(\mathbf{z}), \mathbf{p}(\mathbf{z}), t_f(\mathbf{z})) = 0 \\ \mathbf{r}(\mathbf{x}_i(\mathbf{z}), \mathbf{p}(\mathbf{z}), t_i(\mathbf{z})) = 0 \\ \mathbf{c}_m(\mathbf{x}(\mathbf{z}), \mathbf{u}(\mathbf{z}), \mathbf{p}(\mathbf{z}), t(\mathbf{z})) = 0 \\ \bar{\mathbf{C}}_{eq}(\mathbf{x}(\mathbf{z}), \mathbf{u}(\mathbf{z}), \mathbf{p}(\mathbf{z}), t(\mathbf{z})) = 0 \\ \bar{\mathbf{C}}_{ineq}(\mathbf{x}(\mathbf{z}), \mathbf{u}(\mathbf{z}), \mathbf{p}(\mathbf{z}), t(\mathbf{z})) \leq 0 \end{pmatrix} \tag{3.169}$$

where

$$\mathbf{c}_m(\mathbf{x}(\mathbf{z}), \mathbf{u}(\mathbf{z}), \mathbf{p}(\mathbf{z}), t(\mathbf{z})) = \mathbf{x}_0^m(\mathbf{z}) + \int_{t_{\mathbf{x}_0,m}}^{t_{\mathbf{x}_0,m+1}} \dot{\mathbf{x}}(t)\, dt - \mathbf{x}_0^{m+1}(\mathbf{z}) \tag{3.170}$$
$$\forall m = 1, \ldots, M-1$$

is called multiple shooting defect of segment m.

For multiple shooting, the sensitivity equations from above can be used similarly to the single shooting case. The control and parameter sensitivities do not change but need to be

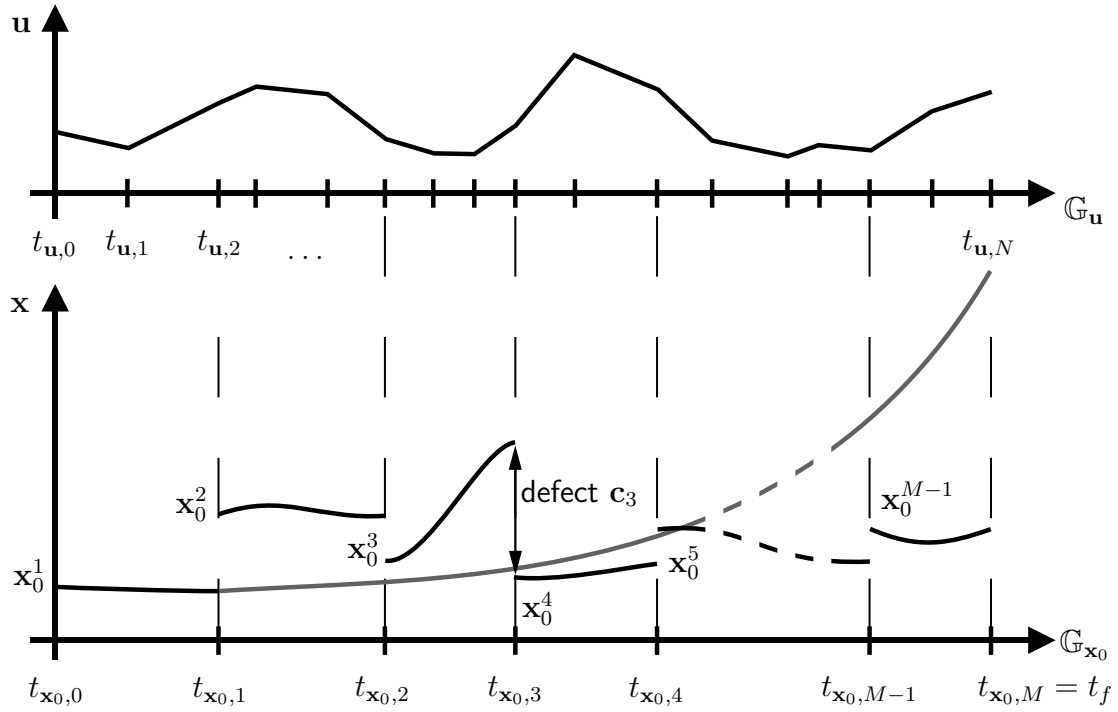

Figure 3.7: *Single and multiple shooting principles.* ***Top graph:*** *Discretized control history over control grid.* ***Bottom graph:*** Gray Line: *Single shooting state history.* Black segments: *Multiple shooting segments before finding a feasible solution. The bottom axis shows the multiple shooting grid, the top one the control grid.*

simulated on each segment separately. The sensitivities with respect to the initial states need to be extended as every segment has its own initial state vector:

$$\frac{d\dot{\mathbf{x}}(t)}{d\mathbf{x}_0^m} = \frac{d}{dt}\left(\frac{d\mathbf{x}(t)}{d\mathbf{x}_0^m}\right) = \frac{\partial \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p})}{\partial \mathbf{x}(t)} \cdot \frac{d\mathbf{x}(t)}{d\mathbf{x}_0^m} \tag{3.171}$$

As no correlation between the different segments exists in the simulation, the initial state sensitivities only need to be calculated within their own respective segment:

$$\frac{d\dot{\mathbf{x}}^m(t)}{d\mathbf{x}_0^k} = \frac{d\mathbf{x}^m(t)}{d\mathbf{x}_0^k} = 0 \qquad \forall k \neq m \tag{3.172}$$

for $\mathbf{x}^m(t) = \mathbf{x}(t)$ for $t \in [t_{\mathbf{x}_0,m}, t_{\mathbf{x}_0,m+1}]$. Consequently, the overall state *Jacobian* can be calculated as:

$$\begin{aligned}\frac{d\mathbf{x}(t)}{d\mathbf{z}} &= \sum_m \frac{d\mathbf{x}(t)}{d\mathbf{x}_0^m} \cdot \frac{d\mathbf{x}_0^m}{d\mathbf{z}} + \sum_k \frac{d\mathbf{x}(t)}{d\mathbf{u}_k} \cdot \frac{d\mathbf{u}_k}{d\mathbf{z}} + \frac{d\mathbf{x}(t)}{d\mathbf{p}} \cdot \frac{d\mathbf{p}}{d\mathbf{z}} \\ &= \sum_m \mathbf{S}_{\mathbf{x}\mathbf{x}_0^m} \cdot \frac{d\mathbf{x}_0^m}{d\mathbf{z}} + \sum_k \mathbf{S}_{\mathbf{x}\mathbf{u}_k} \cdot \frac{d\mathbf{u}_k}{d\mathbf{z}} + \mathbf{S}_{\mathbf{x}\mathbf{p}} \cdot \frac{d\mathbf{p}}{d\mathbf{z}}\end{aligned} \tag{3.173}$$

More details on shooting methods can besides others be found in [Bet09, Ben10, Ger09, Ger12, Fis11, Hol15].

Full Discretization

When implementing the multiple shooting approach very consistently on a fixed integration grid, one could reset the state after every integration step. The resulting optimization parameter vector would then contain a state value at every time step. This is only one way of motivating the full discretization approach, sometimes also called collocation approach (even though in a strict sense collocation denotes a subset of full discretization methods).

We define the (not necessarily equidistant) collocation or state grid as

$$\mathbb{G}_{\mathbf{x}} = \{t_{\mathbf{x},0} < t_{\mathbf{x},1} < \cdots < t_{\mathbf{x},K-1} < t_{\mathbf{x},K} = t_f\} \tag{3.174}$$

with $K+1$ being an arbitrary number of discretization points. Again, the control grid does not have to match the integration grid, even though it is often chosen to do so. Anyway, it is generally recommended to choose $\mathbb{G}_{\mathbf{u}} \subseteq \mathbb{G}_{\mathbf{x}}$ as otherwise changes in the control values may not be resolved accurately by the integration scheme. The control parametrization is again done based on the ideas presented above.

When now collecting all relevant values, the optimization parameter vector

$$\mathbf{z} = \begin{pmatrix} \mathbf{x}_0 & \mathbf{x}_1 & \dots & \mathbf{x}_K & \mathbf{u}_0 & \dots & \mathbf{u}_N & \mathbf{p} \end{pmatrix}^\mathsf{T} \tag{3.175}$$

results. The constraints can be built up similarly as before, now containing one integration defect

$$\mathbf{c}_k(\mathbf{z}) = \mathbf{x}_k(\mathbf{z}) - \mathbf{x}_{k+1}(\mathbf{z}) + h_k \cdot \Phi(\mathbf{x}_k(\mathbf{z}), \mathbf{x}_{k+1}(\mathbf{z}), \mathbf{u}_k(\mathbf{z}), \mathbf{u}_{k+1}(\mathbf{z}), \mathbf{p}(\mathbf{z})) \overset{!}{=} 0 \tag{3.176}$$

for every point on the collocation grid. Herein, $\mathbf{u}_k$ and $\mathbf{u}_{k+1}$ denote the interpolated control values at the respective point on the collocation grid. Again, the formulation does not allow higher order control discretizations as they might be used in higher order collocation schemes. Thus, equation (3.176) is a simplification of a general collocation scheme. Φ designates an arbitrary integration scheme as defined in equation (3.120). Collecting all constraints in one vector,

$$\mathbf{C}(\mathbf{z}) = \begin{pmatrix} \boldsymbol{\Psi}_0(\mathbf{x}_0(\mathbf{z}), \mathbf{p}(\mathbf{z}), t_0(\mathbf{z})) = 0 \\ \boldsymbol{\Psi}_f(\mathbf{x}_f(\mathbf{z}), \mathbf{p}(\mathbf{z}), t_f(\mathbf{z})) = 0 \\ \mathbf{r}(\mathbf{x}_i(\mathbf{z}), \mathbf{p}(\mathbf{z}), t_i(\mathbf{z})) = 0 \\ \mathbf{c}_k(\mathbf{x}(\mathbf{z}), \mathbf{u}(\mathbf{z}), \mathbf{p}(\mathbf{z}), t(\mathbf{z})) = 0 \\ \bar{\mathbf{C}}_{eq}(\mathbf{x}(\mathbf{z}), \mathbf{u}(\mathbf{z}), \mathbf{p}(\mathbf{z}), t(\mathbf{z})) = 0 \\ \bar{\mathbf{C}}_{ineq}(\mathbf{x}(\mathbf{z}), \mathbf{u}(\mathbf{z}), \mathbf{p}(\mathbf{z}), t(\mathbf{z})) \leq 0 \end{pmatrix} \tag{3.177}$$

results, with

$$\begin{aligned} &\mathbf{c}_k(\mathbf{x}(\mathbf{z}), \mathbf{u}(\mathbf{z}), \mathbf{p}(\mathbf{z}), t(\mathbf{z})) \\ &\quad = \mathbf{x}_k(\mathbf{z}) - \mathbf{x}_{k+1}(\mathbf{z}) + h_k \cdot \Phi(\mathbf{x}_k(\mathbf{z}), \mathbf{x}_{k+1}(\mathbf{z}), \mathbf{u}_k(\mathbf{z}), \mathbf{u}_{k+1}(\mathbf{z}), \mathbf{p}(\mathbf{z})) = 0 \\ &\qquad \forall k = 0, \dots, K-1 \end{aligned} \tag{3.178}$$

and $\bar{\mathbf{C}}_{eq}$ and $\bar{\mathbf{C}}_{ineq}$ again being discretized representations of $\mathbf{C}_{eq}$ and $\mathbf{C}_{ineq}$.

Besides, it is also possible to formulate the defect in the time derivative of $\mathbf{x}$. For the simplified formulation presented in equation (3.176) a simple division by h results:

$$\bar{\mathbf{c}}_k(\mathbf{z}) = \frac{\mathbf{x}_k(\mathbf{z}) - \mathbf{x}_{k+1}(\mathbf{z})}{h_k} + \Phi(\mathbf{x}_k(\mathbf{z}), \mathbf{x}_{k+1}(\mathbf{z}), \mathbf{u}_k(\mathbf{z}), \mathbf{u}_{k+1}(\mathbf{z}), \mathbf{p}(\mathbf{z})) \stackrel{!}{=} 0 \tag{3.179}$$

When using higher order collocation schemes, the transformation becomes more complicated and more non-linear. In these cases, the interpretation of the LAGRANGE multipliers associated with the defects also changes. More details on the interpretation and higher order collocation schemes can be found, besides others, in [Bet09, EC92].

As all state values are now part of the optimization vector $\mathbf{z}$, the use of an implicit integration method becomes feasible very easily and without the need to solve an additional non-linear equation in every integration step. The solution of this integration is done by the numerical optimization algorithm in parallel to the solution of the optimization problem, automatically. This may in some cases enable the use of a coarser grid as *stiff dynamics* can in some cases be well integrated using an implicit integration scheme.

The main drawback of the collocation method (full discretization) is the fact that very large numerical optimization problems may result as all state and all control values become part of $\mathbf{z}$. In case a system with many states is considered that requires a simulation on a fine grid with thousands of discretization points, an optimization problem with several ten thousands of variables and constraints can result. The good thing is, that the problem structure is highly sparse in this case as the defects now only depend on some control values and normally only two to three state values. Consequently, when using a full discretization scheme, it is of utmost importance to use a numerical optimization algorithm specially tailored to sparse optimization problems, like those listed in 3.1.5. Chapter 7 gives more details on the exploitation of the sparsity inherent to an optimal control problem when transforming it to a numerical optimization problem.

When using the EULER forward method for performing the numerical integration, the defect equation becomes

$$\mathbf{x}_k(\mathbf{z}) - \mathbf{x}_{k+1}(\mathbf{z}) + h_k \cdot \mathbf{f}(\mathbf{x}_k(\mathbf{z}), \mathbf{u}_k(\mathbf{z}), \mathbf{p}(\mathbf{z})) \stackrel{!}{=} 0, \tag{3.180}$$

while for the implicit EULER backward scheme it is:

$$\mathbf{x}_k(\mathbf{z}) - \mathbf{x}_{k+1}(\mathbf{z}) + h_k \cdot \mathbf{f}(\mathbf{x}_{k+1}(\mathbf{z}), \mathbf{u}_{k+1}(\mathbf{z}), \mathbf{p}(\mathbf{z})) \stackrel{!}{=} 0 \tag{3.181}$$

The implicit trapezoidal rule, which is widely used with collocation schemes, results in the following defect equation:

$$\mathbf{x}_k(\mathbf{z}) - \mathbf{x}_{k+1}(\mathbf{z}) + \frac{h_k}{2} \cdot (\mathbf{f}(\mathbf{x}_k(\mathbf{z}), \mathbf{u}_k(\mathbf{z}), \mathbf{p}(\mathbf{z})) + \mathbf{f}(\mathbf{x}_{k+1}(\mathbf{z}), \mathbf{u}_{k+1}(\mathbf{z}), \mathbf{p}(\mathbf{z}))) \stackrel{!}{=} 0 \tag{3.182}$$

In [HC96], HERMAN and CONWAY present higher order GAUSS-LOBATTO quadrature rules based on a polynomial approximation of the integral. The most accurate scheme they are listing in the work is the fifth-degree GAUSS-LOBATTO rule, which has an order of accuracy

of 8. Therein, every integration step is based on the left sided $\mathbf{x}_k$, the right sided $\mathbf{x}_{k+1}$ and the central state vector $\mathbf{x}_{k+\frac{1}{2}}$ (that all need to be added to the optimization parameter vector) and produces two defects that are calculated from

$$\begin{aligned}\mathbf{c}_{k,1} = \frac{1}{360}\Big((32\sqrt{21}+180)\mathbf{x}_k - 64\sqrt{21}\mathbf{x}_{k+\frac{1}{2}} + (32\sqrt{21}-180)\mathbf{x}_{k+1} + \\ h_k\Big((9+\sqrt{21}\mathbf{f}_k + 98\mathbf{f}_{k,1} + 64\mathbf{f}_{k+\frac{1}{2}} + (9-\sqrt{21})\mathbf{f}_{k+1}\Big)\Big) \stackrel{!}{=} 0 \end{aligned} \tag{3.183}$$

$$\begin{aligned}\mathbf{c}_{k,2} = \frac{1}{360}\Big((-32\sqrt{21}+180)\mathbf{x}_k + 64\sqrt{21}\mathbf{x}_{k+\frac{1}{2}} + (-32\sqrt{21}-180)\mathbf{x}_{k+1} + \\ h_k\Big((9-\sqrt{21}\mathbf{f}_k + 98\mathbf{f}_{k,3} + 64\mathbf{f}_{k+\frac{1}{2}} + (9+\sqrt{21})\mathbf{f}_{k+1}\Big)\Big) \stackrel{!}{=} 0 \end{aligned} \tag{3.184}$$

with the abbreviations

$$\mathbf{f}_k = \mathbf{f}(\mathbf{x}_k(\mathbf{z}), \mathbf{u}_k(\mathbf{z}), \mathbf{p}(\mathbf{z})) \tag{3.185}$$

$$\mathbf{f}_{k+\frac{1}{2}} = \mathbf{f}(\mathbf{x}_{k+\frac{1}{2}}(\mathbf{z}), \mathbf{u}_{k+\frac{1}{2}}(\mathbf{z}), \mathbf{p}(\mathbf{z})) \tag{3.186}$$

$$\mathbf{f}_{k+1} = \mathbf{f}(\mathbf{x}_{k+1}(\mathbf{z}), \mathbf{u}_{k+1}(\mathbf{z}), \mathbf{p}(\mathbf{z})) \tag{3.187}$$

$$\mathbf{f}_{k,1} = \mathbf{f}(\mathbf{x}_{k,1}(\mathbf{z}), \mathbf{u}_{k,1}(\mathbf{z}), \mathbf{p}(\mathbf{z})) \tag{3.188}$$

$$\mathbf{f}_{k,3} = \mathbf{f}(\mathbf{x}_{k,3}(\mathbf{z}), \mathbf{u}_{k,3}(\mathbf{z}), \mathbf{p}(\mathbf{z})) \tag{3.189}$$

$$\begin{aligned}\mathbf{x}_{k,1} = \frac{1}{686}\Big((39\sqrt{21}+231)\mathbf{x}_k + 224\mathbf{x}_{k+\frac{1}{2}} + (-39\sqrt{21}+231)\mathbf{x}_{k+1} + \\ h_k\Big((3\sqrt{21}+21)\mathbf{f}_k - 16\sqrt{21}\mathbf{f}_{k+\frac{1}{2}} + (3\sqrt{21}-21)\mathbf{f}_{k+1}\Big)\Big) \end{aligned} \tag{3.190}$$

$$\begin{aligned}\mathbf{x}_{k,3} = \frac{1}{686}\Big((-39\sqrt{21}+231)\mathbf{x}_k + 224\mathbf{x}_{k+\frac{1}{2}} + (39\sqrt{21}+231)\mathbf{x}_{k+1} + \\ h_k\Big((-3\sqrt{21}+21)\mathbf{f}_k + 16\sqrt{21}\mathbf{f}_{k+\frac{1}{2}} + (-3\sqrt{21}-21)\mathbf{f}_{k+1}\Big)\Big) \end{aligned} \tag{3.191}$$

and $\mathbf{u}_{k,1} = \mathbf{u}(t_{k,1})$ and $\mathbf{u}_{k,3} = \mathbf{u}(t_{k,3})$ being appropriate control interpolations at:

$$t_{k,1} = t_{k+\frac{1}{2}} - \sqrt{\frac{3}{7}}\cdot\frac{h_k}{2} \qquad \text{and} \qquad t_{k,3} = t_{k+\frac{1}{2}} + \sqrt{\frac{3}{7}}\cdot\frac{h_k}{2} \tag{3.192}$$

It can clearly be seen that the defect constraint given in equation (3.176) can only represent a constant or linear interpolation in the controls for calculating the intermediate values $\mathbf{u}_{k,1}$ and $\mathbf{u}_{k,2}$. In general, higher order approximations may also be used. All other constraints in equation (3.177) are not affected by the choice of the collocation scheme.

More details on full discretization methods can besides others be found in [Bet09, Ben10, Ger09, Ger12, Hol15, EC92]

Scaling

To be able to efficiently solve the numerical optimization problem resulting from discretizing the optimal control problem, the numerical scaling of the problem is of utmost importance [Bet09, Fis11]. The solution of the optimization problem can be handled especially easily by the NLP solver, in case all numerical values remain within the same order of magnitude, where a magnitude around 1 is favorable as digital computers normally have the best computational

accuracy in this domain [Gol91]. The purpose of numerical scaling is to bring all values into this domain.

The parameter vector $\mathbf{z}$ can be scaled using a diagonal scaling matrix $\mathbf{T_z}$. The scaled optimization parameter vector $\tilde{\mathbf{z}}$ results from:

$$\tilde{\mathbf{z}} = \mathbf{T_z} \cdot \mathbf{z} \tag{3.193}$$

The scaling of the cost J is similarly done by using a scalar T_J:

$$\tilde{J} = T_J \cdot J \tag{3.194}$$

The constraint vector $\mathbf{C}$ is scaled using another diagonal matrix $\mathbf{T_C}$:

$$\tilde{\mathbf{C}} = \mathbf{T_C} \cdot \mathbf{C} \tag{3.195}$$

The gradient and the *Jacobian* of the discretized problem consequently have to be scaled using:

$$\left(\frac{d\tilde{J}}{d\mathbf{z}}\right) = \frac{d\tilde{J}}{d\tilde{\mathbf{z}}} = T_J \cdot \frac{dJ}{d\mathbf{z}} \cdot \mathbf{T_z}^{-1} \tag{3.196}$$

$$\left(\frac{d\tilde{\mathbf{C}}}{d\mathbf{z}}\right) = \frac{d\tilde{\mathbf{C}}}{d\tilde{\mathbf{z}}} = T_\mathbf{C} \cdot \frac{d\mathbf{C}}{d\mathbf{z}} \cdot \mathbf{T_z}^{-1} \tag{3.197}$$

When changing the scaling of the parameters and the constraints, the LAGRANGE multipliers are also affected. The scaled LAGRANGE function can be written as

$$\tilde{L}(\tilde{\mathbf{z}}, \tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\nu}}) = T_J \cdot J(\tilde{\mathbf{z}}) + \tilde{\boldsymbol{\lambda}}^\mathsf{T} \cdot \mathbf{T_h} \cdot \mathbf{h}(\tilde{\mathbf{z}}) + \tilde{\boldsymbol{\nu}}^\mathsf{T} \cdot \mathbf{T_g} \cdot \mathbf{g}(\tilde{\mathbf{z}}) \tag{3.198}$$

with $\mathbf{T_h}$ and $\mathbf{T_g}$ being the respective parts of $\mathbf{T_C}$ for the equality and inequality constraints in the problem. By comparing $\tilde{L}$ with L

$$\tilde{L} = T_J \cdot L \tag{3.199}$$

the scaling of the multipliers results as:

$$\boldsymbol{\lambda} = \mathbf{T_h} \cdot \tilde{\boldsymbol{\lambda}} \cdot T_J^{-1} \tag{3.200}$$

$$\boldsymbol{\nu} = \mathbf{T_g} \cdot \tilde{\boldsymbol{\nu}} \cdot T_J^{-1} \tag{3.201}$$

It is generally recommended to take the LAGRANGE multipliers into account when considering numerical scaling. However, this cannot always be done easily as the values for the multipliers are normally not known before having found a solution to the problem. When using the methods described below to determine the values of the costates of the original optimal control problem, their order of magnitude may also be considered when scaling the problem.

In most numerical optimization software, the user can specify only one tolerance for all constraints in the problem. If the exact fulfillment of some of the constraints is more important than the fulfillment of some other constraints, numerical scaling may also be used to achieve a weighting. If the user, e.g. selects a numerical tolerance of ε and scales one constraint value by a value in the order of magnitude of ε, then the real unscaled value only achieves an absolute tolerance within the order of 1 for this particular constraint.

An Engineers View on Direct Methods

BETTS and BÜSKENS identify several advantages of direct discretization methods in [Bet09, sec. 4.14], [Büs98, sec. 5.2.4]:

1. The user does not necessarily need to have any knowledge of optimal control theory.
2. Analytical derivatives are not required but can be replaced by numerical finite differences.
3. The constrained-arc sequence does not need to be determined a priori as the NLP solver can determine it.
4. No initial guess for the costates is required. On the other hand, an initial guess for the LAGRANGE multipliers is necessary. Anyway, the method is more robust at this point.
5. The radius of convergence is higher in the direct approach than in the indirect approach.
6. Using the theory of parametric optimization, an extension to parametric optimal control becomes possible.

Finally, from an engineering point of view, if one wants to solve real life optimal control problems containing high fidelity dynamic models, highly non-linear equality and inequality constraints and several phases, the only way to tackle such problems is via a direct discretization scheme. The reasons for this are manifold, as mostly the analytical optimality conditions required for an indirect approach may not be computed for complex systems, the switching structure of the inequality constraints is normally not known in advance, and no closed formulation for multi-phase problems exists in indirect methods. The direct approach can be seen as the "engineering" approach for solving optimal control problems. Anyway, special care has to be taken concerning the convergence to the real solution of the problem in this case. In some cases it is possible that the solution of the discretized problem does not converge to the solution of the original problem. In order to avoid this, the optimality conditions for optimal control problems stated in section 3.2.3 should be checked using the costate estimates calculated from the equations presented in section 3.2.5, whenever possible.

3.2.5 Relations Between the Direct and the Indirect Approach

In this section, the discretization of the optimal control problem using direct methods and the optimality conditions derived using the indirect approach will be put in relation. In general, the relations can be established for the shooting and the collocation approach. Anyway, for the sake of simplicity, here only collocation will be regarded. Moreover, the standard optimal control problem from section 3.2.2 without parameter dependencies is considered. More details on the relations between the direct and the indirect approach can, besides others, be found in [Bet09, sec. 4.11] , [Ben10, p. 170] , [Ger12, sec. 5.4], [Büs98, ch. 7].

In order to be able to distinguish between the values appearing in the continuous optimal control problem (in the *Hamiltonian*) and those of the LAGRANGE function of the discretized problem that are represented by the same symbol, the latter are marked with a $\tilde{\circ}$.

Discrete Minimum Principle

In order to clarify the relationship between the extended *Hamiltonian* $\tilde{H}$ (see equation (3.113)) of the continuous optimal control problem and the *Lagrangian* L of the numerical optimization problem resulting from the discretization (see equation (3.22)), GERDTS derives a discretized version of the minimum principle for a constrained problem in [Ger12, sec. 5.4]. His results are now applied to the problem stated in section 3.2.3, using an implicit EULER backward collocation method with the defect from equation (3.181). Moreover, the following assumptions have to hold:

- The MANGASARIAN-FROMOWITZ constraint qualification (or another appropriate constraint qualification) is fulfilled.
- All functions appearing in the problem definition in (3.91)–(3.93) and (3.112) are continuously differentiable in $\mathbf{x}$ and $\mathbf{u}$. The same holds for the MAYER cost function

$$J = e(\mathbf{x}(t_0), \mathbf{x}(t_f)) = e(\mathbf{x}_0, \mathbf{x}_f) \tag{3.202}$$

  which is used as the only cost function in this derivation (without loss of generality, see section 3.2.2).
- $\mathbf{S}(\mathbf{x}, \mathbf{u})$ is a mixed state and control constraint that is not degraded to a pure state or a pure control constraint.
- $(\mathbf{x}, \mathbf{u})$ is a local minimum of the problem.

Then, there exist multipliers $\boldsymbol{\lambda}_k, \boldsymbol{\mu}_k, \boldsymbol{\sigma}_0$ and $\boldsymbol{\sigma}_f$, being discretized representations of the multipliers in the *Hamiltonian* of the optimal control problem, such that

1. The **discrete optimality conditions** are fulfilled for all $k = 1, \ldots, N$ and all admissible controls $\mathbf{u}$ (see equation (3.115) for the continuous representation)

$$\tilde{H}'_{\mathbf{u}}(\mathbf{x}_k, \boldsymbol{\lambda}_{k-1}, \boldsymbol{\mu}_k, \mathbf{u}_k) = 0 \tag{3.203}$$

2. The **discrete adjoint equation** is fulfilled for all $k = 1, \ldots, N$ (see equation (3.116) for the continuous representation)

$$\boldsymbol{\lambda}_{k-1} = \boldsymbol{\lambda}_k + h_k \cdot \nabla_{\mathbf{x}} \tilde{H}(\mathbf{x}_k, \boldsymbol{\lambda}_{k-1}, \boldsymbol{\mu}_k, \mathbf{u}_k) \tag{3.204}$$

3. The **discrete transversality conditions** are fulfilled (see also equation (3.117))

$$\begin{aligned} \boldsymbol{\lambda}_0 &= -\ (\nabla_{\mathbf{x}_0} e + \nabla_{\mathbf{x}_0} \boldsymbol{\Psi}_0 \cdot \boldsymbol{\sigma}_0) \\ \boldsymbol{\lambda}_N &= \quad\ (\nabla_{\mathbf{x}_f} e + \nabla_{\mathbf{x}_f} \boldsymbol{\Psi}_f \cdot \boldsymbol{\sigma}_f) \end{aligned} \tag{3.205}$$

4. The **discrete complementary conditions** hold for all $k = 1, \ldots, N$ (see also equation (3.118)):

$$\boldsymbol{\mu}_k \geq 0 \tag{3.206}$$

$$\boldsymbol{\mu}_k^{\mathsf{T}} \mathbf{S}(\mathbf{x}_k, \mathbf{u}_k) = 0 \tag{3.207}$$

It is important to mention that the index shift appearing in $\boldsymbol{\lambda}$ is required in order to get the same implicit integration scheme for the system dynamics and the adjoint equations, as the latter need to be solved backwards in time. The discrete representation of the continuous optimality conditions for optimal control problems can be derived from the LAGRANGE function of the discretized problem. In the next section, this derivation will be used for the costate estimation.

Further details on discretized optimality conditions for optimal control problems can besides others be found in [Ger12, sec. 5.4], [Ben10, ch. 6], [Bet09, sec. 4.2], [Str93, Str94].

Costate Estimation for Collocation Methods

In this section, the relations between the costates appearing in the *Hamiltonian* and the LAGRANGE multipliers of the discretized problem will be further examined starting with the derivations for a EULER backward scheme as presented in [Ger12, p. 254], before moving over to a more general full discretization scheme. Besides others, BETTS, BEN-ASHER, and ENRIGHT and CONWAY present equations for costate estimation for a HERMITE-SIMPSON scheme in [Bet09, p. 177], [Ben10, p. 170] and [EC92].

For the EULER backward collocation scheme, GERDTS shows how the discretized optimality conditions can be derived for the discretized problem using its LAGRANGE function. For the formulation used in this work, the *Lagrangian* becomes:

$$\begin{aligned} L(\mathbf{x}, \mathbf{u}, \tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\sigma}}_0, \tilde{\boldsymbol{\sigma}}_f, \boldsymbol{\nu}) = e(\mathbf{x}_0, \mathbf{x}_f) + \tilde{\boldsymbol{\sigma}}_0^\mathsf{T} \boldsymbol{\Psi}_0 + \tilde{\boldsymbol{\sigma}}_f^\mathsf{T} \boldsymbol{\Psi}_f + \sum_{k=1}^{N} h_{k-1} \tilde{\boldsymbol{\lambda}}_{k-1}^\mathsf{T} \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \\ + \sum_{k=1}^{N} \tilde{\boldsymbol{\lambda}}_{k-1}^\mathsf{T} (\mathbf{x}_{k-1} - \mathbf{x}_k) + \sum_{k=1}^{N} \boldsymbol{\nu}_k^\mathsf{T} \mathbf{S}(\mathbf{x}_k, \mathbf{u}_k) \end{aligned} \tag{3.208}$$

with the original LAGRANGE multipliers $\tilde{\boldsymbol{\lambda}}$ being split up to the different multipliers for the equality constraints. Consequently, the derivatives with respect to the states $\mathbf{x}$ and the controls $\mathbf{u}$ evaluate to:

$$\begin{aligned} L'_{\mathbf{u}_k} &= h_{k-1} \tilde{\boldsymbol{\lambda}}_{k-1}^\mathsf{T} \mathbf{f}'_{\mathbf{u}_k}(\mathbf{x}_k, \mathbf{u}_k) + h_{k-1} \cdot \frac{\boldsymbol{\nu}_k^\mathsf{T}}{h_{k-1}} \cdot \mathbf{S}'_{\mathbf{u}_k}(\mathbf{x}_k, \mathbf{u}_k) \\ &= h_{k-1} \cdot \tilde{H}'_{\mathbf{u}_k} \left(\mathbf{x}_k, \tilde{\boldsymbol{\lambda}}_{k-1}, \frac{\boldsymbol{\nu}_k}{h_{k-1}}, \mathbf{u}_k \right) \end{aligned} \tag{3.209}$$

$$\begin{aligned} L'_{\mathbf{x}_k} = e'_{\mathbf{x}_k} + \tilde{\boldsymbol{\sigma}}_0^\mathsf{T} (\boldsymbol{\Psi}_0)'_{\mathbf{x}_k} + \tilde{\boldsymbol{\sigma}}_f^\mathsf{T} (\boldsymbol{\Psi}_f)'_{\mathbf{x}_k} + h_{k-1} \tilde{\boldsymbol{\lambda}}_{k-1}^\mathsf{T} \mathbf{f}'_{\mathbf{x}_k}(\mathbf{x}_k, \mathbf{u}_k) \\ - \tilde{\boldsymbol{\lambda}}_{k-1}^\mathsf{T} + \tilde{\boldsymbol{\lambda}}_k^\mathsf{T} + \boldsymbol{\nu}_k^\mathsf{T} \mathbf{S}'_{\mathbf{x}_k}(\mathbf{x}_k, \mathbf{u}_k) \end{aligned} \tag{3.210}$$

Using the optimality conditions (KKT conditions) from equation (3.24), for $k = 1, \ldots, N-1$, it follows that:

$$\begin{aligned} L'_{\mathbf{x}_k} &= h_{k-1} \tilde{\boldsymbol{\lambda}}_{k-1}^\mathsf{T} \mathbf{f}'_{\mathbf{x}_k}(\mathbf{x}_k, \mathbf{u}_k) - \tilde{\boldsymbol{\lambda}}_{k-1}^\mathsf{T} + \tilde{\boldsymbol{\lambda}}_k^\mathsf{T} + \boldsymbol{\nu}_k^\mathsf{T} \mathbf{S}'_{\mathbf{x}_k}(\mathbf{x}_k, \mathbf{u}_k) \\ &= h_{k-1} \cdot \tilde{H}'_{\mathbf{x}_k} \left(\mathbf{x}_k, \tilde{\boldsymbol{\lambda}}_{k-1}, \frac{\boldsymbol{\nu}_k}{h_{k-1}}, \mathbf{u}_k \right) - \tilde{\boldsymbol{\lambda}}_{k-1}^\mathsf{T} + \tilde{\boldsymbol{\lambda}}_k^\mathsf{T} = \mathbf{0} \end{aligned} \tag{3.211}$$

For the special case of $k = 0$

$$L'_{\mathbf{x}_0} = e'_{\mathbf{x}_0} + \tilde{\boldsymbol{\sigma}}_0^\mathsf{T} (\boldsymbol{\Psi}_0)'_{\mathbf{x}_0} + \tilde{\boldsymbol{\lambda}}_0^\mathsf{T} = \mathbf{0} \tag{3.212}$$

results, while for $k = N$ and $\mathbf{x}_N = \mathbf{x}_f$, one gets:

$$\begin{aligned} L'_{\mathbf{x}_N} &= e'_{\mathbf{x}_N} + \tilde{\boldsymbol{\sigma}}_f^\intercal (\boldsymbol{\Psi}_f)'_{\mathbf{x}_N} + h_{N-1} \tilde{\boldsymbol{\lambda}}_{N-1}^\intercal \mathbf{f}'_{\mathbf{x}_N}(\mathbf{x}_N, \mathbf{u}_N) - \tilde{\boldsymbol{\lambda}}_{N-1}^\intercal + \boldsymbol{\nu}_N^\intercal \mathbf{S}'_{\mathbf{x}_N}(\mathbf{x}_N, \mathbf{u}_N) \\ &= e'_{\mathbf{x}_N} + \tilde{\boldsymbol{\sigma}}_f^\intercal (\boldsymbol{\Psi}_f)'_{\mathbf{x}_N} + h_{N-1} \cdot \tilde{H}'_{\mathbf{x}_N} \left(\mathbf{x}_N, \tilde{\boldsymbol{\lambda}}_{N-1}, \frac{\boldsymbol{\nu}_N}{h_{N-1}}, \mathbf{u}_N \right) - \tilde{\boldsymbol{\lambda}}_{N-1}^\intercal = \mathbf{0} \end{aligned} \tag{3.213}$$

Now, $\tilde{\boldsymbol{\lambda}}_N$ may be defined as:

$$\tilde{\boldsymbol{\lambda}}_N = e'_{\mathbf{x}_N} + \tilde{\boldsymbol{\sigma}}_f^\intercal (\boldsymbol{\Psi}_f)'_{\mathbf{x}_N} \tag{3.214}$$

resulting in:

$$L'_{\mathbf{x}_N} = h_{N-1} \cdot \tilde{H}'_{\mathbf{x}_N} \left(\mathbf{x}_N, \tilde{\boldsymbol{\lambda}}_{N-1}, \frac{\boldsymbol{\nu}_N}{h_{N-1}}, \mathbf{u}_N \right) - \tilde{\boldsymbol{\lambda}}_{N-1}^\intercal + \tilde{\boldsymbol{\lambda}}_N^\intercal = \mathbf{0} \tag{3.215}$$

Focusing on the intermediate steps, by rearranging equation (3.211) to

$$\frac{\tilde{\boldsymbol{\lambda}}_k - \tilde{\boldsymbol{\lambda}}_{k-1}}{h_{k-1}} = -\tilde{H}'_{\mathbf{x}_k} \left(\mathbf{x}_k, \tilde{\boldsymbol{\lambda}}_{k-1}, \frac{\boldsymbol{\nu}_k}{h_{k-1}}, \mathbf{u}_k \right) \tag{3.216}$$

it can be seen that for the EULER backward integration scheme the LAGRANGE multipliers $\tilde{\boldsymbol{\lambda}}_{k-1}$ and $\boldsymbol{\nu}_k$ are connected to the discretized multipliers of the *Hamiltonian* $\boldsymbol{\lambda}_{k-1}$ and $\boldsymbol{\mu}_k$ by:

$$\tilde{\boldsymbol{\lambda}}_{k-1} = \boldsymbol{\lambda}_{k-1} \approx \boldsymbol{\lambda}(t_{k-1}) \tag{3.217}$$

$$\boldsymbol{\mu}_k = \frac{\boldsymbol{\nu}_k}{h_{k-1}} \tag{3.218}$$

Similar derivations can also be done based on equations (3.205) resulting in connections for the remaining multipliers:

$$\begin{aligned} \tilde{\boldsymbol{\sigma}}_0 &= \boldsymbol{\sigma}_0 \\ \tilde{\boldsymbol{\sigma}}_N &= \boldsymbol{\sigma}_N \end{aligned} \tag{3.219}$$

Now, the more general discretization scheme as stated in equations (3.176) and (3.177) will be used, resulting in the following LAGRANGE function for the discretized optimal control problem:

$$L(\mathbf{x}, \mathbf{u}, \tilde{\boldsymbol{\lambda}}, \boldsymbol{\nu}) = e(\mathbf{x}_0, \mathbf{x}_f) + \tilde{\boldsymbol{\sigma}}_0^\intercal \boldsymbol{\Psi}_0 + \tilde{\boldsymbol{\sigma}}_f^\intercal \boldsymbol{\Psi}_f + \sum_{k=0}^{N-1} \tilde{\boldsymbol{\lambda}}_k^\intercal \mathbf{c}_k + \tilde{\boldsymbol{\lambda}}_{\bar{\mathbf{C}}_{eq}}^\intercal \bar{\mathbf{C}}_{eq} + \boldsymbol{\nu}^\intercal \bar{\mathbf{C}}_{ineq} \tag{3.220}$$

In this formulation the global vector of LAGRANGE multipliers $\tilde{\boldsymbol{\lambda}}$ has been split up into the separate multipliers $\tilde{\boldsymbol{\sigma}}_0$, $\tilde{\boldsymbol{\sigma}}_f$, $\tilde{\boldsymbol{\lambda}}_k$ and $\tilde{\boldsymbol{\lambda}}_{\bar{\mathbf{C}}_{eq}}$ for each constraint appearing in equation (3.177).

As the initial and final boundary conditions as well as the cost function formulation do not change when selecting another discretization scheme, the relations from (3.219) are still valid. Anyway, when considering $L'_{\mathbf{x}_k}$ for $k = 1, \ldots, N-1$, equation (3.211) changes to:

$$L'_{\mathbf{x}_k} = \sum_{i=0}^{N-1} \tilde{\boldsymbol{\lambda}}_i^\intercal \frac{d\mathbf{c}_i}{d\mathbf{x}_k} + \tilde{\boldsymbol{\lambda}}_{\bar{\mathbf{C}}_{eq}}^\intercal \frac{d\bar{\mathbf{C}}_{eq}}{d\mathbf{x}_k} + \boldsymbol{\nu}^\intercal \frac{d\bar{\mathbf{C}}_{ineq}}{d\mathbf{x}_k} \tag{3.221}$$

Inserting the general defect equation of the collocation scheme (3.176) into equation (3.221) and simplifying the sum, the following equation results:

$$\begin{aligned} L'_{\mathbf{x}_k} &= \tilde{\boldsymbol{\lambda}}_k^\intercal \cdot \frac{d\left(\mathbf{x}_k - \mathbf{x}_{k+1} + h_k \cdot \Phi(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_{k+1}, \mathbf{u}_{k+1})\right)}{d\mathbf{x}_k} \\ &\quad + \tilde{\boldsymbol{\lambda}}_{k-1}^\intercal \cdot \frac{d\left(\mathbf{x}_{k-1} - \mathbf{x}_k + h_{k-1} \cdot \Phi(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{x}_k, \mathbf{u}_k)\right)}{d\mathbf{x}_k} \\ &\quad + \tilde{\boldsymbol{\lambda}}_{\bar{\mathbf{C}}_{eq}}^\intercal \cdot \frac{d\bar{\mathbf{C}}_{eq}}{d\mathbf{x}_k} + \boldsymbol{\nu}^\intercal \cdot \frac{d\bar{\mathbf{C}}_{ineq}}{d\mathbf{x}_k} \end{aligned} \tag{3.222}$$

$$\begin{aligned} &= \tilde{\boldsymbol{\lambda}}_k^\intercal \cdot \left(\mathbf{I} + h_k \cdot \frac{d\Phi(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_{k+1}, \mathbf{u}_{k+1})}{d\mathbf{x}_k}\right) \\ &\quad + \tilde{\boldsymbol{\lambda}}_{k-1}^\intercal \cdot \left(-\mathbf{I} + h_{k-1} \cdot \frac{d\Phi(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{x}_k, \mathbf{u}_k)}{d\mathbf{x}_k}\right) \\ &\quad + \tilde{\boldsymbol{\lambda}}_{\bar{\mathbf{C}}_{eq}}^\intercal \cdot \frac{d\bar{\mathbf{C}}_{eq}}{d\mathbf{x}_k} + \boldsymbol{\nu}^\intercal \cdot \frac{d\bar{\mathbf{C}}_{ineq}}{d\mathbf{x}_k} \end{aligned} \tag{3.223}$$

The sum can be collapsed as stated above, as for every $\mathbf{x}_k$ only two defects depend on the value of $\mathbf{x}_k$, resulting in $\frac{d\mathbf{c}_i}{d\mathbf{x}_k} = 0, \forall k \neq i \wedge k \neq (i+1)$.

For simplicity, from now on an equidistant grid is used, where it holds that $h = h_k = h_{k-1}$. Now, equation (3.223) can be rearranged to:

$$\begin{aligned} \frac{\tilde{\boldsymbol{\lambda}}_k^\intercal - \tilde{\boldsymbol{\lambda}}_{k-1}^\intercal}{h} &= -\tilde{\boldsymbol{\lambda}}_k^\intercal \cdot \frac{d\Phi(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_{k+1}, \mathbf{u}_{k+1})}{d\mathbf{x}_k} \\ &\quad - \tilde{\boldsymbol{\lambda}}_{k-1}^\intercal \cdot \frac{d\Phi(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{x}_k, \mathbf{u}_k)}{d\mathbf{x}_k} \\ &\quad - \frac{\tilde{\boldsymbol{\lambda}}_{\bar{\mathbf{C}}_{eq}}^\intercal}{h} \cdot \frac{d\bar{\mathbf{C}}_{eq}}{d\mathbf{x}_k} - \frac{\boldsymbol{\nu}^\intercal}{h} \cdot \frac{d\bar{\mathbf{C}}_{ineq}}{d\mathbf{x}_k} \end{aligned} \tag{3.224}$$

The fraction on the left approximates $\dot{\boldsymbol{\lambda}}$ for $h \to 0$ while the first two summands on the right represent a weighted (by the derivative of the increment function) approximation for $\tilde{\boldsymbol{\lambda}}$ within the considered interval. The last two summands of equation (3.224) are a discretized approximation of the constraints appearing in the discretized *Hamiltonian* (see equations (3.203)–(3.207)). The relation (3.218) is now valid for $\bar{\mathbf{C}}_{eq}$ and the active subset of $\bar{\mathbf{C}}_{ineq}$.

Using the aforementioned general integration scheme, the discrete adjoint equation reads:

$$\boldsymbol{\lambda}_{k-1} = \boldsymbol{\lambda}_k + h_k \cdot \boldsymbol{\Phi}_{\nabla_{\mathbf{x}}\tilde{H}}\left(\mathbf{x}_k, \boldsymbol{\lambda}_{k-1}, \boldsymbol{\mu}_k, \mathbf{u}_k, \mathbf{x}_{k+1}, \boldsymbol{\lambda}_k, \boldsymbol{\mu}_{k+1}, \mathbf{u}_{k+1}\right) \tag{3.225}$$

with $\boldsymbol{\Phi}_{\nabla_{\mathbf{x}}\tilde{H}}$ being the increment function for the backward integration of the adjoint equation for the *Hamiltonian* $\tilde{H}$. Here, the same integration scheme as for the state dynamics has to be used. When assuming $\boldsymbol{\Phi}_{\nabla_{\mathbf{x}}\tilde{H}}$ to be a general RUNGE-KUTTA scheme, the summands of the augmented *Hamiltonian* $\tilde{H}$ can be considered separately. Focusing on the costates, the increment function

$$\boldsymbol{\Phi}_{\left(\boldsymbol{\lambda}\cdot\frac{d\mathbf{f}}{d\mathbf{x}}\right)}\left(\mathbf{x}_k, \boldsymbol{\lambda}_{k-1}, \mathbf{u}_k, \mathbf{x}_{k+1}, \boldsymbol{\lambda}_k, \mathbf{u}_{k+1}\right) \tag{3.226}$$

for the model related summand of $\tilde{H}$ can be formulated.

Now, the same comparison between the discretized *Hamiltonian* H (left side of equation (3.227)) and the *Lagrangian* of the discretized problem (right side of equation (3.227)), as also done in equation (3.216) can be performed again. It results that:

$$\begin{aligned}&\boldsymbol{\Phi}_{\left(\boldsymbol{\lambda}\cdot\frac{d\mathbf{f}}{d\mathbf{x}}\right)}\left(\mathbf{x}_k, \boldsymbol{\lambda}_{k-1}, \mathbf{u}_k, \mathbf{x}_{k+1}, \boldsymbol{\lambda}_k, \mathbf{u}_{k+1}\right)\\ &\qquad\approx \tilde{\boldsymbol{\lambda}}_k^{\intercal}\cdot\frac{d\Phi(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_{k+1}, \mathbf{u}_{k+1})}{d\mathbf{x}_k} + \tilde{\boldsymbol{\lambda}}_{k-1}^{\intercal}\cdot\frac{d\Phi(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{x}_k, \mathbf{u}_k)}{d\mathbf{x}_k}\end{aligned} \tag{3.227}$$

This result is valid for every integration scheme considered in this work. Anyway, no further simplification is possible for this general formulation, prohibiting further general statements.

Hence, now the EULER forward scheme is considered, resulting in

$$\frac{d\Phi(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_{k+1}, \mathbf{u}_{k+1})}{d\mathbf{x}_k} = \frac{d\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)}{d\mathbf{x}_k} \tag{3.228}$$

$$\frac{d\Phi(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{x}_k, \mathbf{u}_k)}{d\mathbf{x}_k} = 0 \tag{3.229}$$

and

$$\boldsymbol{\Phi}_{\left(\boldsymbol{\lambda}\cdot\frac{d\mathbf{f}}{d\mathbf{x}}\right)}\left(\mathbf{x}_k, \boldsymbol{\lambda}_{k-1}, \mathbf{u}_k, \mathbf{x}_{k+1}, \boldsymbol{\lambda}_k, \mathbf{u}_{k+1}\right) = \boldsymbol{\lambda}_k\cdot\frac{d\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)}{d\mathbf{x}_k} \tag{3.230}$$

that is finally leading to:

$$\tilde{\boldsymbol{\lambda}}_k = \boldsymbol{\lambda}_k \approx \boldsymbol{\lambda}(t_k) \tag{3.231}$$

Otherwise, using a trapezoidal collocation scheme, things become more complicated, as now

$$\frac{d\Phi(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_{k+1}, \mathbf{u}_{k+1})}{d\mathbf{x}_k} = \frac{1}{2}\cdot\frac{d\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)}{d\mathbf{x}_k} \tag{3.232}$$

$$\frac{d\Phi(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{x}_k, \mathbf{u}_k)}{d\mathbf{x}_k} = \frac{1}{2}\cdot\frac{d\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)}{d\mathbf{x}_k} \tag{3.233}$$

and

$$\begin{aligned}&\boldsymbol{\Phi}_{\left(\boldsymbol{\lambda}\cdot\frac{d\mathbf{f}}{d\mathbf{x}}\right)}\left(\mathbf{x}_k, \boldsymbol{\lambda}_{k-1}, \mathbf{u}_k, \mathbf{x}_{k+1}, \boldsymbol{\lambda}_k, \mathbf{u}_{k+1}\right)\\ &\qquad= \frac{1}{2}\cdot\left(\boldsymbol{\lambda}_{k-1}\cdot\frac{d\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)}{d\mathbf{x}_k} + \boldsymbol{\lambda}_k\cdot\frac{d\mathbf{f}(\mathbf{x}_{k+1}, \mathbf{u}_{k+1})}{d\mathbf{x}_{k+1}}\right)\end{aligned} \tag{3.234}$$

still resulting in:

$$\tilde{\boldsymbol{\lambda}}_{k-1} \approx \boldsymbol{\lambda}_{k-1} \approx \boldsymbol{\lambda}(t_{k-1}) \tag{3.235}$$

3.2.6 Problems with Linearly Appearing Controls and Singular Arcs

Recall the extended *Hamiltonian* from equation (3.113)

$$\tilde{H}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}, \mathbf{u}) = L(\mathbf{x}, \mathbf{u}) + \boldsymbol{\lambda}^{\intercal}\mathbf{f}(\mathbf{x}, \mathbf{u}) + \boldsymbol{\mu}^{\intercal}\mathbf{S}(\mathbf{x}, \mathbf{u}) \tag{3.236}$$

and the optimality condition for constrained optimal control problems from equations (3.114)

$$\mathbf{u}_{opt} = \arg\min_{\mathbf{u}} \tilde{H}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}, \mathbf{u}) \tag{3.237}$$

that may also be formulated using

$$\tilde{H}'_{\mathbf{u}}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}, \mathbf{u}) = L'_{\mathbf{u}}(\mathbf{x}, \mathbf{u}) + \boldsymbol{\lambda}^{\mathsf{T}} \mathbf{f}'_{\mathbf{u}}(\mathbf{x}, \mathbf{u}) + \boldsymbol{\mu}^{\mathsf{T}} \mathbf{S}'_{\mathbf{u}}(\mathbf{x}, \mathbf{u}) = 0 \tag{3.238}$$

according to equation (3.115), whenever $\mathbf{u}$ is either generally unbounded or not at its boundaries in the respective part of the optimal solution.

Now, if the controls $\mathbf{u}$ appear linearly in all summands of the *Hamiltonian* and at least one component of $\mathbf{u}$ is not determined by its boundary value along a particular part of the trajectory, equation (3.238) does not deliver any information on the optimal control history, as all of the gradients $L'_{\mathbf{u}}(\mathbf{x}, \mathbf{u})$, $\mathbf{f}'_{\mathbf{u}}(\mathbf{x}, \mathbf{u})$ and $S'_{\mathbf{u}}(\mathbf{x}, \mathbf{u})$ are independent of $\mathbf{u}$. Consequently, $\nabla_{\mathbf{u}} \tilde{H} = 0$ cannot be solved because $\nabla_{\mathbf{uu}} H$ is singular. These parts of a trajectory where the optimal control $\mathbf{u}(t)$ cannot be determined uniquely using the aforementioned optimality conditions are called singular arcs. In most cases, a unique solution to the optimal control problem exists even when singular arcs appear that cannot be determined. [Ben10, p. 119]

Using the relationship between the continuous optimal control problem and its discretized representation, as shown in section 3.2.5, it can be seen that singular arcs still remain problematic as the NLP problem resulting from the discretization still features a singular *Hessian* in this case. For a collocation approach, this can e.g. be seen from equation (3.208) where the *Hessian* of the *Lagrangian* is singular if the *Hessian* of the *Hamiltonian* and the one of the constraints are singular. If this happens, normally, the numerical algorithm utilized to solve the problem, uses a regularization approach to "fix" the issue. Anyway, as the underlying theoretical problem will not be fixed this way, this approach often results in bad convergence, in chattering control histories or may completely fail in practice.

In [Bet09], BETTS suggests to fix the real underlying problem by imposing the required higher order necessary condition:

$$\frac{d^2}{dt^2}(\nabla_{\mathbf{u}} \tilde{H}) = 0 \tag{3.239}$$

However, he also states that this hybrid technique (that requires the calculation of the *Hamiltonian*) inherits the major drawbacks of the indirect approach as the arc sequence needs to be known in advance.

In the general literature, a widely used approach to overcome the problem of singular arcs is to add an additional penalty LAGRANGE cost function

$$J_{\text{penalty}} = \int_{t_0}^{t_f} \mathbf{u}^{\mathsf{T}} \mathbf{R}_{\mathbf{u}} \mathbf{u} \, dt, \tag{3.240}$$

with $\mathbf{R}_{\mathbf{u}} \in \mathbb{R}^{n_{\mathbf{u}} \times n_{\mathbf{u}}}$ being an arbitrary positive definite weighting matrix, to the problem. This way, the singularity is fixed and the solution to the problem can normally be determined. Anyway, choosing the weighting factors is not a simple task, as numbers too small do not remove the numerical pitfalls, while numbers too large influence the whole solution of the original problem. Besides, this additional LAGRANGE cost function influences the whole trajectory, while only one part of it may be singular.

One way of overcoming this issue is by already respecting it in setting up the model (if possible) and changing it e.g. to another set of controls that do not appear linearly in the problem anymore. In chapter 6, different modeling ideas for aircraft will be discussed also taking singularities related to linear controls into account. [Bet09, p. 125, p. 212].

3.3 Multi Criteria Optimization and Optimal Control

In many real world applications competing objectives make it impossible to determine one single optimal solution for a problem. Instead, these problems can be formulated as multi criteria or multi objective optimization problems. In case the underlying problem was an optimal control problem, one speaks about multi criteria or multi objective optimal control problems. In these problems, the goal is to determine either one or several *"best possible compromises"*.

Mathematically speaking, the previously scalar cost function now becomes vector valued:

$$\min \mathbf{J}(\mathbf{z}) = \min \begin{pmatrix} J_1(\mathbf{z}) \\ J_2(\mathbf{z}) \\ \vdots \\ J_N(\mathbf{z}) \end{pmatrix} = \begin{pmatrix} \min J_1(\mathbf{z}) \\ \min J_2(\mathbf{z}) \\ \vdots \\ \min J_N(\mathbf{z}) \end{pmatrix} \tag{3.241}$$

The *"best possible compromise"* is mathematically expressed as PARETO optimality, where a PARETO optimal point $\mathbf{z}_{opt}$ is defined as a point in which the further decrease of one cost function is only possible by increasing (at least one) other cost function values. The solution $\mathbf{z}_{opt}$ is called PARETO optimal if no $\mathbf{z}$ exists such that

$$\begin{aligned} &J_i(\mathbf{z}) \leq J_i(\mathbf{z}_{opt}), \qquad \forall i = 1, \ldots, N \qquad \text{and} \\ &J_j(\mathbf{z}) < J_j(\mathbf{z}_{opt}) \qquad \text{for at least one index } j \end{aligned} \tag{3.242}$$

Besides, the solution $\mathbf{z}_{opt}$ is called *weak* PARETO optimal if no $\mathbf{z}$ exists such that:

$$J_i(\mathbf{z}) < J_i(\mathbf{z}_{opt}), \qquad \forall i = 1, \ldots, N \tag{3.243}$$

The set of all these "best compromise" solutions is called PARETO set or PARETO front [KM13, Mie01, Lei81]. Figure 3.8 depicts some PARETO points along a convex PARETO front for an optimization problem with two competing cost functions (non-convex PARETO fronts also exist, being computationally more challenging). With the shaded area representing the feasible objective region, one can see that no further improvement of neither one of the cost functions is possible along the PARETO front without increasing the other cost.

The main problem to be solved in multi criteria optimization is the determination of (at least some) points along the PARETO front in order to be able to calculate (at least) an approximation of the front. If one single solution is required in the end, the involvement of a (human) decision maker is inevitable. The decision maker has to define his/her preferences, which then need to be used to sort the PARETO optimal solutions of the problem. MIETTINEN distinguishes between *no-preference methods*, *a posteriori methods*, *a priori methods*, and

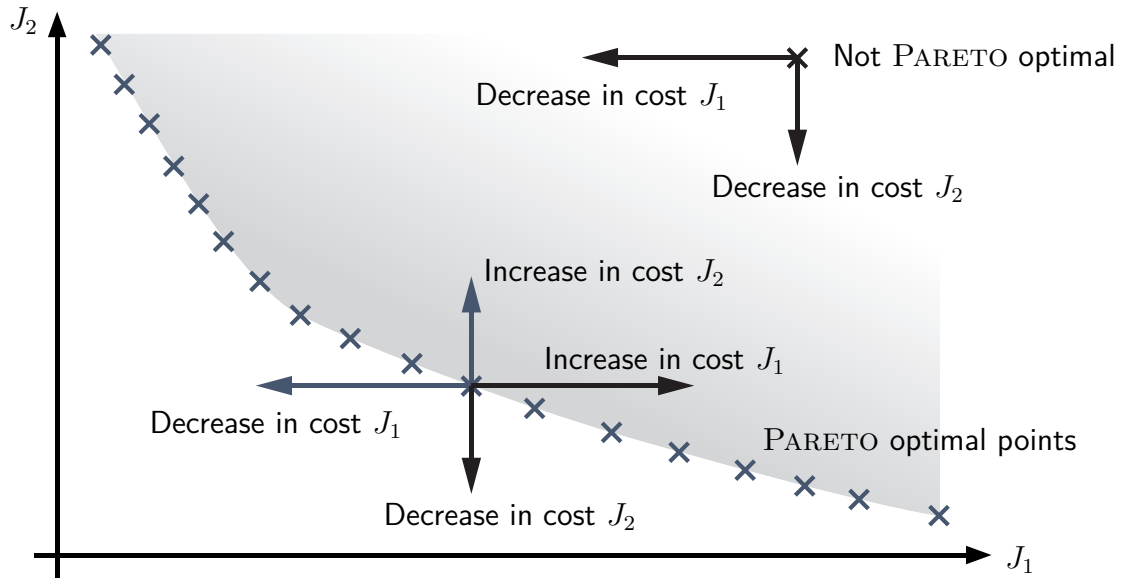

Figure 3.8: *Convex* PARETO *front for a multi-criteria optimization problem with two competing cost functions. The shaded region marks the feasible objective region. A decrease in one cost function always comes with an increase in the other along the* PARETO *front. In all other feasible points, a decrease in both cost directions is possible.*

interactive methods where the name suggests where in the process the decision maker comes into play. In some cases the decision maker may want to combine the cost functions using his/her own scheme as sometimes particular compromises are sought. In most cases, an analytical solution of the problems is not possible, requiring numerical techniques. [Mie01]

The general idea for solving multi criteria optimization problems is based on scalarization, which combines the different cost functions to one scalar cost function using a parametric scalarization scheme. Afterwards, the problem needs to be solved for different parameter sets of the scalarization scheme, hoping to find PARETO points for the original multi criteria optimization problem. In the following sections different scalarization schemes for multi criteria optimization problems are presented, for which more details can e.g. be found in [Mie01].

3.3.1 Weighted Sum Scalarization

The most intuitive idea of determining points along the PARETO front is the weighted sum of cost functions, also called BOLZA scalarization. Therein, the cost values are summed up by:

$$\begin{aligned} J_{sum} &= \sum_{i=1}^{N} w_i \cdot J_i \qquad \text{with} \\ w_i &> 0, \quad \sum_{i=1}^{N} w_i = 1, \qquad \forall i = 1, \ldots, N \end{aligned} \tag{3.244}$$

Anyway, the main drawback of the weighted sum scalarization is that it cannot generate any non-convex parts of a PARETO front as it minimizes the cost function values along a tangent

to the PARETO front, depending on the parameters w_i. Besides, the systematic perturbation of the weights w_i is another challenge, as the relation between the weights and the position of a point along the PARETO front may be highly non-linear. This means that slightly different weights may produce completely different points and vice versa. MIETTINEN references some methods for efficiently selecting the weights in her work. [KM13, Mie01]

3.3.2 Weighted Metrics Scalarization

The weighted metrics scheme uses a target cost value $\mathbf{J}_T$ as reference and tries to minimize the weighted distance in the p-norm between the current solution and the reference cost.

$$J_p = \left(\sum_{i=1}^{N} w_i \left| J_i - J_{i,T} \right|^p \right)^{\frac{1}{p}}, \qquad p \geq 1, \quad w_i > 0, \quad i = 1, \ldots, N \tag{3.245}$$

The method is also called *compromise programming* and the different solutions are produced by weighting the metrics. For positive weights the solution produced by the weighted metrics scheme are weakly PARETO optimal. For convex problems, every PARETO optimal point can be found by this method. The main challenge in applying the method lies in the determination of a target cost value $\mathbf{J}_T$. In the exemplary application in section 10.3 a method for overcoming this challenge in the particular application is proposed. [Mie01]

3.3.3 ε-Constraint Scalarization

In this method, one of the objectives is optimized while all other cost function values are bounded:

$$\begin{aligned} &J_\varepsilon = J_k \qquad \text{subject to} \\ &J_i < \varepsilon_i, \quad \forall i = 1, \ldots, N, \quad i \neq k \end{aligned} \tag{3.246}$$

The problem needs to be solved for all $k = 1, \ldots, N$. The main challenge in this approach is the selection of the upper bounds ε_i for the constrained cost functions. MIETTINEN states that solutions to the problem defined by the ε-constraint scalarization are weakly PARETO optimal. [Mie01]

3.3.4 Tschebyscheff Scalarization

In [KM13], KAYA uses the weighted TSCHEBYSCHEFF scalarization for finding a solution to non-convex multi-criteria optimal control problems. This scalarization uses the minimization of the maximum of the weighted cost components as its cost function:

$$\begin{aligned} &J_\infty = \lim_{p \to \infty} \left(\sum_{i=1}^{N} w_i \left| J_i - J_{i,T} \right|^p \right)^{\frac{1}{p}} = \max_i (w_i (J_i - J_{i,T})) \\ &w_i > 0, \qquad i = 1, \ldots, N \end{aligned} \tag{3.247}$$

In order to be able to find valid solutions for the multi criteria optimization problem, all cost function values have to be positive, which can easily be achieved by modifying the target cost

$J_{i,T}$ appropriately. KAYA further states that a point $\mathbf{z}_{opt}$ is a weak PARETO minimum of the multi criteria optimization problem, if and only if it is a solution to the optimization problem using the scalarized cost function from (3.247).

In order to numerically solve the *min-max* problem that results from the weighted TSCHEBYSCHEFF scalarization, the transformation listed in equation (3.76) can be used, with the constraint that is introduced there applied to all weighted parts of the cost function. This reformulation is also known as the PASCOLETTI-SERAFINI scalarization.

For determining the PARETO front efficiently, a strategy for choosing the weights w_i is required that reduces the number of optimization problems that need to be solved. In [KM13], KAYA presents an algorithm for this purpose. Moreover, in the standard formulation of the TSCHEBYSCHEFF scalarization, all cost function values that are currently eliminated by the $\max$ statement in equation (3.247), are not considered at all, resulting in possible further unused optimization potential. In the exemplary application in section 10.3, this issue is handled by combining several scalarization approaches. More information on the TSCHEBYSCHEFF method and its application can besides other be found in [Mie01, Lin05].

3.3.5 A Priori Methods

Methods in which the decision maker specifies his/her preferences in advance to performing any optimization are called *a priori methods*. The main drawback of those methods is that they do not try to determine the PARETO front (or parts of it), but directly try to find one solution that best fits the needs of the decision maker. Anyway, as the decision maker often does not know what can be achieved within a problem and what not, the application of such methods may be tough. On the contrary, the main benefit is a short solution time as in most of the methods very little optimization problems need to be solved.

In the *value function method*, the decision maker combines the cost functions to one scalar cost best satisfying his/her preferences. In most cases the determination of the value function is complicated to impossible.

When using the method of *lexicographic ordering*, the decision maker selects the most important cost function and solves the problem with respect to this cost. Afterwards, the value of this cost function is constrained and the problem is solved again for the second important cost function. This procedure is carried on until either all cost functions have been minimized or one of the intermediate solutions is unique, not allowing any further improvement in the remaining cost functions. The main drawback of the *lexicographic ordering* method is that no trade off between the cost functions may take place.

The *goal programming* method requires the specification of a goal to be achieved before the deviations to this goal are minimized. The goal should be selected such that not all cost function values in the goal are reachable simultaneously. The method has several variants, most of them requiring the decision maker to specify additional weights or orderings. More details on these and other methods can e.g. be found in [Mie01].

3.3.6 Interactive Methods

As the name suggests, in *interactive methods*, the decision maker uses his/her preferences to interactively influence the optimization process. The main benefit of these methods is the fact that in most cases not the whole PARETO optimal set needs to be determined as the user is able to control the direction into which the PARETO set is evolving. Furthermore, the decision maker only has part of the information at hand in every step potentially simplifying the decision.

In [Mie01], MIETTINEN presents several interactive methods, all having their benefits and drawbacks. The methods presented include the GEOFFRION-DYER-FEINBERG *method*, the *interactive* TSCHEBYSCHEFF *method*, the *Reference Point Method*, the *GUESS Method*, the *Satisficing Trade-Off Method*, the *Light Beam Search* and the *NIMBUS Method*.

3.3.7 Multi Criteria Optimal Control

KAYA distinguishes between different approaches for solving multi criteria optimal control problems. One being the idea of *scalarize-discretize-then-optimize* where the original infinite dimensional optimal control problem is scalarized and afterwards discretized and then solved numerically. On the contrary, he mentions the *discretize-scalarize-then-optimize* technique where the scalarization is performed based on the discretized finite dimensional optimization problem. [KM13]

Considering the *scalarize-discretize-then-optimize* methods, LEITMANN derives necessary and sufficient optimality conditions for PARETO optimality for optimal control problems. He uses the definition of PARETO optimality from (3.242) and combines it with the optimality conditions for optimal control problems from (3.115)–(3.118). [Lei81]

Necessary Conditions for Pareto Optimality

LEITMANN defines the extended *Hamiltonian*

$$\tilde{H}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}, \mathbf{u}, \boldsymbol{\alpha}) = \boldsymbol{\alpha}^\intercal L(\mathbf{x}, \mathbf{u}) + \boldsymbol{\lambda}^\intercal \mathbf{f}(\mathbf{x}, \mathbf{u}) + \boldsymbol{\mu}^\intercal \boldsymbol{S}(\mathbf{x}, \mathbf{u}) \tag{3.248}$$

for a problem with vector valued cost (3.241). Using this function, the necessary conditions can be formulated as:
If the combination $(\mathbf{u}_{opt}(t), \mathbf{x}_{opt}(t))$ is PARETO optimal, then there exist multipliers $\boldsymbol{\lambda}(t)$, $\boldsymbol{\mu}(t)$ and constants $\boldsymbol{\alpha} \in \mathbb{R}^N$ such that the conditions in (3.115)–(3.118) are satisfied for the extended *Hamiltonian* (3.248) and

$$\alpha_i \geq 0, \qquad \forall i = 1, \ldots, N \tag{3.249}$$

holds. Again, the trivial solution $\alpha_i = 0, \forall i = 1, \ldots, N$ is not allowed. More details as well as a proof for these conditions can be found in [Lei81].

Sufficient Conditions for Pareto Optimality

Besides the necessary conditions, LEITMANN also presents sufficient conditions for PARETO optimality for optimal control problems:
The solution $(\mathbf{u}_{opt}(t), \mathbf{x}_{opt}(t))$ is PARETO optimal if there exists a constant $\boldsymbol{\alpha} \in \mathbb{R}^N$ with $\alpha_i \geq 0$ for $i = 1, \ldots, N$ and $\sum_{i=1}^{N} \alpha_i = 1$ such that

$$\sum_{i=1}^{N} \alpha_i J_i(\mathbf{x}(t), \mathbf{u}(t)) > \sum_{i=1}^{N} \alpha_i J_i(\mathbf{x}_{opt}(t), \mathbf{u}_{opt}(t)) \tag{3.250}$$

for all admissible $\mathbf{u}(t)$ with $\mathbf{u}(t) \neq \mathbf{u}_{opt}(t)$.

Compared to the necessary conditions, now, not all components of $\boldsymbol{\alpha}$ need to be non-zero. Even though these conditions are mathematically correct, checking them in real applications is often very tough.[Lei81]

Chapter 4

Implementation of a Discretization Scheme in an Optimal Control Tool

In order to be able to easily solve a wide variety of optimal control problems using gradient based optimization techniques without the need to re-implement the numerical discretization scheme every time, an optimal control tool for *MATLAB* has been developed at the Institute of Flight System Dynamics. This optimal control tool is described here, and is also used to solve some of the examples presented in the remainder of this work. The main contributors during the development of the different versions were Maximilian Richter, Matthias Rieck, Benedikt Grüter, Johannes Diepolder and this author. Besides, several Bachelor and Master thesis as well as student positions have been done on the tool. The current version of the tool is published as *FALCON.m*[1] [Rie+16]. Figure 4.1 shows the logo (designed by Matthias Rieck) of the tool.

Figure 4.1: *Logo of the optimal control tool* FALCON.m *developed at the Institute of Flight System Dynamics. The logo has been designed by* Matthias Rieck.

FALCON.m is mainly an object-oriented *MATLAB* library containing classes for every part required to construct a wide variety of optimal control problems. For ensuring a fast solution of the problems, *FALCON.m* uses several *MATLAB* specific technologies, including

- vectorial evaluations,
- symbolic computation,

[1] The official website for the project can be found at www.falcon-m.com.

- sparse matrices,
- automatic code generation, and
- compiled *MATLAB* API library functions (.mex-functions).

FALCON.m can be divided into several main components, ensuring its high flexibility, where the most important ones are

- the so-called *Model Builder* that can be used to prepare dynamic simulation models as well as cost and constraint functions for their use in an optimal control problem (see section 4.8.4),
- the *Optimal Control Problem Hierarchy* which is used to represent the main structure of the optimal control problem to be solved, also including most of the technologies required to solve it (see section 4.4), and
- the *Numerical Solver Interface* that generally allows the use of *FALCON.m* together with a wide variety of numerical optimization tools such as *IPOPT* [WB06], *SNOPT* [GMS02] or *WORHP* [BW13].

Besides its main core, *FALCON.m* ships with additional tools that allow for the analysis of calculated results including a specialized graphical user interface (GUI) for plotting. Moreover, besides the main classes, additional required external *MATLAB* and *C*-functions are shipped with the tool.

Historically, Maximilian Richter and this author started developing their own separate optimization tools at the Institute of Flight System Dynamics in 2011 and 2012. At some point, they decided to combine their knowledge in one common tool which at that time was called *iFramework*. It was developed and implemented by Maximilian Richter, Matthias Rieck and this author. This first tool contained an implementation of collocation methods as well as multiple shooting methods. The support for both these methods was one of the main drawbacks of the tool, as many aspects were either done twice or not to maximum efficiency in order to cope with the particular specialties of the different methods. In this early version of the code, the required derivatives for the model dynamics and the constraint functions still needed to be provided by the user. Moreover, the tool was able to use dynamics simulations either done in *Simulink* or in *MATLAB* – again, the support of both possibilities lead to suboptimal overall performance.

For the current version of *FALCON.m*, the software structure was revised and the whole code was optimized for maximum execution performance and good usability. Hence, the tool was focused on the use of full discretization (collocation) techniques, reducing code overhead. Anyway, the possibility to integrate shooting techniques was still kept and is also used within this work (see chapter 8 for more details on a combined shooting and collocation approach). Many parts of the code were re-implemented and improved, with the main implementation of the update being done by Matthias Rieck. The main contribution of the author within the development of *FALCON.m* is in the design of the software architecture and the required structure for an efficient implementation.

4.1 Other Available Optimal Control Software

Before *FALCON.m* was developed at the Institute of Flight System Dynamics, other tools that are helpful for solving optimal control problems were already existing. Anyway, all of them had drawbacks for the applications considered here or could not be adapted for the developments done at the Institute. The following list shows the most common ones. The list is non-exhaustive and gives only a very brief overview.

4.1.1 DIDO

The optimal control tool *DIDO* was invented by Michael Ross in 2001 and is based on the *Unified Covector Mapping Theorem* developed by Fariba Fahroo and Michael Ross [FR08]. It is currently commercially available at *Elissar Global*. *DIDO* is a *MATLAB* tool based on the Legendre-Gauss and the Legendre-Gauss-Radau *Pseudospectral Methods*. The tool features a special problem formulation technique called "pencil-and-paper-like" problem formulation. The aforementioned *Covector Mapping Theorem* allows for the estimation of the costates that in some cases allow the verification of the results using the minimum principle. [Eli16]

4.1.2 GESOP

GESOP, the *Graphical Environment for Simulation and Optimization* was developed by *Astos Solutions GmbH*. *GESOP* offers a graphical user interface and uses different direct collocation as well as shooting methods for solving optimal control problems. Besides, the software offers an initial guess generator and a simulation tool. *GESOP* features interfaces to the numerical optimization tools *SNOPT* and *SOS* and supports model and constraint functions provided as Fortran 77, C, or Ada 95 functions or *MATLAB* scripts. *GESOP* uses finite differencing in order to determine gradient information. [Ast16]

4.1.3 GPOPS and GPOPS-II

The first version of the *General Purpose OPtimal Control Software* developed at the *Vehicle Dynamics and Optimization Laboratory* of the *University of Florida* was published in 2008 by Camila Francolin and Anil V. Rao [Fra08]. *GPOPS* represents an implementation of Gauss *Pseudospectral Methods* for *MATLAB* [Rao+10] that is available for free and open source. The required gradients are determined using finite differences.

In 2013, Michael Patterson and Anil V. Rao published the first release of *GPOPS-II*, being the commercial successor of their first tool. *GPOPS-II* uses *hp-adaptive variable-order* Gaussian *quadrature collocation* based on Legendre-Gauss-Radau points and an adaptive grid refinement method. The gradients are still calculated using finite differencing. [PR14]

4.1.4 NUDOCCS

NUDOCCS, short for *"Numerical Discretisationmethod for Optimal Controlproblems with Constraints in Controls and States"* [sic], has been developed by CHRISTOF BÜSKENS in the 1990s. It is implemented in FORTRAN and is based on a multiple shooting method. In order to solve the resulting numerical optimization problem, the routine *E04UCF* from the FORTRAN library *Nag* is used. All required gradients are calculated by different finite differencing schemes incorporated in the tool. [Büs96]

4.1.5 PSOPT

The code *PSOPT* was first published in 2009 and is currently available in version 3. At the moment, the project is maintained by VICTOR M. BECERRA. The open source tool is implemented in C++ and compatible with Linux and Windows systems. It uses direct collocation methods, offering global pseudospectral LEGENDRE and TSCHEBYSCHEFF approaches as well as full discretization methods like the *Trapezoidal* or HERMITE-SIMPSON method. For solving the resulting numerical optimization problem, interfaces to *IPOPT* and *SNOPT* may be used. The required gradients may either be calculated using *ADOL-C*[2], an automatic differentiation code based on operator overloading, or by finite differencing. In case *ADOL-C* is used, information about the problem sparsity is also auto-generated. Furthermore, *PSOPT* offers mesh refinement technologies, that are especially important when using global approximation methods like the *Pseudospectral Method*. [Bec11]

4.1.6 OCPID-DAE1

MATTHIAS GERDTS is the main developer of the optimal control tool *OCPID-DAE1*, short for *Optimal Control and Parameter Identification with Differential-Algebraic Equations of Index 1*, which is another optimal control and parameter identification tool developed in FORTRAN. Even though being mainly developed for Linux, a version that is compatible with Microsoft Windows exists. The tool is based on a multiple shooting discretization method and uses the sequential quadratic programming code from *sqpfiltertoolbox*, another FORTRAN library. The required derivative information may either be calculated using sensitivity differential equations or using finite differences. Available extensions for the tool enable an estimation of the costates or add realtime optimization capabilities based on the sensitivity theory from section 3.1.6. [Ger13]

4.1.7 SOCS

SOCS is the *Sparse Optimal Control Software* mainly developed by JOHN BETTS at *The Boeing Company*. It uses a direct collocation method and features an interface to the sparse

[2] https://projects.coin-or.org/ADOL-C

numerical optimization software *SPRNLP*, developed in the same team. The sparsity patterns of the *Jacobian* and *Hessian* need to be defined by the user in the implementation and are later used to exploit overall problem sparsity. Besides, they are used for the sparse finite difference approximation of the *Jacobian* and the *Hessian*. Furthermore, *SOCS* offers mesh refinement features in order to improve accuracy. The software has been mainly developed in 1993 and 1994 and is implemented in FORTRAN 77. [Uni16]

4.1.8 TOMLAB/PROPT

Per E. Rutquist and Marcus M. Edvall from *Tomlab Optimization* were the main developers of *PROPT*, the optimal control tool offered by that company. *PROP* is commercially available as part of the *TOMLAB* modeling and optimization suite for *MATLAB*. In the latest version, the tool supports direct collocation methods based on Gauss or Tschebyscheff points. *TOMLAB/PROPT* uses the *TOMSYM engine* and its automatic differentiation features for determining the required gradients as well as for code generation. [RE10, Tom11]

4.2 Problem Formulation for FALCON.m

The optimal control problem introduced in section 3.2.1 needs to be transformed to a specific formulation in order to be solved using *FALCON.m*. For that, the transformation techniques from section 3.2.2 may help, depending on the particular task. In general, the tool can determine a time discretized approximation of the box constrained optimal state and control histories

$$\mathbf{x}_{\min} \leq \mathbf{x}(t_k) \leq \mathbf{x}_{\max} \quad \text{and} \tag{4.1}$$

$$\mathbf{u}_{\min} \leq \mathbf{u}(t_i) \leq \mathbf{u}_{\max}, \tag{4.2}$$

with t_k and t_i being points in time on the state and control grids as introduced in section 3.2.4 (see equations (3.174) and (3.149)). Besides, *FALCON.m* can determine additional box constrained parameters

$$\mathbf{p}_{\min} \leq \mathbf{p} \leq \mathbf{p}_{\max} \tag{4.3}$$

influencing the optimal control problem. The software treats every problem as a multi-phase problem, even though the user may specify only one single phase.

In order to be able to cope with free final time problems, the linear time transformation mentioned in section 3.2.2 is directly incorporated in *FALCON.m*. However, this problem formulation requires that all grids and points in time are specified in normalized time $\tau \in [0, 1]$ for each phase of the problem. The parameters required in the transformation for initial and final time are either introduced by the user or automatically. They are anyway added to the parameter vector $\mathbf{p}$ of the overall problem.

The cost to be minimized may be formulated as a differentiable function that is arbitrarily depending on the states, the controls, the initial and final time, and any additional parameters

appearing in the problem:

$$J = J(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}) \tag{4.4}$$

In this formulation, the times t_k and t_i are a set of points in time selected by the user. (The points may not be selected arbitrarily but need to be a subset of the state or control grid introduced before.) The BOLZA cost function introduced in section 3.2.1 can be seen as a subset of this formulation and can consequently be implemented in the software. Besides this general formulation, *FALCON.m* allows the use of LAGRANGE cost functions of the form

$$J = \int_{t_0}^{t_f} L(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}, t)\, dt \tag{4.5}$$

by directly specifying the function L inside the integral and automatically performing a numerical integration.

FALCON.m is able to solve problems containing autonomous and parameter dependent dynamic systems of the following form:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}) \tag{4.6}$$

Non-autonomous dynamic systems may be incorporated using the transformation technique mentioned in section 3.2.2. Moreover, many other dynamic system formulations may be transformed into this standard form using the methods listed above or employing other transformations (like those for discrete controls, see [RRH13, Rie+14]) not described here.

Besides the dynamic constraints, almost every real life optimal control problem features algebraic constraints, like initial or final boundary conditions or path constraints. *FALCON.m* offers functionality to easily add path constraints of the form

$$\mathbf{C}_{\min} \leq \mathbf{C}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}) \leq \mathbf{C}_{\max} \tag{4.7}$$

to each phase of the optimal control problem. These path constraints are either evaluated in every discretization point within a phase or on a user defined subset of the grid. Moreover, the user may specify so-called *Point Constraint Functions* that use an arbitrary set of values appearing in the problem in order to calculate one single constraint value:

$$\mathbf{c}_{\min} \leq \mathbf{c}(\mathbf{x}(t_{k1}), \mathbf{x}(t_{k2}), \ldots, \mathbf{u}(t_{i1}), \mathbf{u}(t_{i2}), \ldots, \mathbf{p})) \leq \mathbf{c}_{\max} \tag{4.8}$$

In this formulation, the times t_k represent user selected points in normalized time that may even be taken from different phases of the optimal control problem. In case interior point constraints need to be specified without knowing their normalized time of appearance, the multi-phase feature of *FALCON.m* must be used. Then, in the problem formulation, the durations of the phases neighboring the respective constraint need to be of variable length and the constraint needs to be introduced as a final or initial boundary condition in these phases.

FALCON.m offers several features to easily add initial and final boundary box constraints of the form

$$\mathbf{x}_{0,\min} \leq \mathbf{x}(t_0) \leq \mathbf{x}_{0,\max} \tag{4.9}$$

$$\mathbf{x}_{f,\min} \leq \mathbf{x}(t_f) \leq \mathbf{x}_{f,\max} \tag{4.10}$$

to every phase of the problem. In case the initial state of one phase is not box constrained but should take the same value as the final state of the previous phase, there is a possibility to directly connect the respective phases.

In all cases mentioned above, equality constraints can be modeled by simply setting the upper limit to the same value as the lower limit:

$$\circ_{\min} = \circ_{\max} \tag{4.11}$$

4.3 Problem Setup Process

The process for setting up and solving an optimal control problem with *FALCON.m* contains the following main steps:

1. **Implement the subsystems of the dynamic models.** The dynamic models used within the optimal control problem need to be implemented in *MATLAB* functions. Large models need to be partitioned into (smaller) subsystems, each implemented separately.

2. **Build the overall dynamic models from the subsystems.** The dynamic models used in the different phases of the optimal control problem need to be combined using the *Model Builder* delivered with *FALCON.m*.

3. **Automatically differentiate all required functions.** At this point, the *Model Builder* will automatically create the analytic derivatives of all subsystems and combine them to the overall gradient of the models using the chain rule.

4. **Auto-generate code for the models and compile.** In order to be able to execute the aforementioned functions many times during the optimization process with maximum evaluation efficiency, the *Model Builder* within *FALCON.m* will automatically create *C*-code from the *MATLAB* functions including their analytic derivatives. Besides, all required wrapper code will be created that allows for the parallelized evaluation of the model dynamics on multi core systems. Finally, the whole code will be compiled into one *MATLAB* executable (.mex-File) that will later be called in the optimization process.

5. **Implement additional constraint functions.** Repeat steps 1 – 4 for all required constraint and cost functions in the problem.

6. **Define the structure of the optimal control problem.** This step may be seen as the main preparation step in solving an optimal control problem with *FALCON.m*. The overall problem hierarchy needs to be set up using the *MATLAB* classes delivered with *FALCON.m*. In this step, the problem itself is created before all required elements, like phases, models, states, controls, parameters, constraint- and cost-functions may be added.

7. **Discretize the optimal control problem.** This step can be seen as the *"discretization"* itself, as now the mapping between the numerical optimization problem and the values appearing in the optimal control problem is created. In *FALCON.m*, the method preparing everything for a fast solution is called *"Bake"*.

8. **Solve the discretized problem using state of the art numerical solvers.** Now, the numerical optimization problem resulting from the discretization of the optimal control problem is solved using an appropriate numerical solver. At the moment, *FALCON.m* features interfaces to *IPOPT* and *SNOPT*.

9. **Analyze the solution and draw conclusions.** After solving the numerical optimization problem, the hierarchy created above contains all information related to the solution of the optimal control problem. The user has free access to all datasets and can post-process them as needed. Moreover, a graphical user interface for plotting the results may help in understanding and interpreting the calculated values.

More details on *FALCON.m* and its usage can – besides some examples – be found in the *FALCON.m* Documentation [Rie+16]. The documentation also lists some more basic ideas that played an important role during the development of the software.

4.4 Problem Structure

Figure 4.2 gives a rough overview of the structure of an optimal control problem in *FALCON.m* (the namespaces of the objects are omitted for better readability). As can be seen, the root element of each optimal control problem is an object of type `falcon.Problem`. Below, the children of the problem can be seen, with the list of *Phases* of type `falcon.core.Phase` being the most important one. (Within the *FALCON.m* code, all classes that are directly accessible by the user and are required to construct an optimal control problem are located in the namespace `falcon`. All classes that are required to represent the optimal control problem, but are not used by the user directly are located in the namespace `falcon.core`.) Besides, each problem contains a list of all relevant parameters that are subject to optimization, a list of all point constraint functions, a list of all MAYER cost functions and a handle to the class containing the specifics of the applied numerical discretization method.

Each phase in the problem contains one object of type `falcon.core.Model` representing the dynamic model used in this phase. The state histories of each phase are stored in the respective *StateGrid* which is of type `falcon.core.Grid`. The time discretization of the *StateGrid* is the lowest common factor for all grids used for the calculation of cost and constraint values. (The mixed collocation and shooting approach presented in chapter 8 introduces a finer grid for the simulation, but not for the constraints.) *FALCON.m* allows the use of different discretization grids for different controls in the *ControlGrids* of each phase (all being of type `falcon.core.Grid`). This enables a reduction in the size of the resulting optimization

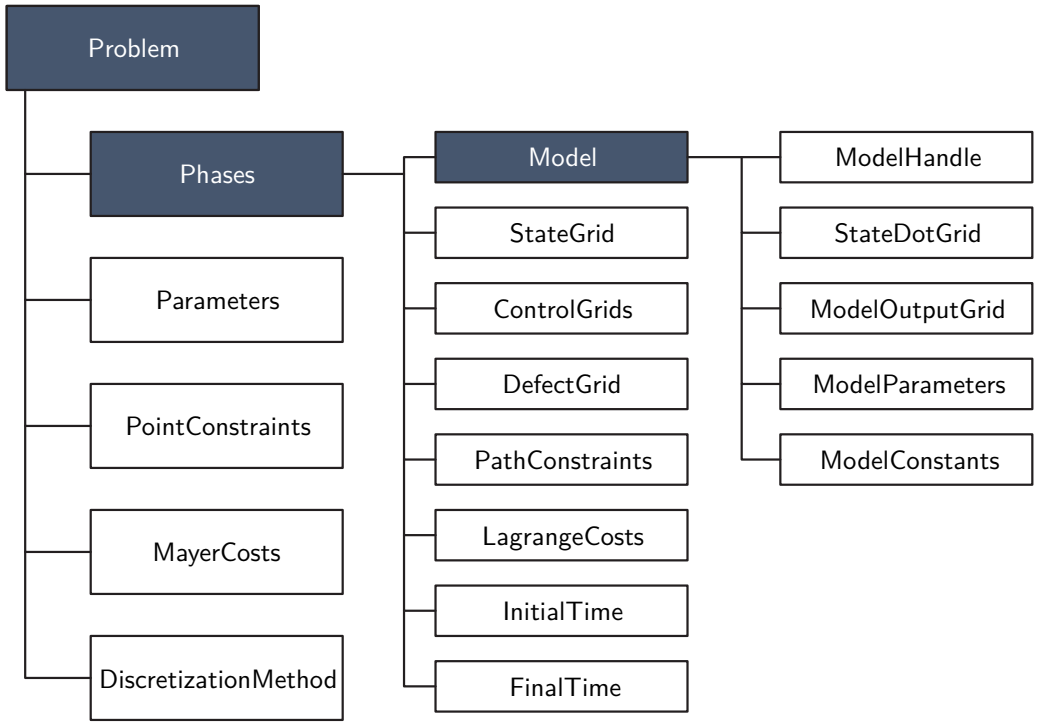

Figure 4.2: *Main elements of the structure of the representation of an optimal control problem within* FALCON.m.

problem in cases where different controls require different resolutions in time. Similar to the *StateGrid*, all defects appearing in the full discretization scheme are stored in the *DefectGrid* of each phase (which is again of type `falcon.core.Grid`). The two arrays *PathConstraints* and *LagrangeCosts* may hold several objects of type `falcon.core.PathFunction` or `falcon.core.LagrangeFunction`, respectively. The ones in the first array specify functions used to calculate the path constraint values together with an upper and a lower boundary. The ones in the latter array are numerically integrated and finally added to the cost function value. The two fields *InitialTime* and *FinalTime* within each phase contain objects of type `falcon.Parameter` and represent the initial and the final time of the respective phase. Problems with fixed initial and/or final time can easily be represented by directly specifying a value for the respective `Parameter` object and setting its `isFixed` property to *true*.

The *Model* of each *Phase* contains all information on the dynamic model used in this phase. First of all, a *ModelHandle* is required that is of type `function_handle` pointing to the *MATLAB* function holding the simulation model and its *Jacobian*. (In most cases, this file has been automatically generated by *FALCON.m* using its *Model Builder* as mentioned before). The values for the time derivatives of the states are stored in the *StateDotGrid* of the *Model*, also being of type `falcon.core.Grid`. Moreover, *FALCON.m* allows the user to specify additional model outputs that may later be used in the cost or constraint functions. The values for these outputs are stored in the *ModelOutputGrid* of type `falcon.core.Grid`. Besides

the states and the controls, the simulation model may also depend on arbitrary numbers of *Parameters* and *Constants*. The *Parameters* are stored in `falcon.Parameter` objects in the *ModelParameters* array, while the numerical values of the required constants are directly stored in the *ModelConstants* field. In general, all parameters appearing in a problem are listed in the *Parameters* array under the top level *Problem* object, but anyway not all parameters may be necessary for evaluating the model and consequently these need to be provided with the model. Anyway, *FALCON.m* takes care of proper sorting and linking of the parameters automatically.

4.5 Derivative Calculation and Code Generation

As *FALCON.m* is designed to mainly support gradient based optimization algorithms, it offers several features for the calculation of gradient information. Overall, the gradient of the cost function and the *Jacobian* of all constraints for the discretized optimal control problem are required. All formulas that calculate the overall gradient and *Jacobian* from the local gradients of the functions involved, are directly integrated into *FALCON.m* not requiring any user input.

All required local gradients like $\frac{\partial \dot{\mathbf{x}}}{\partial \mathbf{x}}$, $\frac{\partial \dot{\mathbf{x}}}{\partial \mathbf{u}}$, and $\frac{\partial \dot{\mathbf{x}}}{\partial \mathbf{p}}$ for the model – or their respective counterparts for constraint or cost functions – are calculated within the *Model Builder* functionality of *FALCON.m*, using the *Symbolic Math Toolbox* from *MATLAB*. As the symbolic computing engine is currently not capable of analytically deriving very complicated expressions, the *Model Builder* calculates the derivatives of each subsystem defined by the user individually, before combining them to the overall model, constraint, or cost function gradient using the chain rule.

In a second step, each function and its respective derivatives are coded into one new common function each, allowing for an easy evaluation during the optimization process. Additionally, these functions feature an evaluation of multiple points in time within one single call. In case *MATLAB Coder* and *OpenMP*[3] are available on the target system, both are used to create *C* code and finally a *MATLAB executable (.mex-file)* that is capable of performing parallelized computations. Besides, *FALCON.m* also allows for the calculation of second order gradient information, normally supplied in form of the *Hessian* matrix. This feature can be used to further improve convergence characteristics in many cases.

To sum up, *FALCON.m* offers a very convenient way of using exact analytic derivatives within the solution of optimal control problems. More details on this particular feature of *FALCON.m* can be found in [Rie+15].

[3] http://openmp.org/wp/

4.6 Sorting and Indexing

Within *FALCON.m*, after the discretization, all values appearing in the optimization parameter vector or in the vector of constraints are sorted according to their time of appearance in the optimal control problem. This means that within the parameter vector $\mathbf{z}$ defined in equation (3.175), first the parameters $\mathbf{p}$ of the problem are listed, then all states and controls of the first phase of the problem are sorted by increasing time. The same holds for all states and controls of the subsequent phases.

The values within the constraint vector $\mathbf{C}(\mathbf{z})$, as defined in equation (3.177) are treated similarly. There, all collocation defects and constraints related to the first phase are listed first, also sorted by their time of appearance in the optimal control problem. Afterwards the same principle is applied to all subsequent phases. The sorting of the values and the calculation of the resulting indices is performed during the preparation of the problem, within the *Bake* function.

The main benefit of this sorting lies in the resulting block diagonal structure of the overall gradient matrix due to the causality of the dynamic systems. Figure 4.3 shows an exemplary sparsity pattern for the gradient matrix of a discretized optimal control problem (with one phase). Further details on problem sparsity and its exploitation can be found in chapter 7.

4.7 Numerical Scaling

Well-formed numerics are one of the core requirements for an efficient solution of a numerical optimization problem. In many cases, bad numerics prevent the optimization algorithms from working properly, sometimes resulting in poor convergence or even no convergence at all. One of the most important aspects regarding the numerics of a discretized optimal control problem is related to the orders of magnitude of the values appearing in the problem. [Bet09, p. 166 ff.]

In order to tackle this issue, *FALCON.m* offers a feature for the numerical scaling of the values appearing in a problem. The user may specify scaling factors M_{scaling} for states, controls, parameters, constraints, and the cost function components. These values will be used to create a linear mapping between the values appearing in the continuous optimal control problem and those appearing in the numerical optimization problem:

$$\circ_{\text{scaled}} = \circ_{\text{original}} \cdot M_{\text{scaling}} \tag{4.12}$$

In case no scaling values are specified, *FALCON.m* tries to determine them automatically from the given bounds and values.

4.8 Software Components

In the following subsections the most important components of *FALCON.m* are described in more detail. However, they are presented rather from a conceptual point of view than

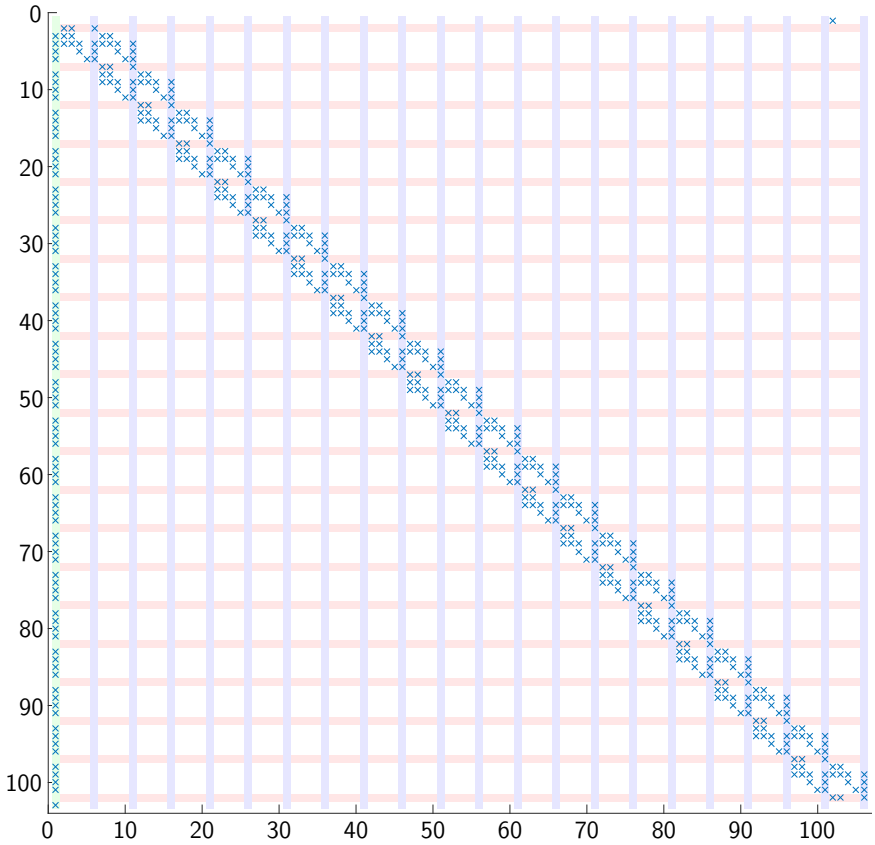

Figure 4.3: *Sparsity pattern of the gradient of a discretized optimal control problem. The first row holds the cost function gradient, all others the constraint* Jacobian. *The rows marked in red represent the* Jacobian *of a path constraint, all other rows hold derivatives of the defects. The first column (green) holds the derivatives with respect to the final time parameter, the blue columns represent derivatives with respect to the controls. All other columns hold derivatives with respect to the states.*

an applicational point of view as details on the usage can be found in the *User Guide* of *FALCON.m* [Rie+16]. All classes described below are *MATLAB handle classes* [The16] that use the *call-by-reference* principle.

4.8.1 Value Containers

When representing an optimal control problem in software, many quantities appearing in the problem formulation may have some commonalities like

- a name,
- a lower bound,
- an upper bound and
- a numerical scaling factor.

Hence, these properties are implemented in the abstract class `falcon.core.OVC` in *FALCON.m*, where *"OVC"* is short for *"Optimization Value Container"*.

Conceptually, all value container classes define the properties of a scalar value. In case a grid of values is required – like e.g. for the states – an object of type `falcon.core.Grid` is used for expanding the properties of a scalar value container object onto a grid over time, avoiding unnecessary duplication of information, finally resulting in less memory consumption. This also implies that the value container objects in general do not hold any numeric values themselves.

Figure 4.4 depicts the child classes derived from `falcon.core.OVC` inheriting its properties that will be described in more detail in the next sections.

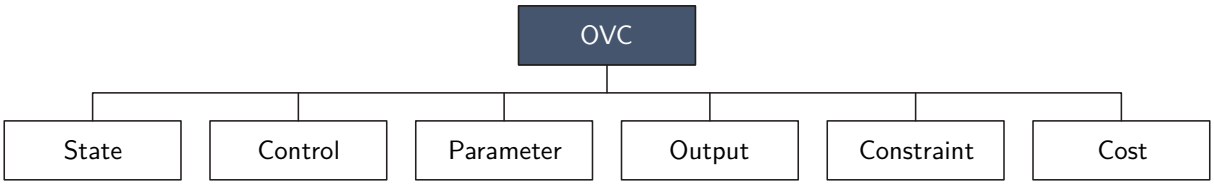

Figure 4.4: *Value container types in* FALCON.m.

States

Each state appearing in an optimal control problem is described by a `falcon.State` object. One *State* object may be used within several phases of a problem as the `falcon.State` class does not hold the solution of the state histories, but it only defines the properties of a single state. In order to represent a state history based on a state definition object, *FALCON.m* automatically expands a `falcon.core.Grid` from a `falcon.core.State` object as soon as a new phase is added to the problem and a state grid is required.

The properties specified for a state, like its name and limits, are automatically applied as the properties for the discretized state within a discretization method. Consequently, all state values calculated on the state grid of a phase are forced to respect the given limits in the final solution of the optimal control problem. When using the combined shooting and collocation method described in chapter 8, the boundaries are only respected by the collocation and multiple shooting nodes, but not by the intermediate values.

Controls

Each control required to model an optimal control problem in *FALCON.m* is described by an object of type `falcon.Control`. Similarly to the states, they may be reused for creating control grids in several phases. The expansion from one single control value to a control grid is done within the method `addNewControlGrid` of the considered phase. This method may be called several times (on the same phase) resulting in several control grids. The benefit

of this feature is that different controls may be discretized on individual grids. Consequently, controls that do not need to be changed as frequently as others can be discretized on a coarser grid resulting in less optimization parameters and a smaller numerical optimization problem.

Moreover, each `falcon.Control` offers an option to fix its value, meaning that the control history based on this object is not subject to optimization. In this case the initial guess values supplied for the control history will not be changed during the optimization process. This feature is especially handy for parameter estimation tasks where the control histories are measured and should not be determined by optimization.

Parameters

Every additional scalar parameter appearing in the optimal control problem that is subject to optimization is represented by an object of type `falcon.Parameter` within *FALCON.m*. In contrast to the states and controls, parameter objects are directly storing their value during the optimization. Consequently, their specifications may not be reused in different places of an optimal control problem except if the same numerical value should be used in different places across the problem.

When solving parameter identification problems, all parameters to be determined within the model need to be represented by a `falcon.Parameter` object that is subject to optimization. Parameters that are not subject to optimization can easily be set fixed within *FALCON.m* using their `isFixed` properties.

Outputs

Besides the state derivatives that every simulation model has to deliver, each of them may also calculate additional model outputs that can later be used within the *cost* or *constraint* calculations. Every model output needs to be specified using an object of type `falcon.Output`. The concept of model outputs has mainly been introduced in *FALCON.m* in order to reduce computational overhead, as in many cases intermediate values appearing during the calculation of state derivatives in the dynamic model, are also required in the calculations of cost or constraint values. By defining these quantities as model outputs their values are made available to other functions within *FALCON.m* without the need to recalculate them.

Contrary to states and controls, model outputs are not automatically constrained within *FALCON.m*. If a user requires a model output to be constrained, a path constraint function turning the `falcon.Output` into a `falcon.Constraint` is required.

Constraints

Objects of type `falcon.Constraint` represent values to be constrained in an optimal control problem. They are calculated by a path constraint function or a point constraint function which are described below. The constraint values are calculated from the states, the controls, the model outputs and the parameters within a problem and feature an upper and

a lower bound that is respected by the final solution of the problem. Similar to parameters and controls, constraints can be switched off, using the `isActive` property defined in the class. Only active constraints are respected when solving a problem, all others are ignored. Equality constraints can be specified by setting the lower and the upper bound of the respective constraint to the same value.

Cost

Objects of type `falcon.Cost` are very similar to those of type `falcon.Constraint`. Anyway, they specify an output of either a MAYER cost function or a LAGRANGE cost function and are not limited by an upper and a lower bound. In the final optimal control problem, all cost values are weighted using their respective scaling factor, summed up and then minimized (or maximized if explicitly configured so).

4.8.2 Structuring Elements

In the following, the concepts of different classes used to structure an optimal control problem within *FALCON.m* are described.

Problem

As mentioned before, the root object for every optimal control problem to be solved in *FALCON.m* is of type `falcon.Problem`. This object holds all phases, functions, the discretization method and other information required to represent an optimal control problem. It is handed over to an object of type `falcon.solver.Optimizer` in order to be numerically solved. Besides, it holds all information about a problem that is of general interest, like a list of all parameters appearing in the problem or the global sparsity pattern of the gradient matrix. Section 4.4 and figure 4.2 give a more detailed overview of the structure of the object.

Phase

Every optimal control problem in *FALCON.m* contains at least one object representing a *phase*, being of type `falcon.core.Phase`. Each phase uses a dynamic model and a state grid. Moreover, if required, one or more control grids, path constraints and LAGRANGE cost functions may be added to each phase.

Upper and lower boundaries for the state values at the initial and final times of a phase may be set using the methods `setInitialBoundaries` and `setFinalBoundaries`. In case the state histories need to be continuous from one phase to another, they can be connected using the `ConnectPhases` or `ConnectAllPhases` methods available in the superior `falcon.Problem` object.

In general, the duration of a phase is free and defined by the two `falcon.Parameter` objects `StartTime` and `FinalTime` stored in the phase. In most applications, the initial

time is fixed while the final time may either be free or fixed. Anyway, all combinations of free and fixed initial and final time are supported by *FALCON.m*.

Grid

All data appearing on a grid over time in an optimal control problem is stored in an object of type `falcon.core.Grid` in *FALCON.m*. The most important grids are *state grids*, *control grids*, *defect grids*, *model output grids* and *path constraint grids*.

The values stored in each grid are specified by the `falcon.core.OVC` objects stored in the field `DataTypes` of the grid. All values stored within one `falcon.core.Grid` object need to be calculated at the same points in time. The grid objects take care of storing and calculating all required indices as well as any data interpolation whenever required.

Model

All required information concerning the simulation model of one phase is stored in a class of type `falcon.core.Model`. This information includes a *function handle* to the dynamic equations, calculating the state derivatives with respect to the independent variable (time), as well as the required gradients with respect to the model inputs. The function is normally created using the automatic differentiation and code generation capabilities of the *Model Builder* of *FALCON.m* as described in sections 4.5 and 4.8.4. The model outputs calculated by each model are stored in the field `ModelOutputGrid` of the `falcon.core.Model` object.

Besides the states and controls, the differential equation of the simulation model may depend on a set of parameters and fixed constants. As not all parameters appearing in the superior `falcon.Problem` may be relevant for all simulation models, each `falcon.core.Model` holds a list of relevant parameters and a list of constants required to evaluate the dynamics.

Path Function

Objects of type `falcon.core.PathFunction` are used in *FALCON.m* to represent path constraint functions that calculate constraint values required to be within their boundaries along the whole trajectory of a phase. Therefore, every `falcon.core.Phase` features the list `PathConstraintFunctions` holding these objects. The `OutputGrid` of a path function is not required to be equal to the state grid but may be any subset thereof.

The `falcon.core.PathFunction` class holds a *function handle* pointing to a *MATLAB* function that calculates the constraint values as well as their gradients with respect to the function inputs. Typically, this function is created using the automatic differentiation capabilities of the *Model Builder* of *FALCON.m* (section 4.8.4). The calculated constraint values are stored in the `OutputGrid` of the path function object. Similar to the dynamic models, path functions may depend on parameters or may require additional constants, that are stored in the object.

Point Function

Contrary to the path constraint functions, `falcon.core.PointFunction` objects represent functions calculating `falcon.Constraint` values that do not need to be within their limits along a complete phase but only at particular points. Moreover, point functions may use any parameter available in the problem as well as any combination of state, control, and model output values from different phases for calculating their constraint values. As point functions consequently cannot be allocated to a phase, they are listed in the field `PointConstraintFunctions` of the problem object.

Besides, `falcon.core.PointFunction` objects can be used to represent MAYER cost functions that are added to the list of `MayerCostFunctions` within the problem. In this case, the output of the function is not of type `falcon.Constraint` but of type `falcon.Cost`.

Lagrange Function

Objects of type `falcon.core.LagrangeFunction` are very similar to the objects of type `falcon.core.PathFunction`, described above. They are used to model a cost function to be integrated along the trajectory of a phase. The main difference regarding their implementation is the fact that instead of only evaluating the user defined function like the path functions do, this class evaluates the function *and* performs the numerical integration of the values.

Additionally, in case a vector valued output of the function is specified, or the user even uses several MAYER and LAGRANGE cost functions within a problem, all values are summed up.

4.8.3 Solution Methods

As described above, the solution of an optimal control problem can be divided into several steps within *FALCON.m*. The following subsections list the main software components involved in the solution of an optimal control problem after it has been fully specified.

Discretization Method

Every `falcon.Problem` contains the field `DiscretizationMethod` that holds a handle to an object of type `falcon.discretization.DiscretizationMethod`. At the moment, three implementations of this abstract class exist, being

- `falcon.discretization.Trapezoidal`, representing a *trapezoidal collocation scheme*,
- `falcon.discretization.BackwardEuler`, representing a *Backward* EULER *collocation scheme* and

- `falcon.discretization.MixedShooting`, representing the mixed collocation and shooting approach presented in chapter 8.

Figure 4.5 shows the abstract base class and the currently available implementations – the three dots illustrate that the object-oriented approach allows for an easy extension of *FALCON.m* to support other discretization methods by deriving additional classes. When defining a problem, the user may specify the discretization method to be used. In case no method is actively selected, *FALCON.m* automatically uses *Trapezoidal Collocation*.

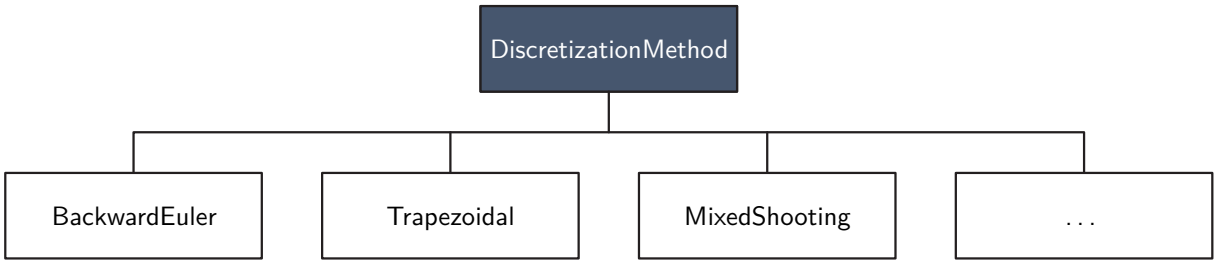

Figure 4.5: *Numerical discretization methods currently available in* FALCON.m.

All discretization method classes include the functionality required to perform the numerical discretization of the previously defined optimal control problem. This mainly includes the numerical integration or collocation scheme, using the model dynamics as inputs and supplying the resulting state defects. The most important methods implemented in these classes are the function `OptiFunc`, which is the function called by the numerical optimization algorithm in each iteration, and the function `Evaluate`, which calculates the state defects for each phase.

First, `OptiFunc` extracts all values from the optimization parameter vector, un-scales them, and distributes them to the objects within the problem definition. Afterwards, the control grid interpolations are updated and the simulation models of all phases are evaluated. Now, having the state derivatives available, the collocation or integration defects, respectively, can be calculated using the `Evaluate` method of the discretization scheme. In the next step, the constraint and cost functions are evaluated, writing their values and their gradients to the problem, and to the constraint vector and the gradient matrix that are handed back to the numerical solver. In the last step, the numerical scaling is applied to the gradient matrix and the constraint vector.

Besides the calculations described above, all child classes of `DiscretizationMethod` need to calculate the indexing and the sparsity pattern for the respective method, as the number of inputs and outputs as well as the sparsity pattern of the gradient may change depending on the method.

Numerical Solver

FALCON.m does not implement its own NLP-solver for determining the solution of the constrained numerical optimization problem that results after discretizing the optimal control problem. Instead, the class `falcon.solver.Optimizer` features a flexible interface allowing the use of many off-the-shelf tools for this purpose. In the current version, *FALCON.m* comes with an implementation supporting the open source interior point method *IPOPT*, that can be found in `falcon.solver.ipopt`, and one allowing for the use of *SNOPT* that is implemented in `falcon.solver.snopt`. Figure 4.6 shows the base class `falcon.solver.Optimizer` with its current implementations as well as a placeholder box for future extensions.

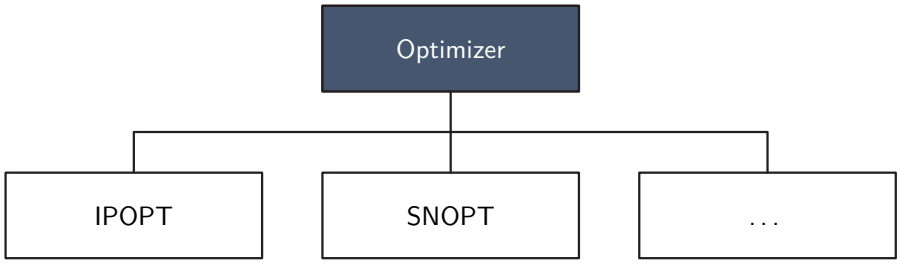

Figure 4.6: *Numerical optimization algorithms available in* FALCON.m.

4.8.4 Model Builder

Within *FALCON.m*, all functions required to evaluate the dynamic model, the constraints and the cost (as listed above) need to supply gradient information. In order to avoid the necessity for the user to implement the JACOBIANS by hand, the so-called *Model Builder* has been introduced in the software. The general scheme used in the *Model Builder* is described in section 4.5, while in the following, the different components of the software will be considered. The *Model Builder* functionalities of *FALCON.m* have mainly been developed and implemented by MATTHIAS RIECK and are described in the publication [Rie+15].

Base Builder

The abstract class `falcon.core.builder.BaseBuilder` represents the root of the *Model Builder*, as all other classes used to construct functions in a problem are either derived from this class or used by this class. This structure avoids code duplication and facilitates maintenance. Currently, three implementations exist, being

- `falcon.SimulationModelBuilder`,
- `falcon.PointConstraintBuilder` and
- `falcon.PathConstraintBuilder`.

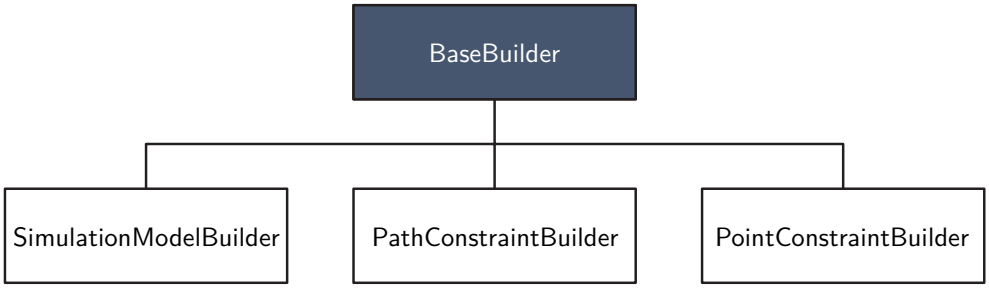

Figure 4.7: *The BaseBuilder class and its child classes.*

As the names suggest, they are used to build the functions required in different places of the optimal control problem. As from a structural point of view, LAGRANGE cost functions are equivalent to path constraint functions and MAYER cost functions are equivalent to point functions, the respective builder classes are used there. Figure 4.7 visualizes the `BaseBuilder` class and its child classes.

Derivative Builder

The creation of the code evaluating the *Jacobian* of a function is outsourced from the code in `BaseBuilder` to the abstract class `falcon.core.builder.DerivativeBuilder`. In the current version of *FALCON.m* the two implementations

- `falcon.core.builder.AnalyticDerivative` and
- `falcon.core.builder.FiniteDifferenceDerivative`

of this abstract class exist. The definition of a common interface for the functionality of providing gradient code offers the possibility to use the *Model Builder* even when no analytic differentiation is possible. This may happen due to the model equations or due to the fact, that the required *MATLAB* toolboxes are not available.

Within *FALCON.m*, by default, the `AnalyticDerivative` implementation is used that creates the derivative code using the *Symbolic Math Toolbox* of *MATLAB*. Anyway, in case a problem occurs or the toolbox is not available, the code falls back to the implementation in `FiniteDifferenceDerivative`. Besides, the user can actively choose to use finite differencing. The left part of figure 4.8 shows the relationship between the `DerivativeBuilder` class and its children.

Derivative Evaluator

Similarly to before, the functionality generating the final code as described in section 4.5 is extracted from the `BaseBuilder` class and is implemented in different child classes derived

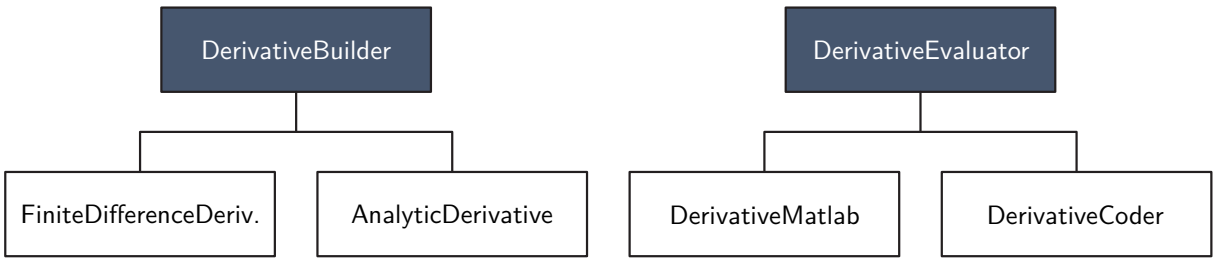

Figure 4.8: *Left: The DerivativeBuilder class and its child classes. Right: The DerivativeEvaluator class and its child classes.*

from the abstract class `falcon.core.builder.DerivativeEvaluator`. The full feature set – using *MATLAB coder* and *OpenMP* – is implemented in

- `falcon.core.builder.DerivativeCoder`, while
- `falcon.core.builder.DerivativeMatlab`

offers some sort of compatibility mode, in case `DerivativeCoder` fails. The latter combines all calculations for a function and its gradients into one common *MATLAB* function (.m-File), without further code generation and compilation. Besides, this approach does not support any multi-threading and consequently runs considerably slower. The relationship between all `DerivativeEvaluator` implementations can be seen in the right part of figure 4.8.

Chapter 5

Improving Robustness by Initial Guess Generation

Any direct solution method for solving optimal control problems, like the ones used in this work and described in section 3.2.4, require the utilization of numerical optimization algorithms such as those from section 3.1. Here, gradient based optimization methods are considered, all only being able to improve a previously found solution iteratively. This property requires the user of such optimization algorithms to specify a starting point for the optimization – the *initial guess*.

In general, the selection of this initial guess is crucial for the convergence properties of an optimization problem, both by means of convergence speed and convergence robustness, where the latter denotes if convergence can be achieved at all. Besides, the specification of an appropriate initial guess is usually not an easy task, as the structure of the solution of an optimal control problem is mostly unknown in advance.

In this chapter, different methods for the (at least partially) automated creation of improved initial guesses for optimal control problems and hence also aircraft trajectory optimization problems are described. In the first section, the idea of *homotopy* is presented, before being, on the one hand, extended to be used in multi aircraft optimization problems and, on the other hand, being improved by sensitivity updates. Second, a method for the creation of initial guesses for high fidelity aircraft trajectory optimization problems is presented that uses different modeling fidelities and a RAUCH-TUNG-STRIEBEL *(RTS)* smoother. Ideas similar to those presented there have already been published for the combination of a KALMAN filter and a *pseudospectral collocation scheme* in [BFH12]. Examples demonstrating the methods can be found in sections 11.6 and 11.7.

5.1 Initial Guess Generation Based on Homotopy

BULIRSCH and PESCH mentioned the general idea of *homotopy* within the framework of optimal control problems in their works [BMP91] and [Pes96] during the nineties. Therein,

they applied it in combination with a multiple shooting scheme which was used to solve the multi-point boundary value problem that results from the optimality conditions that can be derived using the indirect solution approach (see also section 3.2.3). They define a parametric optimal control problem and solve the problem for different values of this parameter – starting with a value that strongly simplifies the problem – until the particular problem, that was to be solved initially, is reached.

In [Fis11], FISCH applies a *homotopy* scheme in combination with a direct multiple shooting method to aircraft trajectory optimization problems focusing on air races. His exemplary problems are very similar to the ones presented in chapter 11. Therein, he stepwise moves the race gates to be passed by the pilots along the race track, in such a way that first a strongly simplified track is considered, before finally the real race track is modeled.

5.1.1 Basic Homotopy

In general, a *basic homotopy* process for optimal control problems works as follows:

1. First, a parameter is identified that may be changed in order to simplify the problem to be solved. The parameter may influence the model, the cost function, the constraints or any combination thereof. In the example presented in section 11.6 the orientation of a race gate, while in the example in section 10.2 the number of aircraft in the scenario is used for this purpose.
2. Next, the solution process is started by altering the parameter value and thus solving a simplified problem. Depending on the parameter, this simplification may be the relaxation of some constraints, simplification of the model or the cost, removing minor aspects of the problem completely, or reducing the problem size and complexity. In the example of section 11.6, the race gate at the center is rotated such that the trajectory through the race track is simplified, while in the example of section 10.2 the trajectory of each aircraft is optimized individually.
3. Afterwards, the problem complexity is stepwise increased again, always using the solution of the previous problem as the initial guess for the consecutive one. For the problem in section 11.6 this means that the race gate is rotated stepwise until it reaches its final orientation and position. More details on how this step is accomplished in the example from section 10.2 can be found in section 5.1.2.
4. Finally, the originally posed problem can be solved based on the solution of the last problem in the homotopy process.

The left diagram in figure 5.1 illustrates the process described above.

Additionally, the solution accuracy that is required when solving the intermediate optimal control problems in the second step of the process does not necessarily need to be as high as the final accuracy. Usually, it is sufficient to solve the problems to a rather rough accuracy which may be two to three orders of magnitude below the finally required accuracy. Anyway,

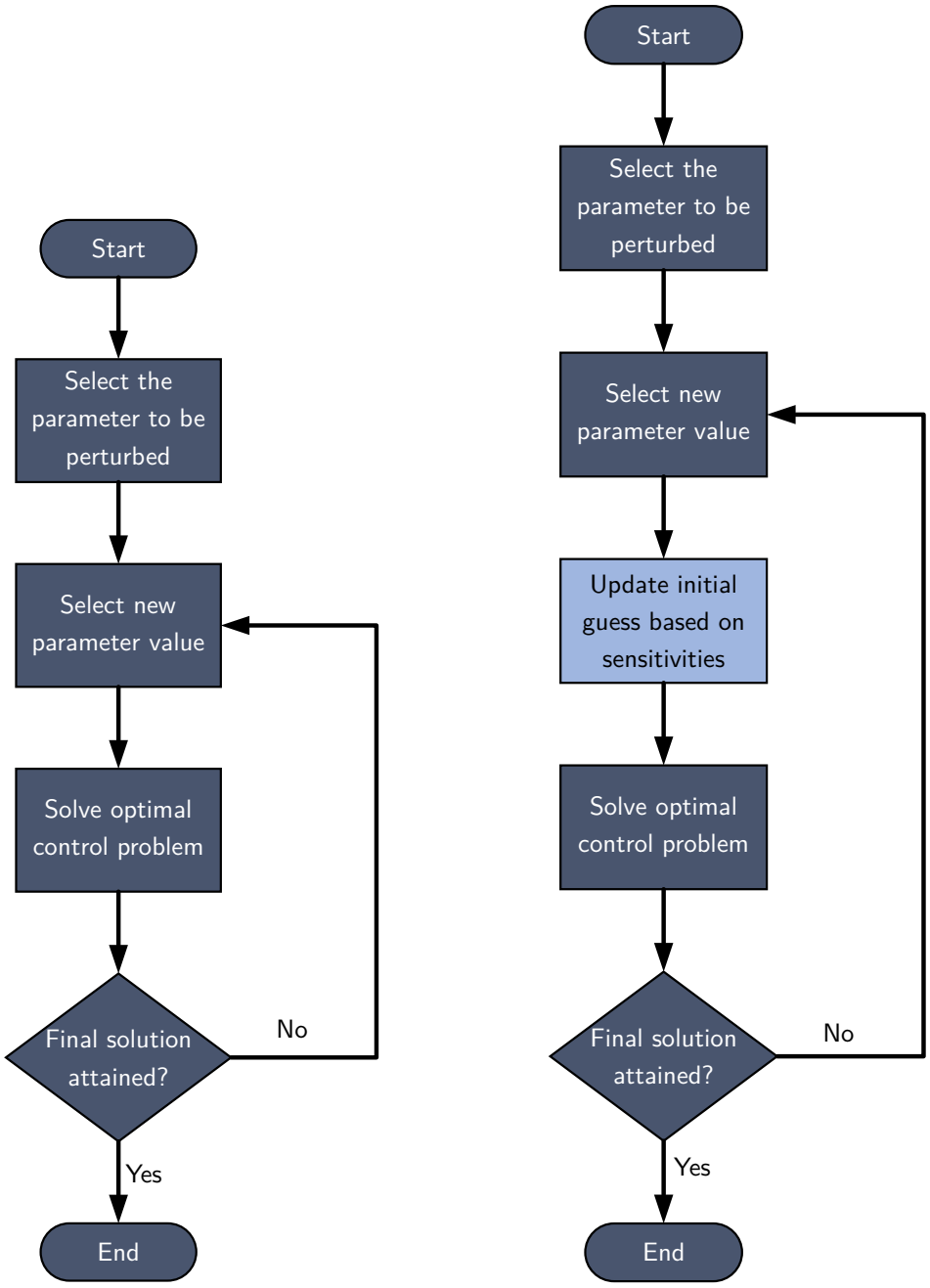

Figure 5.1: *Left: Basic homotopy principle. Right: Extended homotopy principle (see also section 5.1.3).*

in some cases a faster and more robust solution process may be achieved by tightening the accuracy of the intermediate problems.

In the third step of the process – depending on the interface to the numerical solver utilized – one may use different sets of information from one problem as initial guess for the consecutive one. First of all, of course, the optimization parameter vector $\mathbf{z}$ is transferred from one problem to the next. Besides, in some cases, the multipliers $\boldsymbol{\lambda}$ and $\boldsymbol{\nu}$ of the constraints in the problem may also be supplied to the numerical solver. Unfortunately, no general rule

could be determined when the initial guess for the multipliers is helpful and when it is not, based on the results of this work.

Moreover, *grid refinement* – like the approach presented in chapter 9 – may also be seen as a variant of homotopy using the number of grid points as the homotopy parameter. Therein, a problem is solved using a small number of grid points, resulting in less accurate results but a faster and more reliable solution process. Based on a first solution, the state and control histories as well as all parameters appearing in the problem can be extracted. In the next step, the grid is refined while the required initial guess can be calculated by interpolation.

5.1.2 Homotopy for Multi Aircraft Trajectory Optimization Problems

The homotopy idea may also be used with problems that comprise several individual systems, like the multi aircraft problem presented in section 10.2. In problems of this kind, in the first step, an individual optimal control problem for each of the systems can be solved, neglecting the cross-influences between the systems. This way, optimal trajectories – already considering most constraints and the cost function – for each system can be generated. This first step is visualized in the top row of figure 5.2.

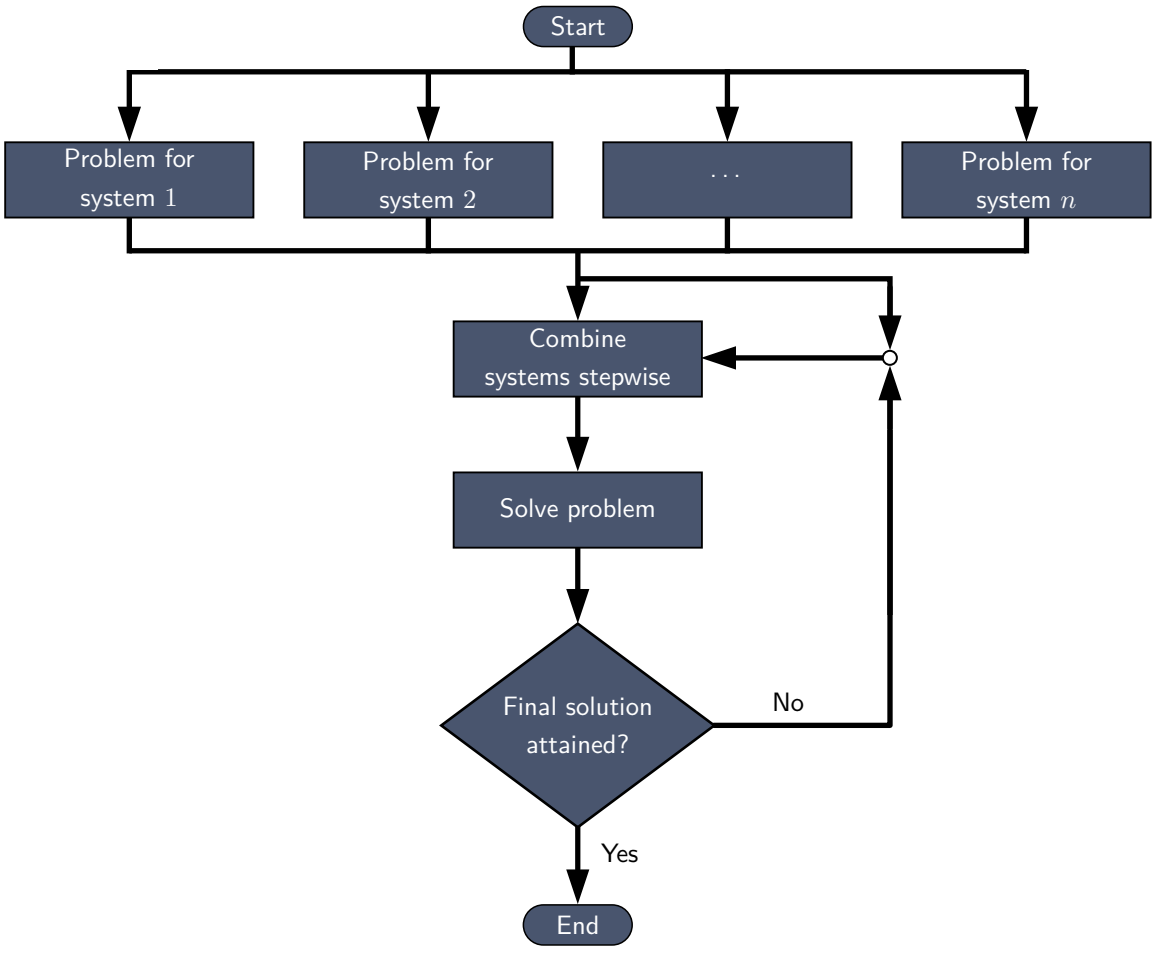

Figure 5.2: *Homotopy scheme for multi aircraft optimization problems.*

In the iterative part of this homotopy approach, the problem size is increased stepwise by adding one system after the other to the optimal control problem. In the multi aircraft problem described in section 10.2, the aircraft are added according to their entry times in the considered airspace, starting with the first arrival. In general, the order of adding the systems to the problem needs to be selected by the user and may be crucial to the process. Again, no general recommendation can be given. Next, similar to the basic homotopy approach described above, each intermediate problem is solved, considering the cross-influences of the different systems. As long as the final number of systems in the problem is not yet reached, the solution for the system to be added is taken from the list of individual solutions and is added to the overall problem. The process stops, when all aircraft are considered and the final solution is attained. The lower part of figure 5.2 visualizes the procedure.

Similar to above, only the optimization parameter vector $\mathbf{z}$ may be supplied from one problem to the next, or the parameters and the multipliers $\boldsymbol{\lambda}$, $\boldsymbol{\nu}$ may be used. In the example in section 10.2, all parameters and the multipliers are used, achieving good results. This is presumably because the trajectories of the individual systems only slightly change from one problem to the next. It has to be noted that the transformation of the optimization parameters as well as the multipliers has to be performed based on the discretized optimal control problem and may not directly be done based on the numerical optimization problem, as the structure of the optimal control problems change quite significantly when adding additional systems between two homotopy iterations. Like before, the solution accuracies of the intermediate problems can be chosen to be less restrictive than those in the final problem.

5.1.3 Extended Homotopy Using Sensitivity Updates

The basic homotopy process described in section 5.1.1 can be further improved based on the sensitivity information of the optimization problem calculated using the equations from section 3.1.6. If continuous parameters are considered during the homotopy process, equation (3.57) can deliver sensitivities for a first order update of the solution of the problem to be solved next in $\mathbf{z}_{upd}$, $\boldsymbol{\lambda}_{upd}$, and $\boldsymbol{\nu}_{upd}$ based on the current solution $\mathbf{z}$, $\boldsymbol{\lambda}$, and $\boldsymbol{\nu}$:

$$\mathbf{z}_{upd} \approx \mathbf{z} + \frac{d\mathbf{z}}{d\mathbf{p}} \cdot (\mathbf{p}_{upd} - \mathbf{p}) \tag{5.1}$$

$$\boldsymbol{\lambda}_{upd} \approx \boldsymbol{\lambda} + \frac{d\boldsymbol{\lambda}}{d\mathbf{p}} \cdot (\mathbf{p}_{upd} - \mathbf{p}) \tag{5.2}$$

$$\boldsymbol{\nu}_{upd} \approx \boldsymbol{\nu} + \frac{d\boldsymbol{\nu}}{d\mathbf{p}} \cdot (\mathbf{p}_{upd} - \mathbf{p}) \tag{5.3}$$

Therein, $\mathbf{p}$ is the homotopy parameter vector of the current problem, while $\mathbf{p}_{upd}$ is the one of the subsequent problem. In this approach, each change in the homotopy parameter may be seen as a disturbance of the last calculated solution. In case discrete parameters are used for the homotopy process – like the number of systems in the problem, the number of discretization points, or the model fidelity – no sensitivities can be calculated, making the proposed sensitivity based homotopy update impossible.

Besides, the step size chosen for the parameter update may not be too large, as otherwise the required smoothness assumptions for equation (3.57) are not fulfilled anymore. The price to pay for the sensitivity based update is the necessity to evaluate the *Hessian* of the problem and to solve the linear system stated in the aforementioned equation. Anyway, when using automatic differentiation, no additional user input is required for the *Hessian* evaluation and the update may be performed automatically. Moreover, the numerical calculation of the *Hessian* is only required for the final solution of each problem and not during the iterations of the numerical solver of the optimization problems.

Similar to above, the solution accuracy for the intermediate problems that need to be solved, has to be chosen with care. In the proposed extended homotopy scheme, this is even more important, as an accuracy of the LAGRANGE multipliers that is too low can add significant inaccuracies to the sensitivity equation (3.57). As a consequence, the linear system may not be solved accordingly, reducing the quality of the updated solution.

Moreover, the numerical solution of the linear system in equation (3.57) may fail, either because the matrix is really singular at the considered point, or because it is of bad numerical condition. The latter may also be influenced by the solution accuracy that is chosen for the intermediate optimization problems. If this happens, the proposed algorithm is falling back to the basic homotopy scheme, using the previously generated solution as the initial guess for the next problem directly. The right diagram in figure 5.1 illustrates the extended process with the additional step added. An example application, demonstrating how the extended homotopy process can accelerate the solution of an air race trajectory optimization problem, can be found in section 11.6.

The idea of using sensitivities to update the solution of an optimal control problem has also been used before for real time optimal control and non-linear model predictive control. Besides others, GERDTS gives a detailed overview of the idea and the algorithms that may be used within this context in [Ger12, p. 293 ff.]. Furthermore, BÜSKENS uses sensitivity analysis for real time optimal control in [Büs98].

5.2 Initial Guess Generation Using Rauch-Tung-Striebel Smoothing

When considering the use of dynamic models of different fidelities for the solution of an optimal control problem, these models may also be used for the determination of initial guesses. In this section, an approach will be presented that uses an extended RAUCH-TUNG-STRIEBEL *(RTS)* smoother in order to approximate the solution of a high fidelity optimal control problem based on a low fidelity solution for the same problem.

The process shown here can be summarized as follows: First, an initial guess for the problem based on the simple model is generated. Second, the optimal control problem based on this simple model is solved, before third, the determined optimal solution is transformed

into an initial guess for a high fidelity optimization using filtering or smoothing techniques. In this case a RAUCH-TUNG-STRIEBEL (*RTS*) smoother is used for this purpose. In the last step, the optimal trajectory for the problem using the high fidelity simulation model can be calculated. One of the required assumptions is that at least a part of the states of the simplified model appear in the more precise model, too.

The basic idea behind using the *RTS* smoother is to interpret the errors of the simplified model (compared to the trajectory of the high fidelity model) as stochastic noise processes. As the *RTS* smoother is an advancement of the well-known KALMAN filter, it is based on similar stochastic assumptions. These include that the dynamics of the model as well as the measurements are influenced by normally distributed, white noise. By using information generated by the simplified model as measurements for the high fidelity model, and by tuning the parameters of the distributions, good smoothing results can be achieved as can be seen in the example in section 11.7 (even though some assumptions are not strictly fulfilled). Another interpretation of the *RTS* smoother in this application is to see it as a controller that tries to keep the trajectory of the high fidelity model as close to the one of the simplified model as possible. For this, all available trajectory information is used and the parameters of the controller are tuned based on the stochastic ideas described before.

In [Fis11], FISCH uses models of different fidelity in order to solve a high fidelity air race trajectory optimization problem, too. In his work he utilizes control algorithms and simulations in order to generate initial guess for the next optimization from the results of the previous one. Besides, a similar idea has been published in [BFH12], before. In the latter, an *Iterated Extended* KALMAN *Filter* has been used in combination with a GAUSS *Pseudospectral Discretization Method* for solving another air race trajectory optimization problem.

5.2.1 Extended Rauch-Tung-Striebel Smoother

In classical estimation theory three main problem types can be distinguished:

- In *prediction* tasks a future state should be estimated based on past measurements.
- In *filtering* tasks the current state of a system should be estimated based on measurements up to the current time step,
- while in *smoothing* tasks the overall time history for a system should be estimated based on measurements along the same time range.

The task to be accomplished here is clearly of the last type, as between two optimization runs all information (system and measurements) is available and the estimation of all values can be performed offline, not posing any runtime requirements. The extended *RTS* smoother used here is an advancement of the classical KALMAN filter that has initially been developed for linear systems. Along this further development, two main improvements have been made, where the *smoother* applies a forward and a backward run to the data, improving the quality of the results by using all available information, while the *extended* version of a filter or

smoother can be used for non-linear dynamics. The extended variants are based on repeated linearization of the underlying dynamic system. Besides, *iterated extended* versions of many of the aforementioned algorithms exist, that iterate over the linearization of the output equations in order to further improve the update step. In the approach presented here, a subset of the states is considered as the model output resulting in linear output equations that remove the benefits of iterated filtering or smoothing techniques. More details on prediction, filtering, and smoothing can besides others be found in [Sim06, CJ12, HG16]. Additionally, GÖTTLICHER uses a similar smoothing approach in the paper [GH16].

In the underlying theory for the KALMAN filter – or any of its derivatives – the deterministic dynamics of the considered system are extended by a stochastic noise process $\bar{\mathbf{w}}$ of appropriate dimensions that can be interpreted as model inaccuracies:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) + \bar{\mathbf{\Gamma}}\bar{\mathbf{w}} \tag{5.4}$$

Here, $\bar{\mathbf{\Gamma}}$ is a constant mapping matrix of appropriate dimensions. Similarly, the output equations are extended by a noise process $\bar{\mathbf{v}}$, which is of the same size as the measurement vector:

$$\mathbf{y} = \mathbf{h}(\mathbf{x}, \mathbf{u}) + \bar{\mathbf{v}} \tag{5.5}$$

In these equations, $\bar{\mathbf{w}}$ and $\bar{\mathbf{v}}$ represent zero-mean, *Gaussian* white noise

$$\bar{\mathbf{w}} \sim (0, \bar{\mathbf{Q}}), \qquad \bar{\mathbf{v}} \sim (0, \bar{\mathbf{R}}) \tag{5.6}$$

with *power spectral densities* of $\bar{\mathbf{Q}}$ and $\bar{\mathbf{R}}$, respectively. The derivations for the smoothing algorithm used here have to be done in discretized time, requiring an explicit numerical integration algorithm (see section 3.2.4 for details) to be directly incorporated. The resulting discrete dynamics are of the form:

$$\mathbf{x}_k = \bar{\mathbf{f}}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) + \mathbf{\Gamma}_{k-1}\mathbf{w}_{k-1} \tag{5.7}$$

$$\mathbf{y}_k = \mathbf{h}(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{v}_k \tag{5.8}$$

Again, the noise processes $\mathbf{w}_k$ and $\mathbf{v}_k$ are zero-mean *Gaussian* with *power spectral densities* of $\mathbf{Q}_k$ and $\mathbf{R}_k$:

$$\mathbf{w}_k \sim (0, \mathbf{Q}_k), \qquad \mathbf{v}_k \sim (0, \mathbf{R}_k) \tag{5.9}$$

It has to be minded that the *covariance* of these noise processes is related to the *power spectral density* via the discretization time step. In the code that is used for performing the data smoothing, the *power spectral density* of the signals is set and transformed to the *covariance* accordingly. [Sim06, CJ12, HG16]

In the following, the double index notation is used to specify quantities appearing during the smoothing. The first index specifies the particular point in time the current value is related to, while the second index specifies the point in time to which the information that is used in the estimation is related to. Consequently, in the forward run of the smoother both indices are either equal or the second index is smaller than the first while in the backward run it is

the other way round. The hat indicates an estimated value while the upper index specifies the forward run f, or the backward run b.

As mentioned before, the smoother uses a repeated linearization of the system dynamics and the output equations around the currently available estimates. In order to simplify the notation, the following abbreviations are used:

$$\mathbf{A}_{k-1} = \left.\frac{\partial \bar{\mathbf{f}}(\mathbf{x}_k, \mathbf{u}_k)}{\partial \mathbf{x}_k}\right|_{\mathbf{x}_k = \hat{\mathbf{x}}_{k-1|k-1}} \tag{5.10}$$

$$\mathbf{C}_k = \left.\frac{\partial \mathbf{h}(\mathbf{x}_k, \mathbf{u}_k)}{\partial \mathbf{x_k}}\right|_{\mathbf{x}_k = \hat{\mathbf{x}}_{k|k-1}} \tag{5.11}$$

The forward run of the *RTS* smoother is initialized using the initial estimates for the states and the state covariance matrix $\mathbf{P}$:

$$\hat{\mathbf{x}}^f_{0|0} = \mathbf{x}_0 \tag{5.12}$$

$$\mathbf{P}^f_{0|0} = \mathbf{P}_0 \tag{5.13}$$

and is then performed according to the extended KALMAN filter that is performing the state estimation in an optimal way, meaning that it is minimizing the covariance matrix of the estimated states. First, the prediction equations need to be applied in every step:

$$\hat{\mathbf{x}}^f_{k|k-1} = \bar{\mathbf{f}}\left(\hat{\mathbf{x}}^f_{k-1|k-1}, \mathbf{u}_{k-1}\right) \tag{5.14}$$

$$\hat{\mathbf{y}}^f_{k|k-1} = \mathbf{h}\left(\hat{\mathbf{x}}^f_{k|k-1}, \mathbf{u}_k\right) \tag{5.15}$$

$$\mathbf{P}^f_{k|k-1} = \mathbf{A}_{k-1}\mathbf{P}^f_{k-1|k-1}\mathbf{A}^\intercal_{k-1} + \mathbf{\Gamma}_{k-1}\mathbf{Q}_{k-1}\mathbf{\Gamma}^\intercal_{k-1} \tag{5.16}$$

Next, the update and correction equations are evaluated:

$$\mathbf{K}^f_k = \mathbf{P}_{k|k-1}\mathbf{C}^\intercal_k\left(\mathbf{C}_k\mathbf{P}_{k|k-1}\mathbf{C}^\intercal_k + \mathbf{R}_k\right)^{-1} \tag{5.17}$$

$$\hat{\mathbf{x}}^f_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\left(\mathbf{y}_k - \hat{\mathbf{y}}_{k|k-1}\right) \tag{5.18}$$

$$\mathbf{P}^f_{k|k} = \left(\mathbf{I} - \mathbf{K}_k\mathbf{C}_k\right)\mathbf{P}_{k|k-1} \tag{5.19}$$

In order to ensure that past and future information is used in the estimation process, the forward filtering is combined with a backward filtering approach in a smoother. In a classical KALMAN smoother one would perform the forward filtering, then the backward filtering and finally combine both results. At this point, the *RTS* smoother features a great performance benefit as it reuses information from the forward run in a combined backward and smoothing run. The equations to be evaluated during the backward run are:

$$\mathbf{P}^b_k = \mathbf{P}^f_{k|k} + \mathbf{M}_k\left(\mathbf{P}^b_{k+1} - \mathbf{P}^f_{k+1|k}\right)\mathbf{M}^\intercal_k \tag{5.20}$$

$$\hat{\mathbf{x}}^b_k = \hat{\mathbf{x}}^f_{k|k} + \mathbf{M}_k\left(\hat{\mathbf{x}}^b_{k+1} - \hat{\mathbf{x}}^f_{k+1|k}\right) \tag{5.21}$$

$$\mathbf{M}_k = \mathbf{P}^f_{k|k}\mathbf{A}^\intercal_k\left(\mathbf{P}^f_{k+1|k}\right)^{-1} \tag{5.22}$$

Obviously, $\hat{\mathbf{x}}^f_{k|k}$, $\hat{\mathbf{x}}^f_{k+1|k}$, $\mathbf{P}^f_{k|k}$, $\mathbf{P}^f_{k+1|k}$, and $\mathbf{A}_k$ need to be saved during the forward run. [Sim06, CJ12, HG16]

5.2.2 Optimization Process

In order to determine initial guesses for high fidelity trajectory optimization problems using the approach based on the *RTS* smoother that is presented here, several steps have to be performed.

First, for the determination of an initial guess for the simplified model, a geometric approach or any other method (like the ones presented above) may be used. Assuming that the simplified problem can be solved relatively easy, this first initial guess may be very rough. In the example presented in section 11.7 a simple linear interpolation between the initial and the final values for the states are used as initial guess. [BFH12]

Second, this simplified optimal control problem needs to be solved. Here, special care has to be taken with respect to the chosen discretization grid to make sure that it is neither too sparse nor too dense. In the first case, the smoothing task that needs to be performed next may be infeasible. In the latter case, the solution of the simplified optimal control problem may be computationally expensive or even impossible. In some cases, an interpolation of the simplified results to a finer grid may help at this point.

In the next step, the result of the simplified solution – containing less states than required in the full model – needs to be "*extended*" to an initial guess for the high fidelity model. For this purpose, the *RTS* smoother presented before is utilized here. The underlying assumption is that at least some of the states $\mathbf{x}_S$ of the simplified model are also part of the states of the high fidelity model $\mathbf{x}_F$. Then, these states (and possibly also the controls of the simplified model $\mathbf{u}_S$) can be seen as the available measurements $\mathbf{y}$ for the high fidelity model. Consequently, in a first step these measurements need to be calculated from the available data:

$$\mathbf{y} = \mathbf{c}(\mathbf{x}_S, \mathbf{u}_S) = \mathbf{h}(\mathbf{x}_S, \mathbf{u}_S) \tag{5.23}$$

Without loss of generality, the function $\mathbf{c}$ can be chosen such that the measurements $\mathbf{y}$ are a subset of the state vector of the high fidelity model $\mathbf{x}_F$. In case the controls of the simplified model match these of the high fidelity model, the high fidelity model $\mathbf{f}_F(\mathbf{x}_F, \mathbf{u}_F)$ can directly be used in the smoother equations, using the previously determined optimal controls as inputs. If this is not the case, the high fidelity dynamic model and the state can be extended by some zero dynamics for the controls:

$$\begin{pmatrix} \dot{\mathbf{x}}_F \\ \dot{\mathbf{u}}_F \end{pmatrix} = \tilde{\mathbf{f}}_F = \begin{pmatrix} \mathbf{f}_F \\ 0 \end{pmatrix} \tag{5.24}$$

In this formulation, the control inputs are mainly driven by the process noise and can consequently be estimated by the smoother. To make sure that the estimation delivers the desired results, the *power spectral density* of the noise processes $\mathbf{w}$ and $\mathbf{v}$ added to the system dynamics and the output equations have to be set up accordingly. In general, a high process noise means that the dynamic model is not reliable, while high measurement noise indicates that the measured output histories are not reliable. Here, only guidelines can be provided, as – depending on the particular system – further tuning might be required. In the example

presented in section 11.7, the coefficients for the *power spectral density* matrices have been estimated based on the orders of magnitude of the values appearing in the problem as follows:

- The *power spectral density* of the process noise for the states that are also part of the measurement is calculated from their order of magnitude times 10^3. This way, a good following of the measurements can be achieved, even if the state dynamic equations slightly differ.
- The *power spectral density* of the process noise for the states that are not part of the measurement is calculated as their order of magnitude times 10^{-1} as for them the state dynamics give the best guidance on their history.
- The *power spectral density* of the process noise for states representing the controls to be estimated (featuring zero dynamics) is calculated from their order of magnitude times 10^6 as in this case the dynamic equations only have dummy character and the overall history should be covered by the noise.
- The *power spectral density* of the measurement noise for the outputs is calculated from the magnitude times 10^{-2} as they should be followed as closely as possible.
- The initial state *covariance* is estimated depending on the initial conditions of the optimal control problem.

One benefit of this process that is worth mentioning is that the gradient information that is required for the *RTS* smoother is also required for the optimization and hence available anyway. Consequently, no additional differentiation of the equations is required.

The process described here works very well for optimization methods using full discretization, because in this case the state histories appear as optimization parameters in the discretized problem directly. Therefore, they are important within the initial guess and need to be calculated as accurately as possible. In contrary to that, the process does not speed up the optimization using shooting methods (especially single shooting) because the state vector only needs to be estimated at the beginning of the segments there (in case of single shooting only once). For that reason a good initial estimate of the states is not very important when using shooting methods.

In the last step of the overall process, the results from the smoother can be used as an initial guess for the high fidelity optimal control problem. After solving this, the optimal state and control histories result. In section 11.7 an air race trajectory optimization example can be found, demonstrating the process using a simplified point mass simulation model and a high fidelity rigid body simulation model.

Additionally, in other applications, the values calculated by the smoother may directly be used to control the high fidelity system, depending on the requirements. In this case, solution times can potentially be significantly decreased as no large scale, high fidelity problem needs to be solved then.

Chapter 6

Improving Performance Based on Modeling Alternatives

When trying to solve optimal control problems, one always has to consider the required fidelity of the corresponding model. In case the model is not detailed enough, relevant effects cannot be reproduced. Otherwise, in case a too detailed model is used, the resulting optimal control problem may be highly non-linear and very large – consequently being hard to solve. In general, the model fidelity is especially relevant when considering trajectories including highly dynamic maneuvers, as otherwise simple models are normally sufficient and do not require much effort on adaptations. In the examples presented in chapter 11 this is especially true as in the air race related applications presented there, the aircraft are mainly operated at their limits.

Besides the pure model fidelity, that is for example the choice between a point mass simulation model, a rigid body simulation model, or one with even more detailed subsystem models, the selected way of modeling the dynamics of a system may strongly influence the solution process of the problem. At this point, the issues from section 3.2.6 become relevant. Therein, it is rehearsed that control variables that appear linearly in all relevant functions of a problem may not be determined using optimal control theory. This also holds for the discretized representation of the problem. The approach presented here first analyses the different available models with respect to their suitability for optimal control problems and then tries to overcome the issues found. It is specially tailored to aircraft models and consequently to aircraft trajectory optimization problems.

The modeling fidelity as well as the way a dynamic system is represented in the optimal control problem has been subject to research for several years now. In [CL98], a pure collocation approach is compared to differential inclusion that tries to remove as many controls from the optimal control problem as possible, replacing them by non-linear constraints in the state derivatives. The control values need to be calculated a posteriori in this approach. Anyway, the authors of the work suggest to not use differential inclusion instead of direct collocation, except for cases where the direct collocation shows bad convergence behavior. The approach presented here also tackles this issue and uses a similar idea.

Similarly, the work [KM03] concentrates on the solution of optimal control problems with linearly appearing controls that additionally feature a parameter dependency. The authors give second order sufficient conditions and present a sensitivity analysis – based on the results of Fiacco – for the optimal bang-bang controls that may appear in such problems. Anyway, the work does not give an overall solution to the issues appearing in combination with singular arcs and bang-bang controls.

In [Fis11] different aircraft models are described that are consecutively used to solve an optimal control problem. The main idea of the work is to start the solution process using a simple point mass simulation model, before adding more and more complex rotational dynamics. Between two consecutive optimizations, simulations are used to calculate the required initial guesses. The work does not focus on linearly appearing controls and overcomes any issues related to them by adding particular Lagrange cost functions that slightly alter the solution.

Another approach for handling the issues coming from linearly appearing controls and singular arcs is to introduce so-called *stages* for partitioning the problem over time, depending on the properties of the controls. Whenever a control value either gets saturated or leaves its limits, a new stage is introduced. The purpose of this is to get a representation of these switching points that is as accurate as possible. In order to be able to use such an approach without manual tuning, a structure detection algorithm is required. Such an algorithm is presented by Schlegel in [SM06a] and extended in [SM06b]. Therein, the authors use a three step procedure, starting with a first solution of the optimal control problem, before detecting the arcs in the solution. Finally, the problem is re-parametrized and solved again as a multistage problem. in [LZZ12] Li extends the idea from Schlegel and applies it to two example problems.

Maurer approaches the problem of bang-bang and singular controls in his work [MTF14]. Therein, the authors use *NUDOCCCS*, *AMPL (A Mathematical Programming Language)* and *IPOPT* to solve the respective problems. Besides, they introduce the points in time when the solution changes from a singular arc to bang-bang or vice versa into the problem formulation as individual parameters. Consequently, their approach is very similar to the multistage idea. Again, the goal of the approach is to be able to exactly hit the switching times in order to avoid chattering due to under- or overshoots.

The remainder of this chapter is structured as follows: First, the general ideas of the model analysis are presented, before the point mass simulation model that is used in the examples of chapter 11 is analyzed from an optimal control point of view. In the next section, the same analysis is performed for the rigid body simulation model, using the aerodynamics presented in section 11.1. Section 6.4 demonstrates a reformulation of the point mass simulation model that significantly improves the convergence properties of the optimization. In the last section of the chapter an approach for transcribing tabulated data by the use of multiple functions and smooth fading is shortly presented.

6.1 Analysis of Simulation Models for Optimal Control

Recalling the extended *Hamiltonian* of a constrained optimal control problem from equation (3.113):

$$\tilde{H}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}, \mathbf{u}) = L(\mathbf{x}, \mathbf{u}) + \boldsymbol{\lambda}^{\intercal}\mathbf{f}(\mathbf{x}, \mathbf{u}) + \boldsymbol{\mu}^{\intercal}\boldsymbol{S}(\mathbf{x}, \mathbf{u}) \tag{6.1}$$

and the optimality conditions from equation (3.115):

$$\tilde{H}'_{\mathbf{u}}(\mathbf{x}_{opt}, \boldsymbol{\lambda}_{opt}, \boldsymbol{\mu}_{opt}, \mathbf{u}_{opt}) = 0 \tag{6.2}$$

combined with the findings presented in section 3.2.6, one can see that the problem of singular controls may appear whenever the *Hessian* of the *Hamiltonian* is singular in at least one of the controls. This means that the *Hessian* of the LAGRANGE cost function, the *Hessian* of the dynamic equations and the *Hessian* of the path constraints may not all be singular in any of the controls in order to avoid these singularities in the solution. Furthermore, if the *Hessian* of the simulation model is singular, it may be impossible to determine the optimal controls even though a path constraint exists that is not singular in the controls, if this path constraint is (at least partially) inactive.

Whenever the control that appears singular in the models is at one of its boundaries, in most cases it is still possible to uniquely determine its value. For all controls that do only appear linearly and still are not at their limits, singular arcs appear that may be hard to calculate from a theoretical point of view. In practical applications, the convergence behavior of the resulting numerical optimization problem is poor in this cases and strong chattering appears in the solution. Depending on the dynamics, this chattering may also influence other states and controls and may lead to an overall noisy solution.

In minimum time optimal control problems, the cost function may be represented by a LAGRANGE cost function:

$$J = \int_{t_0}^{t_f} L(\mathbf{x}, \mathbf{u}) dt = \int_{t_0}^{t_f} 1 dt \tag{6.3}$$

Consequently, the LAGRANGE cost function appearing in the *Hamiltonian* of the problem does not depend on any of the controls:

$$L(\mathbf{x}, \mathbf{u}) = 1 \tag{6.4}$$

Hence, this cost function $L(\mathbf{x}, \mathbf{u})$ is singular in the controls and does not add control dependencies to the *Hessian* of the problem. Consequently, the issue mentioned above is not resolved by this LAGRANGE cost function.

6.2 Analysis of the Point Mass Simulation Model

Based on the theoretical findings, the *Hessian* of the point mass simulation model will be considered for this analysis, first. Figure 6.1 shows the sparsity patterns of the *Jacobian* and the *Hessian* of the augmented point mass simulation model (the regular point mass simulation

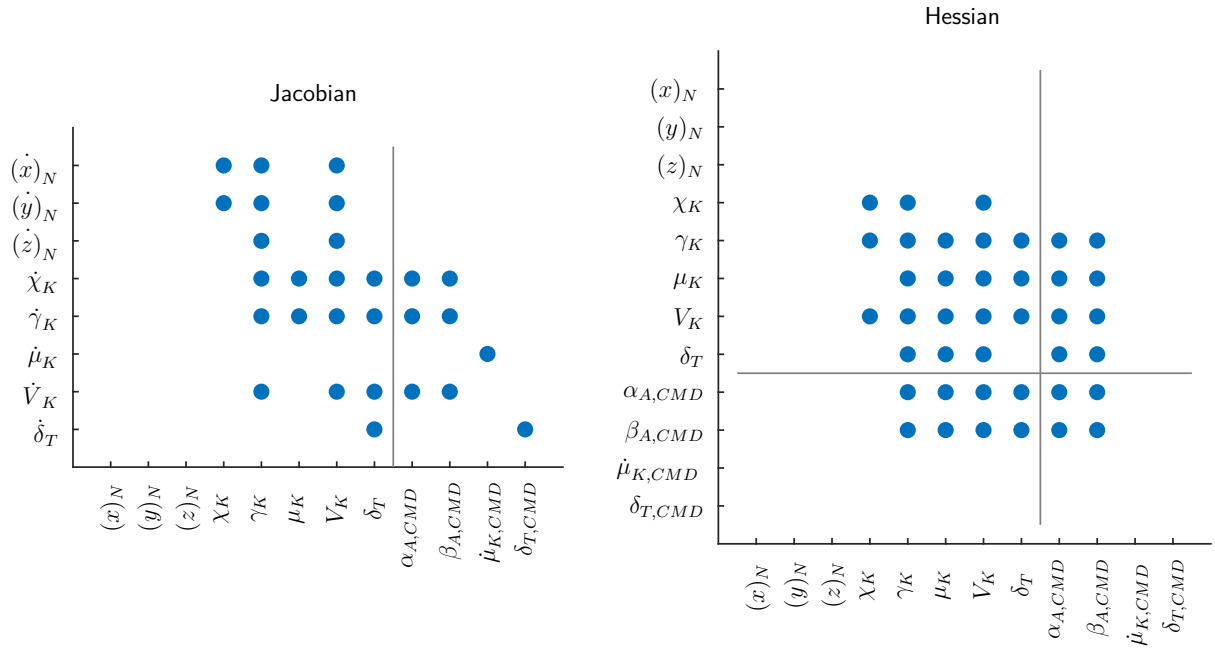

Figure 6.1: *Sparsity patterns of the* Jacobian *and the* Hessian *of the augmented point mass simulation model.*

model with additional, linear roll dynamics, see section 11.1.3) that is used in the air race examples from chapter 11.

It can clearly be seen that the *Jacobian* – left in figure 6.1 – has full rank with respect to the controls, implying that all of them influence the dynamics. However, when looking at the *Hessian* of the dynamic system – right in the figure – it can be seen that $\dot{\mu}_{K,CMD}$ and $\delta_{T,CMD}$ do not influence any of the second derivatives of the model dynamics, meaning that they enter the model linearly. The sparsity pattern for the *Hessian* has been created by summing up the sparsity patterns for all individual state derivatives. This is a valid representation as for this analysis it does not matter which controls influence which state derivatives.

Moreover, the *Jacobian* shows that the states $(x)_N$, $(y)_N$ and $(z)_N$ do not influence the dynamic equations of the system. Such states are sometimes called *ignorable variables* in optimal control theory. In the examples in chapter 11, it will be seen that the *costates* of *ignorable variables* are – in accordance with the theory – constant. [Tor06]

The problems solved in chapter 11 are, besides the state dynamics, constrained in the load factor in the $(z)_B$ direction. This load factor limit is represented by a path inequality constraint in the optimal control problem. Figure 6.2 depicts the sparsity patterns of the *Jacobian* and the *Hessian* of the constraint equation.

The figure shows that the relevant controls $\dot{\mu}_{K,CMD}$ and $\delta_{T,CMD}$ do not influence the path constraint. Consequently, the singularity in the *Hessian* of the *Hamiltonian* of the problem is not removed by the path constraint. It is important to note that an inequality path constraint can anyway only remove the singularity along segments of the trajectory where it is active.

Concluding the analysis in this section, it is expected to experience bad convergence properties and long convergence times for minimum time problems using the model and constraint

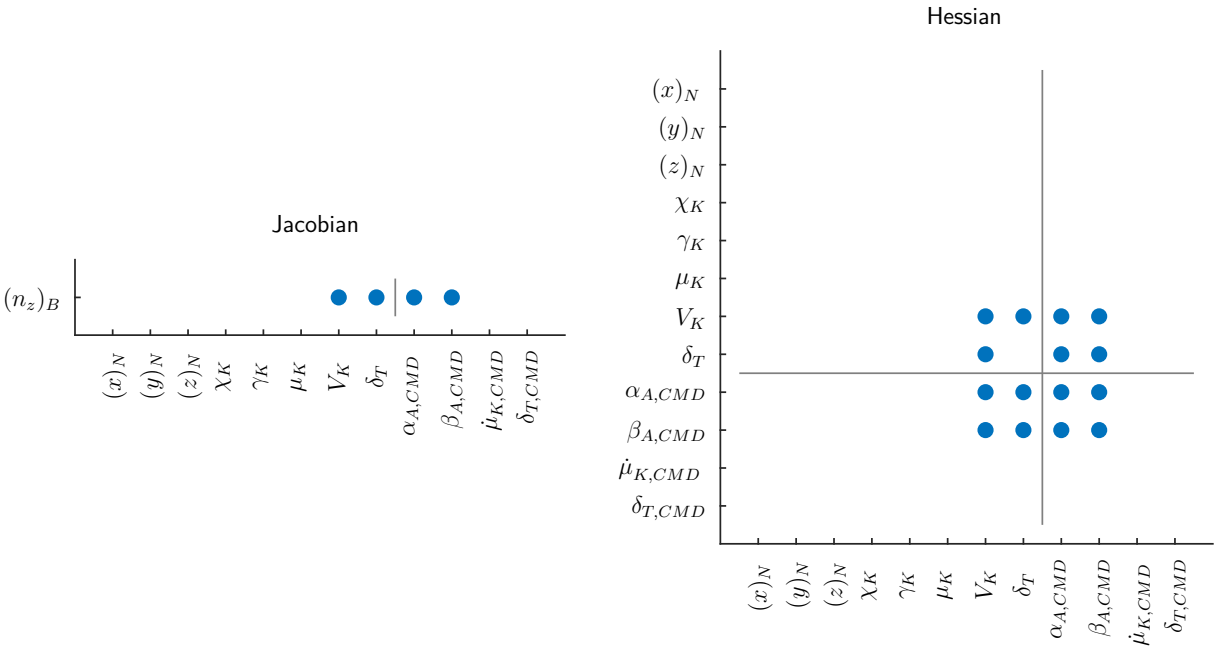

Figure 6.2: *Sparsity patterns of the* Jacobian *and the* Hessian *of the path constraint for the augmented point mass simulation model.*

formulation presented here. However, as minimum time trajectories require maximum speed, one can imagine that from a physical point of view the thrust lever position $\delta_{T,CMD}$ is expected to remain at its maximum most of the time. Consequently, singular arcs are mainly anticipated in the bank rate $\dot{\mu}_{K,CMD}$ which is likely not at its boundaries all the time. Matching results can be found in section 11.2.

6.3 Analysis of the Rigid Body Simulation Model

The analysis that has been performed for the point mass simulation model above can also be conducted for the rigid body simulation model. Of course, the same conditions have to hold in order to be able to construct an optimal control problem that shows good convergence behavior based on the model. Figure 6.3 shows the *Jacobian* of the rigid body simulation model on the left and its *Hessian* on the right. Again, it can be seen that all states – except the position states – and all controls influence the state derivatives. However, in this model, the thrust lever position is the only control that appears linearly in the dynamics. Using the same argumentation as above, no problems are anticipated from the linear thrust dynamics as the thrust lever is expected to remain in full throttle position almost all the time. Matching results can be found in section 11.3.

Figure 6.4 shows the sparsity patterns of the *Jacobian* and the *Hessian* of the load factor constraint for the rigid body simulation model. It can be seen that in this formulation the controls ξ_{CMD}, η_{CMD}, and ζ_{CMD} directly influence the path constraint non-linearly.

The convergence of an optimal control problem using a dynamic model whose *Hessian* has full rank in the controls may be bad even though the sparsity pattern looks promising.

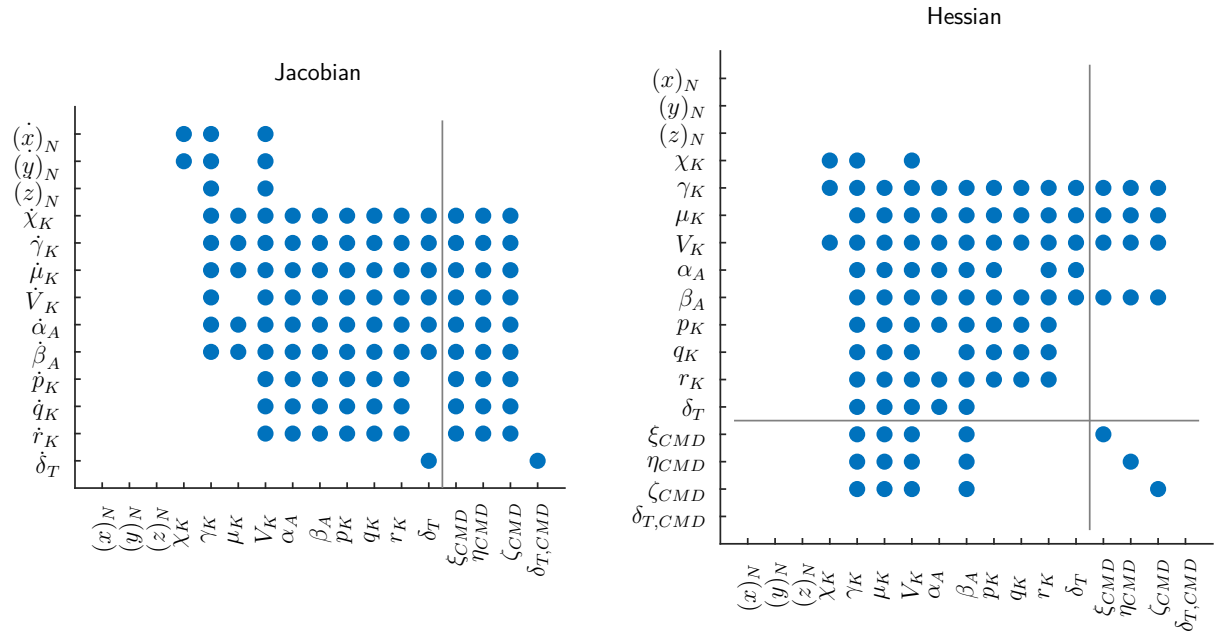

Figure 6.3: *Sparsity patterns of the* Jacobian *and the* Hessian *of the rigid body simulation model.*

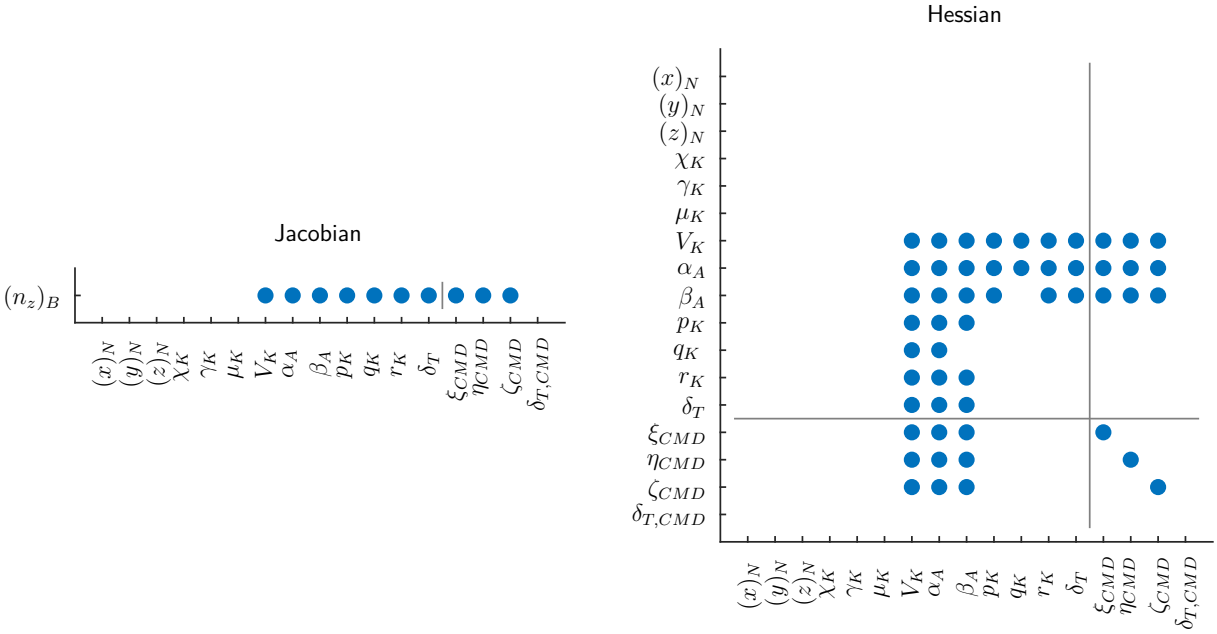

Figure 6.4: *Sparsity patterns of the* Jacobian *and the* Hessian *of the path constraint for the rigid body simulation model.*

The reason for this may be bad numerical conditioning of the *Hessian* resulting in a problem that is solvable from a theoretical point of view, but remains problematic numerically. The numerical conditioning of the matrix cannot be anticipated from the sparsity pattern but has been verified to be unproblematic in the model considered here. However, during the studies it turned out that the control surface drag is one critical factor for good numerical condition of the *Hessian*, and consequently important for good convergence. More details can be found in section 11.3.

6.4 Reformulation of the Point Mass Simulation Model to Avoid Linear Controls

Considering the issue of singular arcs in the augmented point mass simulation model, the most relevant control is $\dot{\mu}_{K,CMD}$. The only purpose of the linear roll dynamics

$$\dot{\mu}_K = \dot{\mu}_{K,CMD} \tag{6.5}$$

added to the model in section 11.1.3 is, to be able to limit the roll rate of the aircraft. When controlling the kinematic bank angle μ_K directly, it may jump, resulting in unrealistic optimal trajectories. Especially in the air race scenarios considered in the examples in chapter 11, where the aircraft have to fulfill bank angle constraints when passing the race gates, jumps in the bank angle are likely to appear in the solutions.

In [Bet09, p. 172], BETTS presents one approach to handle problems with rate constraints in the controls that is favorable to the approach discussed above, as it does not introduce linear controls. Even though, from a theoretical point of view, it may seem that there is no difference between adding a new state μ_K to the dynamics (equation (6.5)) and constraining the associated control

$$\dot{\mu}_{K,CMD,\min} \leq \dot{\mu}_{K,CMD} \leq \dot{\mu}_{K,CMD,\max} \tag{6.6}$$

or constraining the control derivative without adding a state equation

$$\left(\frac{d}{dt}\mu_{K,CMD}\right)_{\min} \leq \frac{d}{dt}\mu_{K,CMD} \leq \left(\frac{d}{dt}\mu_{K,CMD}\right)_{\max}, \tag{6.7}$$

for optimal control problems the two formulations are not equal.

When applying the approach presented by BETTS to the augmented point mass simulation model in the discretized problem, the control $\mu_{K,CMD}$ needs to be constrained by:

$$\left(\frac{d}{dt}\mu_{K,CMD}\right)_{\min} \leq \frac{\mu_{K,CMD}(t_{k+1}) - \mu_{K,CMD}(t_k)}{t_{k+1} - t_k} \leq \left(\frac{d}{dt}\mu_{K,CMD}\right)_{\max} \tag{6.8}$$

When doing so, the *Jacobian* and the *Hessian* of the dynamic model become even smaller than before due to the removed state. Figure 6.5 shows the sparsity patterns of the *Jacobian* and the *Hessian* of this model formulation. It can be seen that now only the thrust lever position appears linearly in the problem.

Similarly like before, figure 6.6 shows the sparsity patterns of the *Jacobian* and the *Hessian* of the load factor path constraint. Here, nothing (except the dimensions) has changed. Anyway, the constraint is not that important here, as the *Hessian* of the *Hamiltonian* is not singular in the control $\mu_{K,CMD}$ in this formulation, anyway.

The results in section 11.5 show that the numerical convergence of the optimal control problem is improved significantly compared to the naive approach of adding an additional state equation. Consequently, convergence time and the number of iterations required to solve the problems can be reduced.

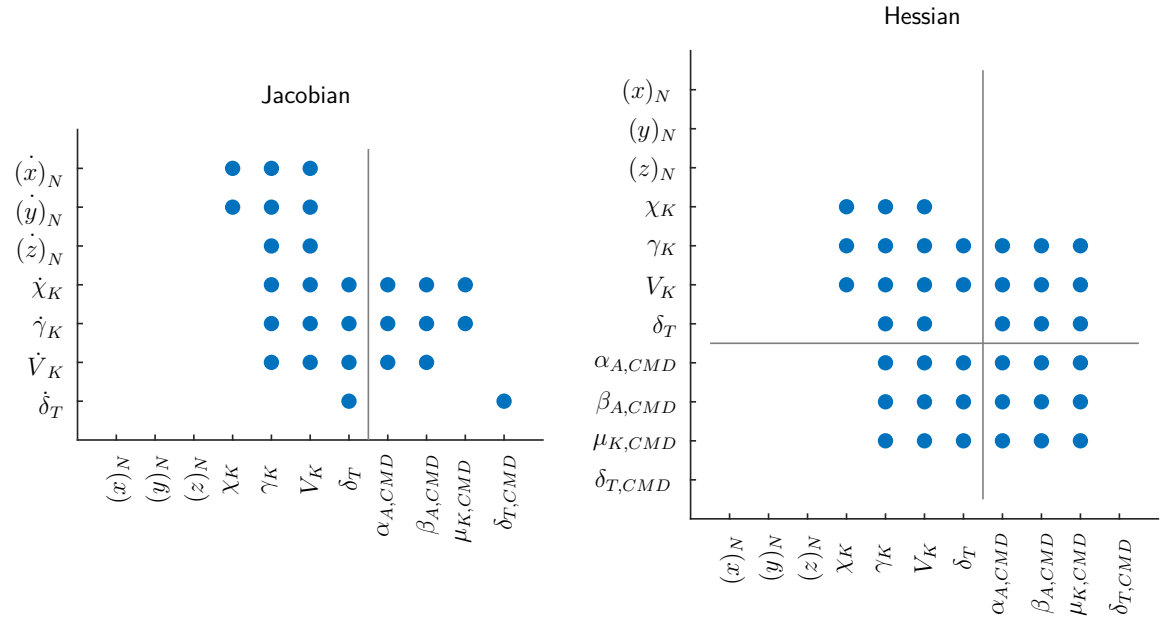

Figure 6.5: *Sparsity patterns of the Jacobian and Hessian of the basic point mass simulation model.*

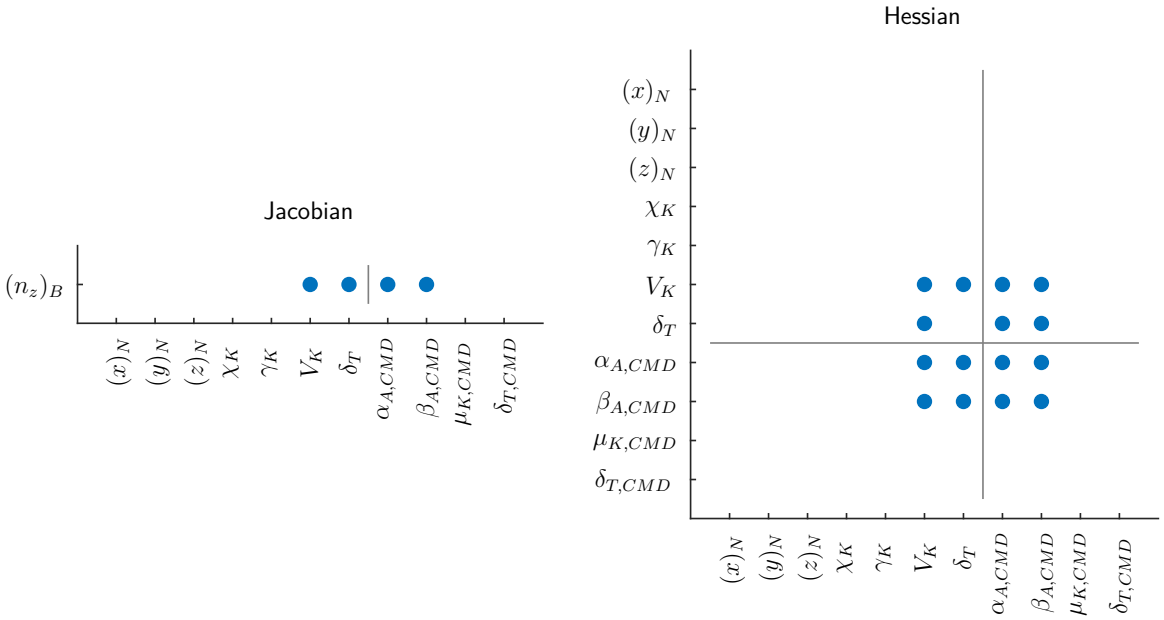

Figure 6.6: *Sparsity patterns of the Jacobian and Hessian of the path constraint for the basic point mass simulation model.*

6.5 Smooth Data Modeling Using Hyperbolic Tangent Functions

Whenever the models in an optimal control problem should represent a real system as close as possible, simple algebraic relationships are often not sufficient. Especially when the model has been determined empirically or by physical experiments, tabulated data results that needs to be incorporated in the dynamics, the constraints, or the cost function.

To be able to use gradient based optimization algorithms together with tabulated data, the data needs to be approximated by continuous and differentiable descriptions. In general, several approaches exist that may be used for the approximation or interpolation of such datasets. These methods include *Spline interpolation*, like e. g. presented in [Ger12, p. 222], where linear interpolation may be seen as a special case thereof, the *Shepard Interpolation* like in [She68, Ren88], and many more.

The basic idea of the approach presented here is to use hyperbolic tangent functions to fade between different piecewise base functions, and hence create a continuously differentiable function that may be used with gradient based algorithms. The general idea of the method is depicted in figure 6.7 where in the diagram at the top some data measurements are plotted

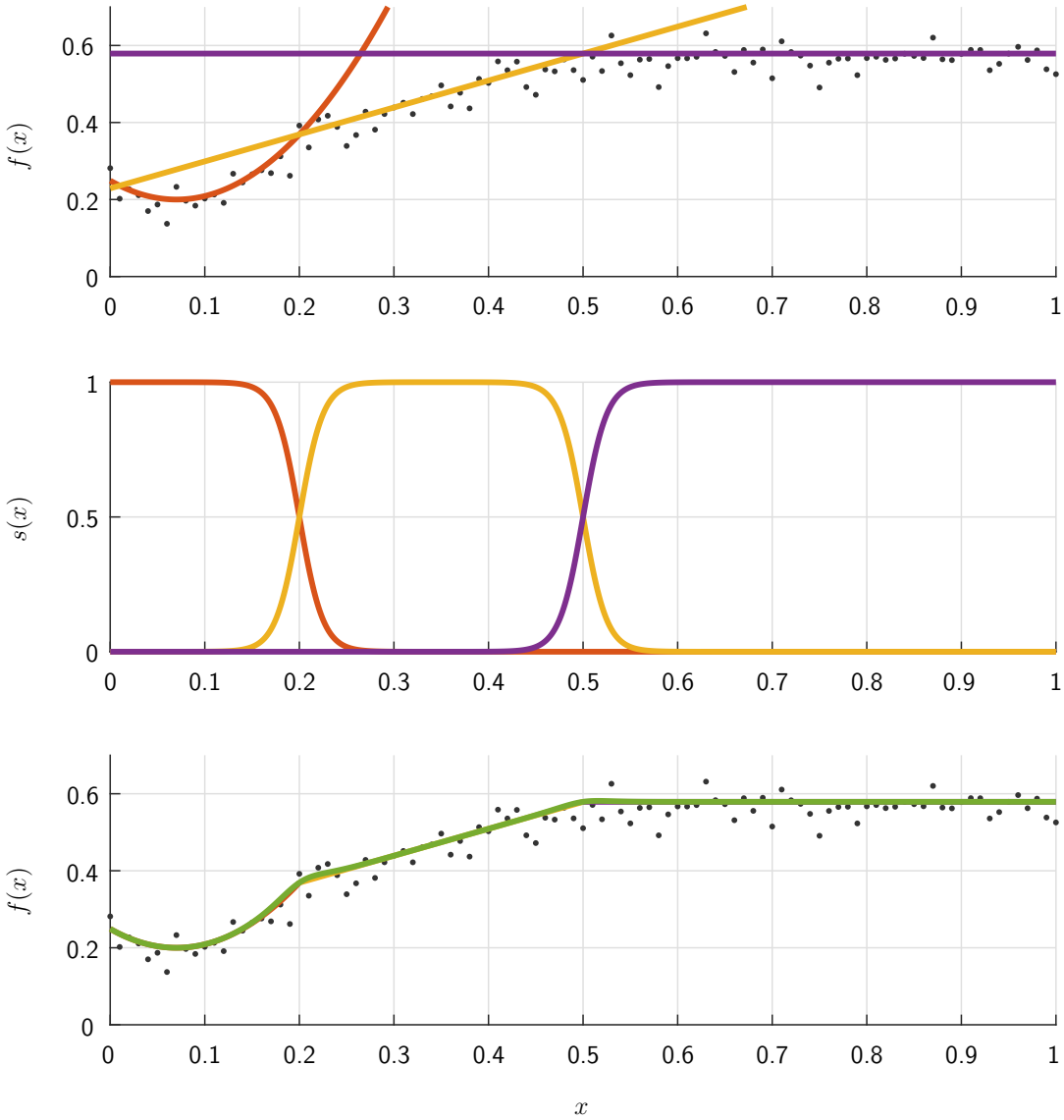

Figure 6.7: *Illustrative example for the hyperbolic tangent data modeling. The top graph shows the data measurements and the three base functions. The second plot shows the transition functions based on the* $\tanh$*. The third graph shows the combined result compared to the piecewise base functions.*

in gray. Now, the red, yellow, and purple lines have been identified as piecewise fits to the data, either manually by the user or by a fitting or approximation algorithm. These fits may be represented by the functions

$$f_1(x), \quad f_2(x), \quad \ldots, \quad f_n(x) \tag{6.9}$$

that are continuous and continuously differentiable. Furthermore, the transition from one function to the next takes place at the transition points:

$$x_1 < x_2 < \cdots < x_{n-1} \tag{6.10}$$

The base functions should be continuous at these transition points, requiring

$$f_1(x_1) = f_2(x_1), \quad f_2(x_2) = f_3(x_2), \quad \ldots, \quad f_{n-1}(x_{n-1}) = f_n(x_{n-1}) \tag{6.11}$$

to hold. In case these conditions are not fulfilled, the hyperbolic tangent functions will create more or less smooth transitions between the base functions, depending on the parameter l_k in equation (6.12). The fading from one function to the next is modeled using the transition function

$$s_k(x) = \left(\frac{1}{2} + \frac{1}{2} \cdot \tanh\left(l_k \cdot (x - x_{in,k})\right)\right) \cdot \left(\frac{1}{2} - \frac{1}{2} \cdot \tanh\left(l_k \cdot (x - x_{out,k})\right)\right) \tag{6.12}$$

where $x_{in,k}$ is the point when a function becomes active and $x_{out,k}$ is the point where it becomes inactive again. The steepness parameter l_k can be tuned for each application, where values too high result in sharp edges of the final function, while values too low smoothen out details. The second graph in figure 6.7 shows the transition functions that are used to fade from one base function to the next in the example.

When combining the different base functions and the transition functions, one overall function results:

$$\begin{aligned} f(x) = f_1(x) \cdot & \left(\frac{1}{2} - \frac{1}{2} \cdot \tanh\left(l_1 \cdot (x - x_1)\right)\right) \\ + f_2(x) \cdot & \left(\frac{1}{2} + \frac{1}{2} \cdot \tanh\left(l_2 \cdot (x - x_1)\right)\right) \cdot \left(\frac{1}{2} - \frac{1}{2} \cdot \tanh\left(l_2 \cdot (x - x_2)\right)\right) \\ + f_3(x) \cdot & \left(\frac{1}{2} + \frac{1}{2} \cdot \tanh\left(l_3 \cdot (x - x_2)\right)\right) \cdot \left(\frac{1}{2} - \frac{1}{2} \cdot \tanh\left(l_3 \cdot (x - x_3)\right)\right) \\ + & \ldots \\ + f_n(x) \cdot & \left(\frac{1}{2} + \frac{1}{2} \cdot \tanh\left(l_{n-1} \cdot (x - x_{n-1})\right)\right) \end{aligned} \tag{6.13}$$

Here, the fade-in term of the first and the fade-out term of the last function have been omitted for simplicity. The last graph in figure 6.7 shows the combined function as an overlay over the piecewise defined base functions. It can be seen that the approximation based on the hyperbolic tangent fits the piecewise functions quite closely, even at the transition points.

The main benefit of the resulting function is that it can be differentiated analytically with respect to the input parameter x. When using these functions in the dynamic model, the

constraints or the cost function of an optimal control problem, the analytic evaluation of the gradient remains possible. Figure 6.8 depicts the derivative of the function from the illustrative example. It can be seen that the function is continuously and smoothly differentiable and no large derivative values appear. All these properties are beneficial for the solution of optimal control problems.

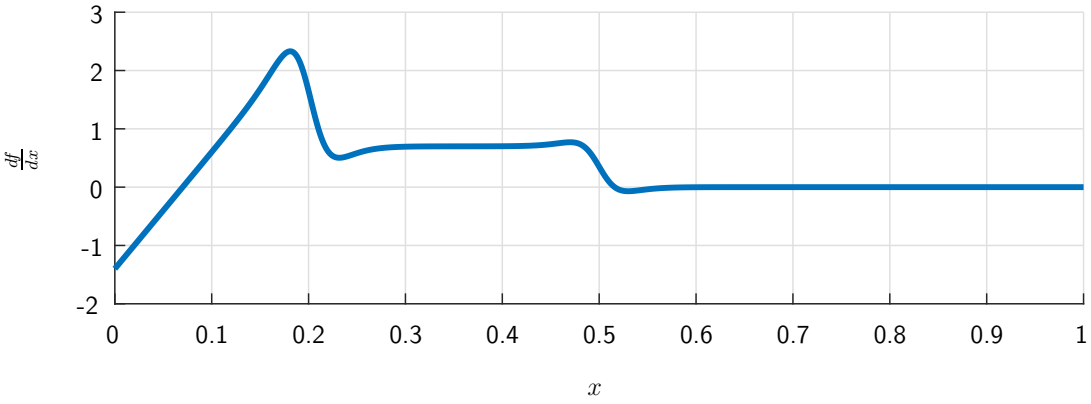

Figure 6.8: *Derivative of the hyperbolic tangent data modeling example.*

The aerodynamics of the example presented in section 11.1.1 use the idea presented here for modeling the dependency of the lift coefficient with respect to the angle of attack in equation (11.4). Therein two constant values are faded using hyperbolic tangent functions. In the example, the fading between the two functions is intentionally used for shaping the resulting relationship.

The principle presented here may be extended to multi dimensional data by using multi dimensional base functions and multi dimensional fading functions. For the multi dimensional fading functions, different methods may be applied:

On the one hand, the fading may be performed as a combination of fading functions in the different coordinate dimensions as

$$\begin{aligned} s_k(\mathbf{x}) = &\left(\frac{1}{2} + \frac{1}{2} \cdot \tanh(l_1 \cdot (x_1 - x_{1,k}))\right) \cdot \left(\frac{1}{2} - \frac{1}{2} \cdot \tanh(l_1 \cdot (x_1 - x_{1,k+1}))\right) \\ &\cdot \left(\frac{1}{2} + \frac{1}{2} \cdot \tanh(l_2 \cdot (x_2 - x_{2,k}))\right) \cdot \left(\frac{1}{2} - \frac{1}{2} \cdot \tanh(l_2 \cdot (x_1 - x_{2,k+1}))\right) \\ &\cdot \ldots \\ &\cdot \left(\frac{1}{2} + \frac{1}{2} \cdot \tanh(l_n \cdot (x_n - x_{n,k}))\right) \cdot \left(\frac{1}{2} - \frac{1}{2} \cdot \tanh(l_n \cdot (x_n - x_{n,k+1}))\right) \end{aligned} \tag{6.14}$$

where x_p represents the different coordinate dimensions of $\mathbf{x} \in \mathbb{R}^n$ and $x_{p,k}$ represent the respective transition points. Figure 6.9 shows an exemplary two dimensional switching function, partitioning the area $[0, 1] \times [0, 1]$ in five regions. Each of the regions may contain another base function.

On the other hand, the transition fading may be performed using a circular shape

$$s_k(\mathbf{x}) = \frac{1}{2} + \frac{1}{2} \cdot \tanh\left(l_k \cdot \|\mathbf{x} - \mathbf{x}_{c,k}\| - r_k\right) \tag{6.15}$$

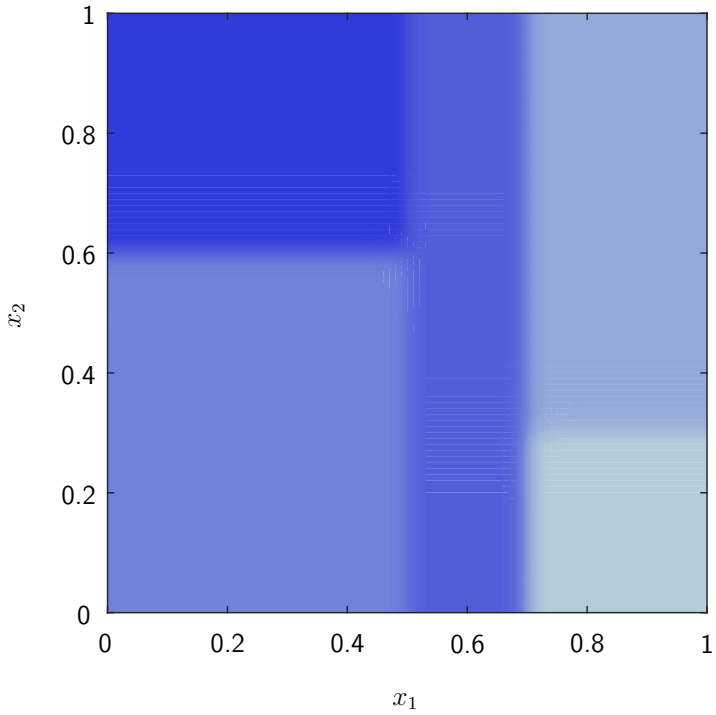

Figure 6.9: *Transition function for multi dimensional hyperbolic tangent modeling.*

based on the *Euclidean* distance, where l_k controls the steepness, $\mathbf{x}_{c,k}$ is the center, and r_k is the radius of the fading region. In this formulation, one base function should be added to the overall function, filling the *gaps* that may possibly exist between two other functions due to the circular fading shapes.

In order to best determine the base functions, the switching points and the steepness of the switching functions, an optimization problem may be set up, fitting the parameters of a user defined ansatz to the available data. For that, a least squares approach may be used in combination with a gradient based optimization approach.

Chapter 7

Improving Performance by the Exploitation of Problem Sparsity

As already mentioned in section 3.2.4, the *sparsity* of a discretized optimal control problem is of utmost importance for the solution process, when using direct discretization techniques. The main reason for this is that especially efficient algorithms for the solution of *sparse* numeric optimization problems exist that are also available in commercial solvers. In this context *sparse* problem means that the matrices characterizing the structure of the respective optimization problem – like the *Jacobian* and the *Hessian* – are *sparse*, implying that they contain a significant relative amount of structural zeros, compared to a small relative amount of non-zero entries. The *sparsity* of a problem is characterized by the pattern of zero and non-zero entries in the corresponding matrices, often described by logical matrices also called *sparsity pattern*. In the following, several generalized and exemplary sparsity patterns will be shown, always depicting non-zero entries as points, and leaving zero entries blank.

In this chapter, the properties of an optimal control problem are examined with respect to sparsity. The goal is to exploit this information to the maximum extent in order to improve the performance – and also the robustness – of the overall solution process. In this section, mostly the *sparsity* of the *Jacobian* of the problems is considered, as the structure of the *Hessian* is closely related to it, anyway. If one wants to exploit the sparsity of an optimal control problem, each aspect of the problem formulation, including the control parametrization, the model dynamics, the cost function, the constraints, the discretization scheme, and the numerical optimization method have to be minded. As shown in section 7.3, the discretization scheme – be it shooting or collocation – has a strong influence on the sparsity of the problem.

In [Bet09, p. 51 ff.], BETTS gives an overview of large and sparse non-linear programming. First, he defines large problems as those with dimensions larger than approximately 1000. Besides, he calls problems sparse if they feature less than 1% of non-zero entries in the *Jacobian* or the *Hessian*. The next issue considered in the book is the calculation of sparse *Jacobian* and *Hessian* information using finite differences. As *FALCON.m* uses analytic differentiation, this issue does not appear in the examples presented here. Anyway, sparsity information has to be calculated and supplied to the numerical solver, too.

The solution algorithms for large and sparse numerical optimization problems are discussed in the next sections of [Bet09, p. 54 ff.] in detail. Besides a description of algorithms for solving sparse QP subproblems, merit functions, and a sparse *Hessian* approximation, BETTS gives an overview over a completely sparse SQP algorithm. Overall, the most important conclusion that can be taken from his work is that large optimal control and numerical optimization problems can be solved a lot easier (sometimes even at all) if being formulated sparse.

Consequently, in section 4.6 of his book [Bet09], BETTS focuses on the sparsity of the non-linear programming problem that results from a discretized optimal control problem. After introducing a standard formulation for the defect constraints in the discretized problem, two reformulations for the trapezoidal scheme and two for the HERMITE-SIMPSON discretization are presented. All of them focus on a reduced number of model evaluations when using finite differences but do not yield a benefit when using analytic differentiation.

FISCH discusses the inherent sparsity properties of different control parameterizations in [Fis11] and illustrates the block structure of a multiple shooting algorithm. Thereby, he uses another sorting logic for the states and the controls as the one implemented in *FALCON.m*, resulting in two triangular blocks per segment compared to one larger block here.

LENZ solves several periodic optimal control problems in his PhD thesis, using multiple shooting as well as full discretization methods. In [Len15, p. 138], he shows the sparsity structures of one of his exemplary problems for a collocation and a multiple shooting scheme. The triangular structure that becomes visible in the work of FISCH cannot be seen in the examples of LENZ as he does not consider any path constraints. For the collocation approach, a structure similar to the one described here results.

The remainder of this chapter is structured as follows: First, a general analysis of an optimal control problem that is solved using a direct discretization method with respect to *problem sparsity* is presented. Afterwards, local influences on the sparsity structure are discussed, before a global picture is drawn in section 7.3. Finally, some special details for multi aircraft optimization problems like the ones presented in chapter 10 are discussed in section 7.4.

7.1 General Problem Sparsity

Figure 7.1 shows the structure of the elements used in the solution process of an optimal control problem when applying a full discretization scheme. (For more details on full discretization see section 3.2.4 and the references therein). In order to be able to fully exploit the sparsity in the process, every element therein needs to be considered. Any element that is not respecting problem sparsity, diminishes the overall problem sparsity and consequently the solution performance (with more or less strong influences).

Within *FALCON.m* all sparse matrices are either stored using the sparse matrix representation of *MATLAB* or by storing the data in *RCV* (row-column-value) format manually, depending on what is more efficient in every particular situation.

The overall sparsity of the discretized problem is constituted by the gradient matrix $\frac{\partial \mathbf{f}}{\partial \mathbf{z}}$

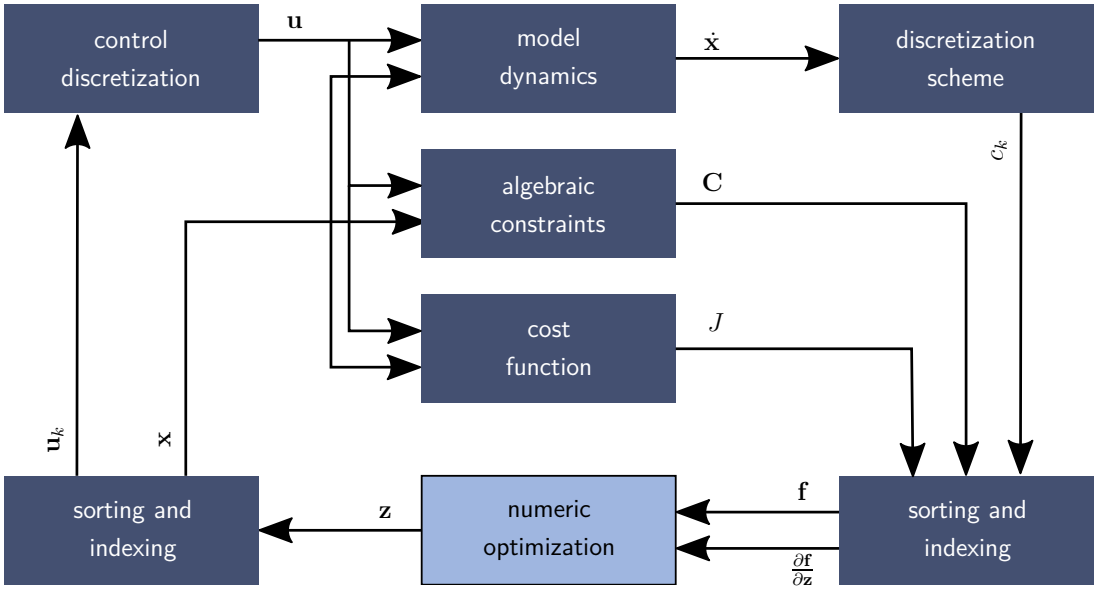

Figure 7.1: *Overview of optimization process, listing all elements relevant for problem sparsity.*

(and the *Hessian* of the corresponding *Lagrangian*) that is characterized by the definitions of the vectors $\mathbf{z}$ and $\mathbf{f}$, entering end exiting the numeric optimization algorithm as can be seen in the lower center of figure 7.1. In order to be able to profit from the sparsity of the problem, the numerical algorithm used needs to be able to exploit the problem sparsity applying means like those described in [Bet09, p. 54 ff.]. For doing so, the sparsity pattern of the overall problem needs to be determined and made available to the numeric optimization algorithm. Commercial solvers that are especially tailored to the solution of sparse numerical optimization problems, like e. g. *IPOPT* [WB06], *SNOPT* [GMS02], *SOCS* [Bet09], and *WORHP* [BW13], feature an interface for specifying this sparsity pattern.

All blocks printed in dark blue in figure 7.1 influence the sparsity of the overall problem and may consequently be used to shape it. Therein, the sorting and indexing of the vectors $\mathbf{z}$ and $\mathbf{f}$ do not influence the number of non-zeros within the *Jacobian* and *Hessian* of the problem, but just their positions within (and the shape of) the sparsity pattern. They may be used to bring all elements as closely to the diagonal of the gradient matrix as possible, resulting in good numerical properties for the optimization. Within *FALCON.m*, all elements are sorted by their order of appearance with respect to simulation time within the phases, leading to a distribution of the non-zero values close to the diagonal of the gradient when assuming causality of the underlying dynamic system (see also section 4.6).

The next aspect that is important for the sparsity of the problem is the discretization scheme used for the controls. Thereby, the sparsity of the discretized numerical problem changes, depending on how the control histories are calculated from the control discretization nodes that are part of the optimization parameter vector $\mathbf{z}$. The control histories are then required for the collocation (or the simulation) of the model dynamics and the evaluation of the constraints and the cost. More details on the control discretization can be found in section 3.2.4, while more on its sparsity can be found in section 7.2.

Having the control values calculated from the control discretization nodes, the cost function, the model dynamics, and the constraint functions can be evaluated, also influencing the sparsity. Considering the latter two, they can mostly be handled point-wise with respect to the simulation time as the sparsity pattern of these functions is usually equal for every point in time to that they apply. Further discussion on these local sparsities can be found in the next section.

Last, the discretization scheme determines how the aforementioned local sparsities are combined to the global sparsity $\frac{\partial \mathbf{f}}{\partial \mathbf{z}}$ of the numerical optimization problem. The general discretization schemes are presented in more detail in section 3.2.4, while sparsity considerations for the methods can be found in section 7.3.

7.2 Local Sparsity Considerations

As already mentioned, the global sparsity of the gradient $\frac{\partial \mathbf{f}}{\partial \mathbf{z}}$ is influenced by all local sparsities of the elements appearing in the problem formulation (see figure 7.1). In this section, first the control parametrization that has been rehearsed in section 3.2.4 is analyzed with respect to its sparsity properties. Afterwards, the sparsity of the model dynamics and the other functions appearing in the problem are discussed.

7.2.1 Sparsity of the Control Discretization

In [Ger12, p. 222], GERDTS reviews the use of *B-splines* for control parametrization as they can be constructed and evaluated comparably easily. Besides, one of their most beneficial properties for optimal control is the fact that they have local support, meaning that a change in a coefficient for one base function does only influence the control history within a limited range in time. However, the lower the order of the *B-spline*, the smaller the range that is influenced by one coefficient and the sparser the mapping. Figure 7.2 shows the base functions of a linear, second order *B-spline* in the top left figure and those of a third order, quadratic one in the top right figure, both using five grid nodes. The sparsity patterns of the gradients of these mappings are plotted below the base functions. The considered mapping is

$$u_k = u(t_k) = \bar{u}_p \cdot B_p(t_k) \tag{7.1}$$

where u_k is the control value at the time t_k and $\bar{u}_p$ is the coefficient corresponding to the p-th base function $B_p(t_k)$. It can be seen that the second order *B-spline* features a very narrow local support, as no more than two segments are influenced by each base function and no more than two base functions are active at any time. The sparsity pattern that can be seen in the bottom left plot in figure 7.2 belongs to the gradient $\frac{\partial u_k}{\partial \bar{u}_p}$ of the second order *B-spline*. The narrow local support is reflected by the sparse band diagonal structure. For the sake of better readability, in the sparsity plot only 21 nodes for the output discretization have been considered. When using third order *B-splines*, the base functions become wider and spread

over three of the four intervals of the discretization, with up to three base functions being active at the same point in time. These facts can also be seen in the sparsity pattern in the bottom right plot in the figure.

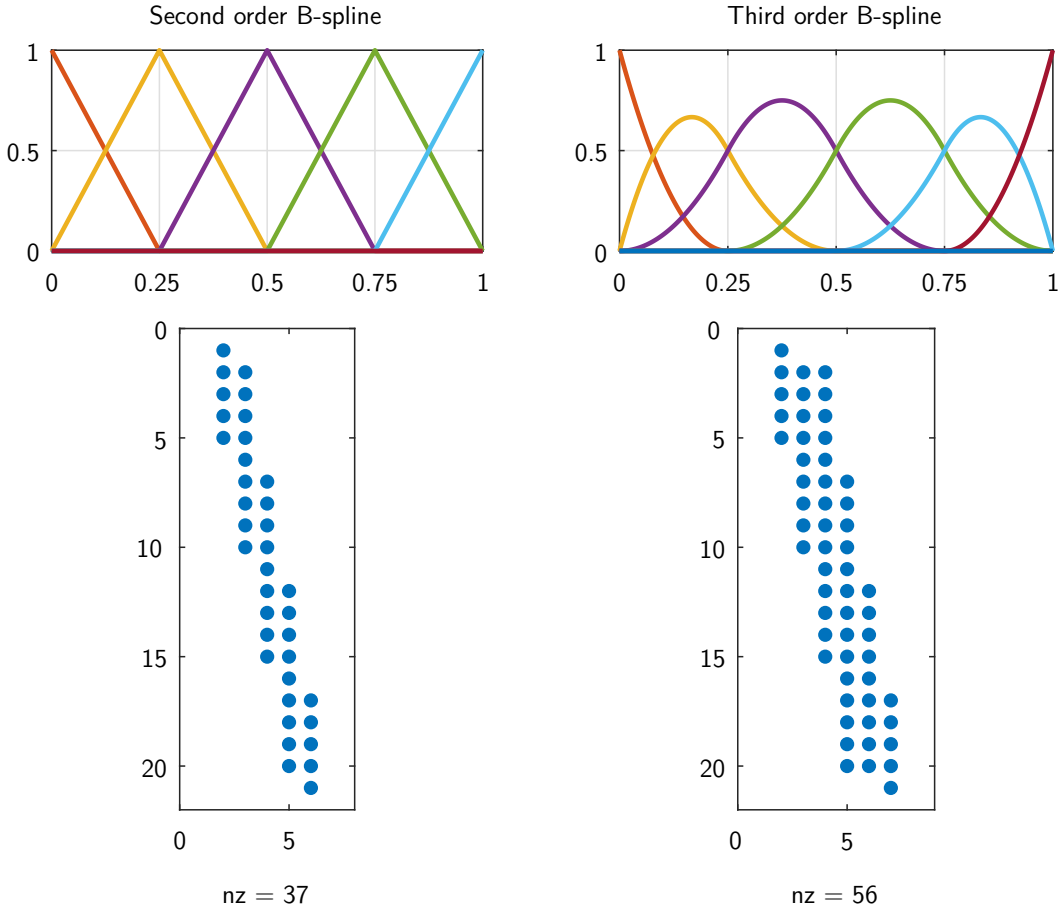

Figure 7.2: *Sparsity patterns of different control interpolation schemes based on* B-splines *(part 1). The top figures show the base functions for the* B-spline *parametrization, the bottom figures show the sparsity patterns of their gradients.*

Figure 7.3 shows that the situation gets even worse, from a sparsity point of view, when using fourth and fifth order *B-splines*. For the fourth order version (left in the plot), the base function in the center already affects all segments of the discretization. It can be seen that in the corresponding sparsity pattern, the fourth column contains only non-zero entries – except for the first and last discretization point. In the fifth order case, the two base functions in the center affect the whole control history. Besides, it can be seen that the definition of *B-splines* requires more base functions for higher order even though the same grid is used. In the examples shown here, the number of base functions ranges from five for the second order *B-splines* to eight for the fifth order *B-splines*.

However, when using far more discretization points, the effects reduce significantly, because the gradient dimensions increase. Anyway, one of the motivations for higher order control parametrizations is a reduction of required grid points. Then, the reduced problem size comes at the cost of decreased sparsity.

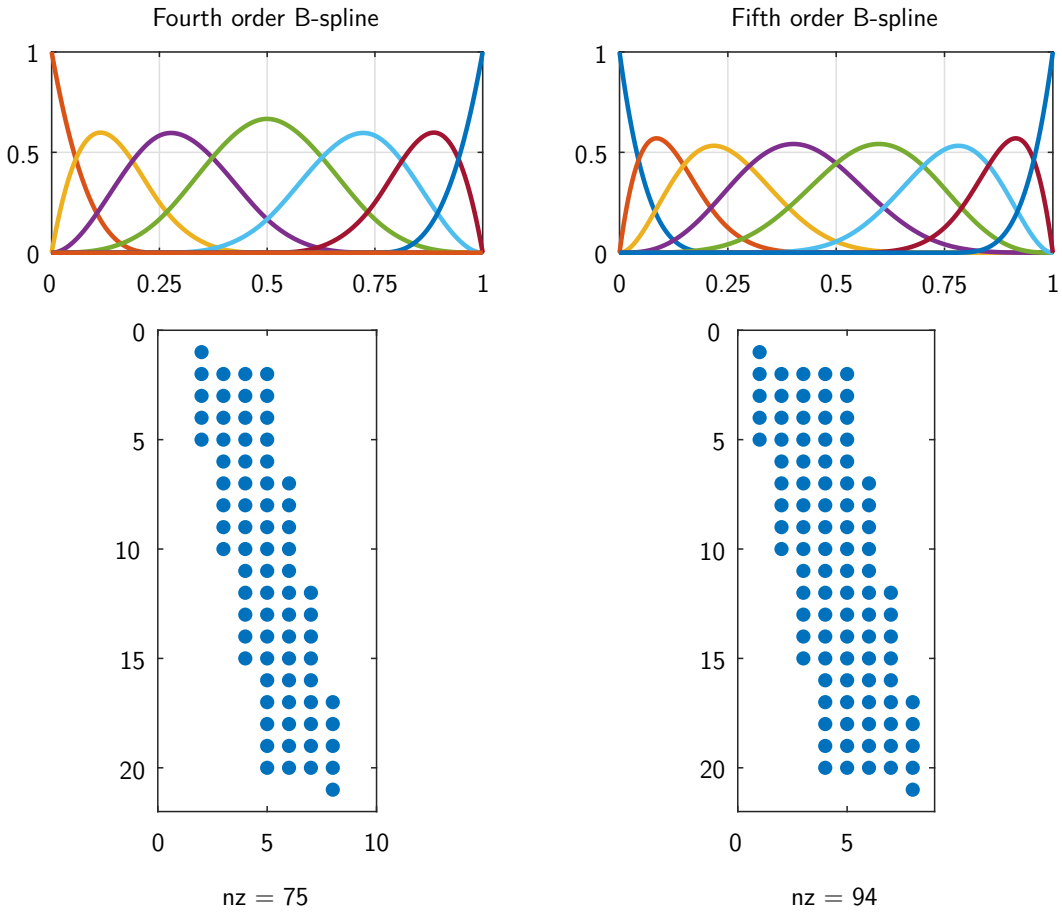

Figure 7.3: *Sparsity patterns of different control interpolation schemes based on* B-splines *(part 2). The top figures show the base functions for the* B-spline *parametrization, the bottom figures show the sparsity patterns of their gradients.*

When using another control parametrization not based on *B-splines* the particular sparsity pattern changes. However, the basic statements remain valid. One may for example think about using *impulse responses* from upstream systems (like control surface actuators) as base functions for the control discretization. Anyway, in this case, base functions with global support result in a discretization where a change at one point in time influences the complete control history (or at least the control history from this point on). When using a control parametrization that does not feature local support, the sparsity pattern of the respective mapping is dense. Depending on the application and the numeric optimization algorithm used, a simpler and more local control parametrization – like linear interpolation – with *more* grid points may then show better performance than a higher order interpolation with less nodes, even though both feature comparable smoothness properties.

7.2.2 Model and Constraint Sparsity

Next, looking at the model dynamics, the path constraints, and the point constraints, local sparsity patterns of the respective functions may be calculated. Figures 6.1–6.6 in chapter 6 show the sparsity patterns of the different model formulations and the load factor path

constraint used in the air race examples, while figure 10.2 shows the sparsity pattern of the model used in the air traffic management example. For the models and the path constraints, local sparsity patterns may be calculated from one time step that can then be incorporated in the overall problem sparsity pattern by using the chain rule and the indexing as described in the next section. Contrary, for any other constraint function it may be required to calculate the overall sparsity pattern directly, as any variable appearing in the parameter vector $\mathbf{z}$ may be connected to any constraint value in $\mathbf{f}$, in this case.

In *FALCON.m* all local sparsity patterns are calculated automatically, when using the automatic differentiation feature.

7.3 Global Sparsity Considerations

In order to obtain the global sparsity pattern to be supplied to the numerical optimization algorithm from the local sparsity information, data about the discretization scheme and the sorting and indexing of the discretized problem is required. In the examples presented in chapters 10 and 11, it turned out that in some cases a larger but more sparse optimization problem can be solved more robustly and faster than a smaller but denser problem. In this section, the sparsity properties of the numerical discretization method (see also section 3.2.4) and the effects of sorting and indexing (see also section 4.6) will be described in more detail.

In order to build the global sparsity pattern from the local ones, the chain rule needs to be applied to the defect equations, combining these two levels. Figure 7.4 shows an exemplary sparsity pattern that results from a single shooting algorithm when all values in $\mathbf{z}$ and in $\mathbf{f}$ are sorted with respect to their time of appearance as described in section 4.6. Along the vertical dimension of the *Jacobian* matrix, first the cost function J appears, followed by all constraints also sorted as described in section 4.6. Last, all constraints that combine more than one point in time appear. Due to the causality of the dynamic systems considered here, no control that becomes effective at one point in time affects any of the states or constraints in the past. The other way round, no constraint is affected by any state or control of a future point in time. Figure 7.4 shows how this results in a lower triangular structure for the single shooting discretization.

During the calculation of the global sparsity pattern from the local ones, a *"fill up"* of the local sparsity may happen in each block (fully or partially) due to the repeated application of the chain rule along the simulation time. This effect depends on the numerical integration algorithm used.

In figure 7.5 the sparsity pattern for a multiple shooting example can be seen. Now, the different multiple shooting segments are decoupled, resulting in structural zeros in the lower triangular part of the matrix, too. Here, the structure of the problem becomes visible even clearer: The columns appearing on the left represent global parameters, like initial and final simulation time that influence the whole trajectory and consequently almost all constraints. The three spots in the figure, where the triangles of the four multiple shooting segments

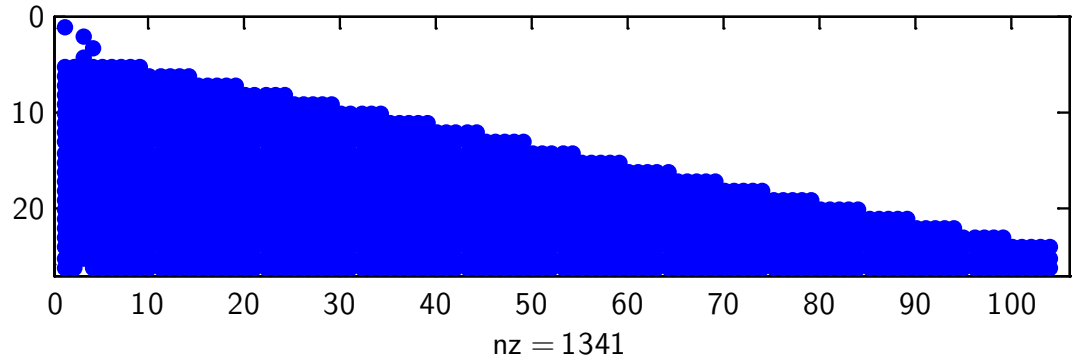

Figure 7.4: *Sparsity of an illustrative example problem using single shooting. [Hol+15]*

overlap vertically, represent the initial multiple shooting states. These influence the simulation in the following phase as well as the final defect of the preceding one. When comparing figure 7.5 to figure 7.4 (which correspond to the exact same optimal control problem), it can be seen how the dimensions of the problem slightly increased – due to the addition of the multiple shooting initial states and the defect constraints – while at the same time the number of non-zero entries in the matrix decreased strongly from 1341 to 534.

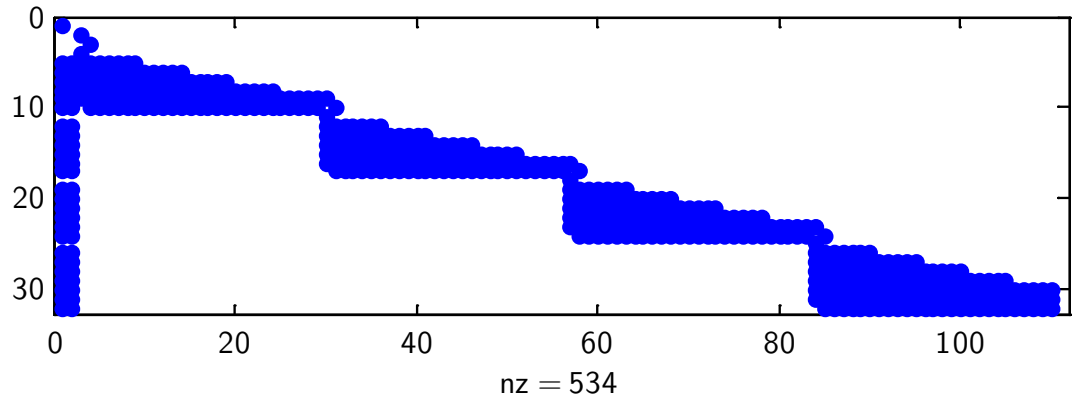

Figure 7.5: *Sparsity of an illustrative example problem using multiple shooting. [Hol+15]*

If a full discretization scheme is used for solving the same problem once more, the sparsity of the gradient matrix rises even further as can be seen in figure 7.6. Now, along the horizontal dimension of the matrix, the parameters, the states, and the controls are listed (again sorted by time of appearance), while on the vertical dimension each path constraint is followed by a collocation defect, connecting the states and controls of one point in time to the next. In this example a trapezoidal collocation scheme is used. Here, the size of the problem increased dramatically in both dimensions, with the number of structural non-zeros reaching approximately the same level as in the single shooting case. Anyway, as the interconnections between the optimization parameters and the constraints are much sparser, the problem may be solved comparably fast and robust by current sparse numerical solvers.

Similar to the sparsity of the different control discretizations in section 7.2, the sparsity of higher order collocation schemes may be not as high as that of lower order ones. When-

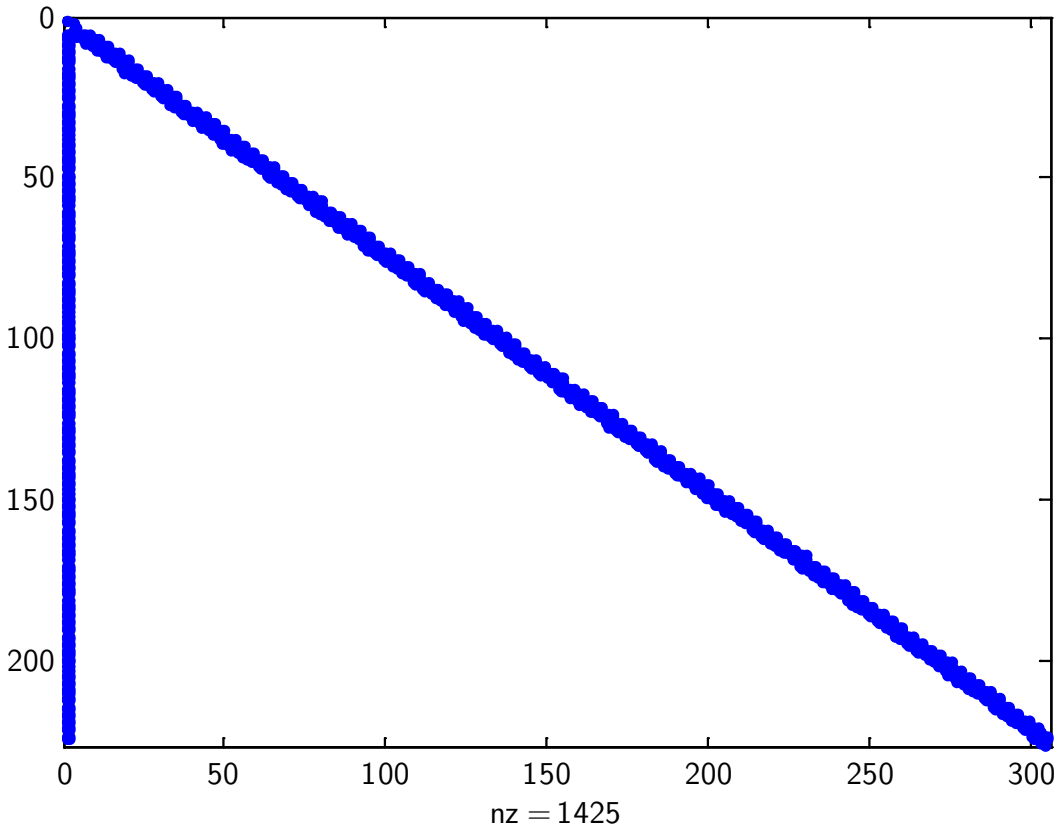

Figure 7.6: *Sparsity of an example problem using full discretization. [Hol+15]*

ever more than two state and control values influence a defect, this connection also becomes visible in the sparsity pattern. In [Bet09, sec. 4.6.5, sec. 4.6.6], BETTS shows an example demonstrating how the sparsity of a HERMITE-SIMPSON collocation scheme can be increased by adding additional optimization parameters for the intermediate states and additional constraints for the defects. In section 4.6.7 of the same work he generalizes the idea to k-stage RUNGE-KUTTA schemes.

7.4 Sparsity in Multi System Problems

In case multiple systems need to be optimized on the exact same grid, the model dynamics need to be combined. The approach used in the examples in chapter 10 combines the individual aircraft simulation models to one large simulation model by concatenating their state and control vectors.

In this case, the sparsity patterns of the combined gradient matrices can also be calculated by combining the sparsity patterns of all individual systems. Figure 7.7 shows the generalized principle. At the very left of the figure, a general sparsity pattern for one dynamic model or one constraint can be seen. In the center, the combination of all individual systems (in this case five) can be seen, resulting in a large sparsity pattern. After this step, the regular discretization – as described in the section above – is performed. When using a collocation

scheme, a sparsity pattern like the one illustrated in the very right diagram of figure 7.7 results. Depending on the collocation scheme used, the sparsity pattern may be copied in this second step, or a mapping may be needed (chain rule).

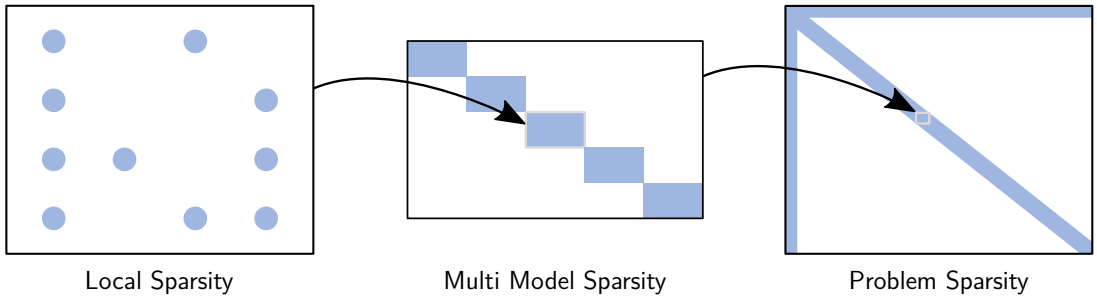

Figure 7.7: *From local to global sparsity patterns in multi system optimization problems.*

Using the aforementioned principle, a highly block diagonal model structure results, also leading to a high sparsity of the overall discretized problem. When looking at the collocation defects from the combined dynamics, the block diagonal structure of the different participants in the problem remains preserved, as no interference between the dynamics may appear, independent of the collocation scheme (the same holds for shooting in this case). The only connection between different systems in the scenario is established through separation and safety constraints that combine states and controls of multiple participants.

A more detailed explanation based on the example can be found in section 10.2.9, and there especially in figures 10.2–10.5 and in table 10.4. Besides, similar considerations have previously also been published in [Bit+15, Bit+16].

Chapter 8

Improving Robustness by the Use of Combined Transcription Methods

One of the issues appearing in optimal control problems with high fidelity simulation models are different time scales that may be inherent to the model, resulting in *stiff differential equations*. Especially in mechanical systems, in many cases, fast dynamics can be recognized as *internal* (small scale) dynamics, while slower dynamics often represent the more visible (large scale) *outer* part of the motion of a body. If one is not interested in a detailed high fidelity solution of such an optimal control problem, the internal and fast dynamics may be considered to be decayed. Hence, they can be removed from the dynamic equations and may be replaced by analytic relationships. The resulting dynamic system features less states and is not *stiff* anymore.

If such simplifications are not acceptable, the full dynamics have to be taken into account. Then, the selection of an appropriate discretization grid is driven by the fast – yet small scale – dynamics that hardly influence the overall results. Consequently, a very fine discretization grid is required for all states in the problem, making the solution computationally expensive. As mentioned in section 3.2.4, implicit integration algorithms can better handle stability issues resulting from *stiff* dynamics. However, they do not deliver *precise and detailed* results if not applied on a very fine grid. In many high fidelity optimal control problems, the required fine grid is computationally more challenging than the additional states.

When considering aircraft trajectory optimization problems, the issue mainly relates to the fast rotational states of the aircraft motion compared to the relatively slow translational movement (see chapter 2). Besides, other subsystem models – like actuators – may be added to the model, creating the same issues. In many scenarios (like the ones presented in section 10), the fast rotational states may be regarded as quasi steady, meaning that the fast decay of any rotational dynamic effects can be ignored. Anyway, this represents a simplification and leads to the point mass simulation model presented in section 2.4.

In this chapter a novel discretization approach is presented that combines a collocation method with a multiple shooting approach. Therein, the dynamic equations of the system are partitioned into a fast and a slow part. The fast dynamics are solved on a dense grid using

the multiple shooting approach, while the slow dynamics are solved on a relatively coarse grid using a direct collocation scheme. The remainder of this chapter is structured as follows: First, an overview of other existing approaches is given. Afterwards, the combination of collocation techniques with multiple shooting is introduced. An example problem using the idea can be found in section 11.4, together with a comparison and discussion of the respective results.

8.1 Overview of Existing Approaches

Different approaches exist for overcoming the aforementioned issue, including the idea of using multiple timescales within one optimal control problem. The following sections present some of the ideas found in the literature.

8.1.1 Multi Rate Runge-Kutta Methods

The issue mentioned before is not only inherent to optimal control problems but similarly appears in pure system simulations. Anyway, in optimal control problems, efficient simulation algorithms are of great importance as the simulations need to be performed many times in order to evaluate the state history within every iteration. The mathematical theory developed for simulation tasks may be useful for optimal control problems, too. Multi rate Runge-Kutta methods have been around for some time, already being mentioned in the sixties [Bra68], before being researched in more detail during the eighties [OS83, GW84].

During the last years, besides others, Günther has published some work on multi rate Runge-Kutta schemes. In [GKR01], a generalized multi rate partitioned Runge-Kutta method is presented that may also be used for the simulation of *stiff* differential equations. The approach assumes that the underlying dynamic system can be partitioned in a (slow) latent part which may be (partially) stiff and a (fast) active part which must not be stiff. The authors of the paper propose a combined integration scheme where for the latent part implicit methods may be used while the active part is solved using an explicit integration scheme. When using multi stage Runge-Kutta methods, the stages of the two numerical integration algorithms are calculated individually, not adding additional coupling.

8.1.2 Multi Timescale Collocation Method

In [DC08b], Desai and Conway present two different two-timescale discretization schemes, based on the fifth degree Gauss-Lobatto collocation method which can also be found in equations (3.183)–(3.192). In the first method, the fast dynamics are collocated on a grid which has half the segment length of the grid for the slow dynamics. In the second method they use a segment subdivision of four, meaning that the high frequency dynamics are discretized using four times as many collocation points as the low frequency dynamics.

In both cases, the number of NLP parameters is reduced compared to a full collocation on a fine grid, as the state values for the slow states are not added as NLP parameters any more.

Anyway, in order to evaluate the fast part of the dynamics, state and control values on the fine grid are required. DESAI and CONWAY considered several approaches for reconstructing the missing state and control values and end up with quartic interpolation polynomials for the controls and quintic interpolation polynomials for the states. Both interpolations are based on the values of the high frequency grid.

The method has been applied to an optimal control problem using a six degree of freedom aircraft simulation model in [DC08a]. Therein, the touchdown velocity of an unpowered reentry vehicle is minimized under certain boundary conditions.

Compared to the method presented in section 8.2, the main drawback of this method is that the two timescales are closely coupled together and cannot be selected independently by the user.

8.1.3 Direct Single-Multiple Shooting Method

BOTTASSO et. al. combine a single shooting and a multiple shooting approach in [BM09, BLM12]. In their work, the main issues identified are

- the amount of NLP parameters required to perform a multiple shooting integration of the fast states, and
- the difficulty to satisfy the multiple shooting defect constraints for the fast states.

Having the potential instability of single shooting – due to the high sensitivity with respect to the initial state – in mind, they suggest to separate the state vector into a fast and a slow part. Then, they use one single shooting integration sweep over the whole time horizon for the fast states, and regular multiple shooting with defect constraints (see also equation (3.170)) for the slow states.

In their formulation, no multiple shooting defect constraints and no multiple shooting initial states are required for the fast states, decreasing the number of optimization variables. Moreover, they state that the multiple shooting of the slow states avoids a blow up in the solution and thereby keeps the overall solution stable. The authors of the paper identify that no matter which initial states are chosen in the optimization, the fast states will not blow up, whenever the slow ones do not do so.

In [BMS10] the optimization of different maneuvers of a rotorcraft are researched. It is investigated how direct collocation and direct multiple shooting may be used in these optimal control problems with models of different fidelities. They state that direct transcription (collocation) is faster and more stable for simple models, while multiple shooting may be used for more complex models – also including fast dynamic components. When using the latter, the size of the discretized problem is independent of the number of integration steps within each multiple shooting segment, enabling the use of a fine grid while avoiding extremely large problems. The authors of the paper draw the conclusion that optimal control problems with detailed simulation models can only be solved well using shooting methods as the fast components require the use of a fine discretization grid.

Another issue that is considered in the work [BMS10] is that the states in between two consecutive shooting nodes cannot be used in the path constraint equations as they are not part of the numerical optimization vector. The proposed solution approach is to detect the segments including a constraint violation after a problem has been solved. Afterwards, these segments should be broken up into smaller ones. This issue also remains in the approach proposed here and leaves room for future research.

8.2 Combined Direct Collocation and Multiple Shooting

The approach for overcoming the aforementioned issues that is proposed here combines a direct collocation scheme with a direct multiple shooting approach. The basic idea is to separate the state vector of the dynamic system into two parts, one containing the fast and the other containing the slow dynamics. The fast dynamics are discretized using a multiple shooting approach and the slow ones using a collocation scheme that is based on the same grid as the multiple shooting *segment nodes*. The number of multiple shooting integration grid points in between these nodes may be chosen freely by the user, forming another, finer grid. In order to be able to evaluate the dynamic equations within the multiple shooting segments, an approximation of the slow states is required. These values can be calculated using different interpolation algorithms, where a cubic and a linear interpolation scheme are suggested here. The required gradients can be evaluated using slightly modified sensitivity equations.

Figure 8.1 visualizes the general idea of the discretization scheme. The two grids inherent to the method can be seen on the abscissa with the large markers representing the collocation and multiple shooting *segment node* grid

$$\mathbb{G}_{\mathbf{x}_s} = \{t_0 < t_1 < t_2 < \ldots < t_N = t_f\} \tag{8.1}$$

and the small markers representing the multiple shooting *integration* grid:

$$\mathbb{G}_{\mathbf{x}_f} = \{\bar{\tau}_{0,0}, \bar{\tau}_{0,1}, \ldots, \bar{\tau}_{0,n_0} = \bar{\tau}_{1,0}, \bar{\tau}_{1,1}, \ldots, \bar{\tau}_{1,n_1} = \bar{\tau}_{2,0}, \ldots, \bar{\tau}_{N,n_N}\} \tag{8.2}$$

In the following, they will be referred to as the *coarse grid* $\mathbb{G}_{\mathbf{x}_s}$ and the *fine grid* $\mathbb{G}_{\mathbf{x}_f}$. The two grids may be chosen almost independent of each other and arbitrarily. The only constraint that needs to be considered is that the multiple shooting *segment nodes* (also being the collocation nodes of the slow states) need to be part of both grids, meaning that

$$t_{k+1} = \bar{\tau}_{k+1,0} = \bar{\tau}_{k,n_k} \tag{8.3}$$

needs to be fulfilled. The coarse grid can hence be seen as an arbitrary subset of the fine grid:

$$\mathbb{G}_{\mathbf{x}_s} \subseteq \mathbb{G}_{\mathbf{x}_f} \tag{8.4}$$

The multiple shooting defect between the k-th and the $k+1$-th multiple shooting segment is visualized in the figure as $d_{f,k}$.

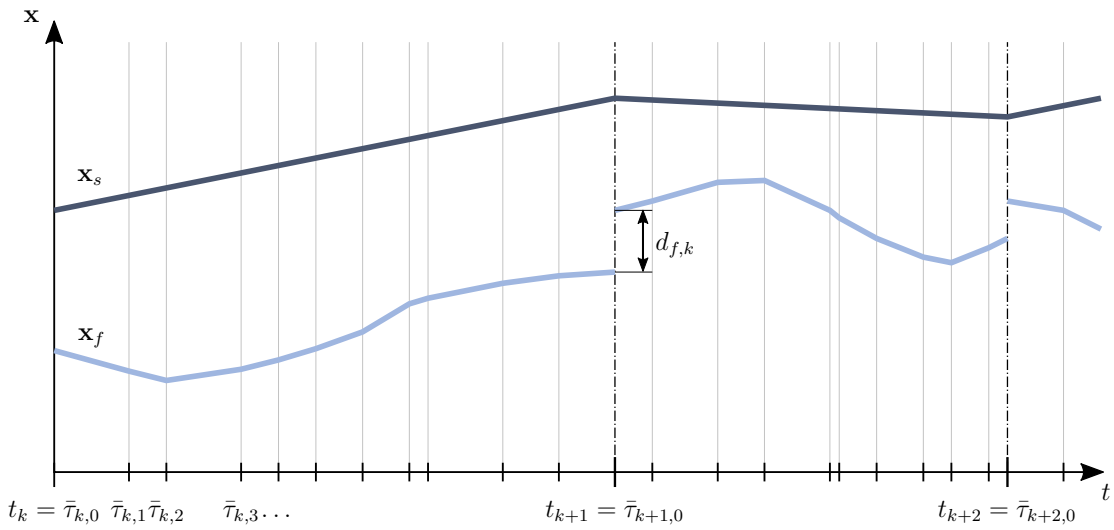

Figure 8.1: *Discretization scheme based on collocation and multiple shooting.*

The dynamics of the system are partitioned into a slow part and a fast part by separating the state vector $\mathbf{x}$ into the sub-vector $\mathbf{x}_s$ for the slow states (shown in dark blue in figure 8.1) and $\mathbf{x}_f$ for the fast states (depicted in light blue in figure 8.1). Consequently, the state dynamics

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{p}) \tag{8.5}$$

also need to be partitioned into the fast dynamics $\mathbf{f}_f$ and the slow dynamics $\mathbf{f}_s$. Equation (8.5) can hence be rewritten as:

$$\dot{\mathbf{x}}_s = \mathbf{f}_s(\mathbf{x}_s, \mathbf{x}_f, \mathbf{u}, \mathbf{p}) \tag{8.6}$$

$$\dot{\mathbf{x}}_f = \mathbf{f}_f(\mathbf{x}_s, \mathbf{x}_f, \mathbf{u}, \mathbf{p}) \tag{8.7}$$

It has to be remarked that neither the slow dynamics nor the fast dynamics can be evaluated without knowledge of the state of each other. As the slow states are implicitly integrated using the full discretization approach from section 3.2.4, the complete history of the slow state $\mathbf{x}_s$ is part of the optimization parameter vector and consequently directly available on the grid $\mathbb{G}_{\mathbf{x}_s}$. The fast states $\mathbf{x}_f$ are integrated using en explicit numerical integration scheme within each multiple shooting segment as described in section 3.2.4. Again, their state history on the coarse grid is part of the optimization parameter vector and consequently known. The required controls are assumed to be known on the coarse grid, too, as they can be determined there by interpolation as discussed in section 3.2.4.

The defects for the slow states $\mathbf{x}_s$ need to be calculated based on the collocation scheme – e. g. from equation (3.182) when using trapezoidal collocation. For the multiple shooting of the fast states $\mathbf{x}_f$, the theory from section 3.2.4 is used, with the defects $d_{f,k}$ presented in equation (3.170). Here, the intermediate values for the slow states, that are required to evaluate $\mathbf{f}_f(\mathbf{x}_s, \mathbf{x}_f, \mathbf{u}, \mathbf{p})$ at the nodes of grid $\mathbb{G}_{\mathbf{x}_f}$, are still unknown. Additionally, the control values are also unknown at these points. However, they can easily be calculated by extending the methods presented in 3.2.4 to the fine grid.

Here, two different approaches for the approximation of the slow states on the fine grid are suggested:

- A linear interpolation based on the state values, and
- a cubic interpolation based on the state values and derivatives.

For both of them, the normalized time $\bar{\tau}_k \in [0, 1]$ is introduced in each segment $[t_k, t_{k+1}]$ of the coarse grid. In the following derivations, the abbreviation

$$\mathbf{x}_s(t_k) = \mathbf{x}_{s,k} \tag{8.8}$$

is used.

The linear interpolation of the slow states can directly be calculated from the initial and the final state values of each multiple shooting segment:

$$\mathbf{x}_s(\bar{\tau}_k) = \mathbf{x}_{s,k} + \bar{\tau}_k \cdot (\mathbf{x}_{s,k+1} - \mathbf{x}_{s,k}) \tag{8.9}$$

When using a cubic interpolation in the segments, the state derivatives at the initial and the final point also need to be considered. They can be calculated from the following state equations:

$$\dot{\mathbf{x}}_{s,k} = \mathbf{f}_s(\mathbf{x}_{s,k}, \mathbf{x}_{f,k}, \mathbf{u}_k, \mathbf{p}) = \mathbf{f}_{s,k} \tag{8.10}$$

$$\dot{\mathbf{x}}_{s,k+1} = \mathbf{f}_s(\mathbf{x}_{s,k+1}, \mathbf{x}_{f,k+1}, \mathbf{u}_{k+1}, \mathbf{p}) = \mathbf{f}_{s,k+1} \tag{8.11}$$

Requiring the state values and the state derivatives to match at the initial and the final point of each segment, the cubic polynomial

$$\begin{aligned} \mathbf{x}_s(\bar{\tau}_k) = {} & (2 \cdot \mathbf{x}_{s,k} + \dot{\mathbf{x}}_{s,k} - 2 \cdot \mathbf{x}_{s,k+1} + \dot{\mathbf{x}}_{s,k+1}) \cdot \bar{\tau}_k^3 \\ & + (-3 \cdot \mathbf{x}_{s,k} - 2 \cdot \dot{\mathbf{x}}_{s,k} + 3 \cdot \mathbf{x}_{s,k+1} - \dot{\mathbf{x}}_{s,k+1}) \cdot \bar{\tau}_k^2 \\ & + \dot{\mathbf{x}}_{s,k} \cdot \bar{\tau}_k + \mathbf{x}_{s,k} \end{aligned} \tag{8.12}$$

results for every grid segment. Hence, the state values at the nodes $\bar{\tau}_{k,l}$ of the fine grid can be calculated using either equation (8.9) or equation (8.12).

Similar to all other discretization schemes, the calculation of analytic gradient information is necessary in order to achieve stable and fast convergence of the discretized optimal control problem. As the slow dynamics are solved using a regular collocation scheme, the calculation of the gradients does not change. The multiple shooting scheme for the fast states was slightly changed because the interpolated slow states need to be added to the equations. Consequently, the sensitivity equations presented in section 3.2.4 need to be adapted accordingly. Starting from equations (3.160)–(3.166), (3.171) and (3.173), the slow states can be seen as an additional external input to the equations featuring their own interpolation scheme. The sensitivity equations (3.160)–(3.162) need to be extended by equations for the slow states

that can be derived based on the chain rule as:

$$\begin{aligned}\frac{d\dot{\mathbf{x}}_f(\bar{\tau}_k)}{d\mathbf{x}_{s,k}} &= \frac{d}{dt}\left(\frac{d\mathbf{x}_f(\bar{\tau}_k)}{d\mathbf{x}_{s,k}}\right) \\ &= \frac{\partial \mathbf{f}_f(\mathbf{x}_s(\bar{\tau}_k), \mathbf{x}_f(\bar{\tau}_k), \mathbf{u}(\bar{\tau}_k), \mathbf{p})}{\partial \mathbf{x}_f(\bar{\tau}_k)} \cdot \frac{d\mathbf{x}_f(\bar{\tau}_k)}{d\mathbf{x}_{s,k}} \\ &\quad + \frac{\partial \mathbf{f}_f(\mathbf{x}_s(\bar{\tau}_k), \mathbf{x}_f(\bar{\tau}_k), \mathbf{u}(\bar{\tau}_k), \mathbf{p})}{\partial \mathbf{x}_s(\bar{\tau}_k)} \cdot \frac{d\mathbf{x}_s(\bar{\tau}_k)}{d\mathbf{x}_{s,k}}\end{aligned} \tag{8.13}$$

$$\begin{aligned}\frac{d\dot{\mathbf{x}}_f(\bar{\tau}_k)}{d\mathbf{x}_{s,k+1}} &= \frac{d}{dt}\left(\frac{d\mathbf{x}_f(\bar{\tau}_k)}{d\mathbf{x}_{s,k+1}}\right) \\ &= \frac{\partial \mathbf{f}_f(\mathbf{x}_s(\bar{\tau}_k), \mathbf{x}_f(\bar{\tau}_k), \mathbf{u}(\bar{\tau}_k), \mathbf{p})}{\partial \mathbf{x}_f(\bar{\tau}_k)} \cdot \frac{d\mathbf{x}_f(\bar{\tau}_k)}{d\mathbf{x}_{s,k+1}} \\ &\quad + \frac{\partial \mathbf{f}_f(\mathbf{x}_s(\bar{\tau}_k), \mathbf{x}_f(\bar{\tau}_k), \mathbf{u}(\bar{\tau}_k), \mathbf{p})}{\partial \mathbf{x}_s(\bar{\tau}_k)} \cdot \frac{d\mathbf{x}_s(\bar{\tau}_k)}{d\mathbf{x}_{s,k+1}}\end{aligned} \tag{8.14}$$

where

$$\mathbf{S}_{\mathbf{x}_f,\mathbf{x}_{s,k}} = \frac{d\mathbf{x}_f(\bar{\tau}_k)}{d\mathbf{x}_{s,k}} \qquad \text{and} \tag{8.15}$$

$$\mathbf{S}_{\mathbf{x}_f,\mathbf{x}_{s,k+1}} = \frac{d\mathbf{x}_f(\bar{\tau}_k)}{d\mathbf{x}_{s,k+1}} \tag{8.16}$$

represent newly introduced sensitivities that need to be determined by integration of the sensitivity equations. Contrary, $\frac{d\mathbf{x}_s(\bar{\tau}_k)}{d\mathbf{x}_{s,k}}$ and $\frac{d\mathbf{x}_s(\bar{\tau}_k)}{d\mathbf{x}_{s,k+1}}$ may be seen as input sensitivities and can be calculated from the respective interpolation scheme. In case the linear scheme from equation (8.9) is used, the required gradients can be evaluated as

$$\frac{d\mathbf{x}_s(\bar{\tau}_k)}{d\mathbf{x}_{s,k}} = 1 - \bar{\tau}_k \tag{8.17}$$

$$\frac{d\mathbf{x}_s(\bar{\tau}_k)}{d\mathbf{x}_{s,k+1}} = \bar{\tau}_k \tag{8.18}$$

while for the cubic interpolation from equation (8.12),

$$\frac{d\mathbf{x}_s(\bar{\tau}_k)}{d\mathbf{x}_{s,k}} = \left(2 + \frac{\partial \mathbf{f}_{s,k}}{\partial \mathbf{x}_{s,k}}\right) \cdot \bar{\tau}_k^3 + \left(-3 - 2 \cdot \frac{\partial \mathbf{f}_{s,k}}{\partial \mathbf{x}_{s,k}}\right) \cdot \bar{\tau}_k^2 + \frac{\partial \mathbf{f}_{s,k}}{\partial \mathbf{x}_{s,k}} \cdot \bar{\tau}_k + 1 \tag{8.19}$$

$$\frac{d\mathbf{x}_s(\bar{\tau}_k)}{d\mathbf{x}_{s,k+1}} = \left(-2 + \frac{\partial \mathbf{f}_{s,k+1}}{\partial \mathbf{x}_{s,k+1}}\right) \cdot \bar{\tau}_k^3 + \left(3 - \frac{\partial \mathbf{f}_{s,k+1}}{\partial \mathbf{x}_{s,k+1}}\right) \cdot \bar{\tau}_k^2 \tag{8.20}$$

results. It can be seen that in the second case the gradient of the slow state dynamics $\mathbf{f}_s$ with respect to the slow states $\mathbf{x}_s$ needs to be evaluated at the initial and the final point of each segment in order to be able to evaluate the gradient of the interpolation scheme.

Depending on which parts of the system dynamics are influenced by the controls, the control grids may be chosen by the user. In general, it does not make sense to have more control discretization points than collocation or integration nodes, as the changes in the controls cannot be resolved then. Additionally, every control grid point should coincide with a point on the state grid. Accordingly, whenever a control is (mainly) entering the slow part of the

dynamic equations, it does not make sense to choose a control discretization that is finer than the one of the coarse (collocation) grid. Contrary, all controls that mainly influence the fast dynamics of the system may be discretized on a grid which may be as dense as the fine grid of the method. Moreover, different control values may be discretized on different control grids. Apart from that, the statements concerning the control discretization from section 3.2.4 are still valid here.

The method as it is presented here, represents a combination of explicit and implicit numerical integration schemes. While for the collocation explicit algorithms as well as implicit ones may be used, for the shooting part the use of implicit method comes at the cost of a very high computational burden. This may be seen as a drawback of the shooting method, as for stiff dynamics implicit methods show much better stability properties. However, the fine integration grid that can be used in the shooting can resolve this issue.

When path constraints are to be calculated from the state histories within an optimal control problem, the state values of the coarse grid can easily be used as they are part of the optimization parameter vector. It may happen that the states in between two nodes of the coarse grid lead to a violation of a path constraint that may not be detected in this case. An example for this can be seen in figure 11.33 of the example problem solved in section 11.4. Technically, it is possible to also evaluate the path constraints based on the states of the fine grid. However, the additional constraints that are introduced to the discretized problem may change the structure of the problem, resulting in convergence issues. Here, further investigation is required. Moreover, the evaluation of the gradients of these path constraints becomes quite cumbersome as they require the sensitivities of the simulation as input.

Concerning the implementation of the proposed scheme on a computer, the independence of all collocation segments and all multiple shooting segments may be exploited for parallelization. This means that the multiple shooting simulations for all segments as well as the evaluation of all collocation defects may be executed in parallel. This feature has been included in *FALCON.m*, resulting in considerable speedup in many cases.

The combined discretization scheme presented here may be extended even more in future research by e. g. estimating the integration errors of the simulation in the multiple shooting segments and using this information for a grid refinement scheme. This way, the number of collocation and shooting nodes may be reduced, further decreasing problem size while increasing solution accuracy and robustness.

The method presented in this section has been applied to an example scenario for air race trajectory optimization in section 11.4.

Chapter 9

Improving Performance Based on Control Grid Refinement

Especially in optimal control problems that feature fast maneuvers compared to the overall time frame considered, a proper selection of discretization grids for the states and the controls is of utmost importance. As one can imagine, the size of the numerical optimization problem resulting from the discretized optimal control problem becomes large if a discretization grid is used that is fine enough to represent fast movements accurately along the whole trajectory – and consequently the problem becomes hard to solve. In order to overcome this issue, it is necessary to wisely place the nodes of the grid at those particular points in time where fast movements take place and remove superfluous grid points from parts where smooth results can be represented using a coarser grid.

In general, two influences have to be distinguished:

- On the one hand the possibility to accurately represent the optimal control inputs based on the *control discretization*,
- and on the other hand the necessity to accurately describe the system dynamics based on the *state grid*.

Of course, the second is related to the first, as fast maneuvers in the controls affect the system dynamics (possibly including some dynamic delay) and consequently require a grid that is fine enough to fully cover the reaction of the system.

9.1 Overview of Mesh Refinement Techniques

In pure numerical simulations, the adaptation of the state grid is normally done based on the discretization error that is estimated by comparing the results of two integration methods of different order (see section 3.2.4 on numerical integration methods and [Ger12, p. 162 ff.]). In case the difference in the results of the two methods remains below a certain limit, the current integration step is accepted, otherwise the step size needs to be reduced. It is possible

to construct RUNGE-KUTTA methods, as described in section 3.2.4, that feature neighboring orders by only using a different set of output vectors $\mathbf{c}$. These RUNGE-KUTTA schemes are called *embedded*. More details on step size selection and *embedded* RUNGE-KUTTA methods can, besides others, be found in [Ger12, p. 185 ff.]. Two of the commonly used *embedded* methods are the RUNGE-KUTTA-FEHLBERG method that has been published by ERWIN FEHLBERG in 1969 in [Feh69], and the DORMAND-PRINCE method [DP80]. Table 9.1 shows the BUTCHER tableau of FEHLBERG's method.

Table 9.1: BUTCHER *tableau for the* FEHLBERG *variable step size integration method. [Feh69]*

0						
$1/4$	$1/4$					
$3/8$	$3/32$	$9/32$				
$12/13$	$1932/2197$	$-7200/2197$	$7296/2197$			
1	$439/216$	-8	$3680/513$	$-845/4104$		
$1/2$	$-8/27$	2	$-3544/2565$	$1859/4104$	$-11/40$	
	$16/135$	0	$6656/12825$	$28561/56430$	$-9/50$	$2/55$
	$25/216$	0	$1408/2565$	$2197/4104$	$-1/5$	0

When using mesh refinement techniques – no matter if the control grid, the state grid or both should be refined – one must distinguish between *dynamic grid refinement methods* and *static grid refinement methods*. In the latter, the optimization and the grid refinement are performed serially, meaning that several optimization runs are done while in between a mesh refinement algorithm is used in order to formulate the next problem to be solved. Several main benefits of these methods can be found: They are relatively stable, as the two processes (optimization and mesh refinement) are not coupled. Besides, they are relatively easy to implement, as they do not require any changes in the discretization scheme of the optimal control problem. The downsides of the methods are the following: Due to the general nature of the methods, it is necessary to solve several optimal control problems which may in some cases not lead to an improvement by means of runtime compared to the solution of the problem on a fine grid right in the beginning (at least if this large problem converges at all).

In contrast to the methods described before, so-called *dynamic methods* try to adapt the required grid (or grids) in parallel to the optimization run. Consequently, they do not require multiple optimization runs, but instead change the grid parameters in between two iterations. The main benefit of the method is obvious as it only requires one optimization run for the solution of the problem. On the other hand, the main drawbacks are as follows: As the method strongly interacts with the optimization itself and consequently also has a strong impact on the local gradients of the problem, a tight coupling between the mesh refinement and the optimization exists. This coupling may lead to a situation where the mesh refinement is somehow "fighting" against the numerical optimization, leading to convergence

problems. Moreover, the implementation of such methods is not easy as they require changes in the discretization scheme and may also impact theoretical foundations of discretized optimal control problems. Due to these drawbacks, in the remainder of this work only static mesh refinement methods will be considered. In [Pol97, sec. 4.7] and [Sch96] more details and references on static and dynamic mesh refinement can be found.

When using a shooting approach for the discretization of an optimal control problem, dynamic grid refinement may be achieved by using a variable step *ODE* solver within each shooting interval. The main problem then is, that the gradient information becomes inaccurate as the numerical effects of the changing integration grid are not reflected in the gradient (neither when using sensitivity equations, nor when using finite differences). Otherwise, static state grid refinement may be achieved by using the methods for step size selection of a variable step integration method in between two optimization runs. This idea may be used with shooting as well as full discretization methods as the collocation approaches presented here may be derived from the RUNGE-KUTTA integration schemes. Consequently, the step size selection normally used in conjunction with RUNGE-KUTTA methods can be transferred to the collocation approaches. BETTS gives an overview of grid refinement methods in [Bet09, Section 4.7, p. 153 ff.]. Regarding optimal control software, state mesh refinement methods are available in *PSOPT* [Bec11] and *GPOPS* [Fra08].

After this brief discussion of state grid refinement methods, the remainder of this chapter focuses on a static control grid refinement method. The main problem is, that the structure of the solution of the optimal control problem is usually not known a priori, making it impossible to select an adequate control grid before running an optimization. The following paragraphs give a short overview of research previously done on static grid refinement methods.

SCHLEGEL et. al. [Sch+05] and BINDER et. al. [Bin+00] use the fast wavelet transformation and the resulting control representations in the wavelet space for grid refinement. Their procedure for the elimination of unnecessary grid points is based on the norm equivalence in the wavelet domain. This enables the elimination of all elements which are smaller than a user defined threshold. The insertion of new grid points works very similar: Wavelet functions with large coefficients are important for an accurate representation of the controls. Therefore, the neighboring wavelets (corresponding to a time scale representation of the wavelets) of these important wavelets, that are not part of the control representation yet, are added. The algorithm presented in [Sch+05] also considers the problem of removing and re-adding the same grid points during the run of the algorithm and therefore keeps track of all possible grid points.

In [JT08] a multiresolution technique based on *Dyadic grids* is presented. The main idea of the proposed algorithm is the use of non-uniform *Dyadic grids* with different resolutions for an iterative grid refinement based on interpolation errors. Therefore, the controls, the states or both are interpolated on a grid finer than the one used in the current iterate. The interpolation error coefficient at these additional points is calculated and then compared to a threshold. If the error is above the threshold the new points are added to the grid.

ZHAO and TSIOTRAS [ZT11] use density functions in their static grid refinement algorithm for redistributing grid points. The idea of density functions has also been used in *Finite Element Methods (FEM)* before [Hug93]. The density function as it is used in [ZT11] describes the distribution of the time points in the control grid. The authors of that paper suggest the third root of the curvature of the respective control as a density function because it yields the best piecewise linear approximation of the controls. The problem for an implementation of this algorithm is that the estimation of the second derivatives of the controls is required in order to calculate their curvature. The authors suggest a spline interpolation to tackle this issue which may not be exact when the underlying grid is coarse. Anyway, the *Density Function Based Mesh Refinement Algorithm (DENMRA)* is the basis for the algorithm suggested here and is described in more detail below.

The results presented in this section have been created with the support of PHILIP BRUHS who did his Bachelor Thesis at the *Institute of Flight System Dynamics* under the supervision of this author. The results have been published similarly in the joint paper [Bit+13a], before.

9.2 From Density Functions to the DENMRA Algorithm

Density functions for grid refinement have been used in other applications for several years now. [Hug93] for example demonstrates the use of density functions within *FEM* applications. For the algorithm presented here, the work of [ZT11] forms the main basis, which itself is mostly based on the algorithm developed in [JT08].

The main idea of density functions for grid refinement in optimal control tasks is to identify in which parts of a trajectory a high density and in which a low density in control grid points is required. In the following, some facts from [ZT11] are recapitulated, considering an interval $\tau \in [0, 1]$ in normalized time.

The density function has to be a non-negative LEBESGUE integrable function satisfying:

$$\int_0^1 \bar{f}(\tau) d\tau = 1 \tag{9.1}$$

In case one wants to use a function that does not fulfill (9.1) normalization is possible as long as the function is non-negative and LEBESGUE integrable on the relevant interval:

$$\bar{f}(\tau) = \frac{f(\tau)}{\int_0^1 f(t) dt} \tag{9.2}$$

The so-called corresponding cumulative distribution function $F(\tau)$ is then defined as

$$F(\tau) = \int_0^\tau \bar{f}(t) dt, \qquad \text{with} \qquad F(0) = 0, F(1) = 1 \tag{9.3}$$

A mesh consisting of N grid points $\{\tau_i\}_{i=1}^N$ with $\tau_1 = 0$ and $\tau_N = 1$ is used. Every point τ_i is located on the grid such that

$$F(\tau_i) = \frac{i-1}{N-1} \tag{9.4}$$

This way, the mesh is dense where the value of $\bar{f}(\tau)$ is large and is coarse where the value of $\bar{f}(\tau)$ is small. Figure 9.1 visualizes the general dependency between the density function, the cumulative distribution function and the distribution of the grid points. It can be seen that the density of grid points is high whenever the density function has large values. Besides, relation (9.4) is visualized on the vertical axis of the plot.

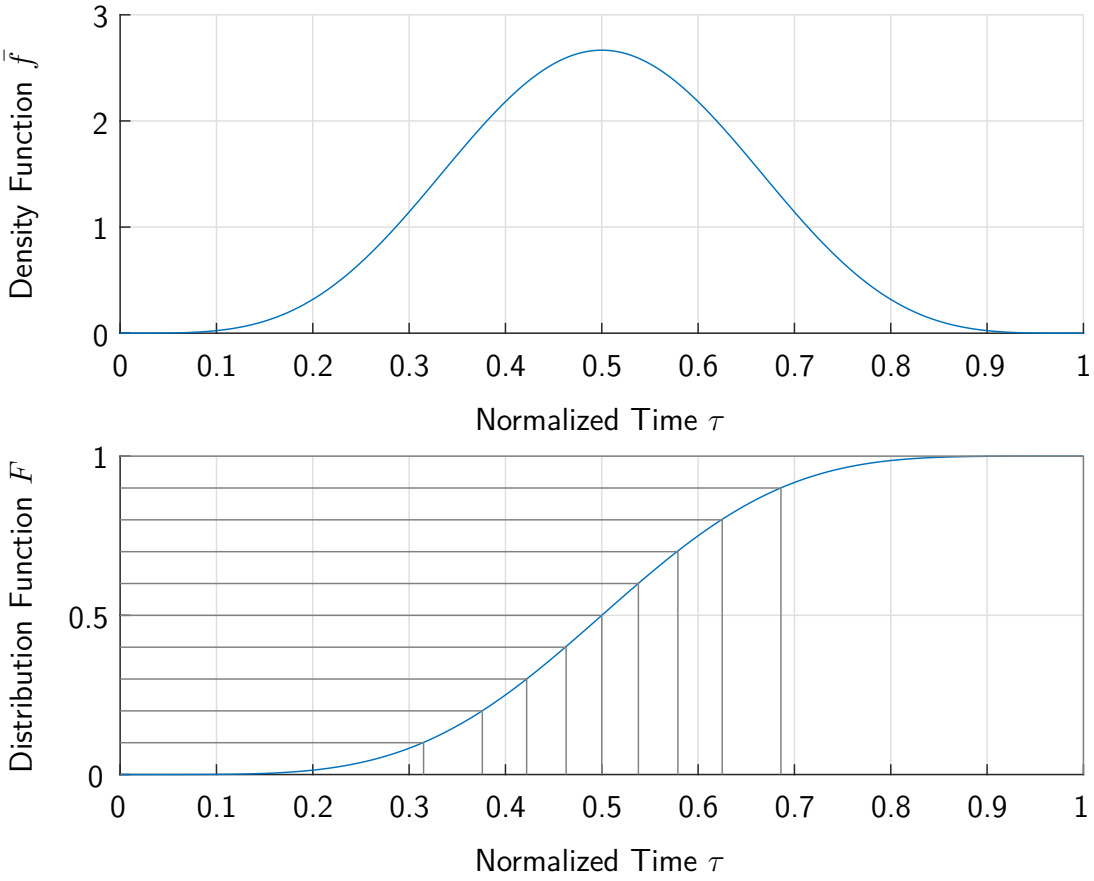

Figure 9.1: *Example for a densitiy function and its coresponding distribution function.*

As the calculation of all τ_i requires the solution of the non-linear equation (9.4) at each grid point – which may be time consuming – ZHAO and TSIOTRAS suggest the use of an interpolation method. Due to $\bar{f}(\tau)$ being non-negative on the relevant interval, $F(\tau)$ is strictly monotone and continuous. This property can be used to approximate the inverse

$$\tau_i = F^{-1}(y_i) \tag{9.5}$$

by interpolation. Therefore, the values of F are evaluated at some a priori selected points τ_k, which can then be used to construct a spline representation of $F^{-1}(y_i)$ and to calculate the required values τ_i.

In [ZT11] the authors use a density function of the form:

$$\rho_u(\tau) = c \cdot \kappa_u(\tau)^{1/3} \tag{9.6}$$

for the control u, where c is a constant and $\kappa_u(\tau)$ is the curvature of $u(\tau)$. As $u(\tau)$ is the result of the previous optimization and only known on the grid points used in that optimization, its curvature $\kappa_u(\tau)$ can not be evaluated exactly. Once again, [ZT11] suggest the use of an interpolating spline function that can then be differentiated analytically. Moreover, they show that the exponent should be selected as $\frac{1}{3}$ because this leads to the best piecewise linear approximation of the considered values. Consequently, their motivation is purely mathematical. [Bit+13a]

9.3 Physically Motivated Density Functions

The main issue when implementing the strategy described in the previous section is the calculation of the curvature of the control histories from their discretized representation. Therefore, a new approach for a density function is introduced that takes physical properties of the point mass simulation model from section 2.4 into account.

First of all, the influence of the thrust lever position is relatively uncritical as its dynamics and its effect on the resulting trajectory is in general rather slow. Hence, the important controls influencing the fast dynamics of a point mass simulation model are the *angle of attack* and the *bank rate* or the *bank angle* (depending on the chosen modeling) which are closely related to the *kinematic flight path inclination angle* γ_K^G and the *kinematic course angle* χ_K^G. Instead of using the curvature of the controls, here the curvature of their "*corresponding*" states is considered. This simplifies the further derivations as the first time derivatives of the states are part of the simulation anyway and do not need to be calculated by interpolation or finite differences. However, the idea that the histories of the states and the controls are precisely approximated by their discretized representations whenever the respective curvature is small, is kept. Additionally, the grids on which the states are evaluated during the optimization are normally finer than the control grids, which reduces the error of finite differencing for the calculation of the second time derivatives, too.

The following combined density function for grid refinement using a point mass simulation model as stated in section 2.4 is used here, while a similar formulation was suggested in [Bit+13a]:

$$f(\tau) = (\kappa_\chi + \kappa_\gamma)^e = (b(\tau))^e \tag{9.7}$$

where κ_χ is the curvature of the *kinematic course angle* χ_K^G, and κ_γ is the curvature of the *kinematic flight path inclination angle* γ_K^G.

9.4 Enhanced Refinement Strategy

When not using the curvature of the controls as a density function, the argument for the exponent $e = \frac{1}{3}$ presented in [ZT11] also becomes invalid. Furthermore, it might be preferable for the user to set limits for the local resolution instead of the number of grid points. This

means that the user of a grid refinement algorithm might not know how many points he would require to resolve the control history adequately, but on the contrary he might be able to state a maximum distance between two grid points as $\Delta\tau_{max}$ while their minimum distance shall be $\Delta\tau_{min}$. The exponent e can then be used to adapt these two values without changing the general shape of the density function and therefore not changing the locations of high or low density but only scaling their values with respect to each other.

This consideration leads to a distribution function

$$F(\tau_i) = \frac{\int_0^{\tau_i} b(t)^e dt}{\int_0^1 b(t)^e dt} = \frac{i-1}{N-1} \tag{9.8}$$

which has remaining degrees of freedom in e, N and the grid point values τ_i. Now, the user defined grid distances $\Delta\tau_{min}$ and $\Delta\tau_{max}$ have to be met by the resulting grid:

$$\Delta\tau_{min} = \min_i(\tau_{i+1} - \tau_i) \quad \wedge \quad \Delta\tau_{max} = \max_i(\tau_{i+1} - \tau_i), \qquad i \in 1, 2, ..., N-1 \tag{9.9}$$

As equations (9.8)–(9.9) cannot be solved analytically, a numerical optimization algorithm is used here to determine e and N in a first step. Therefore, the integral

$$\int_{\tau_i}^{\tau_i+\Delta\tau_{min}} b(t)^e dt \qquad \tau_i \in [0, 1[\tag{9.10}$$

is calculated for all τ_i in the interval. As the minimum distance between two grid points can be found at the position where integral (9.10) takes its maximum, the following equation must hold to achieve a minimum grid spacing $\Delta\tau_{min}$:

$$\max_{\tau_i} \left(\int_{\tau_i}^{\tau_i+\Delta\tau_{min}} b(t)^{e_{min}} dt \right) = \frac{\int_0^1 b(t)^{e_{min}} dt}{N-1} \tag{9.11}$$

It must be mentioned that e_{min} does not denote the minimum exponent, but the exponent belonging to the minimum grid distance. The equation for the maximum distance on the grid has to be fulfilled analogously:

$$\min_{\tau_i} \left(\int_{\tau_i}^{\tau_i+\Delta\tau_{max}} b(t)^{e_{max}} dt \right) = \frac{\int_0^1 b(t)^{e_{max}} dt}{N-1} \tag{9.12}$$

Once again, e_{max} is the exponent belonging to the maximum grid spacing. If the number of nodes N in the grid is fixed, in most cases e_{max} and e_{min} would be different, meaning that either equation (9.11) or equation (9.12) can be fulfilled, but not both. Using the degrees of freedom in changing N and e, the solution of the system of equations is possible in most cases. The quadratic optimization problem to determine e and N may consequently be stated

as:

$$\min \left(\left(\min_{\tau_i} \left(\int_{t_i}^{t_i+\Delta t_{max}} b(\tau)^e d\tau \right) - \frac{\int_0^1 b(t)^e dt}{N-1} \right)^2 + \left(\max_{\tau_i} \left(\int_{t_i}^{t_i+\Delta t_{min}} b(\tau)^e d\tau \right) - \frac{\int_0^1 b(t)^e dt}{N-1} \right)^2 \right) \tag{9.13}$$

In the application presented in section 11.8, this problem can be solved using *SNOPT* [GMS02] in some milliseconds. As this problem is solved in $\mathbb{R}$ for N and e, the number of grid points N resulting from the optimization is not from $\mathbb{N}$. This can easily be corrected by rounding up to the next integer, because the interpolation is getting better with every grid point that is added. This way, the minimum and the maximum grid distance requested by the user are somewhat undercut by the algorithm.

The calculation of the positions of the new set of grid points can now be performed similarly to *DENMRA*. The only difference is that the spline interpolation for the numeric inversion of $F(\tau)$ may in some cases not be necessary anymore, because the distribution function can be calculated on the relatively dense state grid. Hence, the values of the grid points can either be read from $F(\tau)$ directly or calculated by linear interpolation.

The principle described here can be adapted to higher fidelity modeling. There, the same density function might be used or even more sophisticated and more complex density functions may be created by the use of more states. In a rigid body simulation model, for example, all rotational rates of the aircraft are known and can be used to create an appropriate density function. [Bit+13a]

Section 11.8 presents the results that have been achieved for an exemplary air race trajectory using the grid refinement process described here.

Chapter 10

Applications Related to Commercial Air Traffic

In this chapter, optimal control problems from the field of commercial air traffic are considered. Several studies predict a further increase in air traffic above the world and especially above Europe. Even in the already lowered estimations of EUROCONTROL, published in September 2013 [NE13], the number of instrument flight rules (IFR) flight movements in 2019 is expected to be 10.8 million which is 14% more than in 2012. In their long term forecast [EUR10] a further growth is anticipated resulting in 40% more IFR flights in 2030 compared to 2009. Besides this growth in air traffic, EUROCONTROL records more and more problems with the current Air Traffic Management (ATM) system. Currently, the situation at hub airports and especially their Terminal Maneuvering Areas (TMA) is critical as they were growing dramatically during the last decade, with this trend still lasting. [Den01]

The studies presented here have similarly been published in the works [Bit+13b, Bit+14, Bit+15, Bit+16], before. Moreover, the works [FBH12, Ric+14c] from our group were, to some extent, used as a basis.

Currently, many big research projects like *SESAR* in Europe, *NextGen* in the USA or *CARATS* in Japan focus on the development of technologies and processes for the future ATM system. The goal of all these endeavors is to extensively reorganize the air traffic systems in order to bring safety and capacity to the next level while also reaching ecological and economical goals. Even though they are all using different solutions for the challenges, many ideas reappear in different places. These key technologies include, besides many others,

- *User preferred routing* [EE15], where the goal is to remove the currently existing airways and let each participant in the ATM system choose their path freely,
- *i4D trajectories* [EE15], that allow an execution of user preferred routing through an exact planing and prediction of the position of each participant at each moment in time, and
- *Free flight trajectories*, that result from the combination of the two aforementioned technologies.

When being able to perform flight operations based on free flight trajectories, an optimization of each trajectory (in the optimal control sense discussed here) becomes possible. This chapter presents some first results that have been achieved by performing simulation studies around the performance of the air traffic system and the operation of aircraft. [EE15, Bit+16]

In the remainder of this chapter three exemplary applications using the theoretical ideas presented in chapters 4–9 are investigated:

- First, optimal trajectories for an approach scenario are sought. Therein, 18 aircraft approach *Tokyo International Airport* simultaneously under the influence of wind disturbances. Besides, configuration changes in form of different flaps settings are included in the modeling.
- The second problem presented shows results for long haul trajectory optimization problems considering civil aerial refueling for efficiency improvements. The solutions acquired for the two considered scenarios – first one aircraft that is refueled once during the flight, and second two aircraft being refueled by the same tanker one after the other – show a significant decrease in fuel consumption when also considering aircraft that are optimized for long haul flights using aerial refueling.
- In the third and last air traffic related application, the trajectories of several aircraft passing through one common airspace sector are optimized. Thereby, not only the overall cost is considered, but also the attribution of the cost to the parties in the scenario. This way, a fair distribution of the costs should be reached. The scenario uses strongly simplified models and focuses on the application of multi criteria optimization methods on aircraft trajectory optimization problems.

As commercial airliners do not feature highly agile and fast varying dynamics, the models used are point mass models or even further simplified descriptions of aircraft movement. Hence, the available dynamic models from the *Base of Aircraft Data (BADA)* from *EUROCONTROL* can be used. In the examples presented below, models from *BADA Family 3* [EUR14] as well as *BADA Family 4* [EUR12] are utilized. For the multi criteria studies in section 10.4, even more simplified kinematic aircraft models have been used as no dynamic maneuvers are considered there.

The comparably simple dynamic models do not require the use of different modeling fidelities and combined discretization schemes. On the other side, the problems are relatively large, featuring many states and long simulation times. This, and the fact that the dynamics of the different aircraft are connected via separation constraints in the multi aircraft scenarios, require the exploitation of the problem sparsities on a local and a global scale (see chapter 7).

Furthermore, the fairness considerations of the last example presented in this chapter result in the formulation of a multi criteria optimal control problem that can be discretized leading to a multi criteria optimization problem. Consequently, the methodology from section 3.3 is required there.

In all examples described below, the use of appropriate initial guesses is inevitable. In order to be able to supply these initial guesses, methodologies based on the ideas from chapter 5 have been used. Therein, the complexity of the problems to be solved has been increased stepwise, always using the solution of one optimization as the initial guess for the next.

10.1 Literature Review

The computation of optimal trajectories for scenarios including multiple aircraft has received some attention during the last couple of years. In [FBH13] e.g., a very similar method as the one used in the first application here has been applied to optimize an approach scenario including four identical aircraft. There, a combined optimal control problem has been formulated that has been solved by first optimizing each individual trajectory and afterwards calculating the overall solution for the scenario.

The work of Mohan [MMR12] focuses on calculating a solution for obstacle avoidance and the generation of initial guesses based on a rapidly exploring random tree algorithm. Anyway, in the scenarios up to four aircraft landing on the same runway are considered.

The paper [Rag+04] uses a three degree of freedom model without detailed force or fuel flow equations, a collocation scheme for discretization, and *IPOPT* [WB06] to solve the numerical optimization problem. The example scenarios comprise up to eight aircraft and cylindrical separation zones.

[RH02] uses a linear 2D-model featuring four states (planar position and planar velocities) and linear constraints in order to build a *Mixed Integer Linear Program (MILP)*. The safe areas are modeled rectangular and the cost function considered is time. Finally, an example problem containing three aircraft is solved using *IBM ILOG CPLEX Optimization Studio* and the *AMPL* modeling language.

In [SHF04] a framework is presented that can be used for decentralized trajectory planning of multiple aircraft. The method is based on a receding horizon strategy and a MILP problem. The aircraft are modeled using linear dynamic systems with four states. The example problem comprises four aircraft.

Various approaches to automatic conflict resolution have been suggested and investigated in the past. Zhao and Schultz, in [ZS97] e.g., use optimal control methods to resolve conflicting trajectories for two aircraft. They minimize the deviations from the intended flight paths and finally enforce each aircraft back to their trajectory via terminal constraints.

In [Fra+01] Frazzoli et. al. use semidefinite programming to resolve conflicted trajectories in the horizontal plane. In contrary to that, in [HPS02] Hu et. al. consider three dimensional aircraft movement and present an algorithm to calculate optimal solutions for two-aircraft cases and an approximation for the multiple-aircraft case. In [MSS99], once again three dimensional problems are tackled and methods from optimal control are used to calculate free-flight trajectories for multiple aircraft.

In [Vis07] Visser describes a method of optimizing the trajectories for two aircraft in

three dimensional space with respect to noise nuisance on ground. VELA et. al. in [Vel+09] present a conflict resolution algorithm that incorporates a controller work load model and is near-realtime capable. Once again, two dimensional scenarios are considered.

The authors of [CRS11] use a MILP formulation to solve conflicts between arriving aircraft in a 4D approach scenario. Their objective is the minimization of the overall delay while keeping all inbound traffic conflict free.

Previously conducted research in the field of civil aerial refueling for fuel saving has mainly been driven by NANGIA [Nan06a, Nan06b, Nan08], GREEN [Gre02], and HAHN [Hah07]. NANGIA defines multiple efficiency values for measuring the benefits that can be achieved when using air-to-air refueling for civil configurations. Besides, he focuses on the design of new aircraft specifically tailored for long-haul flights using aerial refueling. This means that the aircraft should have more or less the same payload and size as actual long-range planes but have smaller tanks which influence the whole configuration and the aerodynamics. GREEN and HAHN focus on the calculation of the most efficient range as well as a similar redesign of aircraft as NANGIA.

In the work [Sol+12] of SOLER et. al. a conflict avoidance scenario is considered where two aircraft are passing through an airspace sector and a collision needs to be avoided using optimization. The aircraft are modeled using a point mass simulation and different flight modes. Moreover, hazardous weather areas are modeled as avoidance zones. The discrete flight mode switches result in a *Mixed Integer Optimal Control Problem (MIOCP)* that is then reformulated as a *Mixed Integer non-linear Programming Problem (MINLP)*. In [Sol+16] a similar solution technique has been applied, now solving a scenario including seven aircraft. The authors note that the minimization of the overall costs does not necessarily result in a fair distribution of the costs to the participants in the scenario.

ARCHIBALD et. al. in [Arc+08] describe a multi-agent solution for solving conflicts based on *satisficing game theory*. In this theory a decision maker (in this case an aircraft) may sacrifice part of its achievement in case another decision maker can benefit from that. This way, the overall performance may be improved for the cost of several individuals. This problem formulation is close to the problem solved in section 10.4 but tackled quite differently.

In [CHL10] CHALOULOS et. al. present a decentralized model predictive control scheme (MPC) for hierarchical systems and use it to solve collision avoidance problems for unmanned aerial vehicles (UAVs). Besides the high level MPC approach, a low level controller is implemented that takes care of the internal dynamics of the aircraft. Here, a cooperative cost is used to ensure a certain level of fairness between individual and overall costs in the scenarios.

In [TKS13], a multi criteria aircraft trajectory optimization problem is solved using a genetic algorithm combined with the stochastic search optimizer *Tabu search 2*. The goal of the optimization is to understand how environmental impact and operating cost of commercial airliners are related to each other. The scenario that is considered in the work is the climb phase of an Airbus A320. [Bit+14, Bit+15]

10.2 Approach Scenarios Including Several Aircraft, Configuration Changes and Disturbances

In the problem considered in this example, cost-index-optimal trajectories for multiple aircraft approaching an airport in the presence of wind disturbances are calculated. Besides the continuous controls, configuration changes in the form of the extension of flaps – that may be interpreted as discrete controls – are considered with the optimal times of extension being calculated.

The main result presented here is a modeling technique and a multi stage solution process for large scale trajectory optimization problems from the field of ATM. In the first stage of this process, each trajectory is optimized individually, before afterwards multiple problems for an increasing number of aircraft are solved. Finally, optimal trajectories for all aircraft in the scenario result, that adhere to the flight envelopes and separation limits while minimizing the total cost index summed up over all aircraft in the scenario.

In order to be able to solve the large scale optimal control problems that result from the scenario, the methods for initial guess generation presented in chapter 5 and especially those concerning problem sparsity from chapter 7 have been used.

The aircraft dynamics are simulated using point mass simulation models in three dimensional space with the aerodynamic model and the fuel flow models taken from the *Base of Aircraft Data Family 4 (BADA 4)* [EUR12]. The considered scenario is based on real approach trajectories extracted from *MLIT CARATS Open Data* published by the *Japanese Ministry of Land, Infrastructure, Transport and Tourism* [Min15].

Besides, the scenario includes the influence of wind, modeled based on data from the *Earth System Research Laboratory* [Com+11]. In the scenario, the optimal state and control histories for 18 aircraft as well as the optimal points in time for deploying the flaps are determined. The inherent discrete decision problem of sequencing the aircraft is automatically solved by the numerical optimization algorithm in parallel to the calculation of the optimal trajectories. However, as a gradient based optimization algorithm is used, only local optimality can be achieved.

The main results for this example have been published in the conference papers [Bit+15, Bit+16], before.

10.2.1 Aircraft Simulation Model

In this example, the point mass simulation model from section 2.4 for a flat and non rotating earth is used, with the aerodynamics, the fuel flow model and other aircraft specific information taken from the *BADA4* published by EUROCONTROL [EUR12]. The simulation is performed with respect to a locally fixed coordinate frame and includes a dynamic atmosphere model as described in section 2.6.2. The model is directly controlled by the lift coefficient C_L, the aerodynamic bank angle μ_A and the thrust lever position δ_T. Furthermore, the spoiler

deflection δ_{SB} is considered as a continuous model input. The flaps positions – which may be seen as discrete controls – are handled using a hyperbolic tangent function that uses the time of extension as variable, which eventually is continuous. The modeling used for the flaps is described in more detail in section 10.2.4. Overall, the resulting dynamic system for *each* aircraft features seven states and four controls as listed in table 10.1.

The fuel flow model for each aircraft is taken from the *BADA* data [EUR12] and incorporated in the system dynamics using the equations from section 2.8.1. [Bit+15, Bit+16]

Table 10.1: *States and controls of each aircraft in the scenario.*

Name	State	Unit
$(x)_N$	Northward position in locally fixed Navigation Frame N	m
$(y)_N$	Eastward position in locally fixed Navigation Frame N	m
$(z)_N$	Downward position in locally fixed Navigation Frame N	m
V_K	Absolute kinematic velocity	m/s
χ_K	Kinematic course angle	rad
γ_K	Kinematic climb angle	rad
m	Aircraft mass	kg
Name	**Control**	**Unit**
C_L	Aerodynamic lift coefficient	-
μ_A	Aerodynamic bank angle	rad
δ_T	Thrust lever position (normalized)	-
δ_{SB}	Spoiler position (normalized)	-

10.2.2 Wind Model

In this example, the wind field is modeled to be time invariant, yet depending on the current air pressure, which changes with altitude – however, no changes in the lateral extent are considered. The profile of wind speeds used is an approximation of the *Twentieth Century Reanalysis (V2)* wind data from the *Earth System Research Laboratory* [Com+11]. Based on the daily mean wind for the region around Tokyo International Airport, polynomial wind models of sixth order for northward and eastward wind speeds have been derived. The wind speed components resulting from a fit of the aforementioned polynomials to the wind data, results in the following model for the two wind speed components:

$$\begin{aligned} V_{W,u/v} = {} & p_{1,u/v} \cdot (p_H)^6 + p_{2,u/v} \cdot (p_H)^5 + p_{3,u/v} \cdot (p_H)^4 \\ & + p_{4,u/v} \cdot (p_H)^3 + p_{5,u/v} \cdot (p_H)^2 + p_{6,u/v} \cdot p_H + p_{7,u/v} \end{aligned} \tag{10.1}$$

Therein, p_H is the numerically normalized pressure altitude:

$$p_H = \frac{p - 442.9 \text{ mbar}}{327.9 \text{ mbar}} \tag{10.2}$$

This normalization does not have any physical meaning but is only necessary for numerical reasons. After performing a model fit for February 3rd, 2012 the polynomial coefficients given in Table 10.2 result.

Table 10.2: *Coefficients of the approximation polynomials for the wind speed components. All values are calculated for wind speeds in [m/s]. [Bit+16]*

Coefficient	Eastward wind (u)	Northward wind (v)
$p_{1,u/v}$	-3.409	1.584
$p_{2,u/v}$	12.950	-0.08236
$p_{3,u/v}$	-15.310	-9.390
$p_{4,u/v}$	1.831	-1.133
$p_{5,u/v}$	20.220	20.750
$p_{6,u/v}$	-41.320	-2.118
$p_{7,u/v}$	37.700	-7.854

The static air pressure p around a particular aircraft is calculated based on the current flight altitude according to the *Standard Atmosphere* model as presented in section 2.6.1. Using this model, the wind speeds can also directly be mapped to the altitude. Figure 10.1 depicts the result of this mapping for the two wind speed components considered in the scenario.

In order to incorporate the wind model into the flight system dynamics, the relations from section 2.6.2 have been used. [Bit+16]

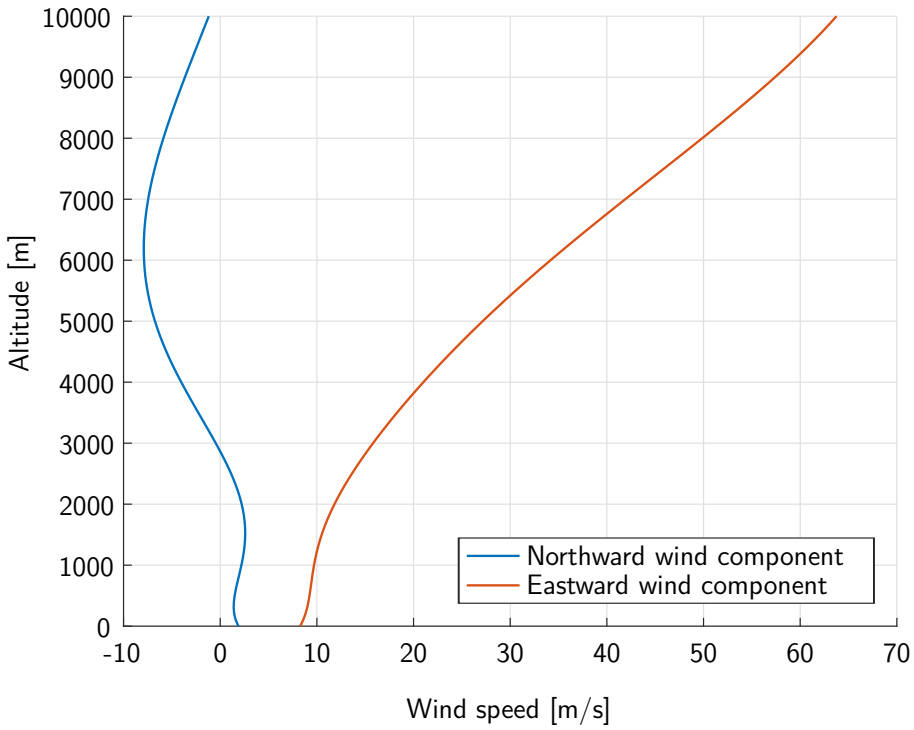

Figure 10.1: *Wind speed components [m/s] over altitude [m].*

10.2.3 Force Model

The aerodynamic forces acting on each aircraft in the scenario are calculated as described in the *BADA* documents [EUR12]. Besides others, the forces depend on the Mach number M, the current speed of sound a, the adiabatic index κ, the universal gas constant R, and the current air temperature T that are all calculated according to atmosphere description from section 2.6.1. [Bit+15, Bit+16]

10.2.4 Discrete Configuration Changes

The extension of the flaps of each aircraft – representing a discrete configuration change – is modeled using a function based on hyperbolic tangent expressions for each position s of the flaps that is considered. In this analysis, the main influencing parameter is the point in time $t_{\text{flaps},s}$ when the change of the flaps position takes place. This parameter is mapped to the relative extension of the respective position $k_{\text{flaps},s} \in [0, 1]$ by:

$$k_{\text{flaps},s} = 0.5 \cdot \tanh(\nu(t - t_{\text{flaps},s})) + 0.5 \tag{10.3}$$

The parameter ν may be used to adjust the speed of fading between two flaps positions. A value of $\nu = 1$ has been used here, as this results in a change time of two flaps positions of approximately four seconds, which is assumed to be realistic. Within the model, the value for any exemplary aerodynamic parameter p_A (e.g. drag coefficient $C_{D,0}$) is calculated by fading between the parameters for the different positions (e.g. position 1 and position 2) of the flaps while the extension is performed:

$$\begin{aligned} p_A = {} & p_{A,\text{flaps, clean}} \\ & + (p_{A,\text{flaps},1} - p_{A,\text{flaps, clean}}) \cdot k_{\text{flaps},1} \\ & + (p_{A,\text{flaps},2} - p_{A,\text{flaps},1}) \cdot k_{\text{flaps},2} \\ & + \dots \\ & + (p_{A,\text{flaps},N} - p_{A,\text{flaps},n-1}) \cdot k_{\text{flaps},n} \end{aligned} \tag{10.4}$$

In equation (10.4), the values $p_{A,\text{flaps},s}$ hold the aerodynamic parameters that are valid in case the flaps are extended to position s. The fading in the parameters takes place linearly – however, as the relative extension parameters $k_{\text{flaps},s}$ are calculated based on an hyperbolic tangent, the resulting aerodynamic parameters are also faded based on hyperbolic tangent functions.

As every aerodynamic parameter for a certain flaps position depends on the parameter of the previous position, the switching times need to be in proper order, which can be achieved by the additional constraints:

$$t_{\text{flaps},1} < t_{\text{flaps},2} < \dots < t_{\text{flaps},n} \tag{10.5}$$

In this example, besides the clean configuration, two flap configurations have been selected and considered for each aircraft. As the final point of the optimization is before the final

approach fix, the aircraft still have time to establish the landing configuration. Consequently, in the optimization no particular final flaps position is required. [Bit+16]

10.2.5 Flight Envelope of Each Aircraft

Along the trajectory of each aircraft, the following path constraints are required in order to obtain realistic flight paths [Bit+15, Bit+16]:

- The load factor in z-direction of the Kinematic Frame K is limited to:

$$0.8 \leq (n_z)_K \leq 1.2 \tag{10.6}$$

  This ensures that a smooth trajectory results that fulfills the requirements for passenger comfort.

- Moreover, the MACH number is limited by:

$$M \leq 0.85 \tag{10.7}$$

  in order to avoid structural damage due to over-speeding.

- For a good pilot acceptance of the trajectories, all aircraft need to be decelerating reasonably all the time by means of calibrated airspeed:

$$-0.8\frac{m}{s^2} \leq \dot{V}_{CAS} \leq 0\frac{m}{s^2} \tag{10.8}$$

- Again, for pilot acceptance as well as passenger comfort, the aircraft must not climb during the approach and the vertical speed is limited to approximately $-2560\frac{ft}{min} \approx -13\frac{m}{s}$:

$$-13\frac{m}{s} \leq \dot{h} = -\dot{z} \leq 0\frac{m}{s} \tag{10.9}$$

- The set of admissible controls for each aircraft is defined by:

$$\begin{aligned} 0 &\leq C_L &&\leq 1.4 && (10.10)\\ -30 \cdot \frac{\pi}{180} &\leq \mu_K &&\leq 30 \cdot \frac{\pi}{180} && (10.11)\\ 0 &\leq \delta_T &&\leq 1 && (10.12)\\ 0 &\leq \delta_{SB} &&\leq 1 && (10.13) \end{aligned}$$

  In this example the lift factor limits are not depending on the flaps configuration but are defined constant along the whole trajectories and for all aircraft. For passenger comfort, the kinematic bank angle μ_K is limited to be between $\pm 30\,\mathrm{deg}$. As the thrust lever position as well as the speed brakes are normalized, their values have to remain within the interval $[0, 1]$.

10.2.6 Combined Dynamic Model

In order to be able to compare the position of all aircraft at the same points in time, a simulation on one common grid is inevitable as the final arrival times of all aircraft are not known in advance and may change during the optimization process. The overall optimal control problem is modeled in one single phase, aggregating all aircraft dynamics in one simulation model. As not all aircraft enter or leave the considered air space at the same time, the respective dynamics are faded in or out using a hyperbolic tangent function:

$$\dot{\mathbf{x}}_i = \mathbf{f}_i(\mathbf{x}_i, \mathbf{u}_i, \mathbf{p}_i) \cdot k_{\mathsf{fade},i} \tag{10.14}$$

$$k_{\mathsf{fade},i} = (0.5 \cdot \tanh(\nu\,(t - t_{ini,i})) + 0.5) \cdot (-0.5 \cdot \tanh(\nu\,(t - t_{end,i})) + 0.5) \tag{10.15}$$

By changing ν, the steepness of the fading functions can be adjusted. In this example, $\nu = 1$ is used, resulting in a fade-out time of approximately four seconds which showed good numerical behavior in the calculations. As the time when the aircraft enter the considered airspace is known, the initial times $t_{ini,i}$ are fixed while the final times $t_{end,i}$ are subject to optimization. The same fading is used for the path constraints.

Consequently, the state, control, and parameter vectors of the multi aircraft optimization problem are a combination of the respective vectors for each single aircraft:

$$\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N)^\mathsf{T} \tag{10.16}$$

$$\mathbf{u} = (\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_N)^\mathsf{T} \tag{10.17}$$

$$\mathbf{p} = (\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_N)^\mathsf{T} \tag{10.18}$$

The same holds for the dynamic equation:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{p}) = [\mathbf{f}_1(\mathbf{x}_1, \mathbf{u}_1, \mathbf{p}_1), \mathbf{f}_2(\mathbf{x}_2, \mathbf{u}_2, \mathbf{p}_2), ..., \mathbf{f}_N(\mathbf{x}_N, \mathbf{u}_N, \mathbf{p}_N)]^\mathsf{T} \tag{10.19}$$

The main drawback of the simulation on the common grid is the size of the resulting simulation model which contains seven states and four controls *per* aircraft in the scenario. Thus, for 18 aircraft an overall simulation model containing 126 states and 72 controls results. However, as the states and controls of the individual aircraft are not connected to each other in the first place, the gradient of the simulation model features a special local sparsity structure (as presented in chapter 7) that will be shown in more detail in section 10.2.9. [Bit+15, Bit+16]

10.2.7 Separation

In addition to the flight envelope constraints for every individual aircraft – specified in section 10.2.5 – temporal and spatial separation between the aircraft has to be ensured for the whole trajectories and especially at the end of the flights prior to landing. The times of arrival at the final point for each pair of aircraft are constrained by:

$$(t_{end,i} - t_{end,j})^2 \geq {t_{min\,Sep}}^2, \qquad \forall (i,j), i \neq j \tag{10.20}$$

with $t_{min\,Sep} = 60s$ in this example. Besides, the spatial distance between each pair of aircraft, modeled as a rotational ellipse, has to remain above the separation limit at all times:

$$((x)_{N,i}(t) - (x)_{N,j}(t))^2 + ((y)_{N,i}(t) - (y)_{N,j}(t))^2 + \left(\frac{R_{xy}}{R_z}\right)^2 ((z)_{N,i}(t) - (z)_{N,j}(t))^2 \geq {R_{xy}}^2,$$
$$\forall (i,j), i \neq j \tag{10.21}$$

The radius R_{xy} in the horizontal plane is selected depending on the aircraft weight classes according to table 10.3, while the vertical separation $R_z = 600ft = 183m$ is constant for all combinations. In contrary to the general separation minima defined in the *ICAO* documents [ICA01], in the model used no distinction with respect to flight directions can be made. This includes that in cases where one aircraft follows the other no differentiation between the leader and the follower is possible. As wake vortices, which have the most critical effect, strongly depend on the order of the aircraft, the modeling is conservative in this respect because for all aircraft combination the worst sequencing is assumed when calculating the separation limits. However, in this approach air traffic safety is not violated, while there may remain some unused performance potential in the solutions. [Bit+15, Bit+16]

Table 10.3: *Aircraft weight categories and separation limits during approach. Based on information taken from [ICA01].*

Weight Category Aircraft A	Weight Category Aircraft B	Minimum Separation [NM]	Minimum Separation [m]
H	H	4	7408
H	M	5	9260
M	M	3	5556

10.2.8 Cost Modeling

The cost function to be minimized in this example consists of three weighted parts, being

- the fuel consumption,
- the flight times,
- and a small auxiliary term penalizing the control effort during the faded-out parts of the trajectories.

The overall fuel consumption is calculated from the initial and the final mass of each aircraft by

$$J_{fuel} = \sum_{i=1}^{N} (m_{ini,i} - m_{end,i}) \tag{10.22}$$

The time cost is calculated from

$$J_{time} = \sum_{i=1}^{N} t_{end,i} \tag{10.23}$$

where the initial times are omitted as they are constant.

The weighting of these first two cost functions is selected such that a cost index

$$CI = \frac{1\text{cost}/1000sec}{1\text{cost}/100kg} = 6\frac{kg}{min} \tag{10.24}$$

results. This very low cost index leads to almost fuel minimal trajectories in the solution.

The third auxiliary part of the cost function is required as otherwise the controls in the faded-out parts of the trajectories would not influence the solution of the optimization problem at all, resulting in undefined values. Consequently, the auxiliary control effort is modeled as

$$J_{con} = \sum_{i=1}^{N} \int_{t_{ini,i}}^{t_{end,i}} \mathbf{u}_i(t)^\intercal \mathbf{u}_i(t) \cdot (1 - k_{\text{fade},i})\, dt \tag{10.25}$$

with k_{fade} being the fading factor from equation (10.15). The auxiliary control cost is weighted by $w_{con} = 0.01$ and does not influence the relevant part of the solution.

The total cost can be calculated by weighting the separate parts and summing up:

$$J = w_{fuel} \cdot J_{fuel} + w_{time} \cdot J_{time} + w_{con} \cdot J_{con} \tag{10.26}$$

Similar cost models have also been used in [Bit+15, Bit+16].

10.2.9 Sparsity

In chapter 7 the effects of problem sparsity on the solution process have been discussed. In order to be able to solve large scale non-linear optimal control problems, like the one presented here, the exploitation of the problem sparsity is inevitable.

In this example, the model dynamics are block-diagonal sparse because of the different dynamic models being concatenated together in one overall simulation model. When using the model builder features of *FALCON.m*, as briefly presented in section 4.8.4, the tool determines the sparsity structure of a model and all constraints automatically together with the automatic differentiation before supplying this information to the numerical solver. In this example, the automatic differentiation has only been used to create the simulation models for the individual aircraft while the combination of the models was done manually.

Figure 10.2 shows the local sparsity pattern of the simulation model, where the dots represent non-zero matrix entries. The first block of the matrix represents the gradient of the state derivative with respect to the states $\frac{\partial \dot{\mathbf{x}}}{\partial \mathbf{x}}$, the second part shows the gradient with respect to the controls $\frac{\partial \dot{\mathbf{x}}}{\partial \mathbf{u}}$ and the third part represents the gradient with respect to the parameters $\frac{\partial \dot{\mathbf{x}}}{\partial \mathbf{p}}$. These are the fade in time $t_{ini,i}$, the fade out time $t_{end,i}$, the extension time of the flaps to the first stage $t_{\text{flaps},1,i}$, and the time when the second stage of the flaps becomes active $t_{\text{flaps},2,i}$.

After combining the simulation models for the 18 aircraft in the scenario, the structure from figure 10.3 results. Each block along the diagonal of this combined sparsity pattern features

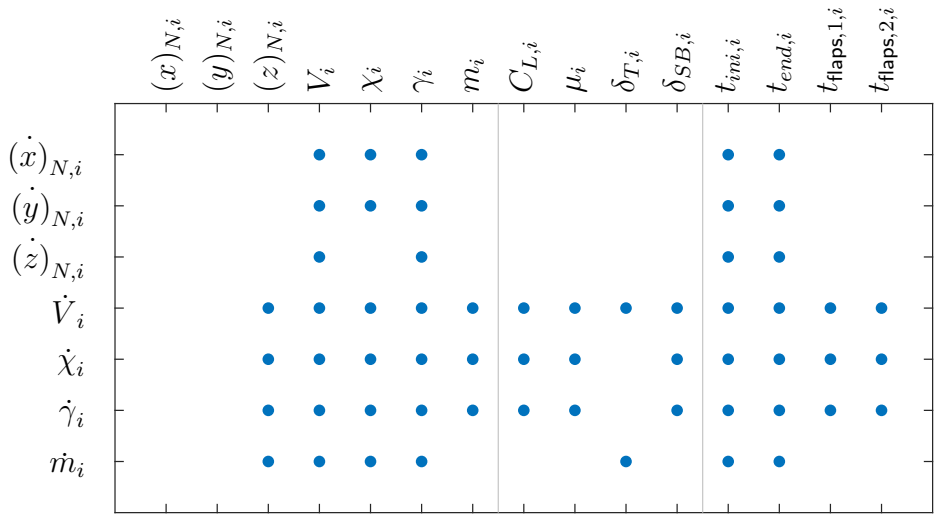

Figure 10.2: *Local sparsity pattern for the gradient of the simulation model consisting of the three components:* $\left[\frac{\partial \dot{\mathbf{x}}}{\partial \mathbf{x}} \frac{\partial \dot{\mathbf{x}}}{\partial \mathbf{u}} \frac{\partial \dot{\mathbf{x}}}{\partial \mathbf{p}}\right]$. *(Compare [Bit+16])*

the structure depicted in figure 10.2. The gradient structure of the flight envelope path constraints can be constructed similarly; that for the separation constraints creates relations between the different aircraft as will be seen further below.

Using the overall model sparsity together with the ones from the constraints, the global sparsity pattern of the problem can be calculated by sorting the variables and applying the discretization scheme. Figure 10.4 shows the global sparsity pattern of the discretized problem with the cost function gradient being located in the first row and all constraints listed below. The constraints contain all path constraints and collocation defects, and are sorted with respect to simulation time. Each dot on the diagonal in the figure comprises all information mentioned above for all aircraft at one point in time. The first columns of the matrix keep the derivatives with respect to the parameters appearing in the problem. The gradient of the problem discretized on a grid with 801 nodes contains $2.911 \cdot 10^6$ non-zero elements of $4.689 \cdot 10^{10}$ possible elements resulting in an approximate sparsity of 99.9938%.

Figure 10.5 depicts the top left corner of the sparsity of the gradient matrix in more detail. Again, the first row contains the gradient of the cost function, while all rows below contain the gradients of the various constraints. Table 10.4 lists the blocks of the matrix based on the row and column structure. It can clearly be seen how the separation constraints link the states of the different aircraft to each other. The separation constraints are sorted such that first the distance between the first and all other aircraft are calculated. Afterwards, the distance between the second aircraft and all subsequent aircraft are calculated (as the distance between the first and the second aircraft has already been evaluated). This is continued until finally the separation between the last two aircraft is constrained. [Bit+16]

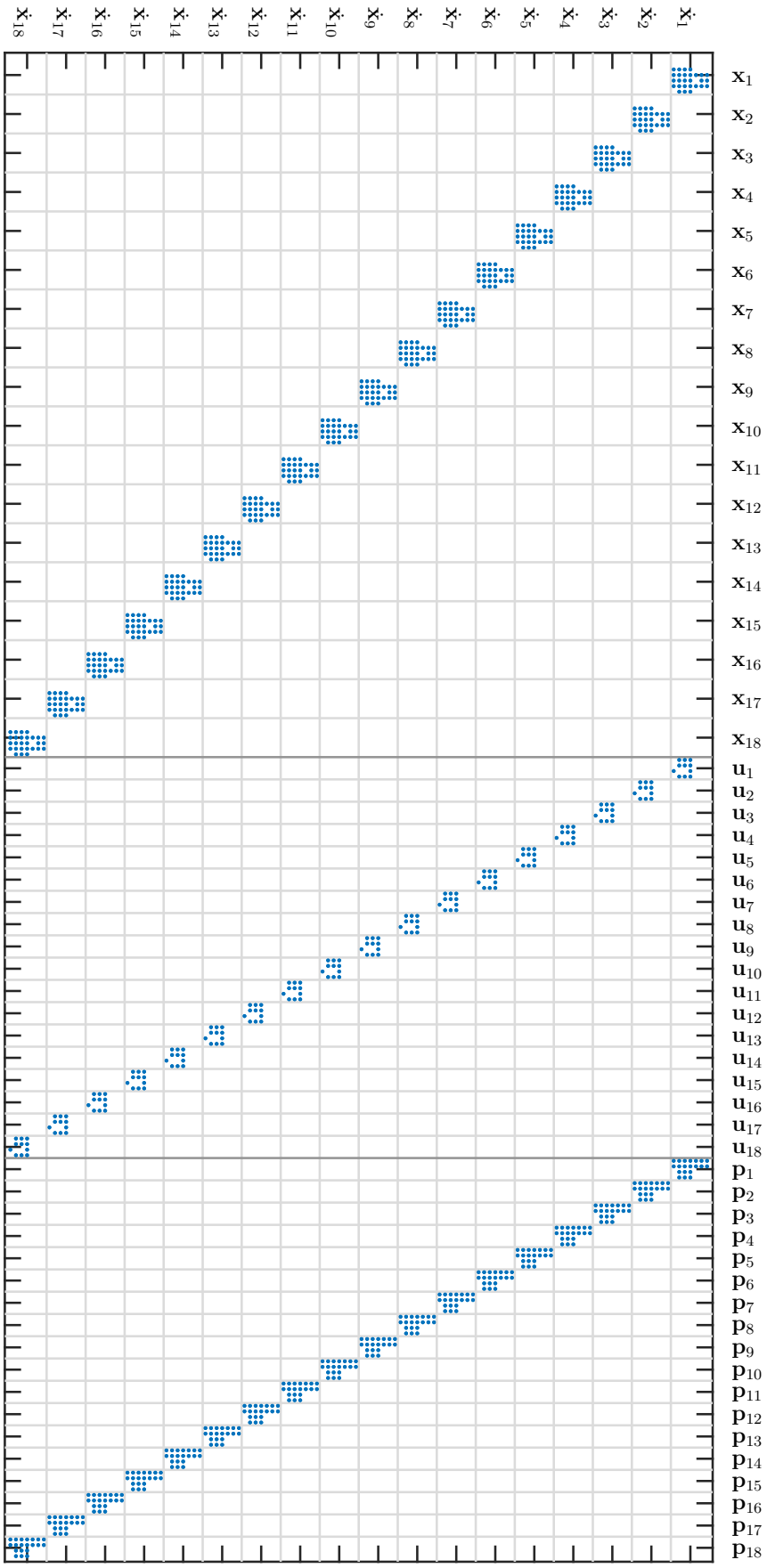

Figure 10.3: *Overall model sparsity. (Figure rotated for better readability.)*

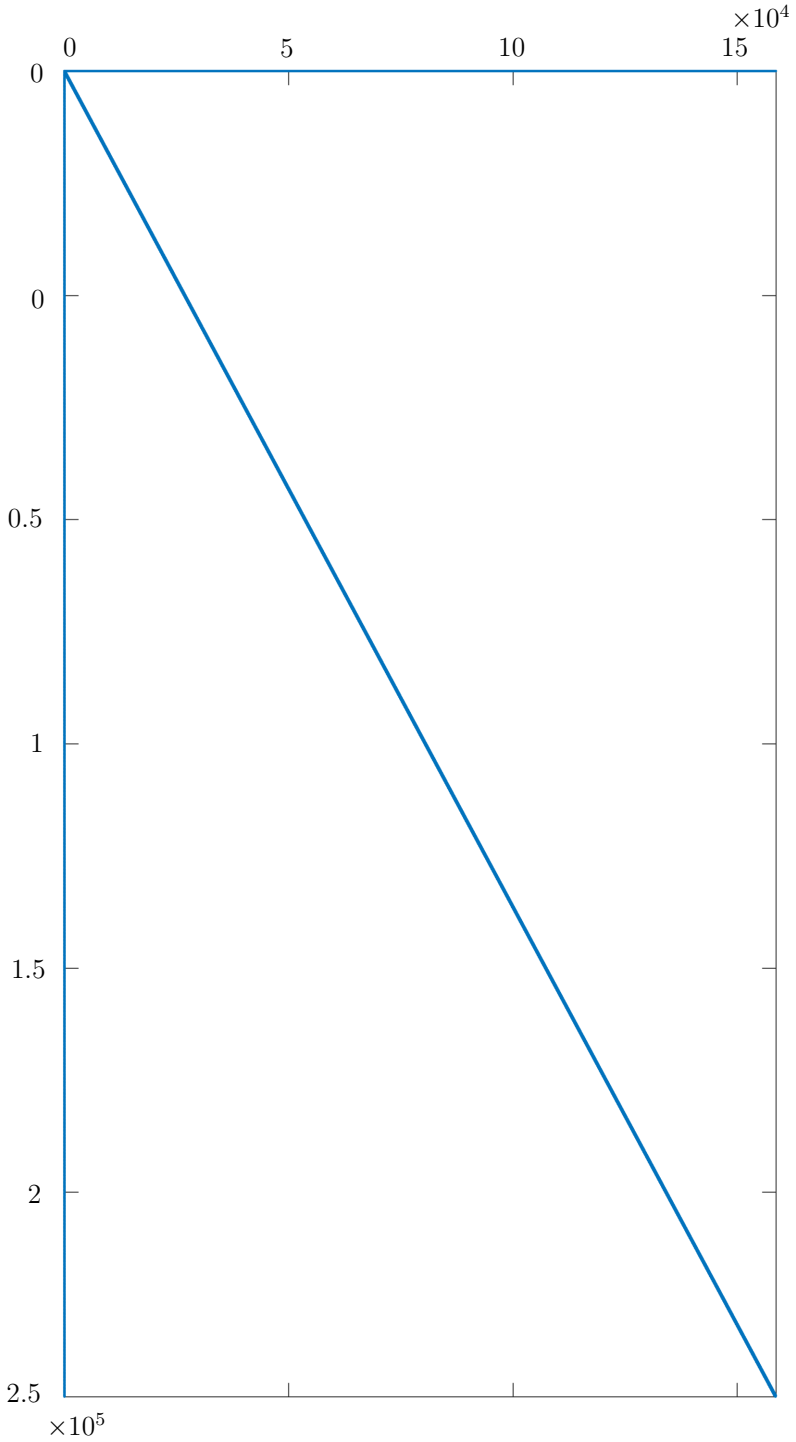

Figure 10.4: *Global sparsity pattern of the gradient of the discretized optimal control problem. The optimization parameters are accounted for from left to right while the cost function and the constraints appear from top to bottom. [Bit+16]*

10.2.10 Solution Process

The combined optimal control problem ensuring separation at all times cannot be directly solved due to its high non-linearity and its large size. Therefore, the following multi-stage homotopy based process is used (see also section 5.1.2):

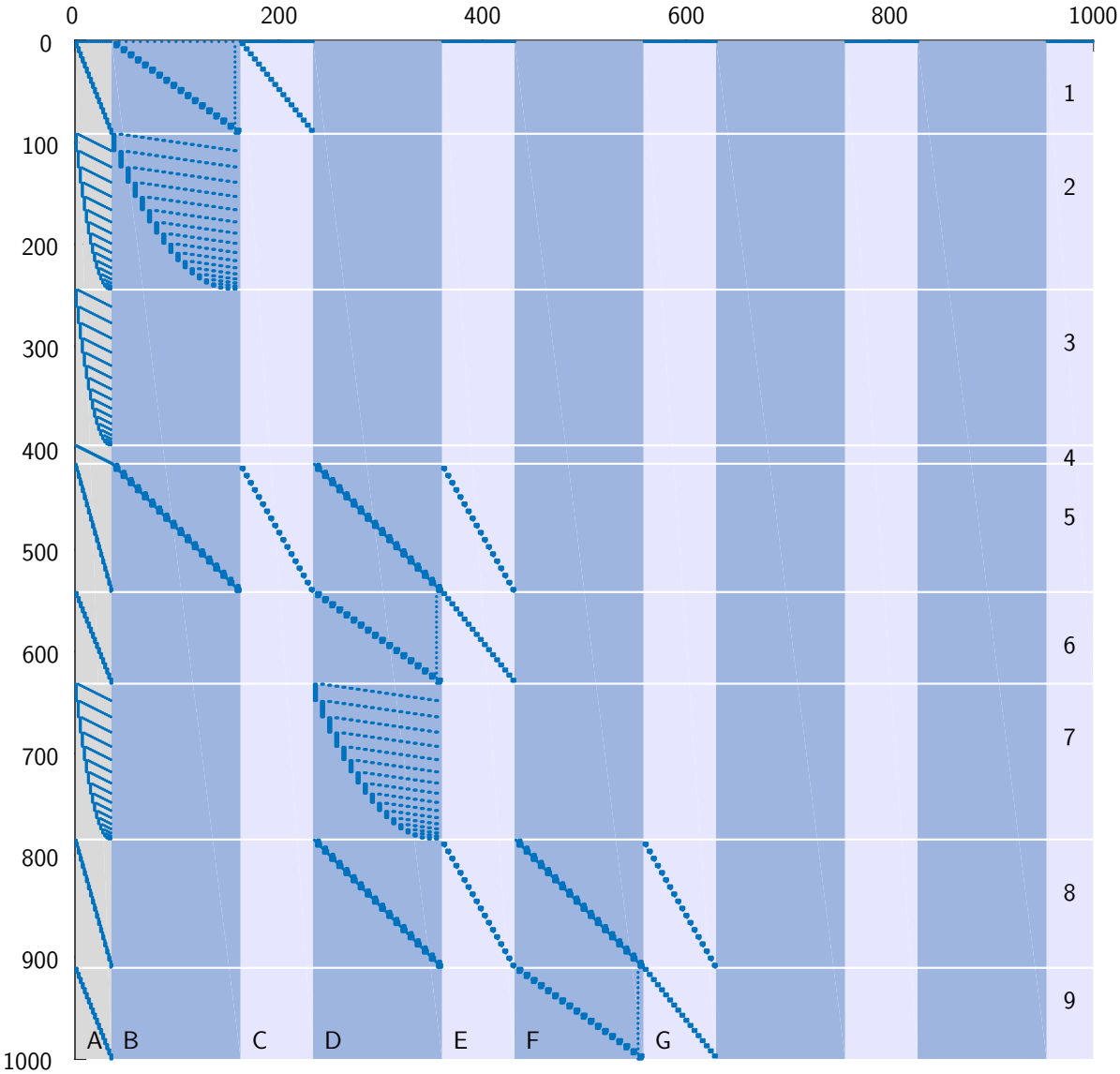

Figure 10.5: *Detail of the upper left corner of the global sparsity pattern of the discretized optimal control problem. The rows define which value is differentiated in the gradient and the columns define with respect to which value. The description of the rows and columns can be found in table 10.4.*

1) Calculate an optimal approach trajectory for each aircraft individually.
2) Start with the first aircraft in the scenario and add the aircraft to the fully constrained problem one after another, until all aircraft are considered.

In each optimization step, the states, the controls, the parameters, and all LAGRANGE multipliers (dual variables) for the collocation defects and the constraints are used as initial guesses for the next problem to be solved. Unfortunately, the process described in section 5.1.3 may not be used here as the addition of an aircraft to the scenario may not be described by a

Table 10.4: *Elements of the gradient of the discretized optimal control problem. The table explains figure 10.5. The rows define which value is differentiated in the gradient and the columns define with respect to which value.*

Row	Description
1	Initial flight envelope constraints
2	Initial separation constraints
3	Final time separation constraints
4	Flap switching time constraints (to ensure proper ordering)
5	First discretization defect
6	Flight envelope constraints at second point in time
7	Separation constraints at second point in time
8	Second discretization defects
9	Flight envelope constraints at third point in time

Column	Description
A	Parameters: $t_{ini,i}$, $t_{end,i}$, $t_{\text{flaps},1,i}$, $t_{\text{flaps},2,i}$, $\forall i \in [1, N]$
B	Initial states
C	Initial controls
D	States at second time step
E	Controls at second time step
F	States at third time step
G	Controls at third time step

continuous parameter. Moreover, every additional aircraft changes the general structure of the problem and it is expected that it also changes the set of active constraints. [Bit+16]

10.2.11 Scenario and Results

The scenario considered in this exemplary study focuses on the approach of 18 aircraft to Tokyo International Airport and has been constructed using *MLIT CARATS Open Data* released by the Japanese *Civil Aviation Bureau of the Ministry of Land, Infrastructure, Transport and Tourism (MLIT JCAB)*. Thereof, track data from the 4$^{\text{th}}$ of March 2013 in the time between noon and 6 o'clock in the evening has been used. From this data, 18 tracks have been extracted for constructing the scenario. Table 10.5 lists the 18 flights that have been considered, together with the aircraft types, the weight categories, the initial positions, and the relevant times. It has to be mentioned here that the wind data and the track data used do not cover the same day as no such data was available to the author. This does not influence the results significantly. However, no comparison of the optimization results with the real track data is possible under these circumstances.

Table 10.5: *Data for the aircraft and the trajectories in the example scenario. [Bit+16]*

AC Type	Weight Cat	Start Lat [deg]	Start Lon [deg]	Start Altitude [m]	Start Time [s]	Final Time [s]	Flap Time [s]
B738W26	M	34.1419	138.2522	8440	0	1420	-
A330-341	H	34.1947	138.3352	10020	0	1266	1238
B738W26	M	34.2130	138.7952	8530	0	1125	-
B772LR	H	34.5399	137.2626	10659	1	1645	-
B738W26	M	36.4880	140.2920	4916	2	1065	-
B772LR	H	34.6585	138.7760	8564	9	835	445
B738W26	M	36.9979	140.1037	7010	23	1360	-
A321-131	M	34.8133	137.0220	10058	33	1820	-
B752WRR40	M	34.7052	137.0034	11278	308	2145	-
B738W26	M	34.0073	137.7211	10052	493	2085	-
B738W26	M	34.7028	137.0076	10659	515	2740	-
B73423	M	34.0081	137.8762	10355	562	2025	-
A320-231	M	36.9949	140.4129	7910	569	1960	-
B738W26	M	34.7031	137.0272	12490	594	2221	-
B772LR	H	34.5353	137.0278	11271	869	2661	-
B763ERGE61	H	34.0140	137.7094	12497	1005	2601	-
B738W26	M	34.0067	137.9777	10058	1024	2431	-
B772LR	H	35.1811	137.0097	8120	1384	3171	-

Results

In figure 10.6 the resulting optimal lateral tracks for all aircraft can be seen. They are mostly straight lines as this is the most time and fuel efficient way to go. Almost all arising conflicts can be solved within the optimization by adjusting the vertical profiles and the velocities along the tracks. Only one aircraft is rerouted by the optimization algorithm in order to ensure final time separation. At this point the main weakness of the algorithm becomes visible as (with every gradient based approach) only local minima can be found and it cannot be determined whether rerouting a different aircraft would lead to another (maybe even better) locally optimal solution. Moreover, other solutions may be found by changing the way the initial guesses are calculated or by altering the solution process described in section 10.2.10. The small arcs in the trajectories of the aircraft coming from the north are related to the partially strong eastward wind and its dependence on the flight altitude.

Figure 10.7 shows a close-up of the last part of the trajectories. Therein, the tracks of all aircraft are visualized using *Google Earth* and the positions at one point in time is plotted. The white ellipsoids have a diameter of $3NM$ being the minimum required separation between two aircraft of weight category M. As this is the minimum of all required separations, the ellipsoids may not interfere for any pair of aircraft along the whole trajectories.

As all aircraft in the scenario have to keep separation distances pairwise to each other, for the considered scenario comprising 18 aircraft, a total of $N \cdot (N-1)/2 = 153$ separation constraints result. Thus, the number of separations grows quadratically with the number of

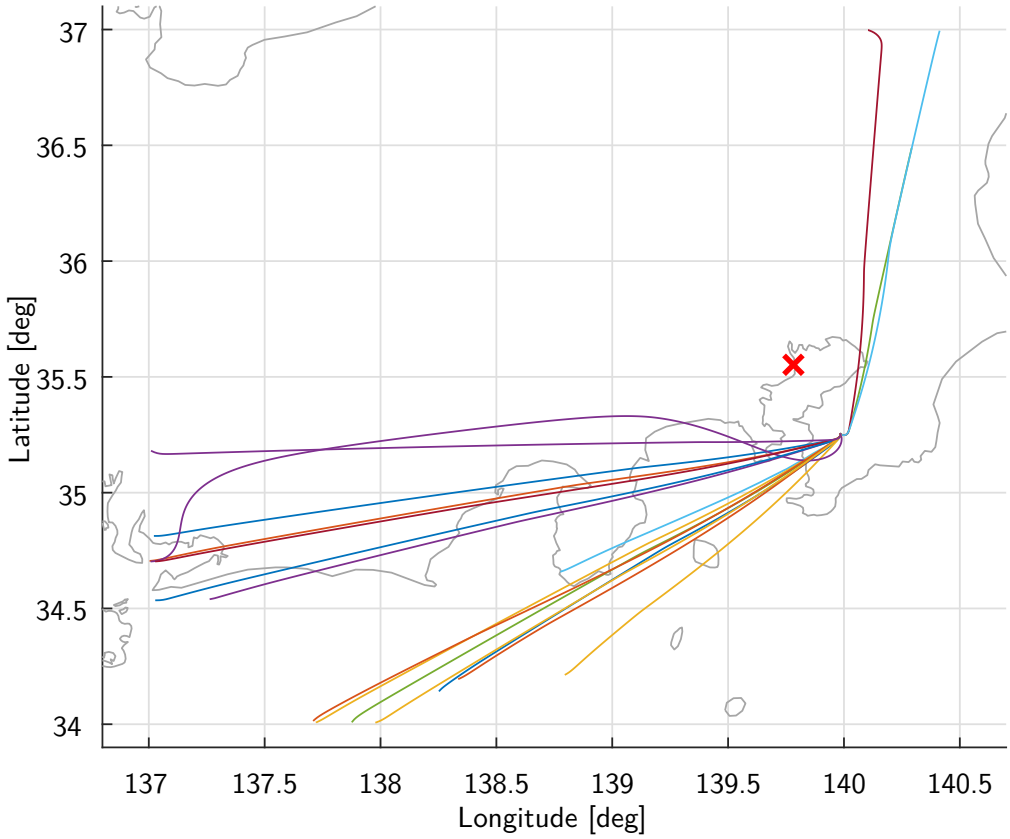

Figure 10.6: *Optimal flight tracks for all aircraft in the scenario. Tokyo International Airport is marked with a red cross. [Bit+16]*

aircraft. The number of separation distance constraints is equal to the number of final time constraints in the problem, even though the distance constraints need to be applied in every time step, while the final time separation only adds one overall constraint per aircraft pairing. The remaining spatial separation margins for all pairs of aircraft, that need to be positive, can be seen in figure 10.8. For some constellations the margin is hit but it is never violated.

The last column of Table 10.5 lists the optimal extension times for the first considered position of the flaps for every aircraft. As can be seen, only the second and the sixth aircraft use their flaps in the considered part of the trajectory to create more lift at lower speeds. For all other aircraft the use of the flaps is neither necessary nor does it create a benefit in the overall cost. This is reasonable as the increased lift of the flaps comes at the price of an increased drag which raises the fuel burn as well as the required time to the destination. No aircraft made use of the second flaps position that was considered here.

Discussion

The results show how cost-index-optimized continuous-descent operations in a highly congested airspace, like the one in the proximity of a strongly frequented airport, may look like. The required aircraft performance envelope as well as the separation limits are thereby maintained at all times. Anyway, controller workload and operational coordination of the different

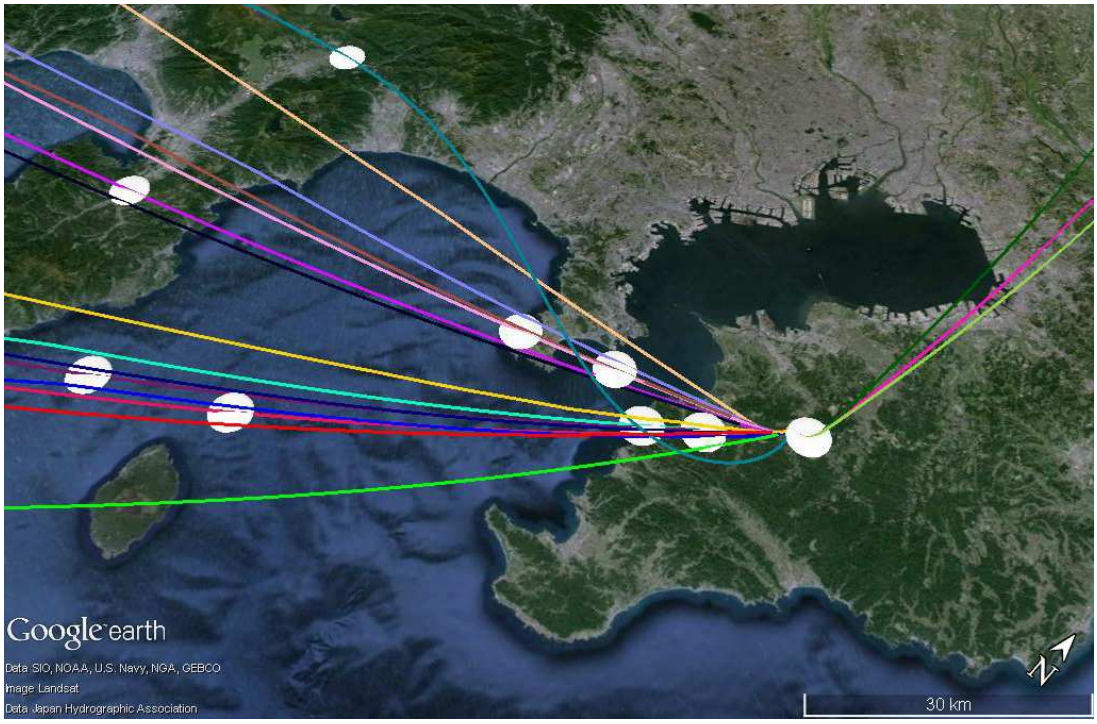

Figure 10.7: *Final part of the trajectories and separation ellipsoids for the aircraft at one point in time. [Bit+16]*

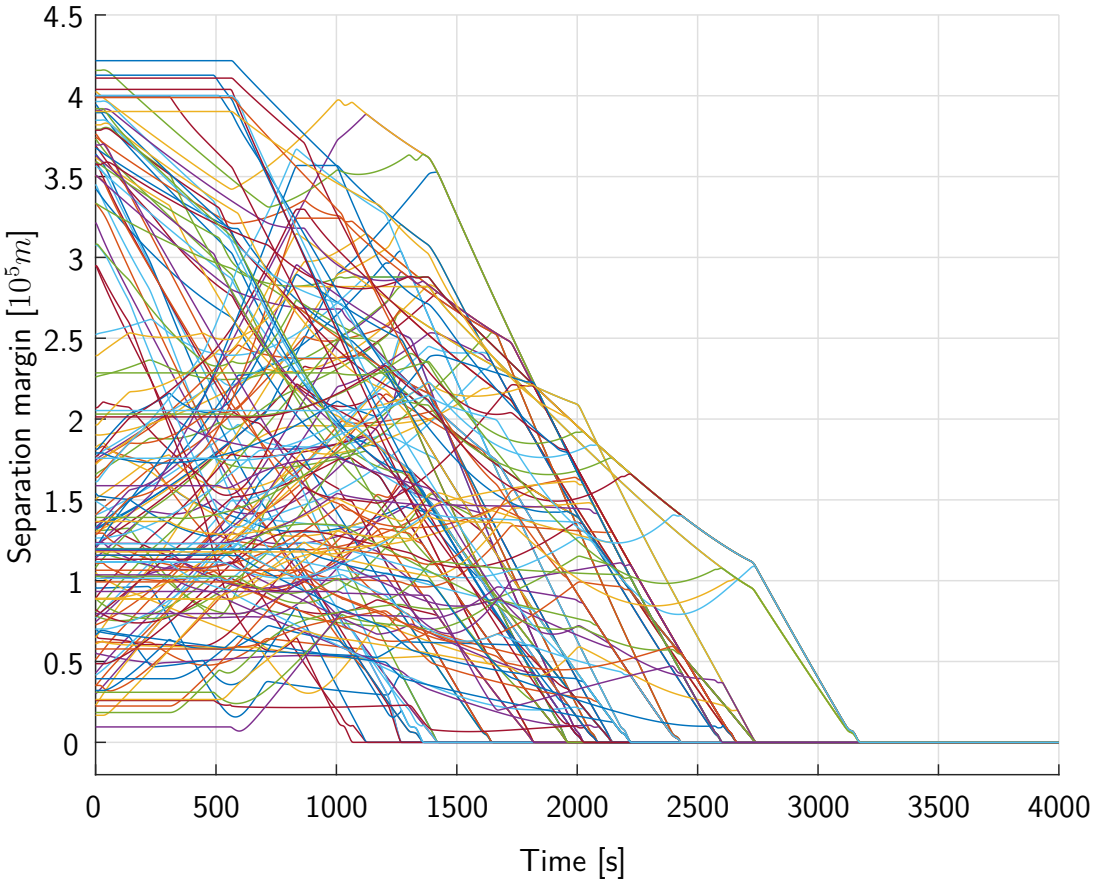

Figure 10.8: *Separation margin between all pairs of aircraft over time. [Bit+16]*

aircraft is not considered here and may put some additional constraints on such a scenario.

In detail, the methods presented here cannot be directly used within the current air traffic system, due to several reasons, with the following being some of the most important ones:

- The required controls for free flight trajectories cannot be commanded to the aircraft by the available set of advisories. On the one hand, exact values (like e.g. a course of $25.9\,\mathrm{deg}$) can currently not be commanded to the aircraft. On the other hand, no transmittable description of free flight trajectories exists [Sol+16]. A possible solution may be the use of *Controller-pilot data link communications (CPDLC)* [RTC05].
- Current air traffic operators need to be confident in the solution of a conflict and consequently also need to have a prediction of the future progress of the scenario without large uncertainties [Sol+16]. This problem may be overcome by using the results from the optimizations in order to provide additional information on the progress of the scenario to the air traffic controller on some kind of a support device.
- No traffic in adjacent airspace is considered in the example that might influence the traffic along the approach. For current operations this is not crucial as spatial separation is used to overcome this issue. This spatial separation can be ensured in the optimization by the definition of no-fly zones.
- The separation is currently constrained conservatively, as the ordering of the aircraft in the calculation of the separation distances is not known. Consequently, a worst-case assumption is used. This may be overcome by defining a new separation measure that takes the flight directions and the weight categories into account. Anyway, this new measure has to remain differentiable if to be used here.
- The current algorithm cannot handle changing scenarios with aircraft appearing or disappearing from the considered area due to external influences (rerouting etc.). In these cases a complete recalculation is necessary. Anyway, this recalculation may use the current solution as an initial guess. However, the idea presented in section 5.1.3 may not be used as the appearing and disappearing of an aircraft may not be modeled using a continuous parameter, but they influence the overall problem structure.
- No disturbances by means of unexpectedly changing environmental conditions like wind and weather are considered. [Sol+16] suggests to use stochastic optimal control methods to overcome this issue.
- In the current implementation, the required computational time is above the time that is normally available between detecting a conflict and resolving it. This may be overcome by parallelizing the algorithm as far as possible and using more powerful computers or even a compute cluster.

The solution of the problem discussed here takes a serious amount of computational time and the method does not show a generally stable behavior when adding more aircraft. This is on the one side related to the mere problem size which is hard to tackle using *IPOPT* [WB06].

This may be overcome by using special large scale solvers like e.g. *WORHP* [BW13] (which comes with other challenges like tuning the settings correctly). On the other side, the addition of further aircraft lets the number of pairwise separation constraints grow even more which at some point will eventually limit this method.

Overall, the method and the solution presented here is (at the moment) not intended as an approach for improving the daily work of ATCs. Conversely, the method and the study is much more suitable to provide a means of benchmarking different solutions for future ATM systems against a locally optimal solution.

In order to overcome the limiting factors for the problem size and the computational time, a hybrid optimization process may be used that separates the problem of determining the optimal trajectory for one aircraft from the optimization of the sequencing and the ensuring of the separations. Different methods may be used within the two levels of the resulting problem, possibly also overcoming the issue of converging to local minima in the sequencing problem.

In a first approach, the remaining operational issues may be overcome by post processing the results created by the optimization. The post processing algorithm needs to make sure that – most likely based on simulations – all operational requirements can be ensured by the resulting trajectories or controller commands. Anyway, post processing is only possible under a deterioration of the overall performance, as any additional constraints posed on an optimization problem will lower the performance (or not influence it in the best case). One possible result of such a post processing algorithm may be aircraft commands that are compatible with the *CPDLC* standard. [Bit+15, Bit+16]

10.3 Optimization of Mid to Long Range Flights Considering Air to Air Refueling

One idea to reduce fuel consumption of long range flights, that has been around a while now, is the use of aerial refueling for civil flights. Therein, tanker aircraft are stationed along the route of the long range flight that climb up to the cruising aircraft in order to refuel them at some point (or several points) along the track. The benefit for the cruising aircraft is that their mean mass is reduced as they do not have to carry the fuel for their complete route. This way, fuel consumption of the cruising aircraft may be reduced. The drawback – obviously – is that additional tanker aircraft are required that also consume fuel as they need to take-off, climb and refuel, before descending and landing again. Besides many technical and safety related challenges one of the questions to be answered is how much fuel can be saved in these scenarios considering optimized operations.

In this section, aerial refueling scenarios are considered as aircraft trajectory optimization problems in order to give a first estimate for the aforementioned question. The optimal control problems to be solved are built up using point mass simulation models based on BADA Family 3 data [EUR14], where available. The models also implement fuel consumption depending on

the aerodynamic situation and the current mass. Besides the models of real, existing aircraft, additional A380 versions with reduced range specially tailored to aerial refueling are introduced. The relevant performance parameters for these redesigned aircraft are estimated using general aircraft design principles. The models for the tankers have also been derived from BADA data with fuel consumption models similar to the cruising aircraft. The exemplary refueling scenarios show the maximum savings that could be achieved. A EULER-Forward collocation scheme is used and the numerical optimization problem is solved using *SNOPT* [GMS02].

Operational aspects like air traffic control, the technical aspects of the refueling maneuver as well as any safety considerations are not part of this study. This also includes that no reserve fuel, that is needed to perform a safe operation, is incorporated in the fuel calculations. However, the achieved results are not deteriorated by these issues as reserve fuel is neither concerned in the reference scenarios nor in the refueling scenarios and finally the comparison between the two is important.

Part of the work presented here has been conducted with the support of CHRISTOPH HORNFECK who has done a semester thesis at the *Institute of Flight System Dynamics of TUM*. The redesign of the long range aircraft for flights with refueling has been performed by MALTE SCHWARZE from the *Institute of Aircraft Design at TUM*. The results have been published in similar form in the paper [Bit+13b], before.

10.3.1 Simulation Models

The simulation models used in this aircraft trajectory optimization task are extended point mass simulation models like those presented in section 2.4. The aerodynamic parameters, the thrust model as well as the fuel consumption models are taken from the *BADA Family 3*, where available [EUR14]. For the derived short range versions of a wide body aircraft, that are introduced in section 10.3.3, the parameters are based on lookup tables that are then approximated by a continuously differentiable spline representation allowing for the analytic calculation of all gradients. The models are controlled by the angle of attack α_A, the bank angle μ_A and the thrust lever position δ_T. No wind is considered in the scenarios, rendering all aerodynamic and kinematic quantities equal.

To calculate the actual fuel flow from the maximum and the idle fuel flow given in the *BADA* model, a linear interpolation with respect to the thrust lever position is performed:

$$\dot{m}_{fuel} = \dot{m}_{fuel,idle} + \delta_T \cdot (\dot{m}_{fuel,max} - \dot{m}_{fuel,idle}) \tag{10.27}$$

Overall, the model for each participant Pi contains seven states and three controls. The considered states for one participant are

$$\mathbf{x}_{Pi} = \begin{pmatrix} \lambda_{Pi} & \varphi_{Pi} & h_{Pi} & V_{K,Pi} & \chi_{K,Pi} & \gamma_{K,Pi} & m_{Pi} \end{pmatrix} \tag{10.28}$$

where a detailed description of the quantities can be found in table 2.1 in section 2.1. In the remainder of this example, the cruiser aircraft are marked by indices Ai while the tankers are marked by indices Ti instead of the general placeholder Pi. [Bit+13b]

10.3.2 Flight Envelope Constraints

In order to reach realistic aircraft trajectories that can be flown by the aircraft, several flight envelope constraints are added to the problem formulation. The angle of attack α_A is limited to:

$$-4\,\mathrm{deg} = \alpha_{A,min} \leq \alpha_A \leq \alpha_{A,max} = 10\,\mathrm{deg} \tag{10.29}$$

The bank angle μ_A is mainly limited by the passenger comfort to:

$$-30\,\mathrm{deg} = \mu_{A,min} \leq \mu_A \leq \mu_{A,max} = 30\,\mathrm{deg} \tag{10.30}$$

The normalized thrust lever position has to be from:

$$\delta_T \in [0, \ldots, 1] \tag{10.31}$$

Additionally, the load factor in vertical direction of the *Body Fixed Frame* B has to remain within:

$$0.85 \leq (n_z)_B \leq 1.15 \tag{10.32}$$

The flight envelope by means of speed is on the one side limited by the calibrated airspeed limits from the BADA model and on the other side by the MACH number:

$$V_{CAS,min} \leq V_{CAS} \leq C_{CAS,max} \tag{10.33}$$

$$Ma \leq 0.85 \tag{10.34}$$

Similar constraints have been used in the work [Bit+13b].

10.3.3 Redesign of a Long Haul Aircraft for Aerial Refueling

This section gives a brief overview of the redesign of an Airbus A380 for short to mid-range flights not exceeding a maximum distance of 3000NM. Assuming that carriers wanted to offer existing connections but with the benefits of aerial refueling, they would require aircraft being able to carry the same amount of payload but over a far shorter distance. The approximate calculation of masses and aerodynamic parameters for such a plane has been conducted by MALTE SCHWARZE from the *Institute of Aircraft Design* at the *Technische Universität München*. The results of this aircraft redesign have only been published in the joint research paper [Bit+13b] but not in more detail separately.

The following assumptions have been made:

- The newly designed A380 will have the same payload,
- fuselage weight and
- design point (by means of thrust-weight ratio and area loads) as the existing one.
- Moreover, the basic performance requirements like cruise speed or
- T/O and landing distance remain the same.

In a first step, the original A380-800 has been recalculated to validate the underlying numerical models. Table 10.6 lists the data from the recalculation and the dimensioning of the short range aircraft.

The general relations used for the redesign of the A380 for short range are the following:

- The desired reduction in maximum range requires less fuel capacity which also significantly reduces the overall weight when considering unchanged payload.
- As this means that only the fuel weight portion decreases, all other mass ratios, consisting of the share of structural weight and the share of payload for the plane have to rise as compensation.
- Both, the overall mass change and the mass relation change make the required lift decrease and hence allow the use of smaller wings which also results in a reduced weight of the wings, although they have to be reinforced because of the decreased amount of fuel inside.
- This, in reverse, shifts the structural weight and its share again such that the problem has to be solved iteratively.

The procedure for the calculation of the required masses, the thrust and the aerodynamic parameters is divided into three major parts that can be seen in figure 10.9. The central element of the algorithm is the calculation of the weights and their shares based on a given *Maximum Take-Off Weight (MTOW)*, the required fuel weight and the required payload. Therein, the overall structural weight, the *Zero Fuel Weight (ZFW)* and the weight of the aircraft components are estimated recursively based on general aircraft design principles as they can for example be found in [Tor82, Ray12, Ros04, Lof80]. For the fuel calculation the *BADA Fuel Calculator* [EUR14] and the aircraft performance program *Piano-X*[1] have been used. The reference area – that can be calculated from the weight when assuming that the area load remains unchanged – and the ZFW from the main calculation are fed into a separate algorithm which estimates the wing weight based on [Tor82]. The result generated there is afterwards compared to the result of the main computation. The whole process can be terminated when a given tolerance in the results is reached. Thereafter, the masses and the wing area are fed into the last part of the algorithm that assesses the basic aerodynamic parameters as well as the required thrust based on the similarity of the aircraft performance. This includes the design point in terms of area loading, the thrust-to-weight ratio, and the specific fuel consumption.

The procedure described above has been performed for two different redesigns of the A380. The first one (A380-388) is tailored for a payload of $59t$ (which is the design payload of the original A380-800) and therefore has a *MTOW* of $388t$. The second approximation (A380-457) was done for a version *designed* to carry a payload of about $90t$ (the maximum payload of the original A380-800) resulting in a *MTOW* of $457t$. Table 10.6 lists the basic parameters

[1] http://www.piano.aero/

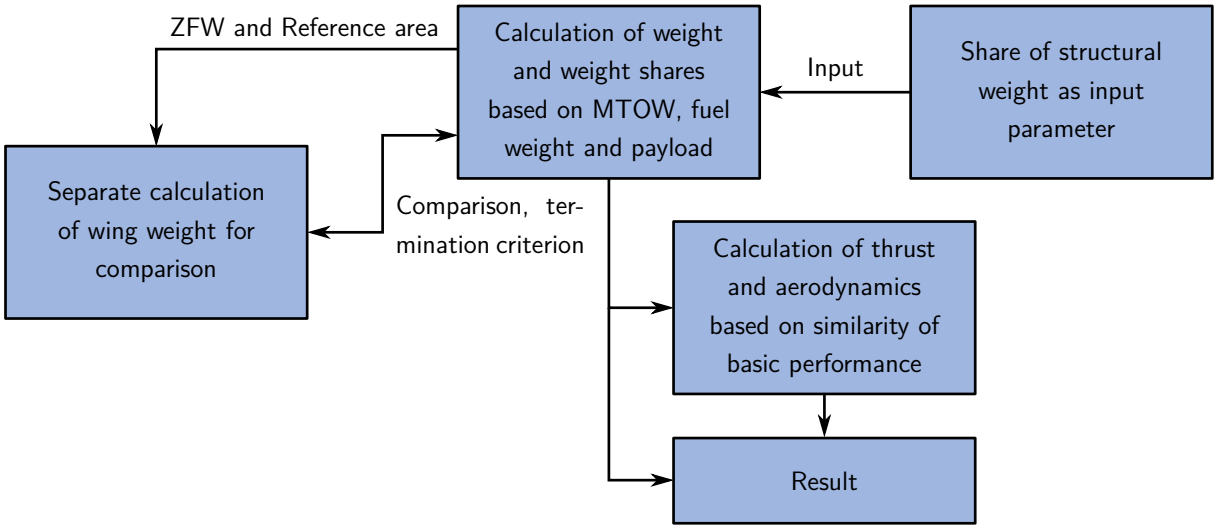

Figure 10.9: *Structure of the calculation process of aircraft masses and parameters. [Bit+13b]*

for the original A380-800, for the recalculation that was used to verify the model and the two new redesigns for a shortened range of $3000NM$. The data required for the redesign is taken from [Air15]. [Bit+13b]

Table 10.6: *Basic data of the aircraft redesigns. The base data that has been used here is mainly taken from [Air15]. [Bit+13b]*

		A380-800 Original	**A380-800 Evaluation**	**A380-388 (59t payload)**	**A380-457 (90t payload)**
Maximum take-off weight (MTOW)	kg	569000	568390	388000	457000
Operating weight empty (OWE)	kg	270364	270600	226790	247770
Maximum fuel weight	kg	253938	252020	101250	119570
Wing reference area	m^2	845	845	608	719
Range	NM	7700	7700	3000+R	3000+R
Payload	kg	59000	59110	59000	90000
C_{L0}	-	n.p.	5.24	5.51	5.51
$C_{L\alpha}$	-	n.p.	0.2744	0.2885	0.2885
C_{D0}	-	n.p.	0.015	0.017	0.016
C_{D2}	-	n.p.	0.051	0.045	0.045

10.3.4 Multi-Aircraft Optimal Control Problem

The goal of the optimization is to minimize the overall fuel consumption of all aircraft involved, leading to the cost function:

$$J = \sum_{i=1}^{N} m_i(t_{0,i}) - m_i(t_{f,i}) \tag{10.35}$$

The initial and final boundary conditions for every aircraft are set to the departure or arrival runway, including the direction of the runway. The climb angle is set to $\gamma_K = +3\,\mathrm{deg}$ for the departure and to $\gamma_K = -3\,\mathrm{deg}$ for the arrival. No configuration changes are considered in this example, not allowing a detailed modeling of the climb out and the landing. As the overall fuel consumption along the whole flight is considered, the particular optimization potentials in these phases are negligible. Moreover, the initial mass of each aircraft is subject to optimization and has to remain between the zero fuel weight and the maximum takeoff weight.

To be able to model the refueling, the problem is divided into multiple phases, with constraints on the aircraft positions and the phase times enforced. These constraints assure that the tankers and the traveling aircraft are close enough during refueling. Figure 10.10 shows the general phase structure used in the calculations. The two horizontal lines depict the tracks of the two long haul aircraft traveling from two adjacent airports to two different destinations. Their flight tracks are assumed to be roughly parallel such that the same tanker (or tankers) may be used for refueling both flights. In the example, two tankers are used to refuel both aircraft, twice each.

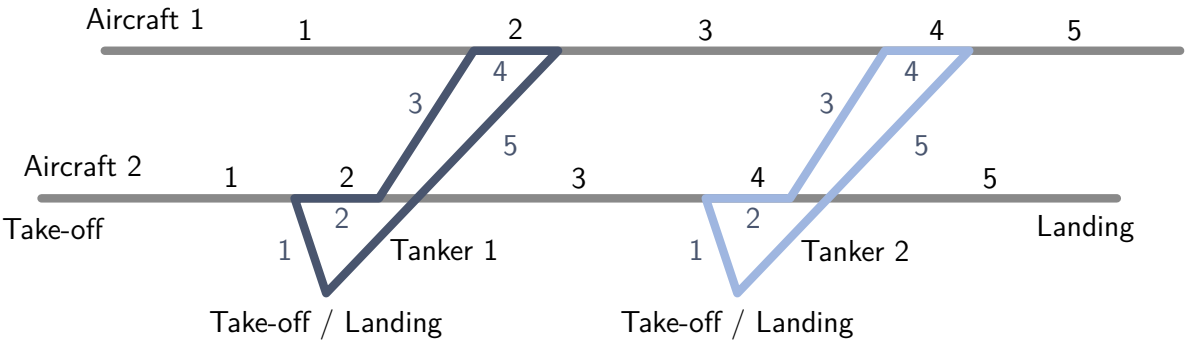

Figure 10.10: *Phase structure for aerial refueling problem. The numbers along the line represent the phase numbers. [Bit+13b]*

The length of each phase j of a participant Pi is defined by the time $T_{Pi,j}$, with every second phase being a refueling phase and every other phase being a normal travel flight phase. In figure 10.10, for the airliners, odd phases are pure travel flight phases which for the tankers are phases where they either travel from their base airport to the first airliner, from one airliner to the next, or back to their base. For both the tankers and the airliners, even phases are refueling phases. The length of the refueling phases have to have the same duration for the airliner as for the tanker, which is assured by the constraint:

$$T_{Ai,2j} = T_{Tj,2i} \tag{10.36}$$

Furthermore, time constraints are required that ensure that the tanker and the aircraft to be refueled are at the same location at the same time even after a different number of preceding phases.

In the phases of the problem where the refueling takes place, the position of the aircraft to be refueled and that of the tanker refueling have to be close enough such that the refueling is technically possible. Here, the positions of the two aircraft are set equal during refueling, as a precise relative position control is not part of this study. [Bit+13b]

10.3.5 Results

Multiple scenarios have been investigated using the methods described above, where two of them are presented here in more detail. Afterwards, some general trends that have been realized will be briefly pointed out.

The first example involves one Airbus A380 that is intended to go from *Munich in Germany (MUC)* to *São Paulo in Brazil (GRU)*. In the reference scenario without refueling, the model of a regular A380 is used. For the flight with refueling the redesigned A380-388 with a shorter range of $3000NM$ is used, while a model of a Boeing KC767-300 is used as tanker. The tanker is based at *Dakhla in Morocco (VIL)*. Table 10.7 lists the initial and final boundary conditions for the airliner and the tanker. Figure 10.11 shows the optimized trajectory for the flight on a map of the earth. The optimized refueling maneuver takes place close to the tanker base in *Dakhla*.

Table 10.7: *Initial and final conditions for an example refueling scenario from Munich, Germany to São Paulo, Brazil. Refueling takes place close to Dakhla, Morocco. [Bit+13b]*

		A380 / A380-388	**KC 763**
Initial Airport		Munich, MUC	Dakhla, VIL
Initial longitude	λ_0	$11.7510\,\mathrm{deg}\,E$	$15.9320\,\mathrm{deg}\,W$
Initial latitude	φ_0	$48.3407\,\mathrm{deg}\,N$	$23.7183\,\mathrm{deg}\,N$
Initial altitude	h_0	$453m$	$11m$
Initial course angle	χ_0	$263\,\mathrm{deg}$	$30\,\mathrm{deg}$
Destination airport		São Paulo, GRU	Dakhla, VIL
Final longitude	λ_f	$46.4695\,\mathrm{deg}\,W$	$15.9320\,\mathrm{deg}\,W$
Final latitude	φ_f	$23.4320\,\mathrm{deg}\,S$	$23.7183\,\mathrm{deg}\,N$
Final altitude	h_f	$750m$	$11m$
Final course angle	χ_f	$254\,\mathrm{deg}$	$30\,\mathrm{deg}$

Figure 10.12 depicts the fuel masses on board the two aircraft during the flights. It can be seen that the airliner does not take-off fully fueled but with enough fuel on board to reach the tanker, as this way the mass along the first part of the trajectory of the aircraft is reduced, consequently also reducing fuel consumption.

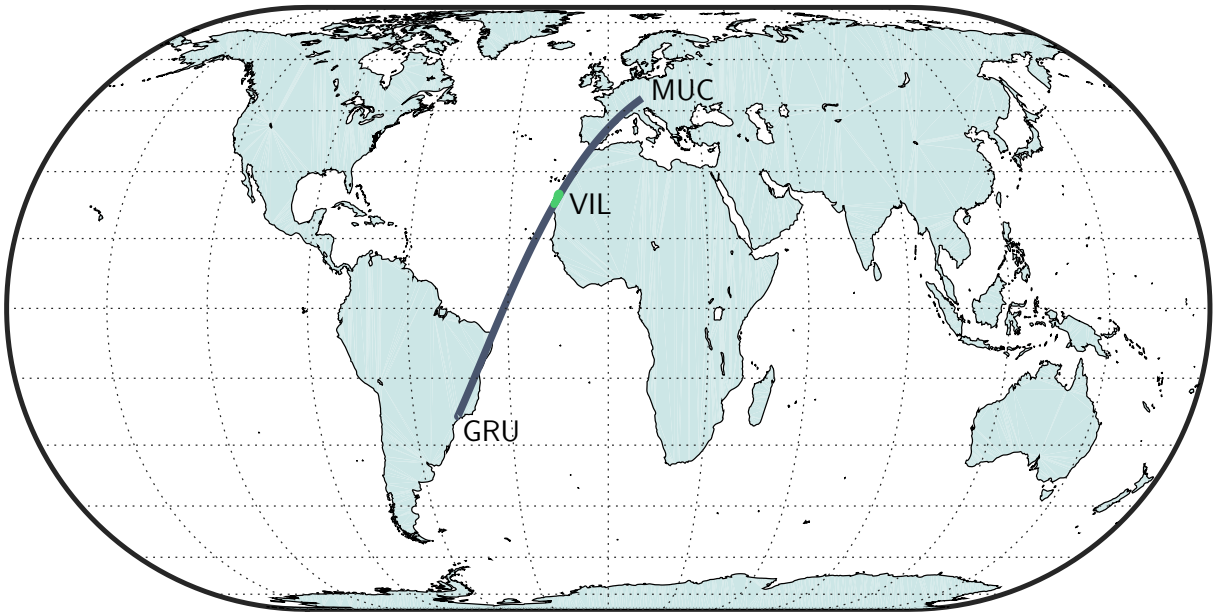

Figure 10.11: *Trajectories for the airliner and the tanker in the first example scenario. [Bit+13b]*

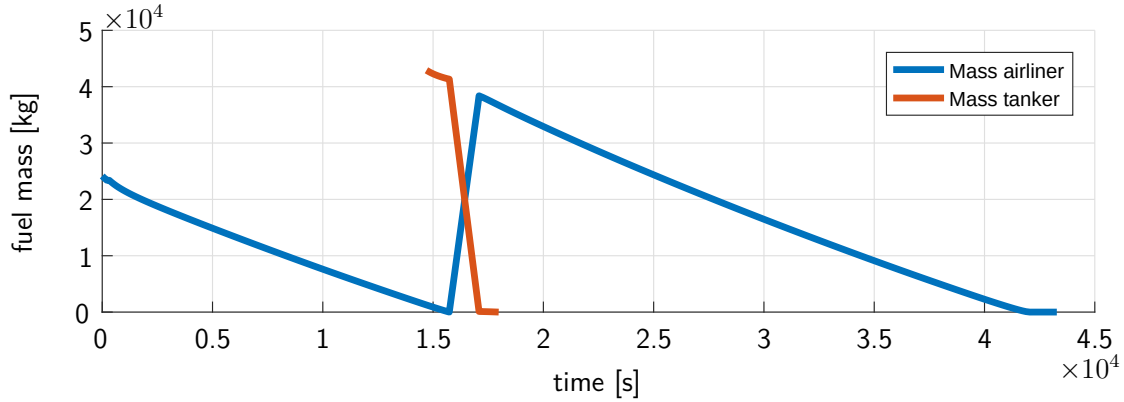

Figure 10.12: *Fuel masses on board the aircraft in the first example scenario. [Bit+13b]*

Figure 10.13 is an image created with *Google Earth* showing the trajectories while the refueling maneuver takes place. It can be seen that the tanker climbs until it reaches the airliner before both are flying synchronously while the refueling is performed. Afterwards, the tanker descends to return to its base airport.

In this example fuel savings of approximately 14% with respect to the nominal scenario without refueling could be achieved. While the regular Airbus A380 consumed approximately $156t$ of fuel in the simulation, the redesigned A380-388 only consumed $127t$ of fuel. Of course, the fuel consumption of the tanker also has to be taken into account which was roughly $7t$ in the example simulation. Overall, without refueling, $156t$ of fuel were consumed while with refueling only $134t$ of fuel have been burnt. Table 10.8 lists the detailed masses again.

The second example involves two airliners: one flying from *Munich in Germany (MUC)* to

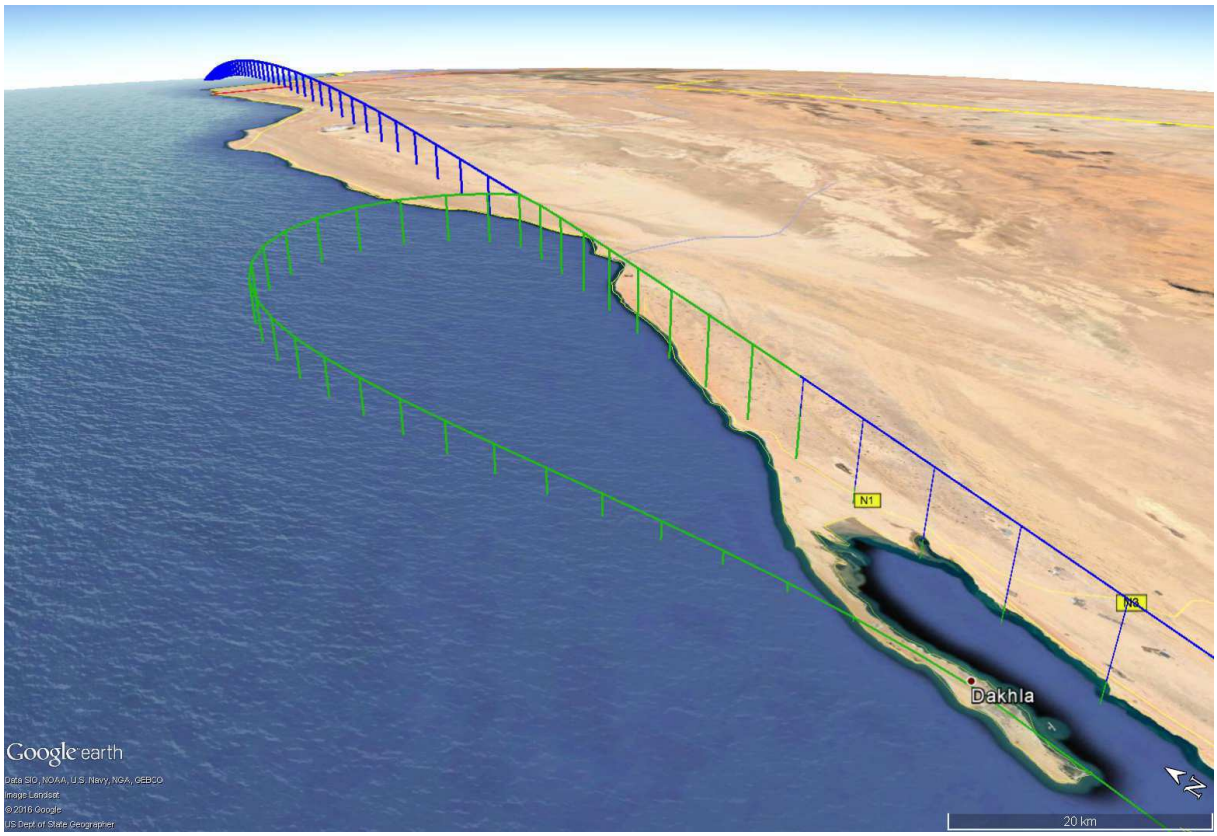

Figure 10.13: *Part of the trajectory while refueling above Dakhla, Morocco. [Bit+13b]*

Table 10.8: *Results for refueling a flight from Munich to São Paulo by a tanker based in Dakhla, Morocco.*

Fuel consumption	**Without refueling**	**With refueling**
Airliner, A380, MUC-GRU	$156t$	$127t$
Tanker, KC 763, VIL	-	$7t$
Sum	$156t$	$134t$

San Francisco in the USA (SFO) and another one traveling from *Frankfurt in Germany (FRA)* to *Los Angeles in the USA (LAX)*. In the reference scenario, regular A380 models are used for both flights, while for the refueling scenario, the redesigned A380-388 models have been used. In the latter calculations another Airbus A380 is considered as a tanker that is stationed in *Sisimiut in Greenland (JHS)*, assuming all its payload to be additional fuel. Table 10.9 lists the initial and final conditions for the two flights and the tanker.

The optimized flight tracks for the aircraft in the scenario with refueling can be seen in figure 10.14, where the first flight going from MUC to SFO is depicted in gray, the second flight going from FRA to LAX is depicted in blue and the trajectory of the tanker stationed in JHS is printed in green.

Figure 10.15 illustrates the fuel on board of each of the aircraft showing that the tanker

Table 10.9: *Example civil refueling scenario involving two airliners. The airliners are travelling from MUC to SFO, and from FRA to LAX. The tanker base is in this case located in JHS. [Bit+13b]*

		A380 / A380-388	**A380 / A380-388**	**KC 763**
Initial Airport		Munich, MUC	Frankfurt, FRA	Sisimiut, JHS
Initial longitude	λ_0	$11.7510\,\mathrm{deg}\,E$	$8.5342\,\mathrm{deg}\,E$	$53.7293\,\mathrm{deg}\,W$
Initial latitude	φ_0	$48.3407\,\mathrm{deg}\,N$	$50.1428\,\mathrm{deg}\,N$	$66.9513\,\mathrm{deg}\,N$
Initial altitude	h_0	$453m$	$100m$	$9m$
Initial course angle	χ_0	$263\,\mathrm{deg}$	$250\,\mathrm{deg}$	$62\,\mathrm{deg}$
Destination airport		San Francisco, SFO	Los Angeles, LAX	Sisimiut, JHS
Final longitude	λ_f	$122.3807\,\mathrm{deg}\,W$	$118.4190\,\mathrm{deg}\,W$	$53.7293\,\mathrm{deg}\,W$
Final latitude	φ_f	$37.6068\,\mathrm{deg}\,N$	$33.9338\,\mathrm{deg}\,N$	$66.9513\,\mathrm{deg}\,N$
Final altitude	h_f	$3m$	$36m$	$9m$
Final course angle	χ_f	$208\,\mathrm{deg}$	$263\,\mathrm{deg}$	$62\,\mathrm{deg}$

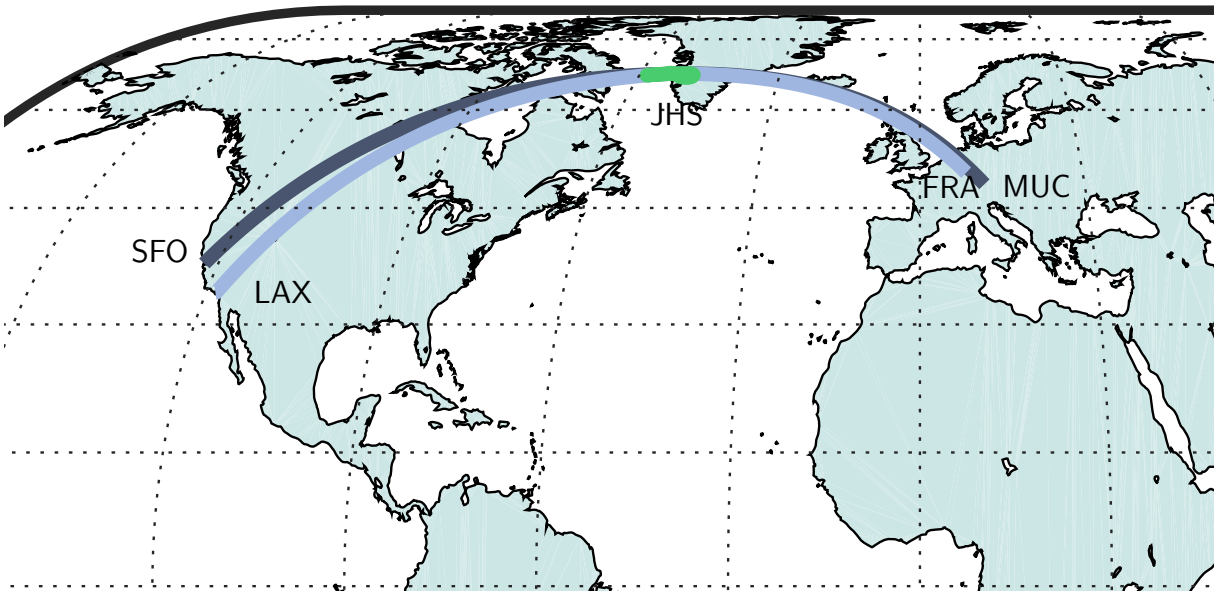

Figure 10.14: *Trajectories of the aircraft in the second aerial refueling scenario. The flight from MUC to SFO is depicted in gray, the flight from FRA to LAX in blue and the track of the tanker in green. [Bit+13b]*

in this example cannot carry enough fuel to refuel both aircraft completely for their remaining routes. This leads to an optimized solution where the second aircraft does not start its flight with the minimum amount of fuel required until the refueling (the aircraft still has fuel left at this point), but with the minimum fuel required for the first segment plus that part of the fuel for the second segment that cannot be delivered by the tanker.

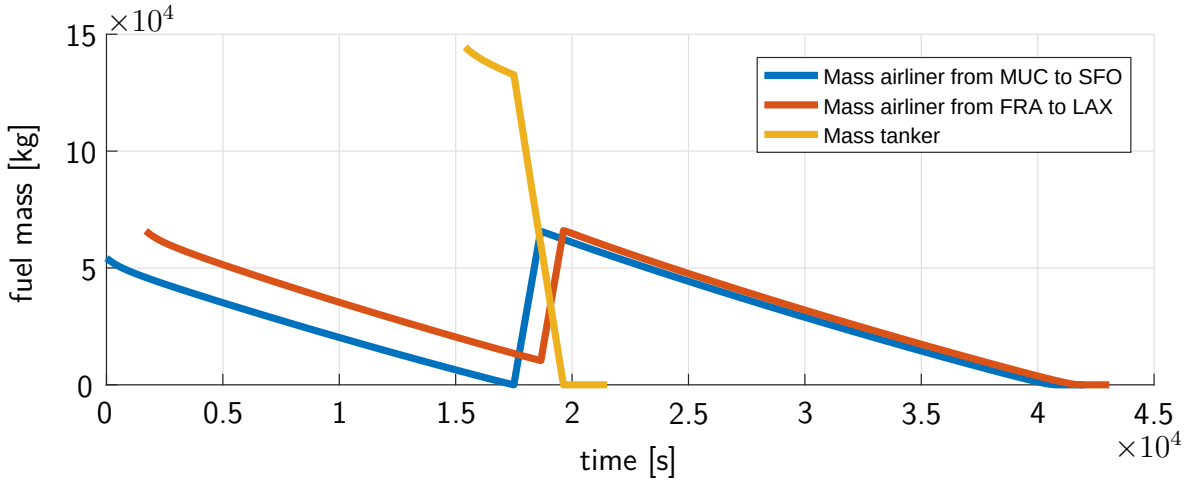

Figure 10.15: *Fuel masses on board the aircraft in the second example scenario. [Bit+13b]*

The overall savings that could be achieved in the second example are listed in table 10.10. The two long haul flights in the example consumed $296t$ of fuel, while the flights using the redesigned models and being refueled only burnt a total of $247t$ of fuel. In this case, an additional $20t$ of fuel are needed by the tanker, in total leading to a reduction of fuel consumption of approximately 9.8% or $29t$ relative to the solution without refueling.

Table 10.10: *Results for the masses in the second example scenario. [Bit+13b]*

Fuel consumption	**Without refueling**	**With refueling**
Airliner 1, A380, MUC-SFO	$149t$	$118t$
Airliner 2, A380, FRA-LAX	$147t$	$129t$
Tanker, A380, JHS	-	$20t$
Sum	$296t$	$267t$

In the studies on the use of aerial refueling for reducing fuel consumption based on aircraft trajectory optimization, generally, fuel savings due to refueling have been in an approximate range of 5% to 20%. Thereby, without redesigning the aircraft, fuel savings of approximately $5-7\%$ could be achieved, while another $5-10\%$ are possible when also redesigning the aircraft for shorter ranges. Overall, the achievable savings very strongly depend on the mission profiles to be optimized and on the size of the tanker aircraft. Moreover, the number of refueling maneuvers per flight and per tanker have a strong influence on the results of the optimization.

In the examples presented here, the tanker bases have been chosen in advance and were not subject to optimization. This would be one possible enhancement for future research on the topic. It is expected to save even more fuel, when the location of the tanker bases is also subject to optimization. Here, two possible versions might be investigated as one might consider only existing airports or all possible airport locations.

Moreover, no safety constraints like reserve fuel remaining on board each aircraft are considered in this work. These constraints would in some way be dynamic, when considering

refueling scenarios, as the refueling maneuver itself may fail and the remaining fuel should nevertheless always be sufficient to return to a safe airport. This would lead to constraints on the remaining fuel at each point in time depending on the distance to the nearest suitable airport.

Furthermore, in the examples no wind is considered that has a strong influence on the optimal trajectories and consequently can also make the aerial refueling process more complicated. In case strong winds move the optimal trajectories far away from the tanker bases, the fuel consumed by the tankers will increase, reducing the benefit that is created. Anyway, if many tanker bases are operated at the same time it may become possible to always use a tanker located at the best suited airport, finally weakening the aforementioned drawback. However, the effect wind creates on the fuel consumption along the trajectory can most likely be neglected as the reference trajectories will be affected similarly to trajectories with aerial refueling. Further studies may be conducted that include the influence of wind and multiple tanker bases.

The redesign that has been performed in order to estimate the parameters for the short range Airbus A380 versions is a very rough redesign and it is almost impossible to determine the inherent uncertainties. The fact that the redesign of the currently available Airbus A380 fits the data that is publicly accessible quite well, is a good indication that the uncertainties within the models are below the improvements that can be achieved by civil aerial refueling. Anyway, in order to make sure that significant benefits can be achieved further studies are necessary. Moreover, in the particular examples presented in detail here, only the smaller version of the redesigned aircraft has been used, while the larger Airbus A380-457 has not been considered. If doing so, the comparison of the results becomes a lot more complicated as the change in fuel consumption in the scenarios needs to be put in relation to the payload that can be carried. This may be done using the efficiency metrics developed in [Nan06b].

Of course, aerial refueling in civil applications is in a very early stage of research and a lot of technical and safety-related issues have to be tackled before a technology readiness level could be achieved that would allow an implementation of such maneuvers into everyday procedures. Nevertheless, aerial refueling is quite well established in military applications and might also be used in other scenarios involving for example UAS or other modern – civil or military – flying equipment. [Bit+13b]

10.4 Fairness considerations in ATM scenarios

In every conflicting air traffic scenario multiple aircraft – mostly operated by different airlines that are competing on the market – are involved. Thus, every airline is interested in not only a safe solution of the conflict but also a cost efficient solution for their aircraft. Fairness in this environment means that the cost increases that occur by resolving the conflict are equally distributed to all participants in the considered scenario.

In some scenarios the resolution of a conflict may be almost fully paid by one aircraft, or

at least a small portion of the parties involved, if the optimal solution for the overall costs is calculated. Therefore, in these examples, air traffic scenarios are not only optimized for the overall costs but also for the individual costs of every aircraft and the fair distribution of them. The resulting mathematical problem can be formulated as a multi criteria optimization problem as revisited in section 3.3. In the considered scenarios, N aircraft enter an airspace at different times and fixed positions before flying to fixed exit positions.

The results presented in this example have been created with the support of Benjamin Fleischmann who did his diploma thesis at the *Institute of Flight System Dynamics* under the supervision of this author. The results have been published in a similar form in the joint paper [Bit+14], before.

10.4.1 Aircraft Simulation Model

In the scenarios considered here (as well as in [Bit+14]), the influences of the inherent aircraft dynamics are simplified as the considered maneuvers are relatively slow and far away from the flight envelopes of the aircraft. Besides, no wind influence is incorporated here. Thus, the aircraft dynamics are modeled using pure kinematic relations. As a flight level change is always the last measure to be taken by an air traffic controller when resolving conflicts, the scenarios investigated here do not take level changes into account but are modeled in two-dimensional space. The formulation used allows for an extension to three dimensions, for the cost of a significant increase in computational burden. Consequently, the position equations of motion in the locally fixed Navigation Frame N for each aircraft $i = 1, \ldots, N$ are given by:

$$(\dot{x})_{N,i} = V_{K,i} \cdot \cos \chi_{K,i} \qquad \forall i = 1, \ldots, N \tag{10.37}$$

$$(\dot{y})_{N,i} = V_{K,i} \cdot \sin \chi_{K,i} \qquad \forall i = 1, \ldots, N \tag{10.38}$$

where N is the total number of aircraft considered.

The models are controlled by the kinematic velocity $V_{K,i}$ and the kinematic course angle $\chi_{K,i}$. This leads to the following state and control vectors $\mathbf{x}_i$ and $\mathbf{u}_i$ for each aircraft:

$$\mathbf{x}_i = ((x)_{N,i}, (y)_{N,i})^\intercal \tag{10.39}$$

$$\mathbf{u}_i = (V_{K,i}, \chi_{K,i})^\intercal \tag{10.40}$$

10.4.2 Separation Constraints

In order to maintain safe separation between all pairs of involved aircraft, a path constraint is added to the scenarios. This constraint has to be fulfilled along the whole trajectories of all aircraft. The minimum separation is given as:

$$d_{\min} = 5NM = 9260m \tag{10.41}$$

In the examples here no differentiation between the weight classes of the aircraft is made and consequently a fixed separation distance is used. As only the horizontal motion of aircraft is

considered here, the distance between two aircraft can be calculated from:

$$d = \sqrt{((x)_{N,i} - (x)_{N,j})^2 + ((y)_{N,i} - (y)_{N,i})^2} \tag{10.42}$$

which is the two dimensional representation of equation (10.21). In order to avoid the square root, the distance constraint can be reformulated and implemented as

$$(d_{\min})^2 \leq ((x)_{N,i} - (x)_{N,j})^2 + ((y)_{N,i} - (y)_{N,i})^2 \tag{10.43}$$

which is numerically more stable.

In case of a scenario containing N aircraft, N_d pairs of distance constraints have to be fulfilled, where N_d is the number of possible aircraft combinations and evaluates to:

$$N_d = \frac{N \cdot (N-1)}{2} \tag{10.44}$$

Similar to the example presented in section 10.2, the number of separation constraints grows quadratically with the number of aircraft and is consequently the main limiting factor when adding aircraft to the scenarios. [Bit+14]

10.4.3 Multi-Aircraft Optimal Control Problem

The combination of all aircraft in the scenarios considered here is performed similarly to the formulation presented in section 10.2.6, because the same problem concerning unknown and different initial and final times needs to be solved. Again, in oder to be able to evaluate the spatial separation constraints at the correct points in time, a simulation on one common grid is used. Another way of solving the problem may be the partitioning of the whole scenario into multiple phases in order to use different combinations of active aircraft in each phase [Ric+14c].

Here, the aircraft dynamics are extended by *three* fading factors. One is used to activate the dynamics at the beginning of the trajectory based on time and the other one is used to fade out the dynamics at the final position based on the aircraft location:

$$\dot{\mathbf{x}}_i = \begin{pmatrix} (\dot{x})_{N,i} \\ (\dot{y})_{N,i} \end{pmatrix} = \begin{pmatrix} V_{K,i} \cdot \cos\chi_{K,i} \\ V_{K,i} \cdot \sin\chi_{K,i} \end{pmatrix} \cdot \delta_{x,i} \cdot \delta_{y,i} \cdot \delta_{t,ini,i} \tag{10.45}$$

The three fading factors are modeled according to

$$\delta_{x,i} = \frac{1}{2}\tanh\left(a \cdot |(x)_{N,i} - x_{f,i}|\right) + \frac{1}{2} \tag{10.46}$$

$$\delta_{y,i} = \frac{1}{2}\tanh\left(a \cdot |(y)_{N,i} - y_{f,i}|\right) + \frac{1}{2} \tag{10.47}$$

$$\delta_{t,ini,i} = \frac{1}{2}\tanh\left(\nu \cdot (t - t_{ini,i})\right) + \frac{1}{2} \tag{10.48}$$

with $x_{f,i}$ and $y_{f,i}$ being the final position of aircraft i and $t_{ini,i}$ being its initial time. As in the examples considered here all aircraft start at the same point in time, $\delta_{t,ini,i} = 1$ can be used

(also compare equation (10.48) to equation (10.15)). Additionally, depending on the final flight direction, one of the two location dependent factors is also set to one. The splitting of the fading factor for the final position in two factors for the coordinate directions represents a simplification of a general model formulation using the Euclidean distance. The drawback of the simplified formulation is that the final direction of movement has to be known for all aircraft, its benefit is an increase in solution stability as no squared distance is required. The parameter a used in (10.46) and (10.47) controls the steepness of fading, where values that are too high result in very high gradients which is numerically problematic and values that are too low result in a very long and unrealistic fading. In the examples considered here, a value of

$$a = 20\frac{1}{NM} = 1.0799 \cdot 10^{-2}\frac{1}{m} \tag{10.49}$$

turned out to be a good compromise after testing several values for $a \in [10^{-3}\frac{1}{m}, \frac{1}{m}]$. Using this position based fading, final boundary conditions cannot be given as equality constraints anymore but have to be specified as small ranges. If these ranges and the steepness are tuned accordingly, they both do not affect the solutions in practice. [Bit+14]

10.4.4 Cost Functions and Fairness

In the examples presented here, the costs generated by each flight are approximated by the flight time required to pass through the given sector. As no fast maneuvers are performed, no height changes are considered and the velocities are strictly limited here, this assumption is reasonable. To be able to calculate the flight time even when considering the fading of the dynamics presented above, the time t_i for each aircraft is added as an additional state according to the equation:

$$\dot{t}_i = 1 \cdot \delta_{x,i} \cdot \delta_{y,i} \cdot \delta_{t,ini,i} \tag{10.50}$$

The final values for this newly introduced state, $t_{f,i}$ can then be used to calculate the overall cost for a scenario as:

$$t_{sum} = \sum_{i=1}^{N} t_{f,i} \tag{10.51}$$

In general, at this point, other more detailed cost functions like operational cost, fuel consumption, emissions, or a combination thereof can also be integrated into the simulation.

Within the scope of this work, fairness is represented by an even distribution of costs to all aircraft inside a scenario. As not all aircraft have the same distance to cover inside the considered sector, the absolute costs for crossing the sector are not a suitable means of comparison. Instead, the relative cost change with respect to the case where every aircraft can cross the sector freely, is used. These relative increases of costs for resolving the conflicts, compared to the reference case, are calculated from

$$c_i = \frac{t_{f,i} - t_{f,i,\min}}{t_{f,i,\min}} \tag{10.52}$$

where $t_{f,i}$ is the crossing time of the aircraft i in the scenario with all conflicts being resolved and $t_{f,i,\min}$ is the minimum crossing time of the aircraft i if all other aircraft were neglected. A scenario is considered fair if these relative cost increases are evenly distributed.

Hence, the goal of the optimization is to minimize every single cost increase c_i. Thus, the problem can be interpreted as a multi criteria optimization problem, like the ones presented in section 3.3, with the cost function vector containing one entry for each participant:

$$J_A = \begin{pmatrix} c_1 \\ \vdots \\ c_N \end{pmatrix} \tag{10.53}$$

When looking at the problem from a statistical point of view, the sample mean c_m and the sample variance c_{var} of the aforementioned cost increases can be calculated from

$$c_m = \frac{1}{N} \cdot \sum_{i=1}^{N} c_i \tag{10.54}$$

and

$$c_{var} = \frac{1}{N-1} \cdot \sum_{i=1}^{N} (c_i - c_m)^2 \tag{10.55}$$

The variance of the relative cost increases of all aircraft may be considered as an alternative means of fairness which leads to another multi criteria optimization problem with the cost function vector

$$J_B = \begin{pmatrix} c_{sum} \\ c_{var} \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{N} c_i \\ \frac{1}{N-1} \cdot \sum_{i=1}^{N} (c_i - c_m)^2 \end{pmatrix} \tag{10.56}$$

In this formulation, the minimization of the sum c_{sum} is equivalent to the minimization of the mean c_m and the minimization of the variance c_{var} is equivalent to the minimization of the standard deviation $c_{std} = \sqrt{c_{var}}$. [Bit+14]

10.4.5 Multi Criteria Optimization

Several methods for solving multi criteria optimization and optimal control problems exist, where a short overview of the most relevant scalarization techniques can be found in section 3.3.

Here, the problem is investigated using two different cost function formulations as stated in (10.53) and (10.56). For these two formulations different solution algorithms are applied and presented below.

Weighted Sum Scalarization

Using the method from section 3.3.1 and the cost formulation from equation (10.53), the scalarized cost function becomes:

$$J_{sum} = \sum_{i=1}^{N} w_i \cdot c_i \tag{10.57}$$

As the *relative* cost increases ensure a certain fairness level in this formulation, all weights in equation (10.57) can be chosen as:

$$w_i = 1 \qquad \forall i \in 1, \ldots, N \tag{10.58}$$

Overall, this optimization will result in the minimum summed cost increases without considering fairness in more detail. From a multi criteria optimization problem point of view, fixing the weights may result in one point in the approximation of the PARETO front. [Bit+14]

Weighted Metrics Scalarization

The weighted metrics scalarization from section 3.3.2, which is based on a p-Norm, amplifies big cost increases and reduces small ones for $p > 1$, potentially increasing fairness in the scenario. When omitting a target cost value, the cost function (10.53) becomes:

$$J_{\|c\|_p} = \left(\sum_{i=1}^{N} w_i \cdot |c_i|^p \right)^{\frac{1}{p}}, \quad p > 1 \tag{10.59}$$

As the scenarios considered here get more constrained when the separation needs to be ensured compared to the case when free flight is possible, the cost functions can only be equal to or greater than the reference costs. Thus, all relative cost changes are positive:

$$c_i \geq 0, \qquad \forall i = 1, \ldots, N \tag{10.60}$$

by choosing all weights $w_i = 1, \forall i = 1, \ldots, N$, again, and with (10.60), (10.59) becomes:

$$J_{\|c\|_p} = \left(\sum_{i=1}^{N} (c_i)^p \right)^{\frac{1}{p}}, \quad p > 1 \tag{10.61}$$

On the other side, when using a target cost value c_T, the optimization criterion (10.53) becomes:

$$J_{T,p} = \left(\sum_{i=1}^{N} w_i \cdot (c_i - c_T)^p \right)^{\frac{1}{p}} \tag{10.62}$$

To be able to use this cost function formulation, a value for c_T is required, whose estimation may be challenging for ATM scenarios. Thus, an optimization minimizing the overall costs is performed here as a first step, before the minimum optimal cost value (regarding one aircraft) is scaled by k_T and used as target:

$$c_T = \min c_i \cdot k_T, \qquad k_T > 0 \tag{10.63}$$

The factor k_T can be tuned to shift the weight between overall costs and cost distribution inside a scenario and therefore can be used to estimate points close to the PARETO front of the problem – at least when chosen arbitrarily. In the example considered below, $p = 1$ is chosen and all weights are set to $w_i = 1$ using the same reasoning as before. The resulting cost function is:

$$J_T = \sum_{i=1}^{N} w_i \cdot (c_i - c_T) \tag{10.64}$$

In this formulation, deviations from the target value to lower cost values are penalized equal to deviations to higher cost values. [Bit+14]

Tschebyscheff Scalarization

Using the scalarization from section 3.3.4, setting all weights to $w_i = 1$, and omitting the target cost, the following cost function can be constructed from the cost function formulation (10.53):

$$J_{\|c\|_\infty} = \lim_{p\to\infty} \left(\sum_{i=1}^{N} (c_i)^p \right)^{\frac{1}{p}} = \max_i (c_i) \tag{10.65}$$

In order to be able to numerically solve the resulting TSCHEBYSCHEFF problem, the transformation from section 3.2.2 is used. A problem that will arise when using this solution strategy is the fact that only the maximum cost function is minimized. All other cost functions may stay above their respective optimal values (but below the maximum cost function value, of course) and will not be further pushed down by the optimization algorithm. To tackle this problem the approach may be extended by the *Limited Minimum Sum Optimization* that is presented in the next section.

Limited Minimum Sum Optimization

When using the weighted sum scalarization, the overall costs become minimal without considering a fair cost distribution between the different participants. When using the TSCHEBYSCHEFF scalarization the opposite holds. In this approach these two techniques are combined: First, the problem based on the TSCHEBYSCHEFF scalarization is solved, before afterwards the resulting maximum individual cost is used as a limit for all individual costs in the problem:

$$c_i \leq c_{max}, \qquad i = 1, \ldots, N \tag{10.66}$$

In a second optimization, a minimization of the overall cost function is performed to lower the costs for all participants that are not at their optimum yet. This method can be further extended by loosening the upper limit for any individual cost based on a parameter k_s:

$$c_i < k_s \cdot c_{max}, \quad \forall i, j \in 1, \ldots, N \tag{10.67}$$

By tuning k_s, a reduction in the overall costs may be achieved and the weighting of the overall costs to fairness may be shifted. Consequently, for different values of k_s an approximation of some points along the PARETO front may be calculated.

Weighted Sum Scalarization in Mean and Variance

When considering the second cost function formulation for the problem as stated in (10.56), another weighted sum formulation can be derived:

$$J_{mv} = w_m \cdot \left(\frac{J_{Sum}}{N} \right)^2 + w_{var} \cdot J_{var} = w_m \cdot \bar{c}^2 + w_{var} \cdot \frac{1}{N-1} \sum_{i=1}^{N} (c_i - \bar{c})^2 \tag{10.68}$$

Here, w_m and w_{var} are the weights for the two terms of the sum and have to be chosen arbitrarily. As only the relation between the two components is relevant the two scaling

factors only represent one degree of freedom. To be able to achieve a meaningful comparison between both values, the mean has been squared, as the variance also is of second order.

One possibility to choose the weights is setting one of them to zero and the other to one. When doing so, either a minimization of the mean, which is equal to the minimization of the sum presented before, or a minimization of the variance results. The latter has also been implemented in the considered scenarios. To ensure the quality of the solution by means of overall costs, an additional condition has been added to the calculations that limits the sum and therefore also the mean of the costs

$$c_{sum} \leq c_{sum,min} \cdot k_c, \qquad k_c \geq 1 \tag{10.69}$$

where $c_{sum,min}$ is calculated by minimizing the overall costs. This condition ensures a maximum in the summed cost and therefore also a maximum in the mean. Different values for the scaling k_c have been used in the examples leading to different optimal points.

Another possibility for the weights would be to set both to one, which leads to an equivalent weighting of the mean and the variance but in some cases might not be numerically favorable. Thus, the weights might also be calculated based on a previously run minimization of the mean (indicated by the asterisk $*$). Then both parts of the sum can be weighted such that they lie within the same numerical range by choosing:

$$w_m = \frac{1}{(\bar{c}^*)^2} = \frac{1}{\left(\frac{1}{N}\sum_{i=1}^{N} c_i^*\right)^2} \tag{10.70}$$

$$w_{var} = \frac{1}{\mathrm{var}(c_i^*)^2} = \frac{1}{\left(\frac{1}{N-1}\sum_{i=1}^{N}(c_i^* - \bar{c}^*)^2\right)^2} \tag{10.71}$$

The calculations based on the two latter formulations only lead to one point each on the approximated PARETO front. Anyway, choosing different combinations of w_m and w_{var} makes it possible to calculate multiple points. [Bit+14]

10.4.6 Scenarios and Results

Two example scenarios have been set up and solved in [Bit+14] in order to demonstrate how the methods presented before can be used in ATM applications considering a fair distribution of costs. The results are shown in detail again, here. All scenarios have been discretized using a trapezoidal collocation scheme while the numerical optimization is performed using *SNOPT* [GMS02] or WORHP [BW13], respectively.

Description of Scenario 1

The first scenario consists of four aircraft that are positioned pairwise opposite to each other. The scenario can be seen in figure 10.16. The aircraft starting points are marked by triangles, their destinations by crosses. The intersection of the trajectories is located at the origin of the locally fixed frame; the starting positions are 40 NM and 45 NM out and the final positions

are 60 NM and 65 NM out. All boundary points are either located on the $(x)_N$-axis or on the $(y)_N$-axis, as can be seen in the figure. Even though in current day operations, two-way traffic is separated vertically in order to avoid such conflicts, the example demonstrates the capabilities of the algorithm.

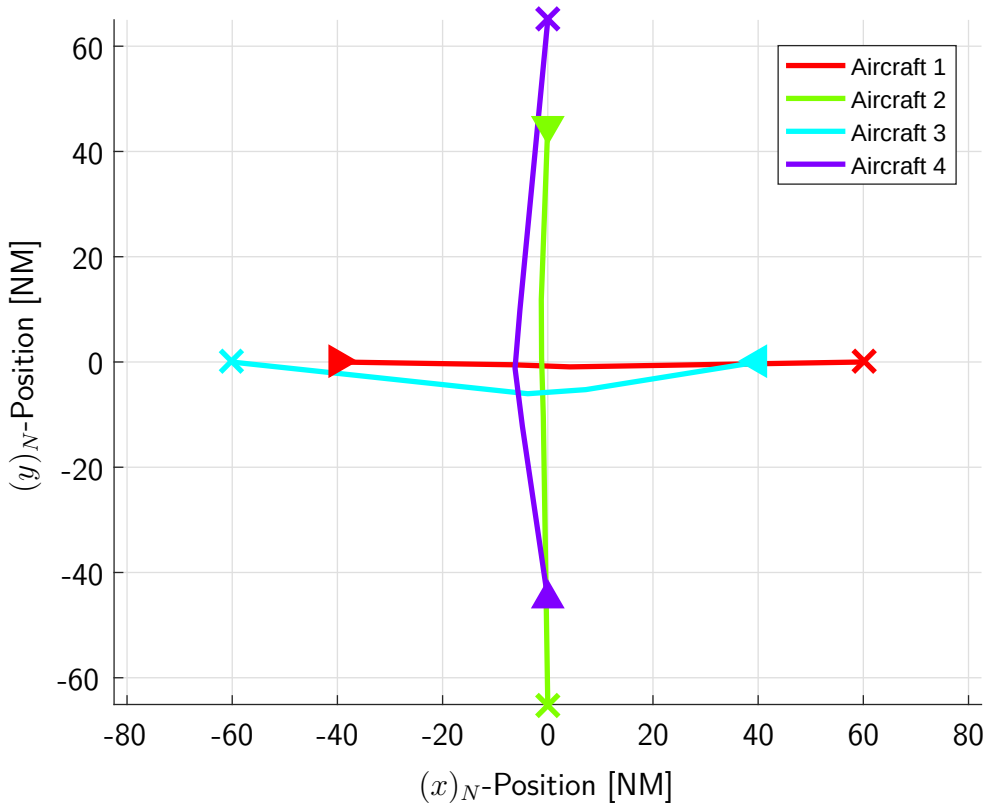

Figure 10.16: *Overall cost minimal solution for scenario 1. [Bit+14]*

The tracks in figure 10.16 show the solution for the scenario when minimizing the overall costs and ignoring any fairness aspects. It can clearly be seen, that aircraft 3 and 4 have to fly a noticeable detour while aircraft 1 and 2 are hardly affected by the solution of the conflict. Hence, in this scenario the cost minimal solution is not desirable for all parties. The numerical results for the relative increases in the flight time – compared to the conflict free scenario – and the total time for each aircraft can be seen in table 10.11. The cost increases have a mean of $c_m = 0.465\%$ and a standard deviation of $c_{std} = 0.339\%$. [Bit+14]

Results for Scenario 1

When using the aforementioned weighted metrics scalarization with $p = 2$ the mean of the costs in the first scenario becomes $c_m = 0.468\%$ while the variance is $c_{var} = 0.327\%$. As can be seen from figure 10.17, the results are very similar to that created using the weighted sum scalarization. The same holds for the minimization of the mean and the variance for a weighting of $w_m = w_{var} = 1$. The results created using the TSCHEBYSCHEFF scalarization, which in this case is equal to the limited minimum sum optimization if $k_s = 1$, tend to be

Table 10.11: *Cost function values for scenario 1 for the minimization of the mean compared to the reference scenario without conflict resolution.*

Aircraft Number	Flight Time	Cost increase
1	$1066s$	0.170%
2	$1172s$	0.106%
3	$1074s$	0.917%
4	$1178s$	0.668%

slightly better in fairness and a little worse in the overall costs. The results for the minimum target optimization as well as the results for the minimization of the variance strongly depend on the chosen parameters as can be seen in figure 10.18 and figure 10.19.

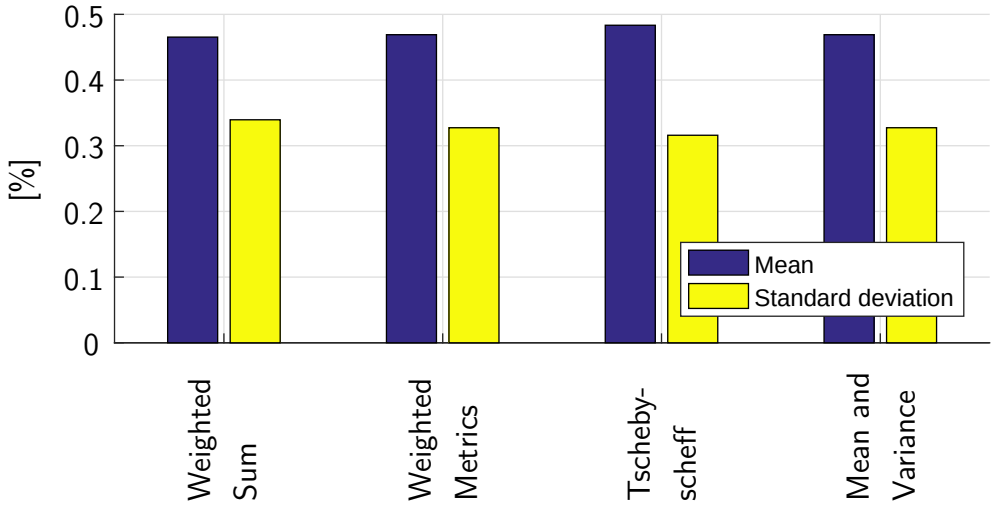

Figure 10.17: *Results in mean and standard deviation for scenario 1. [Bit+14]*

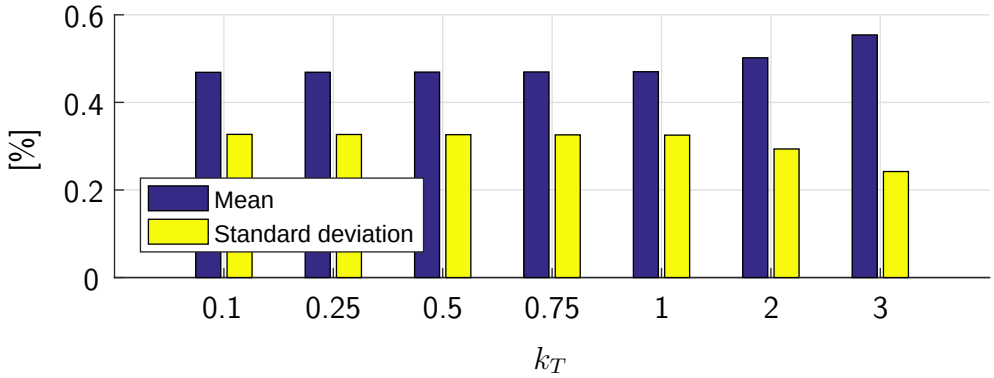

Figure 10.18: *Optimal mean and standard deviation for different parameters k_T in scenario 1 using the minimum target optimization. [Bit+14]*

When plotting the mean of the cost increases over its standard deviation for the different optimization methods, an approximation of the PARETO front of the problem may be drawn.

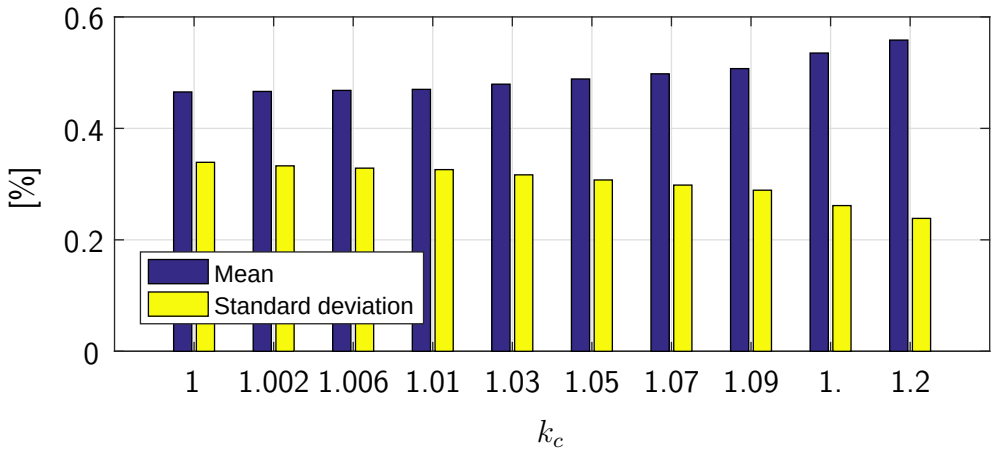

Figure 10.19: *Optimal mean and standard deviation for different parameters k_c in scenario 1 using the minimum variance optimization. [Bit+14]*

As stated in section 3.3, the methods used in the example are not capable of generating the exact PARETO front of the problem as potentially non-convex parts may not be covered. Figure 10.20 shows the results for the different methods in different colors. It can clearly be seen that no single optimal solution in mean and standard deviation exists but that it is only possible to improve one of them while the other gets worse. For means of comparison, the results for mean and variance are shown here, although they are not the real criterion for PARETO optimal points in the first problem formulation (10.53). [Bit+14]

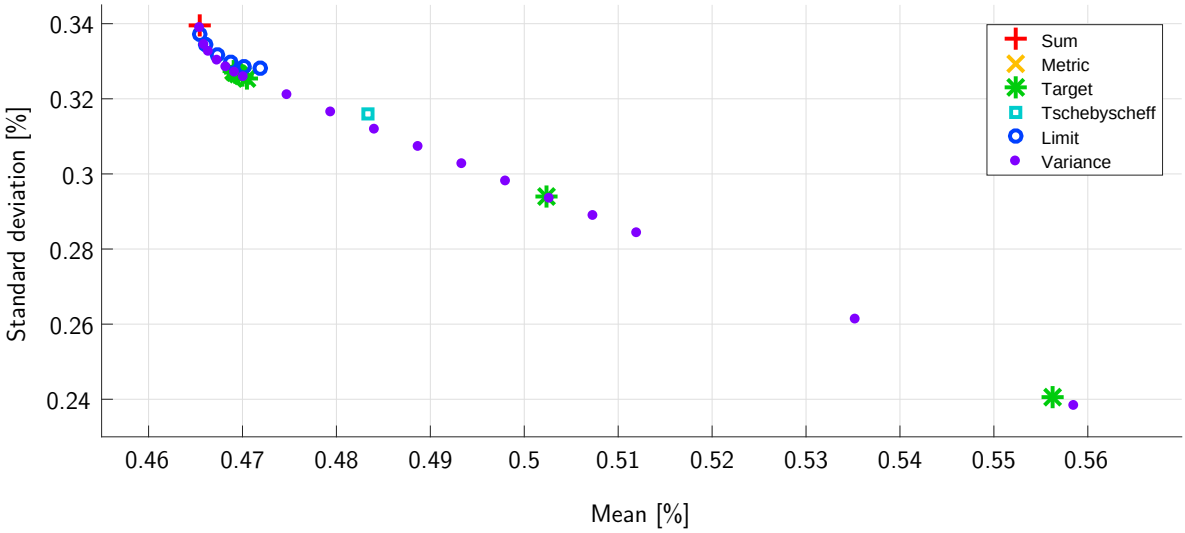

Figure 10.20: *Approximation of the PARETO front in mean and standard deviation for scenario 1. [Bit+14]*

Description of Scenario 2

Three aircraft are involved in the second scenario. They all start at an $(x)_N$-position of $x_0 = -50NM$. Their $(y)_N$-positions are spread regularly in $\delta_x = 10NM$ steps with the middle aircraft being positioned on the $(x)_N$-axis. They are all starting at the same time and need to mirror their $(y)_N$-positions until the end of the sector that is located at $x_{final} = 50NM$.

Figure 10.21 shows the geometry of the scenario, including the solution minimizing the overall costs. In this scenario the relative increase of costs for aircraft 1 is $c_1 = 3.724\%$ the relative increase for aircraft 2 is $c_2 = 0.452\%$ and the increase for aircraft 3 also is $c_3 = 3.724\%$. Consequently, the mean in this scenario is $c_m = 2.627\%$ while the standard deviation is $c_{std} = 1.537\%$. Table 10.12 lists the results for the minimization of the mean for the second scenario. [Bit+14]

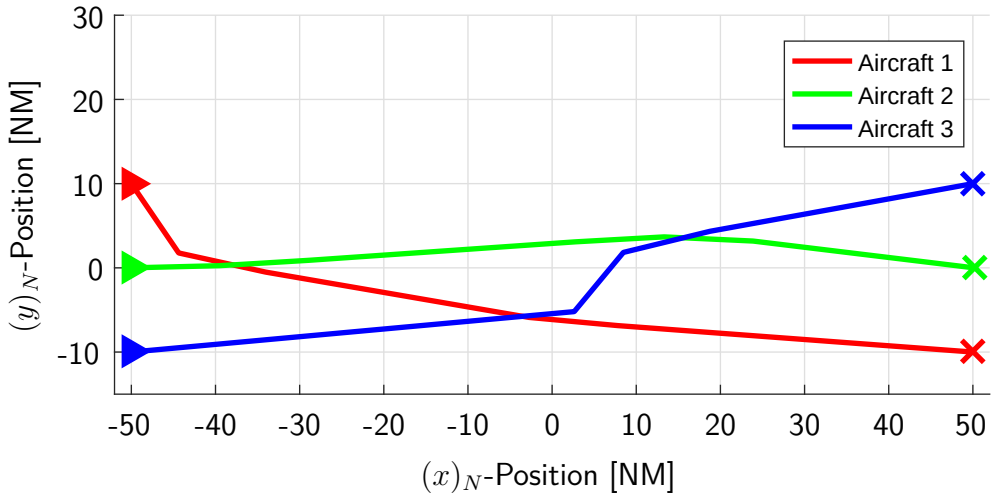

Figure 10.21: *Overall cost minimal solution for scenario 2. [Bit+14]*

Table 10.12: *Cost function values for scenario 2 for the minimization of the mean compared to the reference scenario without conflict resolution.*

Aircraft Number	Flight Time	Cost increase
1	$1126s$	3.724%
2	$1069s$	0.452%
3	$1126s$	3.725%

Results for Scenario 2

The results for scenario 2 are quite different from those of scenario 1 as the problem is more interconnected and the cost increases are higher. The minimum mean in the costs can again be achieved by minimizing the overall costs (of course), this time reaching a value of $c_m = 2.63\%$ with a standard deviation of $c_{std} = 1.54\%$. Figure 10.22 shows the mean and the standard

deviation for the minimization of the weighted metric with $p = 2$, the TSCHEBYSCHEFF method and the optimization of mean and variance both weighted by one $w_m = w_{var} = 1$. It is striking that the TSCHEBYSCHEFF optimization is able to reduce the variance to zero with a comparably small increase in the mean of the costs.

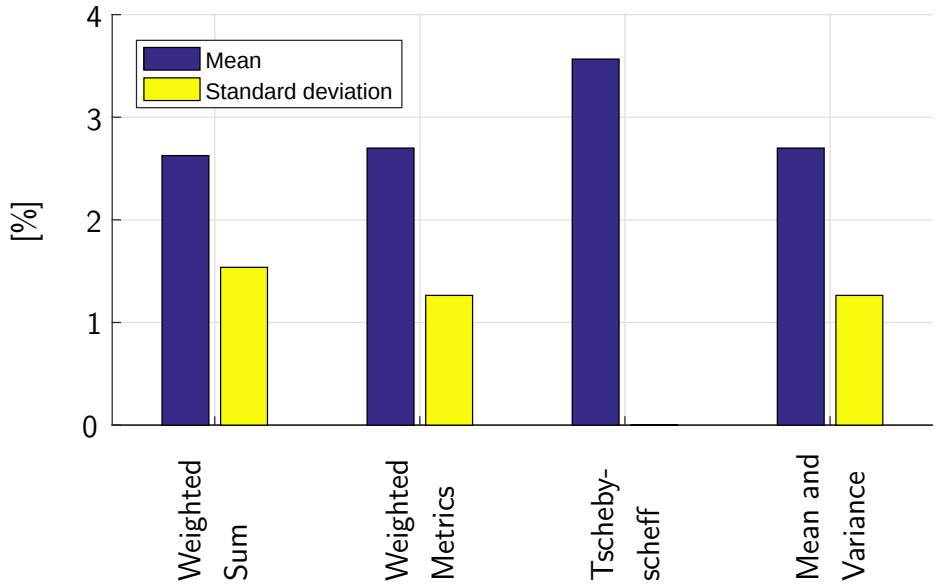

Figure 10.22: *Results in mean and standard deviation for scenario 2. [Bit+14]*

When using the minimum target optimization for the second scenario with different parameters (figure 10.23), the results are quite similar to the results calculated in the first example. Once again, the cost increases with increasing scaling factors k_T while the standard deviation simultaneously decreases.

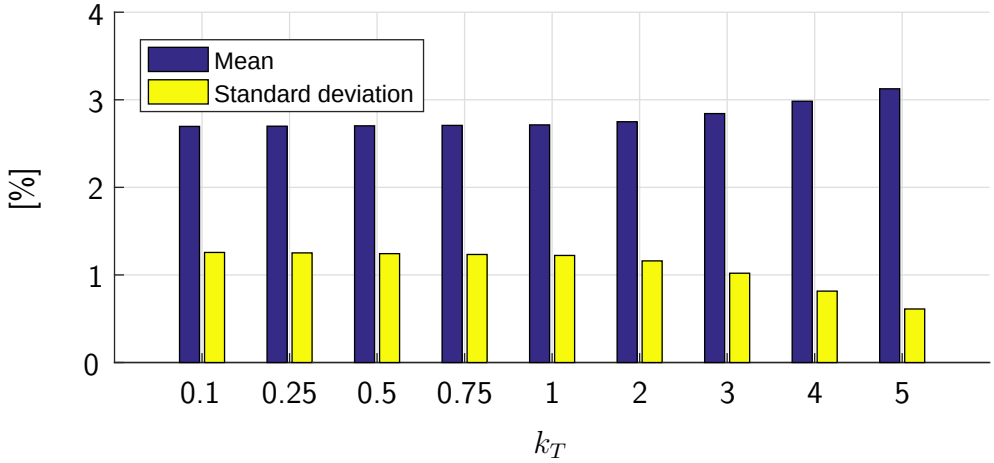

Figure 10.23: *Optimal mean and standard deviation for different parameters k_T in scenario 2 using the minimum target optimization. [Bit+14]*

The results for the minimization of the variance with different parameters k_c look very similar to the results in scenario 1 and can be seen in figure 10.24. Anyway, the numeric

values are bigger, as the conflict is generally more expensive to resolve. Thus, in this scenario an increase in fairness can be achieved with a comparably low relative increase in the mean.

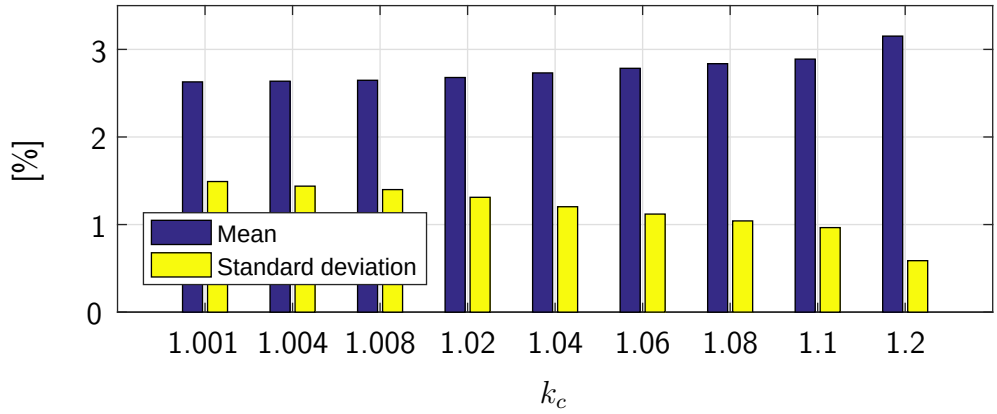

Figure 10.24: *Optimal mean and standard deviation for different parameters k_c in scenario 2 using the minimum variance optimization. [Bit+14]*

When comparing all methods for scenario 2 and plotting the mean of the costs against their standard deviation, the result is quite similar to the result created in scenario 1. These results can be seen in figure 10.25. This time, all PARETO optimal points are located on one clean line which is very likely to be the real PARETO front for the problem formulated in (10.56). Moreover, no *significant* point (meaning that a huge decreases in one dimension brings a relatively small increases in the other) can be found. So, the human decision maker is challenged and no preferable point can be detected. [Bit+14]

Discussion

The results of the calculations show that no method is superior to any other – neither regarding the results nor with respect to computational time. Instead, all methods are able to calculate some points that are somewhere close to the PARETO front of the overall problem. Using a combination of the methods presented, a good approximation for the PARETO front may be found – depending on the considered scenario. In many scenarios no preferable solution may exist in which either the mean or the variance could be lowered without significantly increasing the other. Finally, the decision for one point inside the solution still remains with the decision maker, i.e. the human controller.

The presented study was a first work on the topic as no other publication was found that combines multi criteria optimization with optimal control to calculate optimized solutions for ATM conflicts. This may also be connected to the fact that the cost increases are marginal and may not matter to the airlines as long as there is other aspects where much higher savings can be achieved. Anyway, there is potential for future research: First of all, further investigations on PARETO optimality of the calculated points can be done. Furthermore, the cost functions used here are only based on time which is a first approximation of the real costs and can be extended to much more realistic models. Besides the aircraft cost functions, also controller

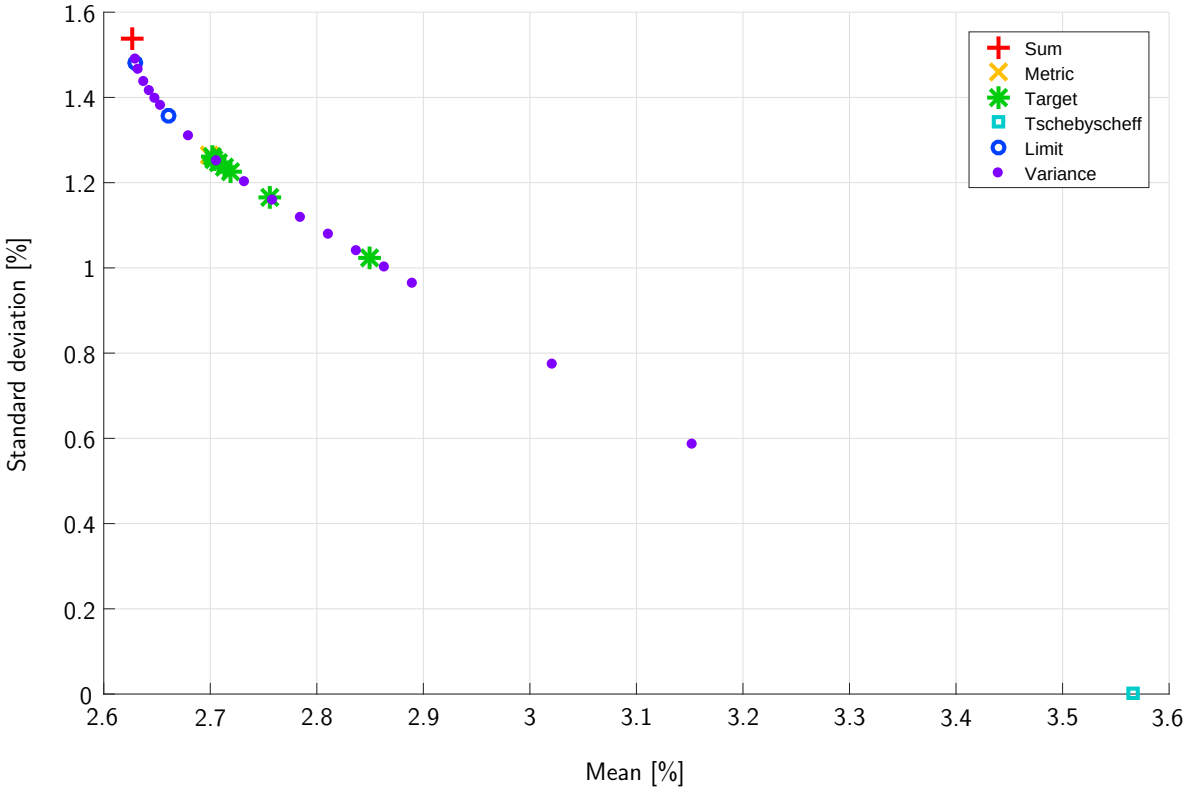

Figure 10.25: *Approximation of the* PARETO *front in mean and standard deviation for scenario 2. [Bit+14]*

workload and the number of maneuvers might be considered as both are crucial to safety. Moreover, uncertainties in the scenarios like wind may be added. Finally, studies may be conducted analyzing the multi criteria optimal control problem in more detail as introduced in section 3.3.7. [Bit+14]

Chapter 11

Air Race Related Applications

The calculation of optimal air race trajectories is one of the most challenging applications of aircraft trajectory optimization as the aircraft are mostly operated at their limits. This requires the use of highly non-linear, high fidelity simulation models together with comparably fine discretization grids for the controls and the states. Moreover, the different time scales of the fast rotational dynamics (the aircraft in the examples feature roll rates of up to $420\frac{\text{deg}}{s}$) and the comparably slow translational dynamics make the resulting differential equations *stiff*, representing another challenge for solving respective optimal control problems.

Thus, problems from the field of air race trajectory optimization are used in this chapter to demonstrate the results presented in chapters 5–9. The first scenario considered in the examples contains an air race track, defined by seven race gates that are located at the positions given in table 11.1. All positions are measured with respect to the start gate (Gate 1) – which at the same time is the final gate – and are given in meters. The orientation for each gate is the direction of the first passing – later in the race it may also be reversed. The aircraft competing in the race have to pass the gates several times, where the race gate order is given as:

$$1-2-3-4-5-4^*-6^*-7^*-1^*-7-6-4-5-4^*-3^*-2^*-1^*$$

Here, an asterisk indicates that the respective race gate needs to be passed from the opposite direction. Consequently, the overall race course is defined by 17 interior point conditions that are transformed to boundary conditions using the transformation from section 3.2.2, resulting in a total of 16 phases in the optimal control problem.

Figure 11.1 depicts the optimal trajectory for the full rigid body simulation model with the race gates being represented by red crosses. The starting gate, also forming the final gate, can be seen in the bottom right corner of the plot, located at $(0, 0, 0)$.

Moreover, the theory presented in section 3.2.5 is used in the examples to perform a further analysis of the results. In this analysis, the costates (estimated based on the LAGRANGE multipliers of the discretized problem) and the multipliers for the controls are used to approximate the *Hamiltonian*. Furthermore, the derivative of the *Hamiltonian* with respect to the controls is derived from the discretized problem and is used in the analysis. Overall, the numerical

Table 11.1: *Position and orientation of the race gates relative to the start gate.*

Gate	Gate Position			Orientation
No.	$(x)_N\ [m]$	$(y)_N\ [m]$	$(z)_N\ [m]$	$\chi_G\ [rad]$
Gate 1	0	0	0	-0.174
Gate 2	237.63	-58.55	0	-0.174
Gate 3	565.00	54.45	0	-0.172
Gate 4	880.01	-29.25	0	0
Gate 5	1127.08	-79.46	0	0
Gate 6	610.39	-110.02	0	3.142
Gate 7	299.40	35.62	0	3.140

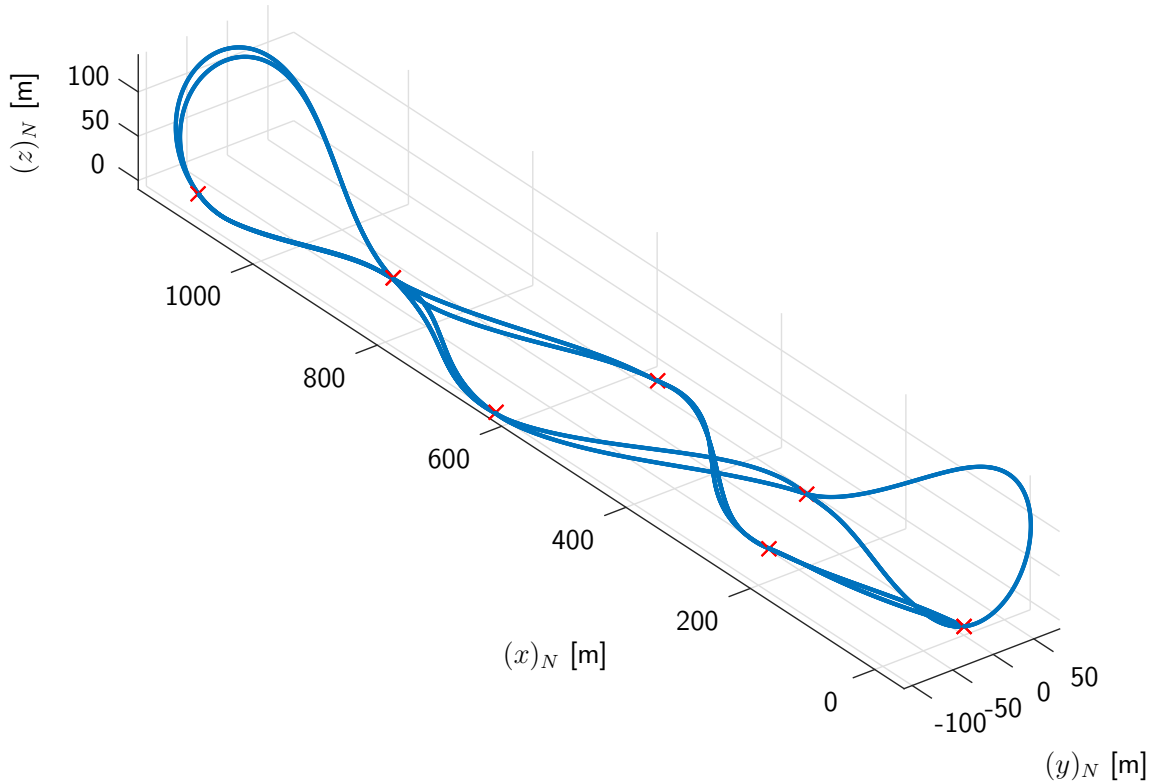

Figure 11.1: *Time optimal spatial trajectory through the air race course.*

results show very good accordance with the underlying theory, confirming good accuracy of the computations.

The optimization of air race trajectories has some tradition at the *Institute of Flight System Dynamics* of TUM. Florian Fisch worked on the topic for some years and published his results on several conferences [Fis+08, Fis+09, FLH12], in a scientific journal [FBH12] and in his dissertation [Fis11]. Results similar to the ones shown here were partially also presented in the conference papers [BFH12, Bit+13a], before.

In [Fis11] different aircraft models are described that are consecutively used to solve an optimal control problem. The main idea of the work is to start solving the respective optimal

control problem using a simple point mass simulation model before adding more and more complex rotational dynamics. Between two consecutive optimizations, simulations are used to calculate the required initial guesses. The work does not focus on linearly appearing controls and overcomes any issues related to them by changing the problem formulation by adding particular LAGRANGE cost functions.

11.1 Simulation Model

The simulation model used in the optimization studies presented here has been constructed based on the work [Fis11] and the experience of HERBERT WEIRATHER who is an aerobatic pilot also flying the *Zivko Edge 540T*. Based on his expert knowledge about the aircraft behavior, the parameters of the simulation model have been tuned in order to match the behavior of the real aircraft as closely as possible. Anyway, no real parameter estimation has been performed. HERBERT WEIRATHER did his Bachelor Thesis at the *Institute of Flight System Dynamics* under the supervision of this author. [Wei14]

The models are based on the theory presented in chapter 2. In all examples a locally fixed navigation frame is used for determining the position of the aircraft as the spatial extent of air race trajectories is comparably small and does not require the use of any global coordinate reference. Consequently, the position propagation equations (2.25) were used. For the same reason, a flat and non-rotating earth is used in the models, like in equation (2.39), (2.59) and (2.73). Furthermore, no wind is considered in the examples, rendering all aerodynamic and kinematic quantities equal. Anyway, the addition of wind to the simulation model is easily possible, even though it cannot be determined in advance how the performance of the optimization algorithm may change when doing so. The origin of the locally fixed coordinate frame used for determining the position of the aircraft is located at the center of the start gate of the race course. As only races above water are considered here, all race gates are located at the same altitude in the examples.

The basic parameters of the aircraft model are listed in table 11.2, while the aerodynamic parameters are described in the next sections. [Wei14]

11.1.1 Aerodynamic Model

The aerodynamic forces are calculated based on the force equation (2.103), using the following relationships for the aerodynamic coefficients:

$$C_L = C_{L,0} + C^*_{L,\alpha}(\alpha_A) + C_{L,q} \cdot \tilde{q} + C_{L,\eta} \cdot \eta \tag{11.1}$$

$$C_Q = C_{Q,\beta} \cdot \beta_A + C_{Q,p} \cdot \tilde{p} + C_{Q,r} \cdot \tilde{r} + C_{Q,\zeta} \cdot \zeta \tag{11.2}$$

$$\begin{aligned} C_D = {} & C_{D,0} + k \cdot (C_{L,\alpha} \cdot \alpha_A)^2 + k_\beta \cdot (C_{Q,\beta} \cdot \beta_A)^2 + \\ & (C_{D,\xi} \cdot \xi_{CMD})^2 + (C_{D,\eta} \cdot \eta_{CMD})^2 + (C_{D,\zeta} \cdot \zeta_{CMD})^2 \end{aligned} \tag{11.3}$$

Table 11.2: *General model parameters for the aircraft used in the examples.*

Parameter name	Symbol	Unit	Value
Aircraft mass	m	kg	693.0
Wing reference area	S	m^2	8.928
Wing span	b	m	7.5
Mean length of aerodynamic chord	$\bar{c}$	m	1.44
Propulsion reference velocity	V_R	m/s	30
Maximum engine thrust	T_R	N	$0.8 \cdot m \cdot g$
Moments of inertia	I_{xx}	$kg \cdot m^2$	420.3035
	I_{yy}	$kg \cdot m^2$	726.7184
	I_{zz}	$kg \cdot m^2$	827.3201

As the flights are partially conducted in the proximity of stall, a model for the lift coefficient that features a linear dependency on the angle of attack is not sufficient. Hence, a non-linear function $C^*_{L,\alpha}(\alpha_A)$ is used in the lift equation, that is modeled as

$$C^*_{L,\alpha} = a \cdot \tanh\left(\alpha_A \cdot \frac{180}{\pi} \cdot b\right) - e \cdot \left(1 + \tanh\left(\alpha_A \cdot \frac{180}{\pi} \cdot d - c\right)\right) \tag{11.4}$$

using the idea of combining different functions and fading them by a hyperbolic tangent that is presented in section 6.5. The parameters for equation (11.4) are listed in table 11.3. Figure 11.2 shows a plot of the relationship. As a decrease in lift close to stall does not come with a decrease in drag, a quadratic lift-drag relationship may not be used here. Instead, a basic linear lift model incorporating the constant value $C_{L,\alpha}$ is used to calculate the lift that is used in the quadratic drag equation (11.3). The remaining aerodynamic force coefficients can be found in table 11.4.

Table 11.3: *Parameters for the lift coefficient depending on the angle of attack.*

a	b	c	d	e
2.763	0.0443	8.1722	0.4473	0.791

The aerodynamic moments of the model are calculated using the relationship from equation (2.104). The coefficients appearing therein are modeled using the following relationships:

$$C_l = C_{l,\xi} \cdot \xi + C_{l,\zeta} \cdot \zeta + C_{l,p} \cdot \tilde{p} + C_{l,r} \cdot \tilde{r} + C_{l,\beta} \cdot \beta_A \tag{11.5}$$

$$C_m = C_{m,0} + C_{m,\eta} \cdot \eta + C_{m,q} \cdot \tilde{q} + C_{m,\alpha} \cdot \alpha_K \tag{11.6}$$

$$C_n = C_{n,\xi} \cdot \xi + C_{n,\zeta} \cdot \zeta + C_{n,p} \cdot \tilde{p} + C_{n,r} \cdot \tilde{r} + C_{n,\beta} \cdot \beta_A \tag{11.7}$$

As can be seen, the model for the aerodynamic moments is linear, using the coefficients that can be found in table 11.5.

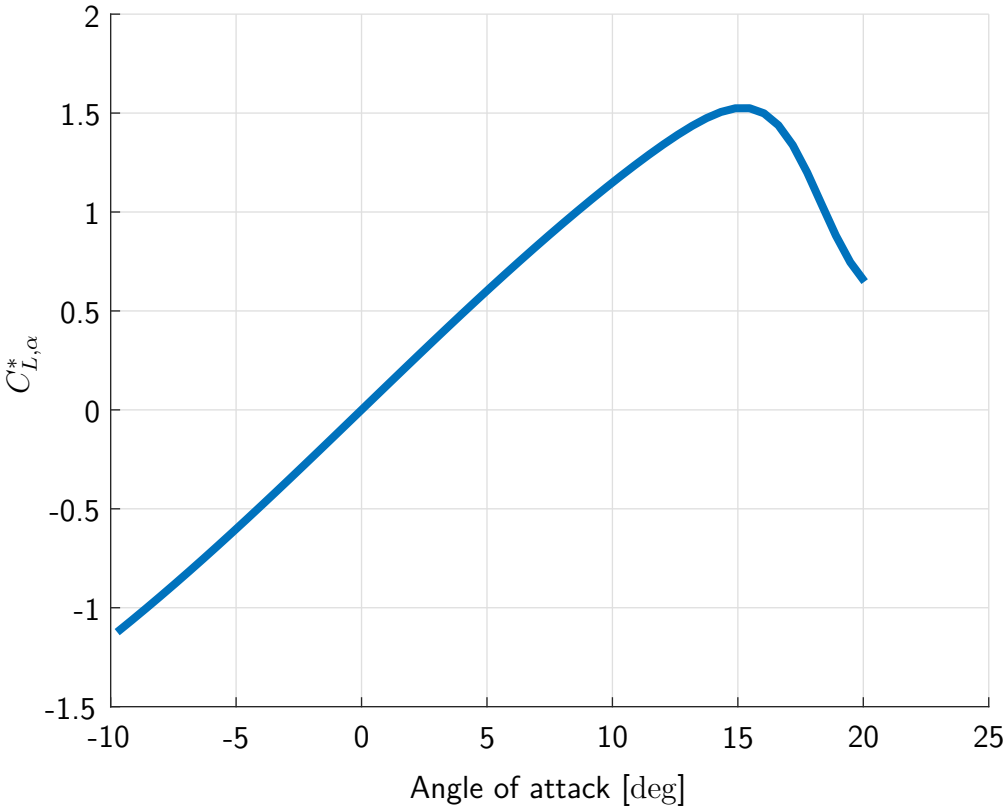

Figure 11.2: *Part of the lift coefficient depending on the angle of attack.*

Table 11.4: *Aerodynamic force coefficients.*

$C_{D,0}$	0.0295	$C_{Q,\beta}$	-0.589355	$C_{L,0}$	0.055
k	0.05134	$C_{Q,p}$	0.042480	$C_{L,\alpha}$	4.75
k_β	1.696770	$C_{Q,r}$	0.048340	$C_{L,\eta}$	-0.073242
$C_{D,\xi}$	0.05	$C_{Q,\zeta}$	-0.195313	$C_{L,q}$	-3.479492
$C_{D,\eta}$	0.05				
$C_{D,\zeta}$	0.05				

Table 11.5: *Aerodynamic moment coefficients.*

$C_{l,\beta}$	0.024902	$C_{m,\eta}$	-0.317383	$C_{n,\beta}$	0.149902
$C_{l,p}$	-0.583008	$C_{m,q}$	-16.930176	$C_{n,p}$	0.014648
$C_{l,r}$	0.087891	$C_{m,\alpha}$	-0.145406	$C_{n,r}$	-0.4731
$C_{l,\xi}$	-0.2126	$C_{m,0}$	-0.004883	$C_{n,\xi}$	-0.0073
$C_{l,\zeta}$	0.001			$C_{n,\zeta}$	0.170898

11.1.2 Engine Dynamics

Even though in realistic air races the pilots are hardly ever touching the thrust lever, its position is considered as an input to the optimal control problem here for the sake of completeness. As the thrust created by the engine cannot follow the commanded thrust instantaneously, a dynamic model is incorporated here. The engine dynamics are directly applied to the thrust lever position command, using the dynamic equation

$$\dot{\delta}_T = \frac{1}{T_{Engine}} \cdot (\delta_{T,CMD} - \delta_T) \tag{11.8}$$

with $T_{Engine} = 0.5$. The resulting thrust force is then modeled depending on the current air speed and linear in the thrust lever position:

$$\left(F^G_{P,x}\right)_B = \delta_T \cdot \frac{V_R}{V_A} \cdot T_R \tag{11.9}$$

11.1.3 Augmented Point Mass Simulation Model

In the pure point mass formulation, the bank angle is directly used as a control, allowing discontinuities if not further constrained. In order to prevent jumps in the bank angle, strongly simplified linear roll dynamics may be added to the point mass equations as:

$$\dot{\mu}_K = \dot{\mu}_{K,CMD} \tag{11.10}$$

In this augmented model formulation, $\dot{\mu}_{K,CMD}$ is used as a command that is integrated along with the model dynamics to get a continuous behavior of the bank angle μ_K. The limits in the roll dynamics of the aircraft can then be accounted for by limiting the admissible range of $\dot{\mu}_{K,CMD}$.

Alternatively, μ_{K_CMD} can directly be used for controlling the bank angle, where the same roll rate limits can be achieved using the dynamic constraint formulation from section 6.4.

11.1.4 Path Constraints

Besides the *dynamic* constraints formed by the aircraft simulation model presented before, algebraic constraints have to hold along the whole trajectory. Table 11.6 lists the limits for these path constraints.

As the $(z)_N$-axis is pointing downwards and the origin of the locally fixed frame is located at the center of the clearance area of the initial race gate, the limit in the $(z)_N$ position of the aircraft is positive and $10m$ below the center of this area. The regulations of air races limit the load factor in $(z)_B$ direction of the body fixed frame as stated in table 11.6. In order to prevent the aircraft from stalling, the stall speed limit needs to be satisfied. On the other hand, the maximum speed is limited by the regulations of the air race and should ensure safety. The aerodynamic angle of attack and the aerodynamic angle of sideslip need to be limited as the aerodynamic model is only valid within the given range. Extreme aerodynamic situations

may not be described correctly by the formulation introduced above and consequently must be avoided. The roll rate limits given in table 11.6 are motivated by the structural limits of the aircraft and the pilot.

Table 11.6: *Path constraints applied in the air race examples.*

Description	Symbol	Unit	Value
z-Position limit due to ground clearance	$(z)_{N,max}$	m	10
Load factor limits due to regulations	$(n_z)_{B,min}$	-	-2
	$(n_z)_{B,max}$	-	10
Stall speed limit	$V_{A,min}$	m/s	25
Maximum speed due to regulations	$V_{K,max}$	m/s	102.9
Aerodynamic angle of attack limits	$\alpha_{A,min}$	rad	-0.17
	$\alpha_{A,max}$	rad	0.35
Aerodynamic angle of sideslip limits	$\beta_{A,min}$	rad	-0.17
	$\beta_{A,max}$	rad	0.17
Roll rate limits	p_{min}	rad/s	-7.33
	p_{max}	rad/s	7.33

11.1.5 Race Gate Constraints

Besides the path constraints that have to be fulfilled along the whole trajectory, additional constraints need to be respected whenever the aircraft is passing through a race gate. Table 11.7 lists the constraints applicable there. As the time of passing through a race gate is unknown before solving the respective optimal control problem, the race gate constraints need to be incorporated in the problem formulation using the phase approach described in section 3.2.2.

At every race gate, the position of the aircraft has to match the center of the clearance area of the race gate in all three spatial dimensions. Here, no tolerance in the position is allowed resulting in equality constraints. The same holds for the kinematic course angle χ_K of the aircraft which is required to exactly match the orientation of the race gate. Furthermore, the regulations of the air races require the pilots to pass through the race gates wings level, meaning that the bank angle of the aircraft needs to be zero: $\mu_k = 0$. Here, the rules allow a tolerance of $\pm 10\,\mathrm{deg}$ that is also used in the optimizations.

11.1.6 Cost Function

The cost function to be minimized is the overall race time. This cost function may be formulated as the LAGRANGE cost function:

$$L(\mathbf{x}(t), \mathbf{u}(t), t) = 1 \tag{11.11}$$

Table 11.7: *Constraints applied at the race gates in the air race examples.*

Description	Symbol	Unit	Tolerance
x-Position of the race gate	$(x)_N$	m	± 0
y-Position of the race gate	$(y)_N$	m	± 0
z-Position of the race gate	$(z)_N$	m	± 0
Orientation of the race gate	χ_K	rad	± 0
Bank angle in the race gate	μ_A	deg	± 10

resulting in the following overall cost function:

$$J = \int_{t_0}^{t_f} 1 \, dt = t_f - t_0 \tag{11.12}$$

The initial time is fixed to $t_0 = 0$ as it represents a redundant degree of freedom.

11.1.7 Initial Guess

In the first examples, the initial guess that is required for solving the optimal control problem is determined based on two dimensional DUBINS paths [Dub57] for the position of the aircraft – unless specified otherwise. Figure 11.3 shows the initial trajectory for the race course that has been defined in table 11.1. All other values are initialized according to table 11.8.

Table 11.8: *Initial guess for the states and the controls of the air race examples.*

Variable	Type	Value	Variable	Type	Value
$(x)_N$	Dubins path	-	β_A	constant	0
$(y)_N$	Dubins path	-	p	constant	0
$(z)_N$	constant	0	q	constant	0
χ_K	from Dubins path	-	r	constant	0
γ_K	constant	0	δ_T	constant	1
μ_K	constant	0	ξ	constant	0
V_K	constant	$100\frac{m}{s}$	η	constant	0
α_A	constant	0	ζ	constant	0

11.2 Low Fidelity Trajectory Optimization

First, the results for the trajectory optimization using low fidelity simulation models will be presented and discussed. These include

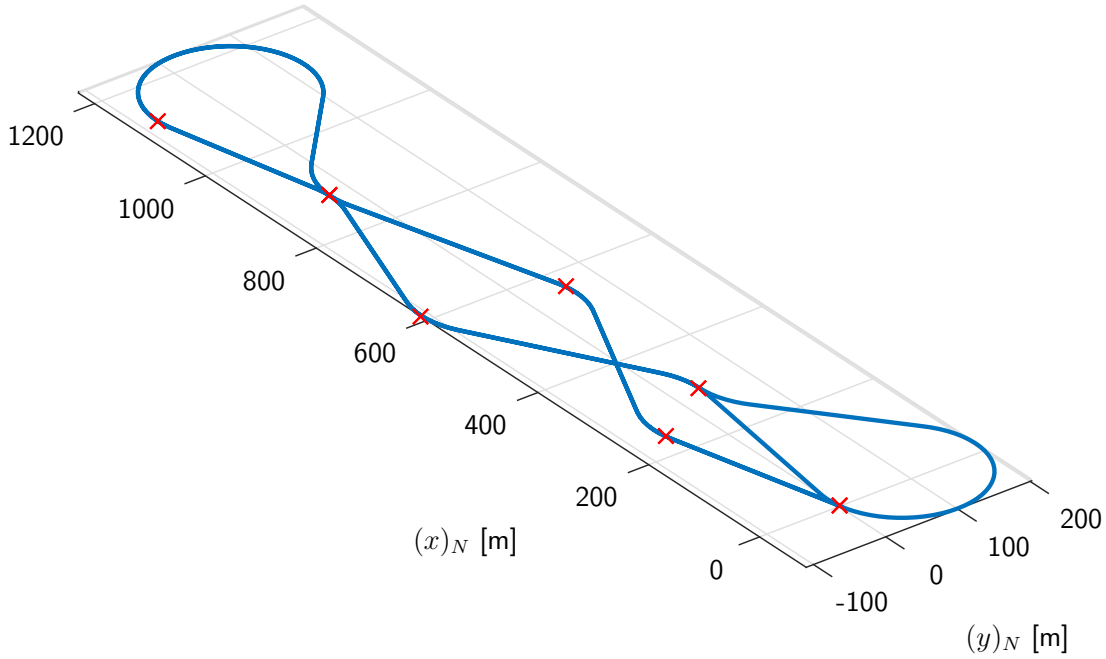

Figure 11.3: *Initial guess for the spatial trajectory through the air race course.*

- the straight forward approach using the augmented point mass simulation model formulated before, and
- the idea for the model reformulation avoiding singular controls, discussed in chapter 6 and especially presented in section 6.4.

Both problems were solved on a grid containing 501 discretization points per phase, resulting in a total of 8016 nodes. The states and the controls were discretized on the same grid. As the race gates are not distributed equally along the race track, not all phases in the optimal control problem feature the same simulation time. With a fixed number of grid points per phase, this results in different resolutions of the discretization with respect to the simulation time. In the examples shown here, the distance between two discretization points varies between:

$$0.0048s = \Delta t_{min} \leq \Delta t \leq \Delta t_{max} = 0.0129s \tag{11.13}$$

The grid is intentionally chosen this fine for two reasons: First, the results of the low fidelity optimization should be compared with the results from the high fidelity optimization presented in the next section, which requires a very fine grid due to the fast internal dynamics. Second, and more important, the highly dynamic race course requires the pilots to perform the right maneuvers at the exact right time, leading to the necessity of a very fine control grid. Consequently, the state grid also needs to be chosen very fine, as it may not be coarser than the control grid. This phenomenon will be seen in the discussion of the resulting control histories below.

For the solution of the discretized numerical optimization problems, *IPOPT* [WB06] is used with the constraint tolerance set to $tol_C = 10^{-6}$ and the optimality tolerance set to $tol_J = 10^{-5}$ for the augmented point mass model. The relatively low optimality tolerance is

required due to the bad numerical properties of the dynamic model (see section 6.2) which makes it impossible to solve the problem to a higher accuracy. When using the formulation with the dynamic constraints from section 11.2.2, an optimality tolerance of $tol_J = 5 \cdot 10^{-6}$ is feasible and used. All optimizations were performed on a personal computer equipped with an Intel Core i7-950 CPU with $3.07GHz$ and $20GB$ of RAM. The collocation code of *FALCON.m* features multi-threading and consequently makes use of all four CPU cores installed in the machine.

11.2.1 Augmented Point Mass Simulation Model

In this first section, the results for the optimization using the augmented point mass simulation model without any model reformulations are presented. Figure 11.4 shows the overall optimal trajectory through the given race course. The positions of the race gates are indicated by the red crosses. As most of the race gates have to be passed several times, the trajectory partially overlaps itself.

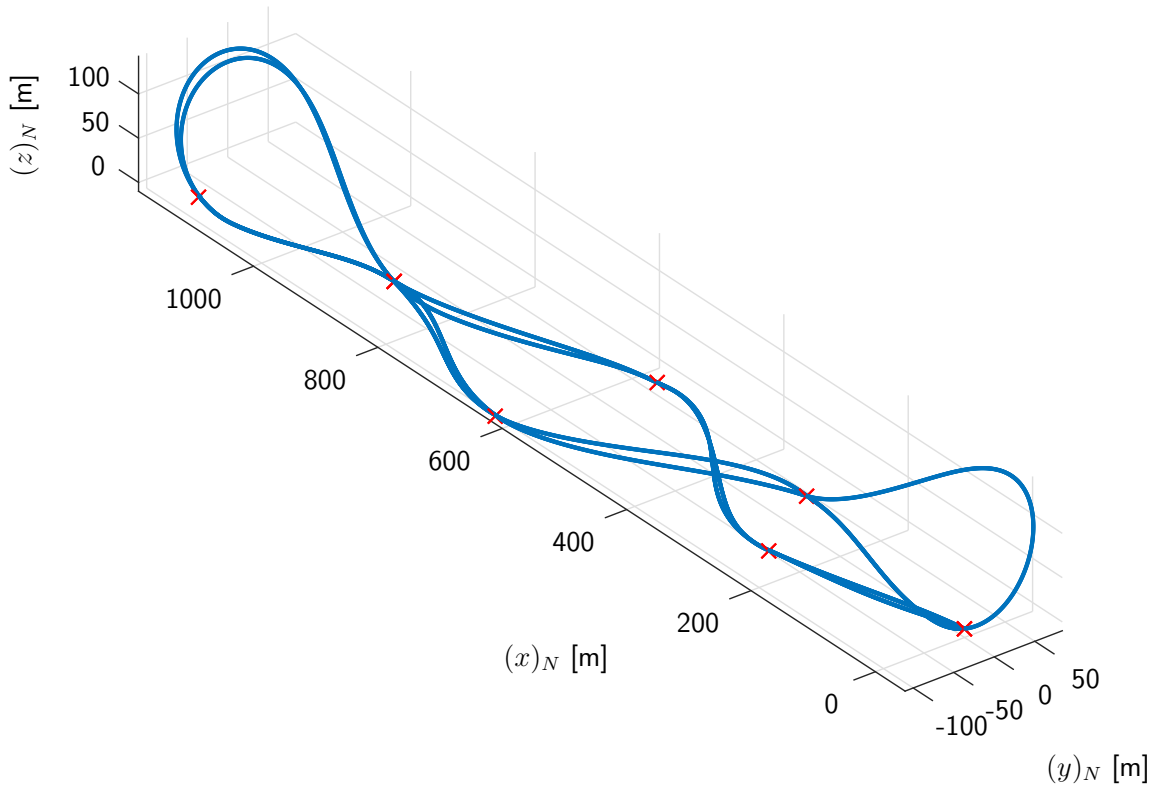

Figure 11.4: *Time optimal spatial trajectory for the augmented point mass simulation model.*

After discretization, the numerical optimization problem contained 96208 optimization variables and 72152 constraints. Hence, the gradient matrix of the discretized problem might contain up to $6.9416 \cdot 10^9$ elements but has a sparsity ratio of 99.9901%, resulting in 684336 structural non-zero entries. The numerical optimization took 13219 iterations to converge which equals approximately $10782s$ on the aforementioned machine. The resulting optimal cost function value is a race time of $t_f = 60.2253s$.

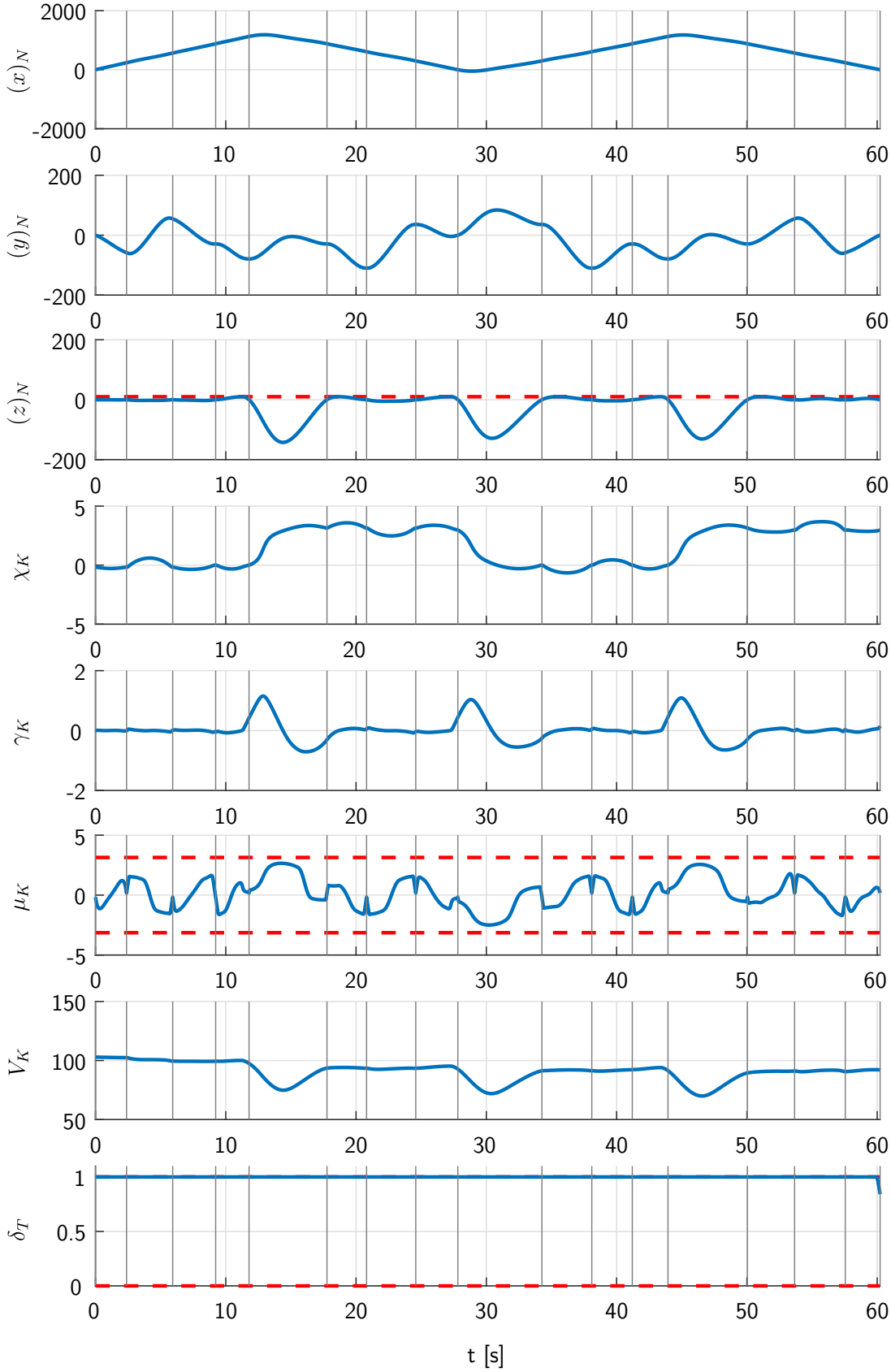

Figure 11.5: *Optimal state histories for the augmented point mass simulation model.*

Figure 11.5 shows the state histories along the optimal trajectory through the air race. In this and all forthcoming plots, the phase boundaries are illustrated by the gray, vertical lines. The dashed red lines represent the state boundaries to be respected along the race trajectory as given before. It can be seen that only the limits for $(z)_N$ and δ_T become active during the race. Furthermore, the three half Cuban eights – performed by the aircraft to achieve quick turns – become clearly visible in the $(z)_N$, V_K and γ_K states. The kinks that can be seen in the history of μ_K are required to fulfill the bank angle constraints when passing the race gates. At these points the aircraft is using the full dynamic envelope in the roll motion.

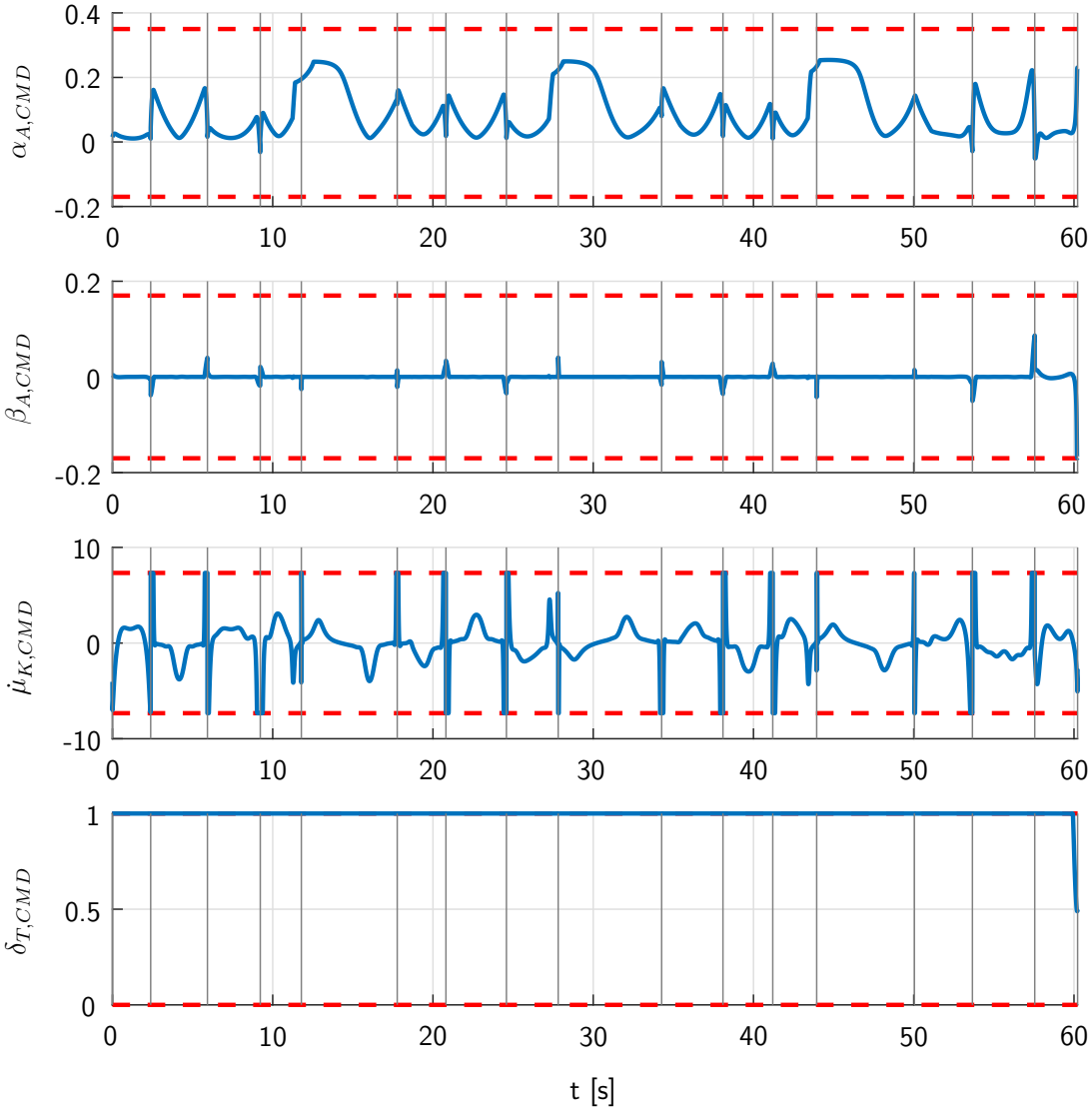

Figure 11.6: *Optimal control histories for the augmented point mass simulation model.*

The optimal control histories can be found in figure 11.6. It can be seen that only the bank rate $\dot{\mu}_{K,CMD}$ and the thrust lever position δ_T reach their limits. The angle of attack $\alpha_{A,CMD}$ is not at its boundaries except during the half Cuban eight maneuvers, where it reaches its maximum allowed value. The angle of sideslip $\beta_{A,CMD}$ is zero almost everywhere as it induces too much drag that slows the aircraft down and eventually causes it to loose time. Only when

passing the race gates, the angle of sideslip is used to be able to continue turning even though no bank angle is allowed. At these points the faster turning of the aircraft compensates for the loss in speed. The bank rate $\dot{\mu}_{K,CMD}$ reaches its limit whenever the aircraft is passing through a race gate, as mentioned before. The thrust lever $\delta_{T,CMD}$ remains in full throttle position throughout the whole race. Only at the very last instant in time a decrease can be seen which is a numerical artifact as the thrust at the very final discretization point does not influence the solution anymore. Especially in the angle of attack and the bank rate, quick jumps in the control inputs can be seen. The determination of the precise moment in time where these jumps take place require the aforementioned fine control grid. When too coarse grids are used in optimal control problems, one can often see chattering in the solution which can be interpreted as a repeated overshooting of the dynamics around the true location of the jump. When the resulting control history should be used in a real application, jumps are often not possible and an additional smoothing or filtering of the histories is required.

In figure 11.7 the vertical load factor in the body fixed frame $(n_z)_B$ is plotted together with its boundary values. The load factor constraint becomes active when performing the half Cuban eight maneuvers due to the high velocity of the aircraft in combination with the requirement to turn as quickly as possible. As soon as the velocity of the aircraft drops due to climbing, the load factor experienced by the pilot also drops and the limit becomes inactive again. As the velocity of the aircraft is decreasing along the race track (due to the drag induced by the agile maneuvers), the activation duration of the constraint decreases from each Cuban eight maneuver to the next because the critical velocity is undercut faster each time.

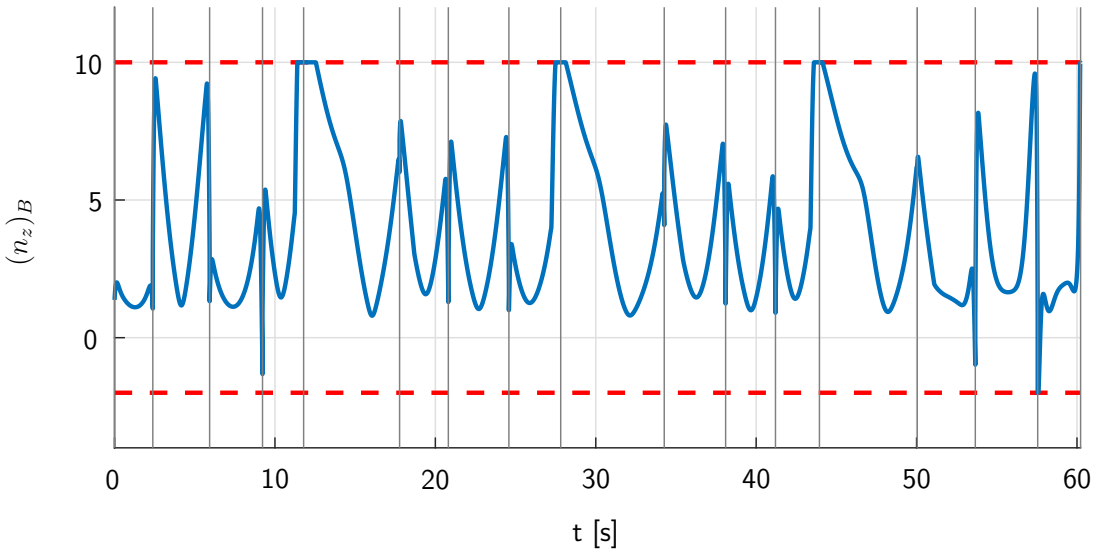

Figure 11.7: *Constrained load factor and its limit for the augmented point mass simulation model.*

Figure 11.8 shows the costates (or adjoint variables) that have been estimated by *FALCON.m* based on the LAGRANGE multipliers determined by *IPOPT* as described in section 3.2.5. Multiple theoretical findings can be recovered from the costate histories of the problem.

First of all, the costates for the position states $\lambda_{(x)_N}$, $\lambda_{(y)_N}$, and $\lambda_{(z)_N}$ are constant almost everywhere within each phase but jump between the phases. This makes sense as the position states do not influence the dynamics and consequently the derivative of the *Hamiltonian* H with respect to them is zero. As this derivative equals the negative time derivative of the respective costates (equation (3.101)), these have to be constant. The only exception is the costate of the $(z)_N$-position, that jumps whenever the vertical position constraint becomes active or inactive. The jumps appearing when a race gate is passed (visualized by the vertical, gray lines in the plots) can easily be explained by the constraints that apply there: All costates related to states that are forced to a value at the gates, feature a jump at this point due to the transversality conditions (equation (3.102)). The costate history of the course angle λ_{χ_K} can be interpreted using the same argumentation. For all other states a continuous costate history results – which is in perfect compliance with optimal control theory. Additionally, it can be seen that all costates related to states that are not constrained at the final point in time become zero at the end. Using optimal control theory and the interpretation of the costates as sensitivities of the cost (see section 3.2.3), they have to be zero, as a change in these final *optimal* states does not influence the cost function anymore.

As described in section 3.2.3, using the costate estimates, the state derivatives, and the constraint values in the problem, the *Hamiltonian* can be approximated. Figure 11.9 shows the time history of the *Hamiltonian* of the problem that contains four parts here:

$$\tilde{H}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}_\mathbf{S}, \boldsymbol{\mu}_\mathbf{C}, \mathbf{u}) = \underbrace{L(\mathbf{x}, \mathbf{u})}_{A} + \underbrace{\boldsymbol{\lambda}^\mathsf{T}\, \mathbf{f}(\mathbf{x}, \mathbf{u})}_{B} + \underbrace{\boldsymbol{\mu}_\mathbf{S}^\mathsf{T}\, \mathbf{S}(\mathbf{x}, \mathbf{u})}_{C} + \underbrace{\boldsymbol{\mu}_\mathbf{C}^\mathsf{T}\, \mathbf{C}(\mathbf{x})}_{D} \tag{11.14}$$

A The LAGRANGE cost function,

B the costates times the state derivatives,

C the multipliers for the path constraints times the constraint margin, and

D the multipliers for the state inequality constraints times their margin.

Compared to the *Hamiltonian* in equation (3.113), here – besides the mixed state control constraints – pure state constraints are present making the underlying theory a little more complicated. However, this issue is not discussed here and in this analysis the two constraints $\mathbf{S}$ and $\mathbf{C}$ may be seen as a separation of the ones presented in equation (3.113), with $\boldsymbol{\mu}$ being partitioned into $\boldsymbol{\mu}_\mathbf{S}$ and $\boldsymbol{\mu}_\mathbf{C}$.

As can be seen in figure 11.9, the portions coming from the constraints are all zero as either the constraint margins (the difference between the value and the related limit) or the respective multipliers are zero. The portion coming from the state dynamics sums up to a value of $\boldsymbol{\lambda}^\mathsf{T}\, \mathbf{f}(\mathbf{x}, \mathbf{u}) = -1$, constant over time. Using the LAGRANGE cost function from equation (11.11), the overall *Hamiltonian* may be estimated accordingly and is also shown in the plot. As the whole problem formulation is not time dependent (the problem is autonomous), the *Hamiltonian* needs to be constant. Additionally, for free final time problems, the *Hamiltonian* is required to be $H(t_f) = 0$ at the end. With both conditions combined, the *Hamiltonian*

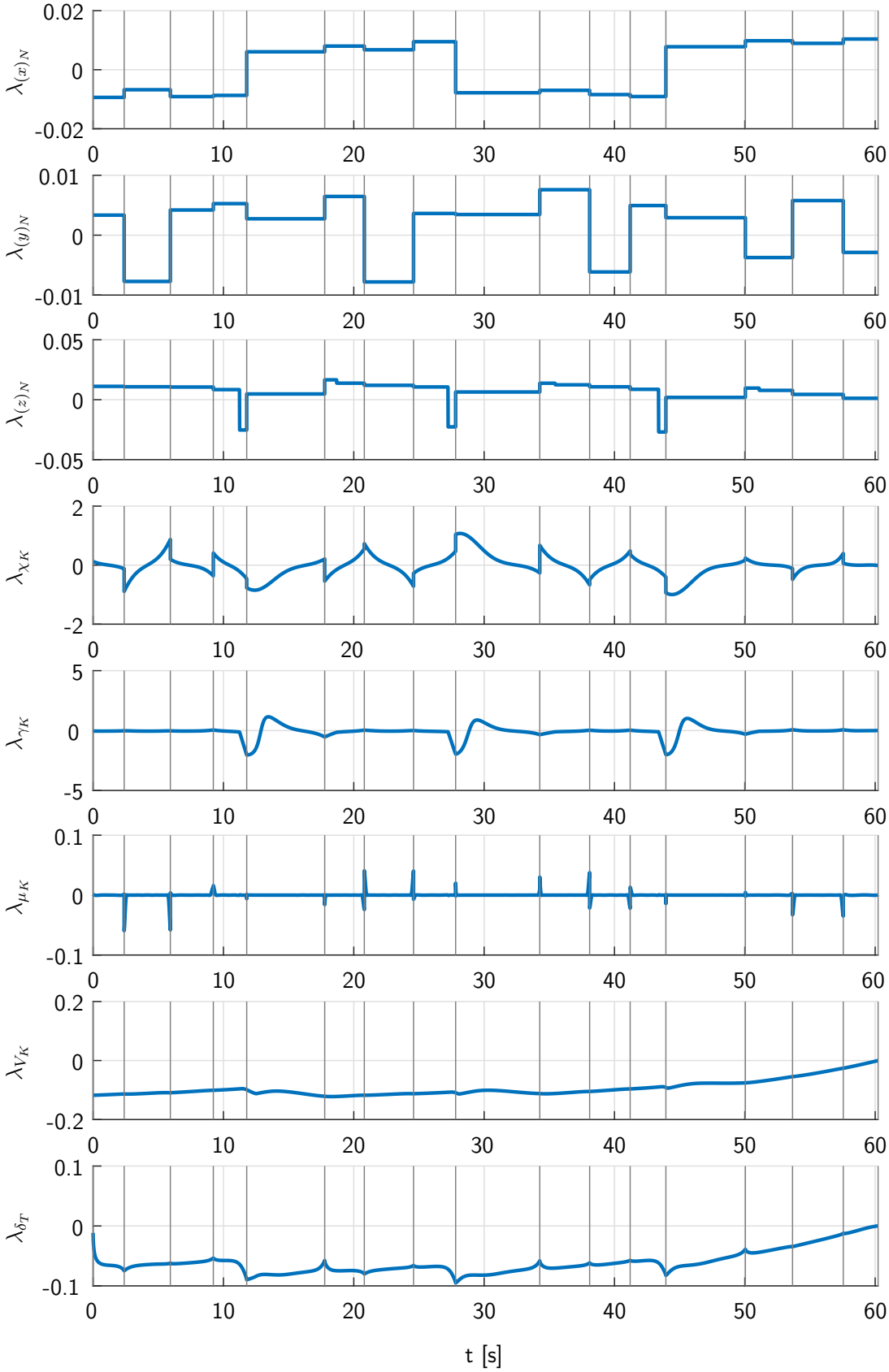

Figure 11.8: *Costate histories for the augmented point mass simulation model.*

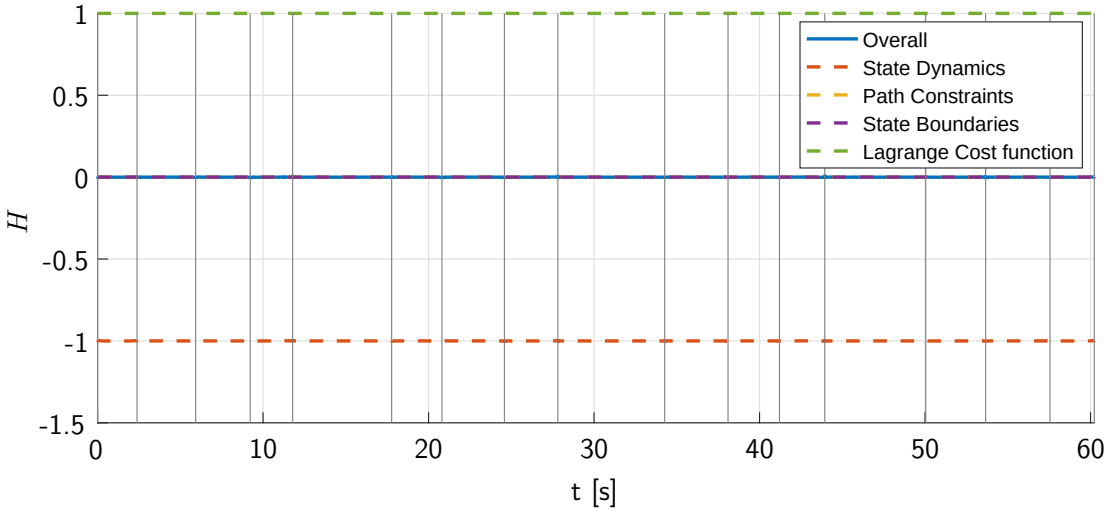

Figure 11.9: *Augmented* Hamiltonian *over time for the augmented point mass simulation model.*

is required to be constant zero along the whole trajectory. Again, nice accordance with the underlying optimal control theory from section 3.2.3 becomes visible, confirming the optimality of the numerical solution.

In the last step, the derivative information – that has been automatically calculated by *FALCON.m* – is used to estimate the derivative of the *Hamiltonian* with respect to the controls along the trajectories. The theory of optimal control states that this derivative tells where the respective control is saturated and where not. In optimal control problems with linearly appearing controls, this derivative is called switching function. For the nonlinear cases considered here, the derivative of the *Hamiltonian* with respect to the controls is called *extended switching function* as it delivers similar information. Whenever the *extended switching function* is negative, the respective optimal control has to be chosen to be at its upper bound, while it needs to be at its lower bound for positive values. This can also be seen from equation (3.98) for bounded controls. In case the *extended switching function* is zero, the related control cannot be determined directly but needs to be calculated by solving the underlying equations. Then, its value is not at the boundaries. Besides, as all discretized controls enter the numerical optimization problem as optimization variables, *IPOPT* also calculates multipliers for the control values. Now, the derivative of the *Hamiltonian* with respect to the controls can be compared to the multipliers from *IPOPT*. The plots in figure 11.10 show the respective histories, each calculated by the two different approaches, where good compliance can be seen (except for some small spikes at the phase boundaries coming from numerical effects). Additionally, when comparing the plots to the control histories in figure 11.6, it can clearly be seen that whenever a control hits one of the boundaries, the related derivative is non-zero, fulfilling the aforementioned sign condition.

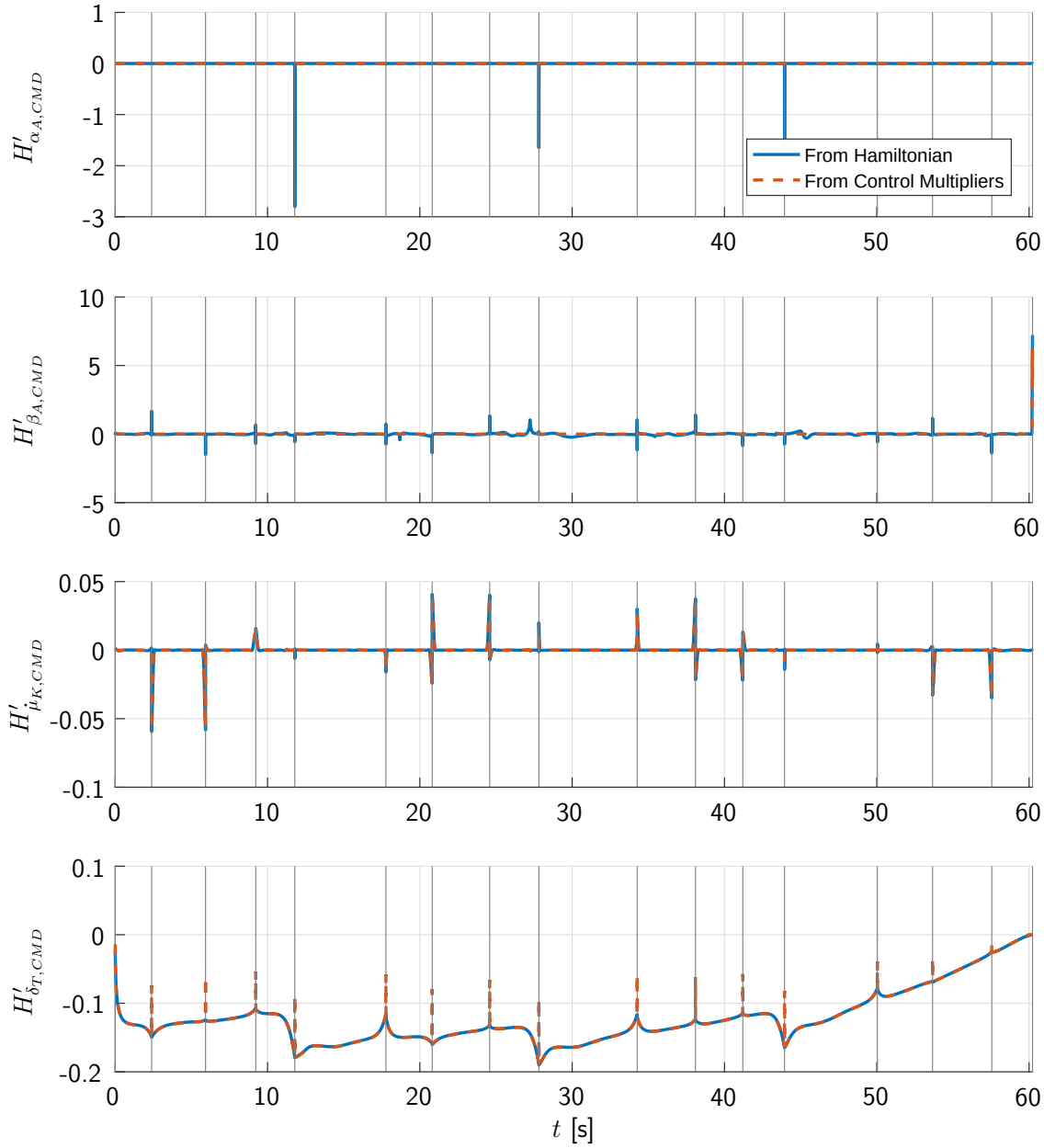

Figure 11.10: *Derivative of the* Hamiltonian *with respect to the controls for the augmented point mass simulation model.*

11.2.2 Point Mass Model with Dynamic Control Constraints

In this section, the results for the optimization using the reformulated model based on the idea presented in section 6.4 are shown. Now, the overall race trajectory is depicted in figure 11.11 showing good agreement with the previously presented results.

In the model used in this example, the linear dynamic equation for the bank angle has been removed and replaced by a discrete, finite difference approximation. Hence, along the whole trajectory the condition

$$-7.33\frac{rad}{s} = \dot{\mu}_{K,min} \leq \bar{\dot{\mu}}_{K,CMD} = \frac{\mu_K(t_k) - \mu_K(t_{k-1})}{t_k - t_{k-1}} \leq \dot{\mu}_{K,max} = 7.33\frac{rad}{s} \tag{11.15}$$

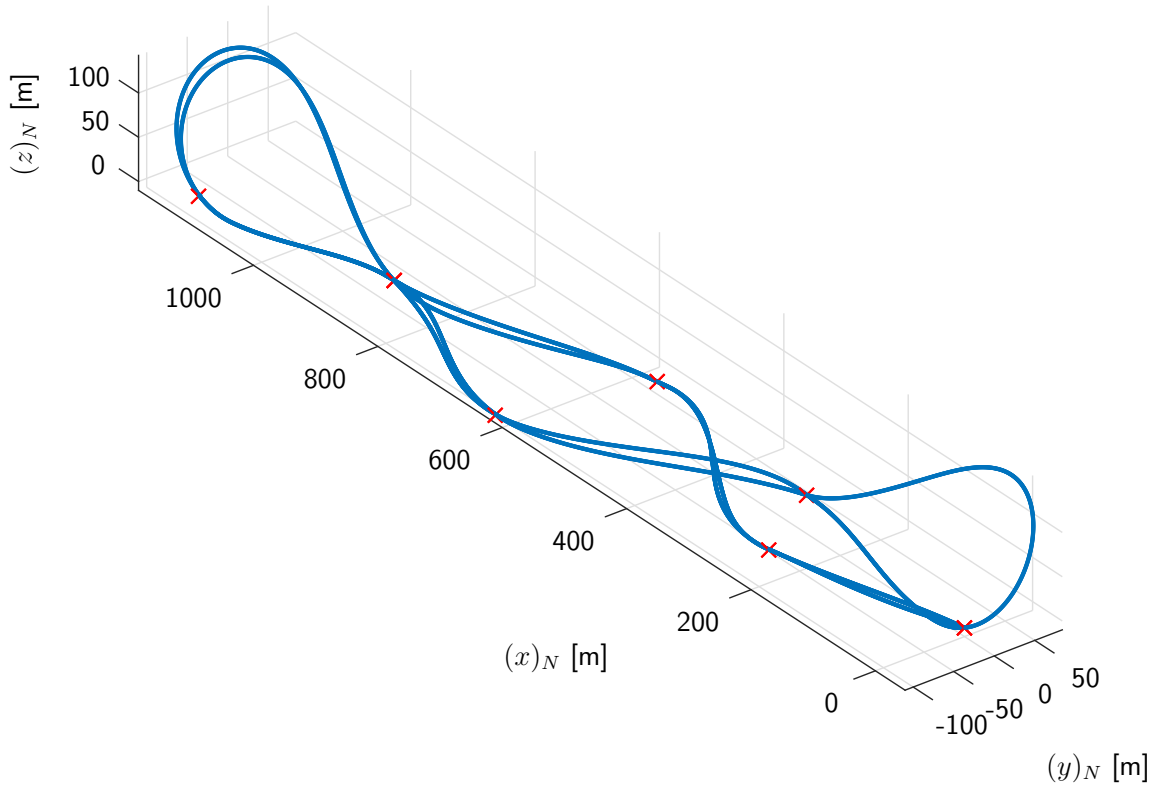

Figure 11.11: *Time optimal spatial trajectory for the model including dynamic control constraints.*

has to hold $\forall t_k \in [t_1, t_f]$. Similarly, the dynamic equation for the thrust lever has been removed because the thrust lever control also appears linearly in the problem. Anyway, $\delta_{T,CMD}$ is almost always at its boundaries, not making it the critical control in the problem. However, for the sake of completeness of the transformation, the dynamic equation (11.8) was replaced by the constraint:

$$-1 = \dot{\delta}_{T,min} \leq \bar{\dot{\delta}}_{T,CMD} = \frac{\delta_T(t_k) - \delta_T(t_{k-1})}{t_k - t_{k-1}} \leq \dot{\delta}_{T,max} = 1, \qquad \forall t_k \in [t_1, t_f] \tag{11.16}$$

The selection of the limits for $\bar{\dot{\delta}}_{T,CMD}$ is based on the time delay $T_{Engine} = 0.5$ and the respective range $\delta_T \in [0, 1]$ of the removed linear dynamic and represents an approximation thereof.

After the discretization, the numerical optimization problem contained 80176 optimization variables and 72184 constraints. Hence, the gradient matrix of the discretized problem might contain up to $5.7875 \cdot 10^9$ elements but has a sparsity ratio of 99.9887%, resulting in 652368 structural non-zero entries. The numerical optimization took 1927 iterations to converge which equals approximately $1730s$ on the aforementioned machine. The resulting optimal cost function value is a race time of $t_f = 60.2304s$

In figure 11.12 the state histories for the optimal trajectory can be seen which in general are very similar to the results presented in section 11.2.1. Again, the phase boundaries are illustrated by the gray, vertical lines. In this formulation only the altitude constraint in $(z)_N$ becomes active – all other states do not reach their boundaries.

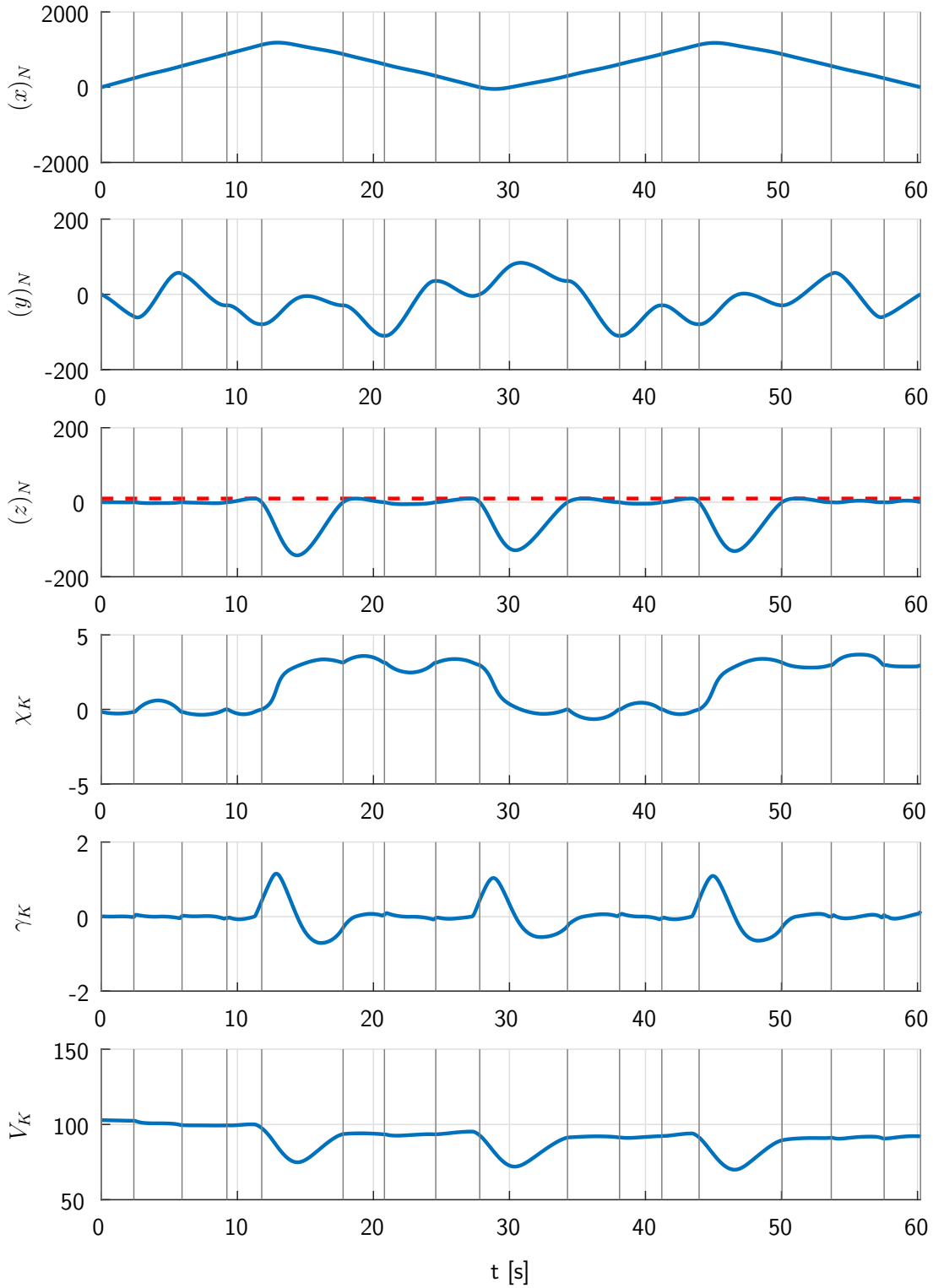

Figure 11.12: *Optimal state histories for the model including dynamic control constraints.*

The optimal control histories for the problem are depicted in figure 11.13. When comparing the results to the ones for the augmented point mass simulation model in figure 11.6, no differences in the angle of attack $\alpha_{A,CMD}$ or the angle of sideslip $\beta_{A,CMD}$ can be seen. Moreover, the newly added control $\mu_{K,CMD}$ shows a great similarity to the respective state in

figure 11.5. The thrust lever $\delta_{T,CMD}$ again remains at full throttle position, not even showing the decrease at the final time. The reason for this is that with the thrust dynamics being removed, the final thrust lever control influences the trajectory and consequently the solution of the problem.

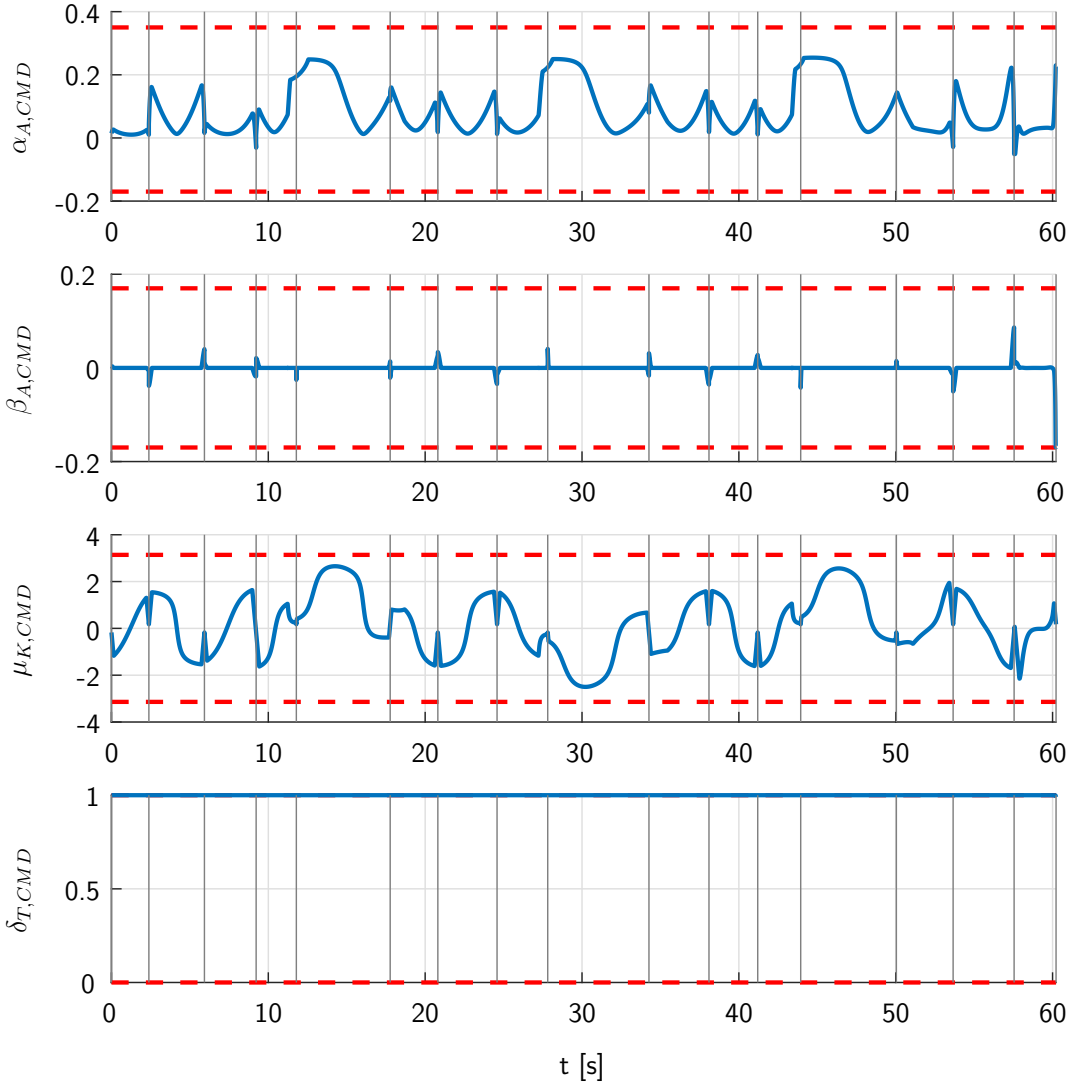

Figure 11.13: *Optimal control histories for the model including dynamic control constraints.*

The history of the load factor constraint is plotted in figure 11.14 and shows very good agreement with the one in figure 11.7. Thus, the explanation given there is also valid here.

Figure 11.15 depicts the costate histories, again showing the same results as already visible in figure 11.8 and explained there.

Similarly to the previous section, it is possible to estimate the *Hamiltonian* of the problem. Figure 11.16 shows the parts of the *Hamiltonian* as described in equation (11.14). In the theory used therein, the dynamic control constraints are not covered, resulting in errors in the estimation whenever the dynamic control constraints are active. Figure 11.17 shows the two dynamic control limits as described in equations (11.15) and (11.16). One can see that the bank rate is hitting its limits at almost the same points as in the example presented before (see

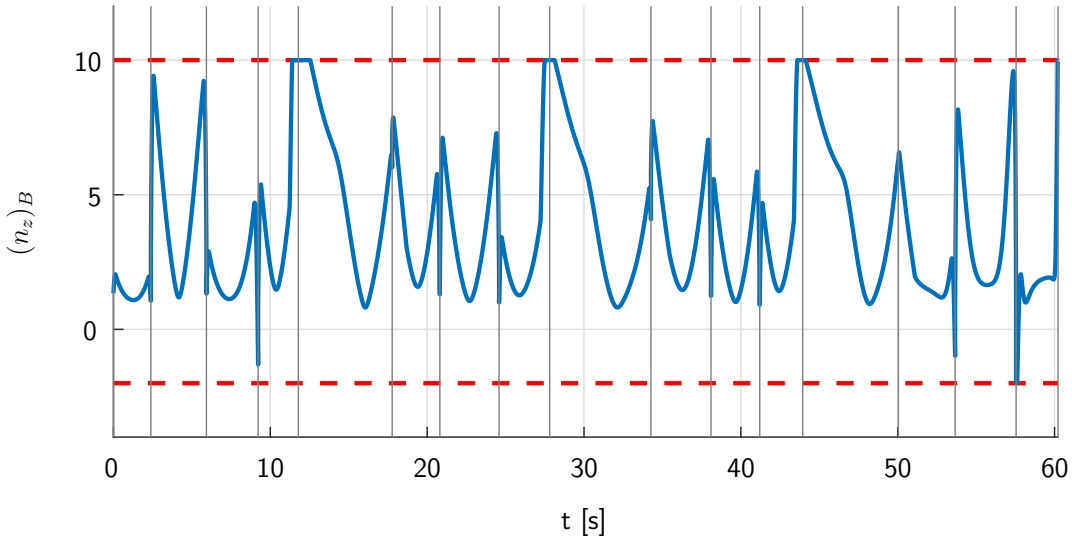

Figure 11.14: *Constrained load factor and its limit for the model including dynamic control constraints.*

figure 11.6). Moreover, it becomes visible how the approximation of the *Hamiltonian* is not constantly equal to zero whenever the dynamic constraints are active. This becomes visible when comparing figures 11.16 and 11.17. This perfectly fits the theory as the approximation of the *Hamiltonian* does not cover the constraint and consequently is not precisely correct at these points.

Figure 11.18 shows the derivative of the *Hamiltonian* with respect to the controls, again estimated twice: On the one hand from the approximation of the *Hamiltonian*, and on the other hand from the control multipliers calculated by *IPOPT*. In this case the two approaches deliver slightly different results – again as expected, because of the errors in the estimate of the *Hamiltonian*. Whenever the dynamic control constraint is not active and the approximation of the *Hamiltonian* is consequently good, the two results for the function match. Besides, it can again be seen that all controls – except the thrust lever position – are not at their limits almost everywhere.

11.3 High Fidelity Trajectory Optimization

In this section, the results for the optimization of air race trajectories using the full rigid body simulation model are presented. More details on the model can be found in chapter 2 and in section 11.1. Similar to before, the problem was solved on a grid containing 501 discretization points per phase, resulting in a total of 8016 nodes. Again, due to the different lengths of the phases, the distance between two nodes varies from $0.0048s$ to $0.0129s$. *IPOPT* [WB06] was used to solve the problem to a feasibility tolerance of $tol_C = 10^{-6}$ and an optimality tolerance of $tol_J = 5 \cdot 10^{-6}$ on the previously mentioned machine.

After discretization, 136288 optimization variables and 112227 constraints need to be considered. The sparsity ratio of the gradient matrix is 99.9864%, resulting in 2074082 structural

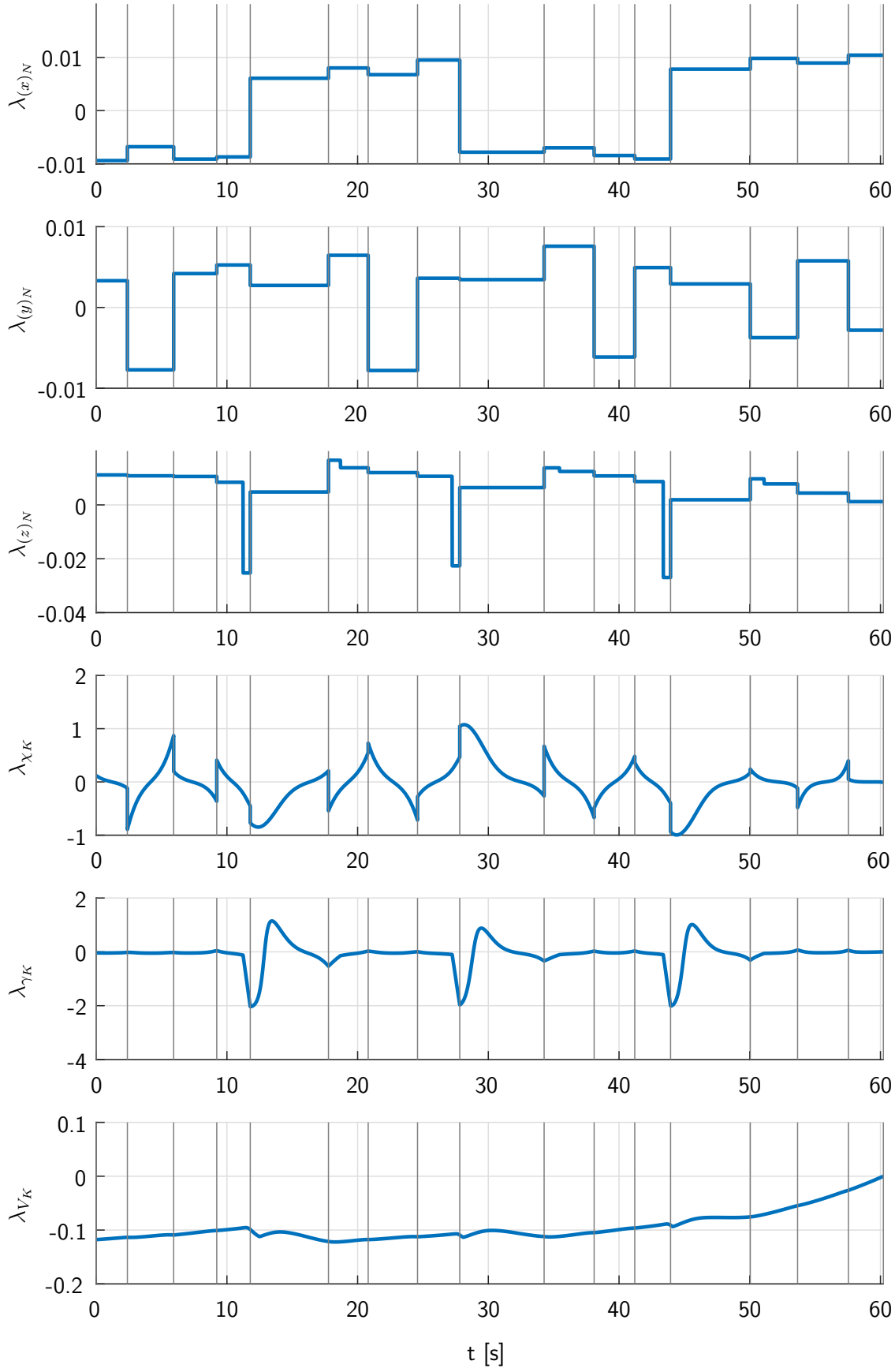

Figure 11.15: *Costate histories for the model including dynamic control constraints.*

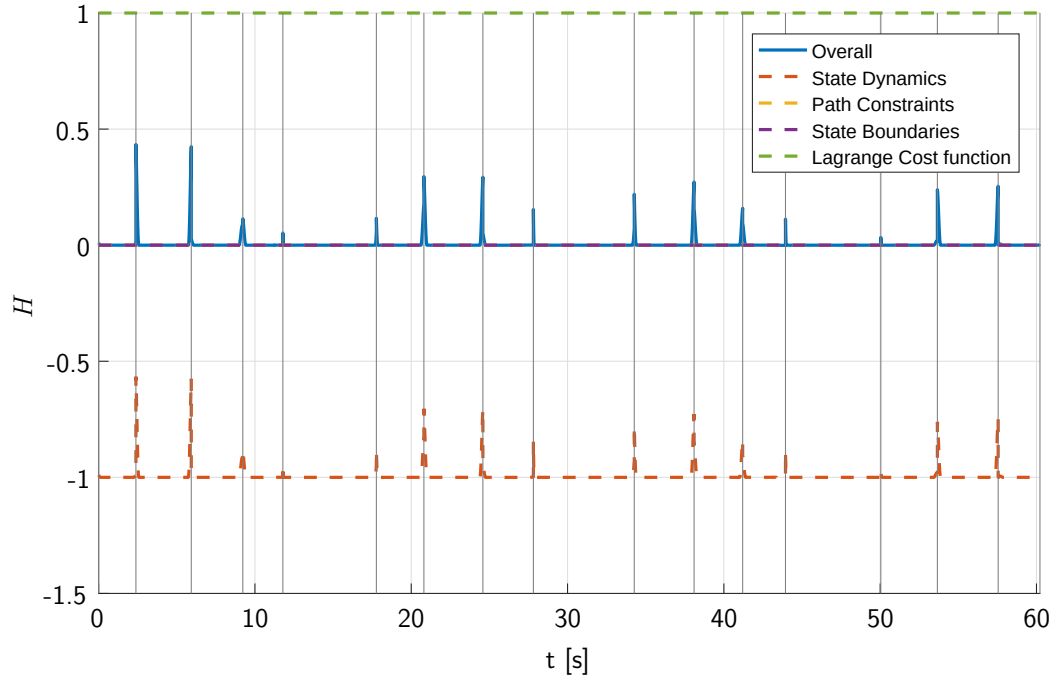

Figure 11.16: *Augmented* Hamiltonian *over time for the model including dynamic control constraints.*

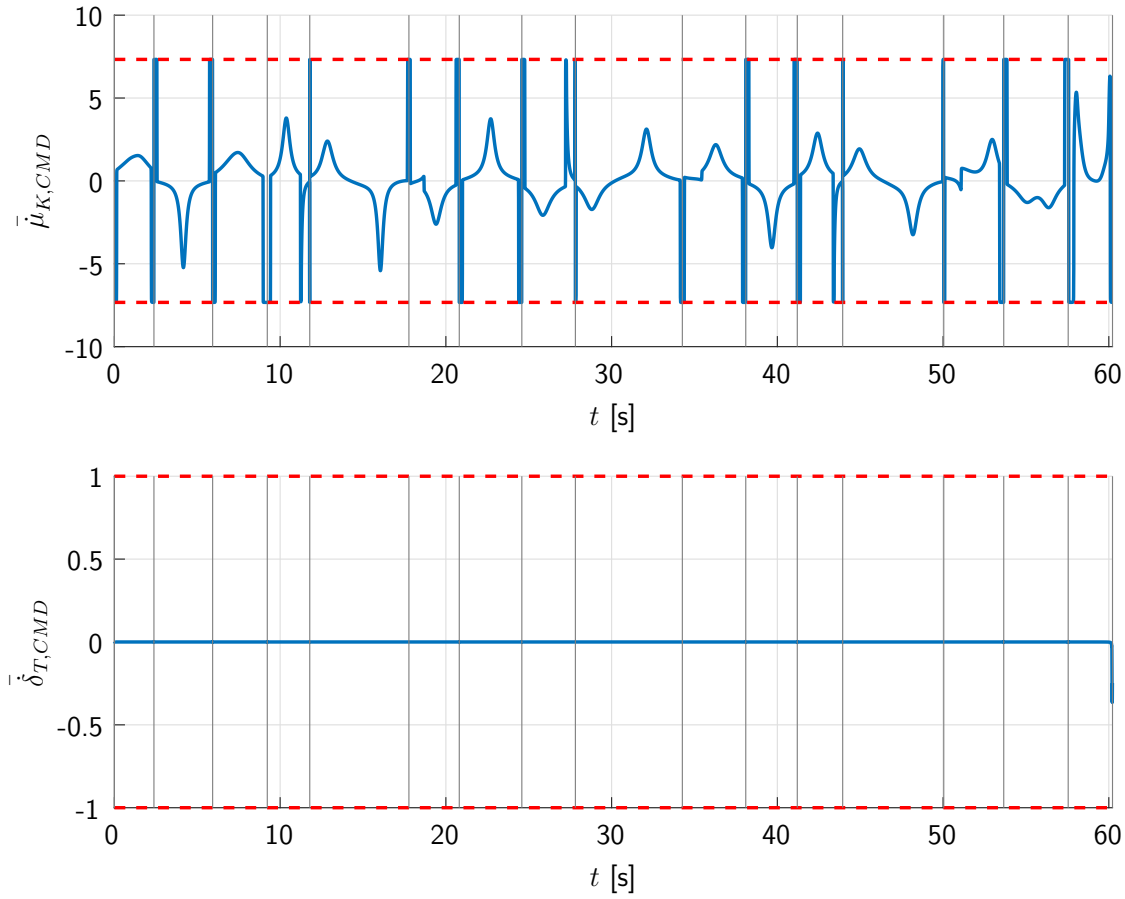

Figure 11.17: *Dynamic Control Constraint for limiting the control rates.*

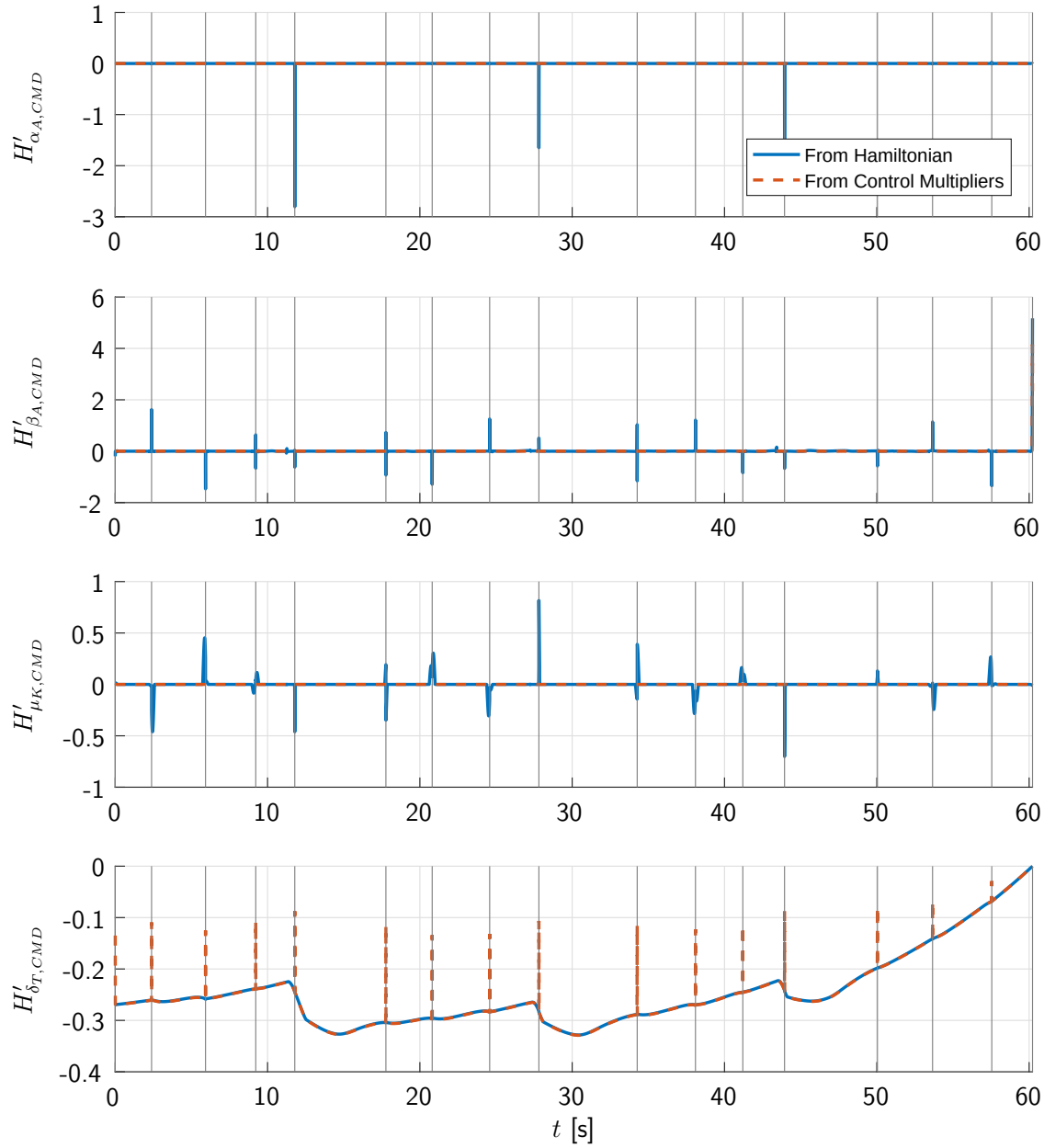

Figure 11.18: *Derivative of the* Hamiltonian *with respect to the controls for the model including the control constraints.*

non-zero entries in the overall gradient matrix. The numerical optimization converged to a solution after 883 iterations which equals approximately $2290s$.

The spatial trajectory that results from the optimization can be found in figure 11.1, above. The minimal race time calculated in the optimization is $60.3437s$. Figures 11.19 and 11.20 show the state histories along the air race, with the phase boundaries being represented by the gray, vertical lines. Now, besides the limits for $(z)_N$, also the limits for p_K become active during the race. The remainder of the analysis is similar to that of the point mass simulation model, except that now, when passing the gates, the aircraft is rotating around the bank axis using the full envelope in the rotational dynamics, pushing the roll rate p_K to the limits.

The histories of the optimal controls can be found in figure 11.21. It can be seen that fast

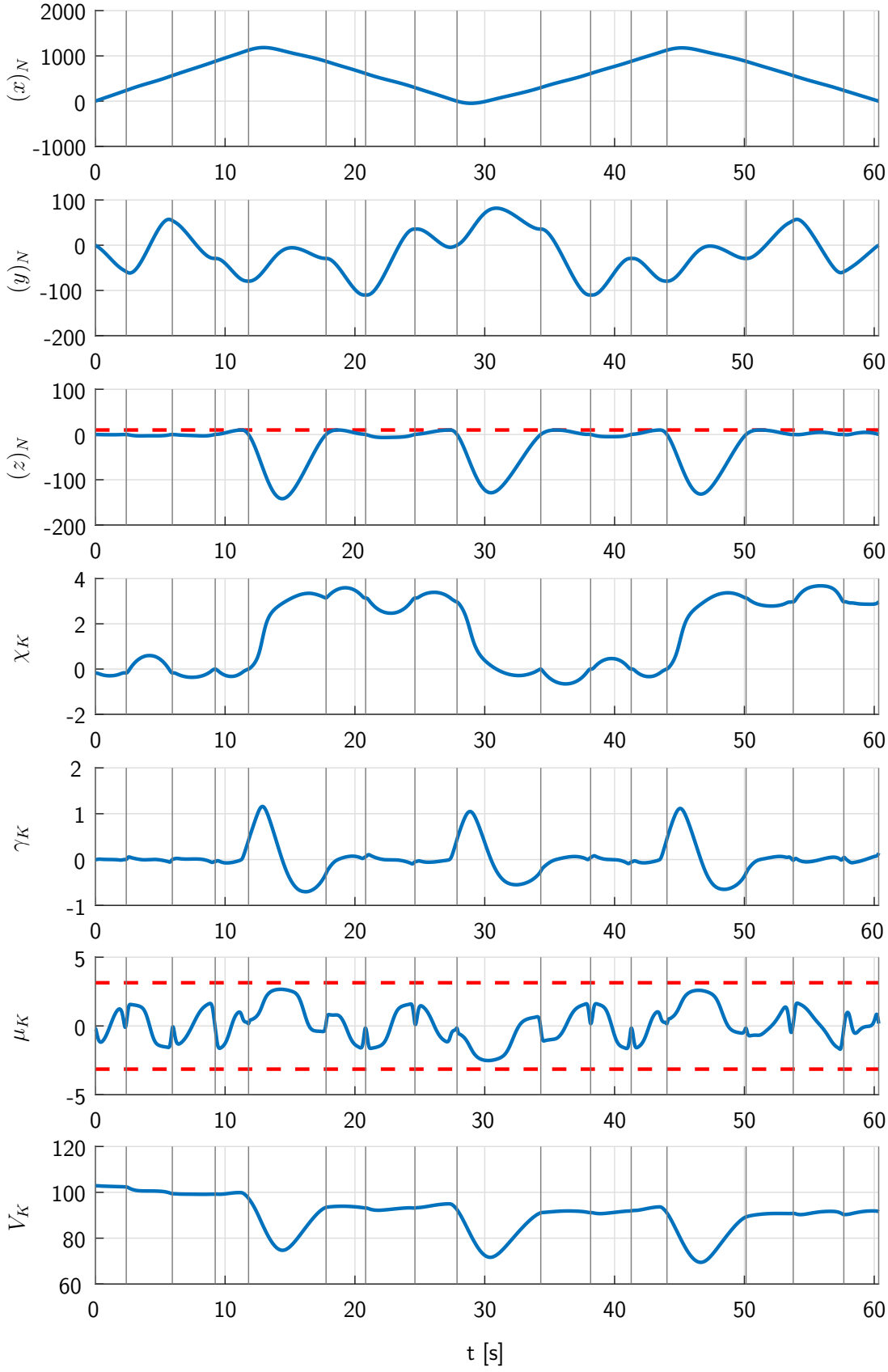

Figure 11.19: *Optimal state histories for the high fidelity simulation model.*

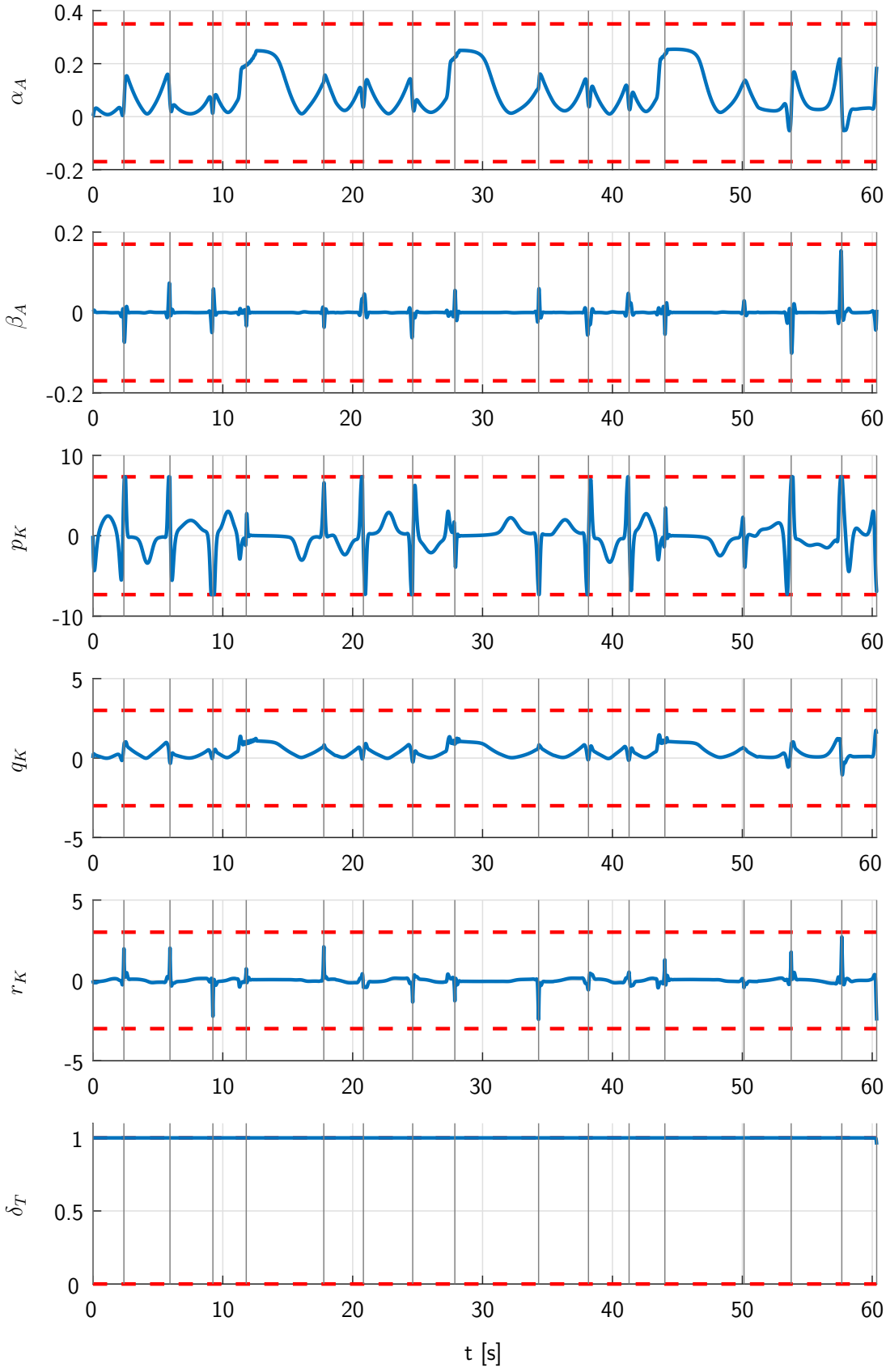

Figure 11.20: *Optimal state histories for the high fidelity simulation model.*

and full movement of the aileron ξ_{CMD} is required in the proximity of the race gates – again for fulfilling the bank angle constraints there. The elevator η_{CMD} is only hitting its boundaries at the second last race gate. Moreover, the rudder ζ_{CMD} is only used when passing the race gates. Again, this behavior is attributed to the bank angle constraint which renders a small angle of sideslip valuable that is built up and relieved using the rudder. As can be seen in the last graph in the figure, the thrust lever $\delta_{T,CMD}$ remains in full throttle position all the time except at the very last instant. The argumentation from section 11.2.1 applies again.

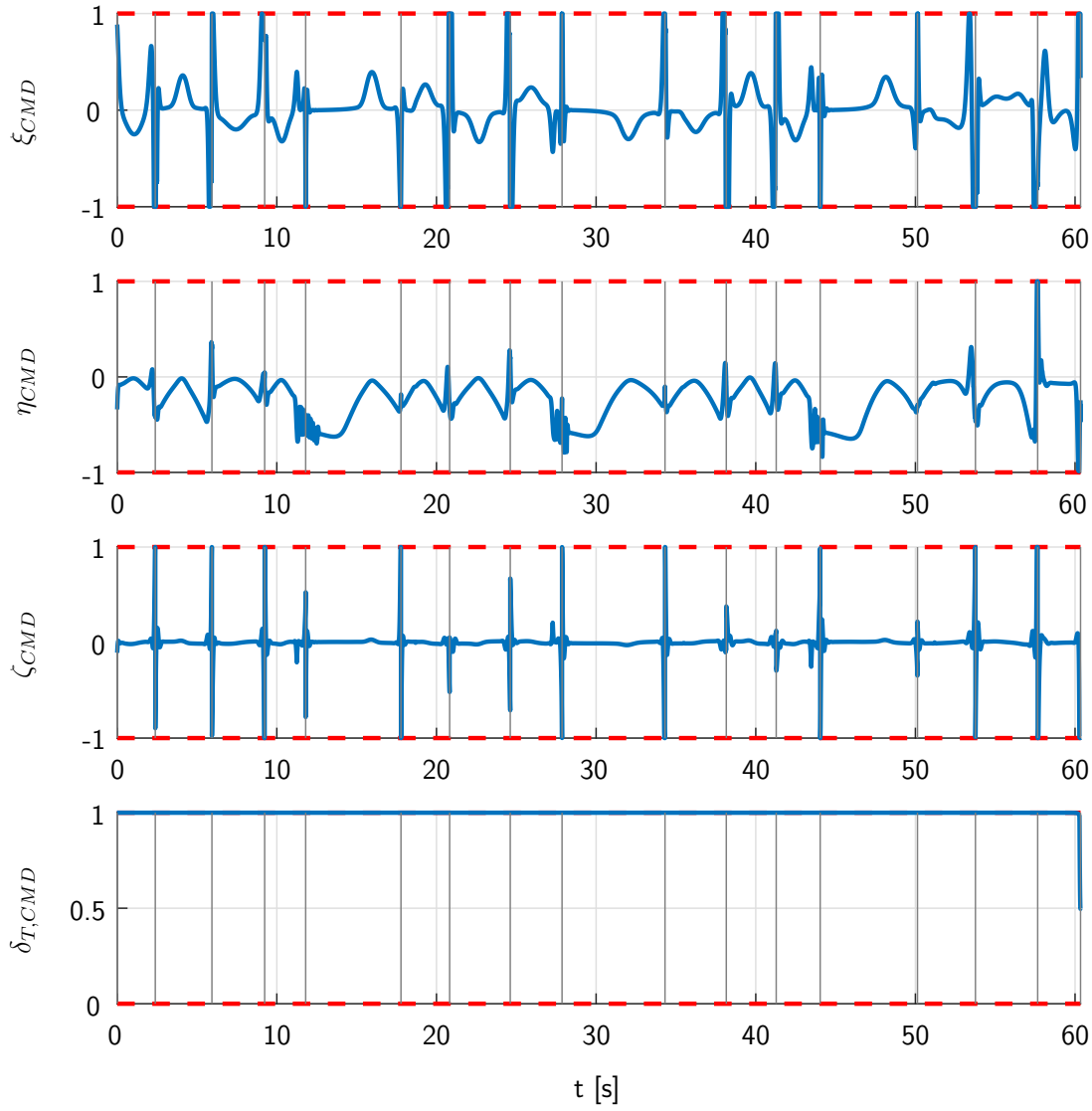

Figure 11.21: *Optimal control histories for the high fidelity simulation model.*

It is noticeable that there is slight chattering in the elevator control when passing the race gates right before the half Cuban eights. Neither a purely mathematical nor a numerical reason was found during the analysis of the results which leads to the assumption that the aircraft may gain something in the cost function by the slightly increased drag that results from the elevator movement (see (11.3) for the drag equation). In [Ben10, p. 110] Ben-Asher shows how an analysis of the *Hodograph* plot of the state derivatives may be used to determine if chattering may appear or not. In order to get the *Hodograph*, one has to plot the set of

reachable state derivatives for the whole set of possible controls at the state values under consideration. In case this set is *not strictly convex*, chattering may appear. In the example presented here the state space – and consequently also the state derivative space – features 13 dimensions, not allowing for a simple visual convexity check. Anyway, different projections of the *Hodograph* to three dimensional space can be drawn. Then, the overall set can only be convex if every projection to the subspaces is convex. On the other hand, if one such projection is not strictly convex, the original set cannot be so, neither. Figure 11.22 shows two such projections, clearly indicating that the set is not strictly convex at the state values where chattering appears. As not *all* possible combinations of control inputs may be checked, the controls have been discretized in order to create the figures. The left plot in figure 11.22 shows the linear (and consequently not strictly convex) influence of the controls on the body rates, while the graph on the right show a clearly non-convex shape in direction of the time derivative of the velocity. Anyway, this is not proving that the slight chattering is part of the real solution to the problem, but another strong indication. Of course, this slight chattering also becomes visible in other places throughout the solution.

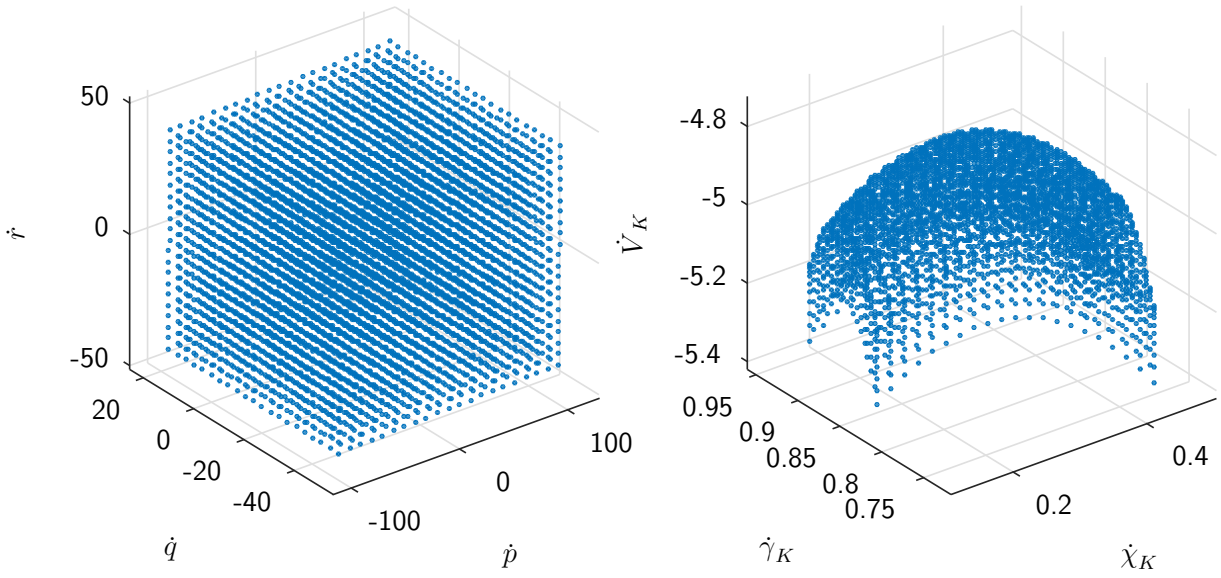

Figure 11.22: *Projection of the Hodograph to different three dimensional subspaces during chattering.*

In figure 11.23, the vertical load factor $(n_z)_B$ is plotted. It can be seen that the history of the values is similar to the results calculated before, rendering the same explanation valid.

The figures 11.24 and 11.25 show the costate estimates from *FALCON.m*. The behavior of the costates for the position states $\lambda_{(x)_N}$, $\lambda_{(y)_N}$, and $\lambda_{(z)_N}$ is found to be similar to above, with the same explanation being applicable. The jumps in all costates related to states that are constrained at the phase boundaries can also be found again here (compare section 11.2.1). The remainder of the analysis is also similar to above, except that here, the slight chattering that appears in the elevator also becomes visible in some of the costates.

Figure 11.26 shows the time history of the augmented *Hamiltonian* of the problem con-

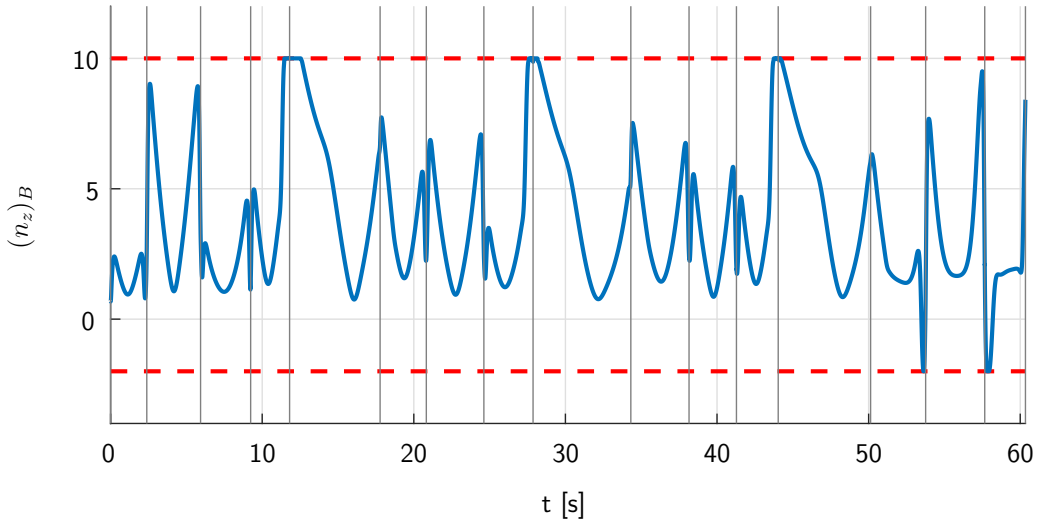

Figure 11.23: *Constrained load factor and its limit for the high fidelity simulation model.*

structed from the four parts presented in equation (11.14). As in this example no differential constraints are present, all effects may be covered by the theory, resulting in a good approximation of the *Hamiltonian*. Again, a nice accordance to the underlying optimal control theory becomes visible, supporting the optimality of the numerical solution.

The derivative of the *Hamiltonian* with respect to the controls for the optimization based on the rigid body simulation model can be found in figure 11.27. On the one hand, the plots show that in this case the derivation based on the approximated *Hamiltonian* matches the multipliers calculated by *IPOPT* nicely. Besides, the location of the arcs where the controls are not saturated, that may be determined from the histories, matches those of the control. Finally, again, good compliance between the numerical results and the underlying optimal control theory can be seen.

11.4 Combined Collocation and Shooting

Next, the results to the air race problem described above that have been obtained using the method suggested in section 8.2 will be discussed. The approach uses a collocation scheme for the slow states of the aircraft motion while the fast states are integrated using a multiple shooting scheme. According to the statements in chapter 2, the rotational states are to be considered fast, while the translational ones represent the slow dynamics of the aircraft. Consequently, the fast states in the model from above are

$$\mathbf{x}_f = [\mu_K, \alpha_K, \beta_K, p, q, r]^\mathsf{T} \tag{11.17}$$

while the remaining states are considered to be slow:

$$\mathbf{x}_s = [(x)_N, (y)_N, (z)_N, \chi_K, \gamma_K, V_K, \delta_T]^\mathsf{T} \tag{11.18}$$

The dynamic equations of the system are partitioned similarly to the state vector, resulting

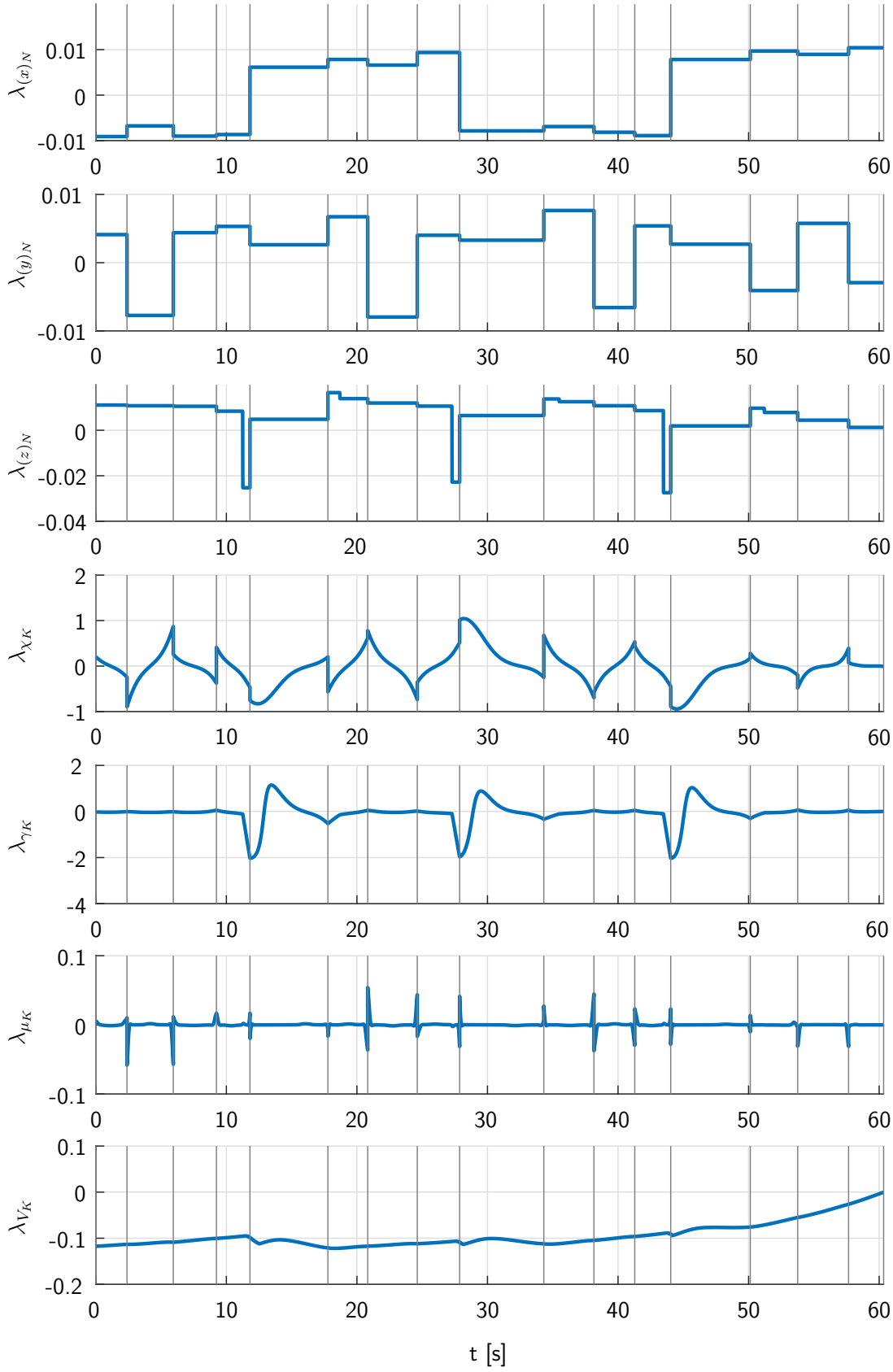

Figure 11.24: *Costate histories for the high fidelity simulation model.*

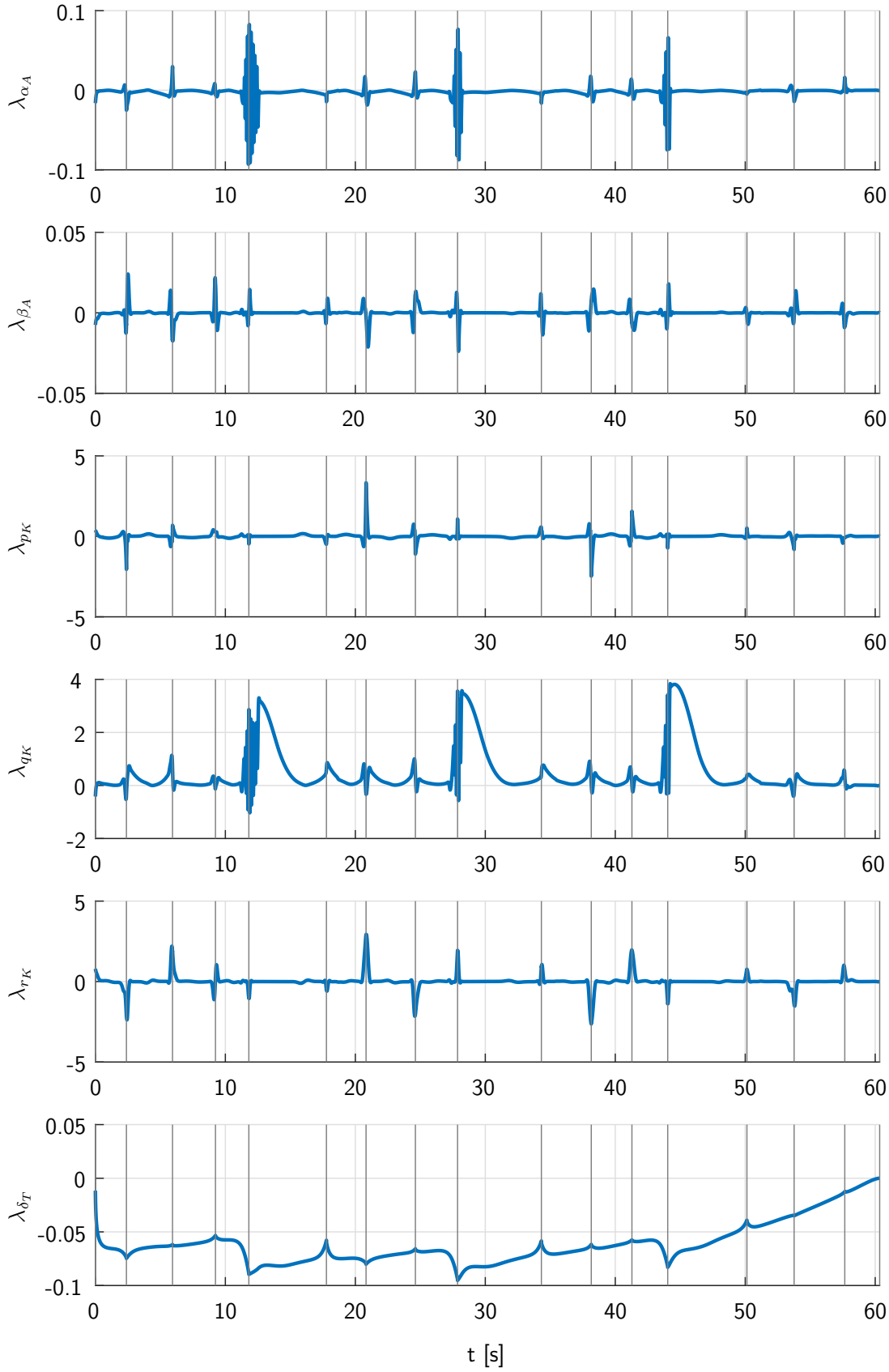

Figure 11.25: *Costate histories for the high fidelity simulation model.*

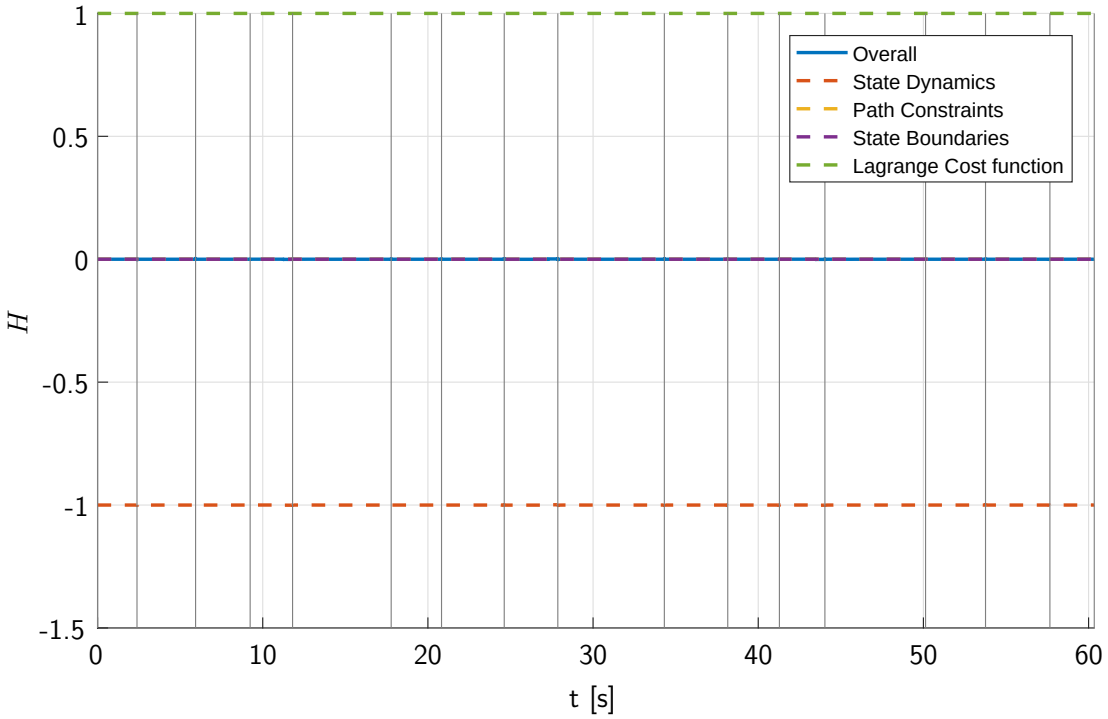

Figure 11.26: *Augmented* Hamiltonian *over time for the high fidelity simulation model.*

in an equation of motion for the fast states

$$\dot{\mathbf{x}}_f = \mathbf{f}_f(\mathbf{x}_s, \mathbf{x}_f, \mathbf{u}, \mathbf{p}) \tag{11.19}$$

and one for the slow states:

$$\dot{\mathbf{x}}_s = \mathbf{f}_s(\mathbf{x}_s, \mathbf{x}_f, \mathbf{u}, \mathbf{p}) \tag{11.20}$$

In this example, the slow states were discretized on an equidistant grid containing 101 points per phase. For the fast states a subdivision of five grid points was used, resulting in a total of 501 grid points per phase. Overall, for the slow states 1616 collocation nodes and for the fast states 8016 multiple shooting nodes result. For the approximation of the slow states onto the grid of the fast states, the linear interpolation from equation (8.9) has been implemented. The controls are also discretized on the coarse collocation grid formed by 101 points per phase and have been linearly interpolated to the fine multiple shooting integration grid. The simulation within the multiple shooting segments was done using an explicit EULER forward integration scheme.

Due to the different lengths of the phases, the distance between two collocation points for the slow states varies from $0.024s$ to $0.065s$ while for the fast states the grid spacing is between $0.0048s$ and $0.0129s$. Again, *IPOPT* [WB06] was used to solve the problem to a feasibility tolerance of $tol_C = 10^{-6}$ and an optimality tolerance of $tol_J = 5 \cdot 10^{-6}$.

The number of optimization variables decreased to 27488 while the number of constraints dropped to 22627 in the discretized problem, as less grid points are required in the combined discretization scheme. Even though the number of 764082 structural non-zero elements in the gradient is also comparably low, the sparsity ratio of 99.8772% is not as high as it is with pure

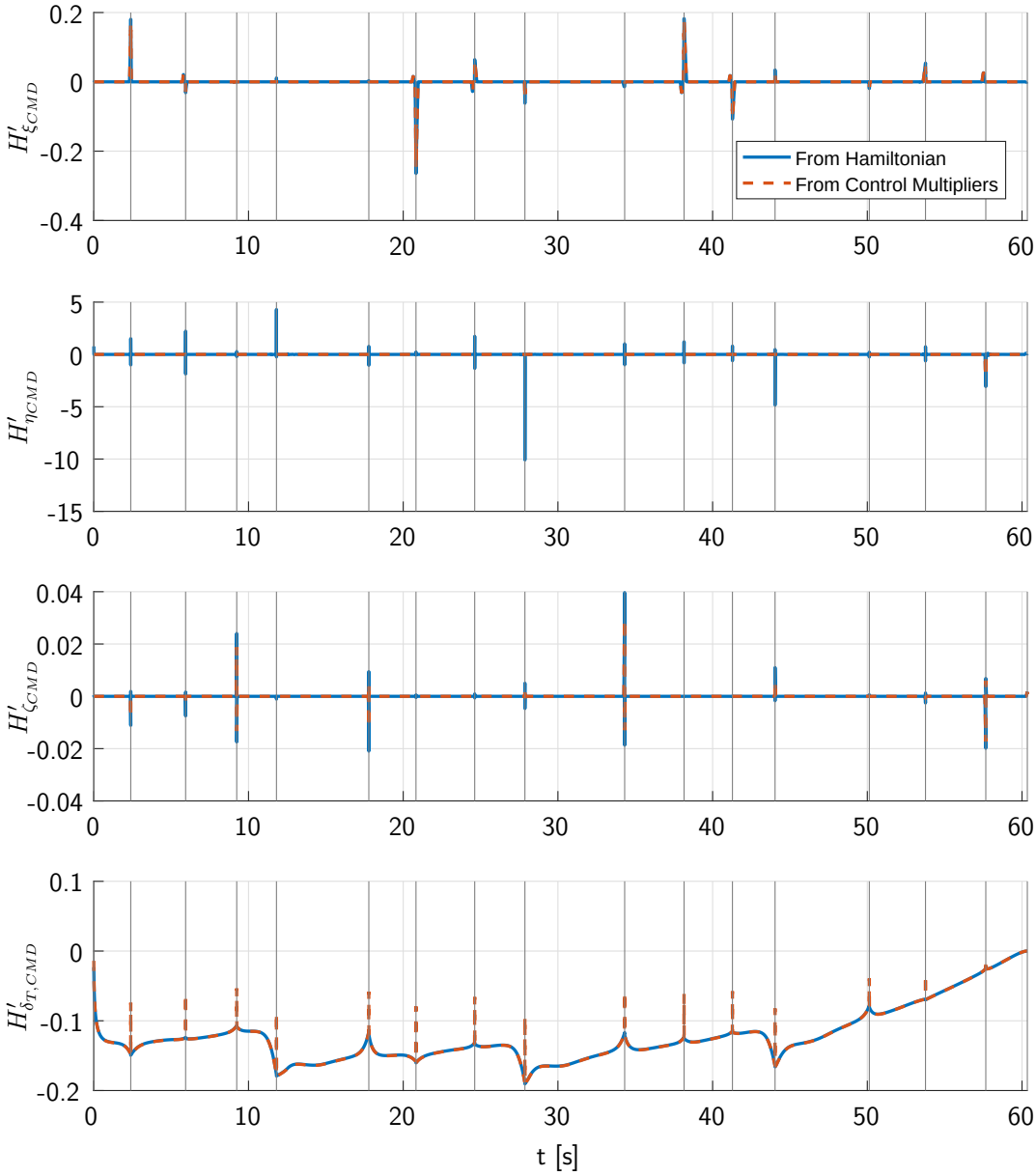

Figure 11.27: *Derivative of the* Hamiltonian *with respect to the controls for the high fidelity simulation model.*

collocation – as expected due to the higher interdependence in the shooting scheme and the smaller overall problem size.

On the aforementioned machine, the problem converged in approximately $2144s$ and 1355 iterations. Contrary to the pure shooting code, the code used in this mixed approach has not yet been tuned for maximum efficiency, leaving room for further improvements by means of execution time. The trajectory that results from the optimization can be found in figure 11.28. The minimal race time calculated in the optimization is $60.3512s$.

The state histories along the trajectories are depicted in figures 11.29 and 11.30. It can be seen that similar to before, the limits for $(z)_N$ and for p_K become active during the race.

When comparing figures 11.21 and 11.31 no large differences can be observed, keeping the analysis from section 11.3 valid here.

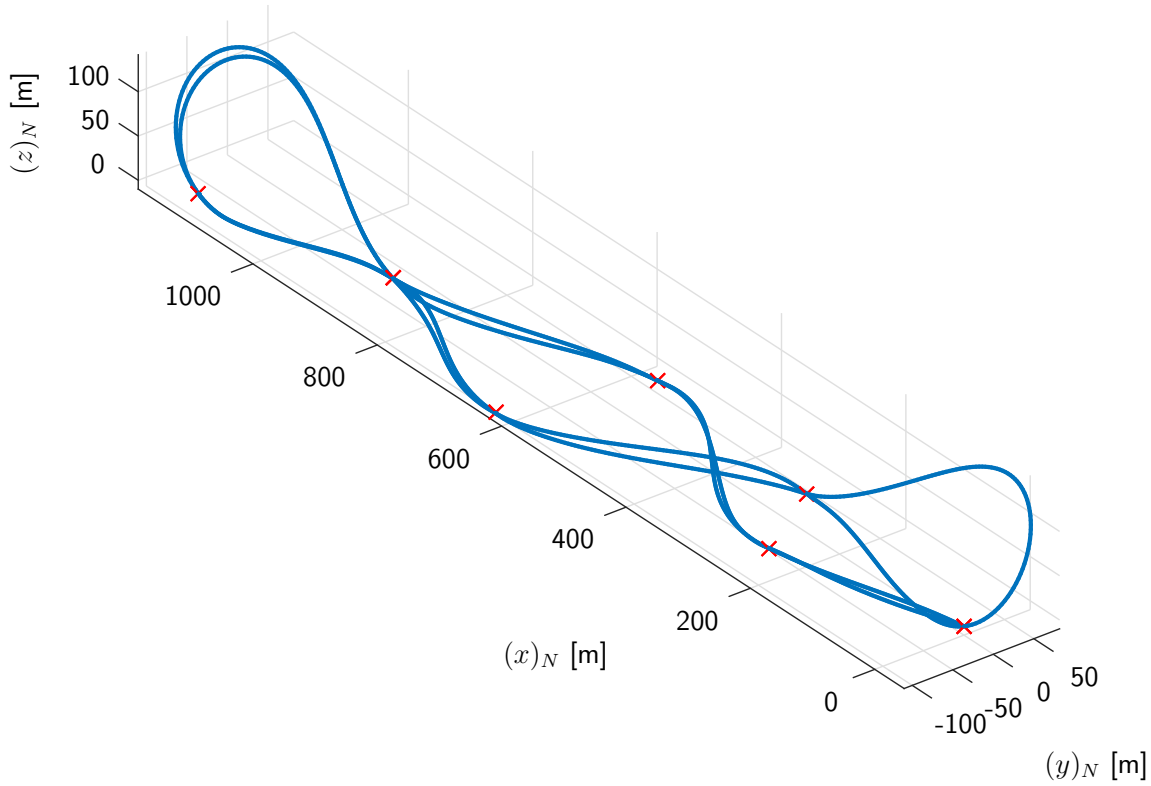

Figure 11.28: *Time optimal spatial trajectory for the combined simulation model.*

The same is true for the path constraint on the load factor in $(z)_B$ direction that is depicted in figure 11.32. Therein, no violation of the constraints in between two grid points can be observed. Contrary, this is not true for the roll rate limit as can be seen in figure 11.33. In the interval between $24.5s$ and $24.6s$ the roll rate limit is violated by the states resulting from the multiple shooting integration. As only the states at the collocation nodes (marked by the crosses in the figure) are considered in the path constraint, the violation cannot be detected.

Figure 11.34 shows a magnified part of the history of the vertical position of the aircraft. It can clearly be seen that the position dynamics can smoothly be represented by the collocation nodes marked by the crosses. Opposite to this, the detail of the bank angle history in figure 11.35 shows that the collocation grid points are not sufficient for a smooth representation of the history of this fast varying state. Hence, the blue line shows the state history calculated by the multiple shooting approach.

As the theory for the costate estimation from section 3.2.5 is only valid for collocation approaches, no costate histories can be given for the fast states here. The derivation of this theory for multiple shooting schemes remains an open issue for future research. Hence, the approximation of the *Hamiltonian* is not possible.

The aforementioned control discretization based on the coarse grid of the method is not ideal for all the controls as the control surface deflections ξ_{CMD}, η_{CMD} and ζ_{CMD} mainly influence the fast rotational dynamics and consequently also would require faster control inputs. Therefore, a future extension of the algorithm may be the use of a control grid which is similar to the multiple shooting integration grid, containing 501 nodes per phase. Thus, the problem

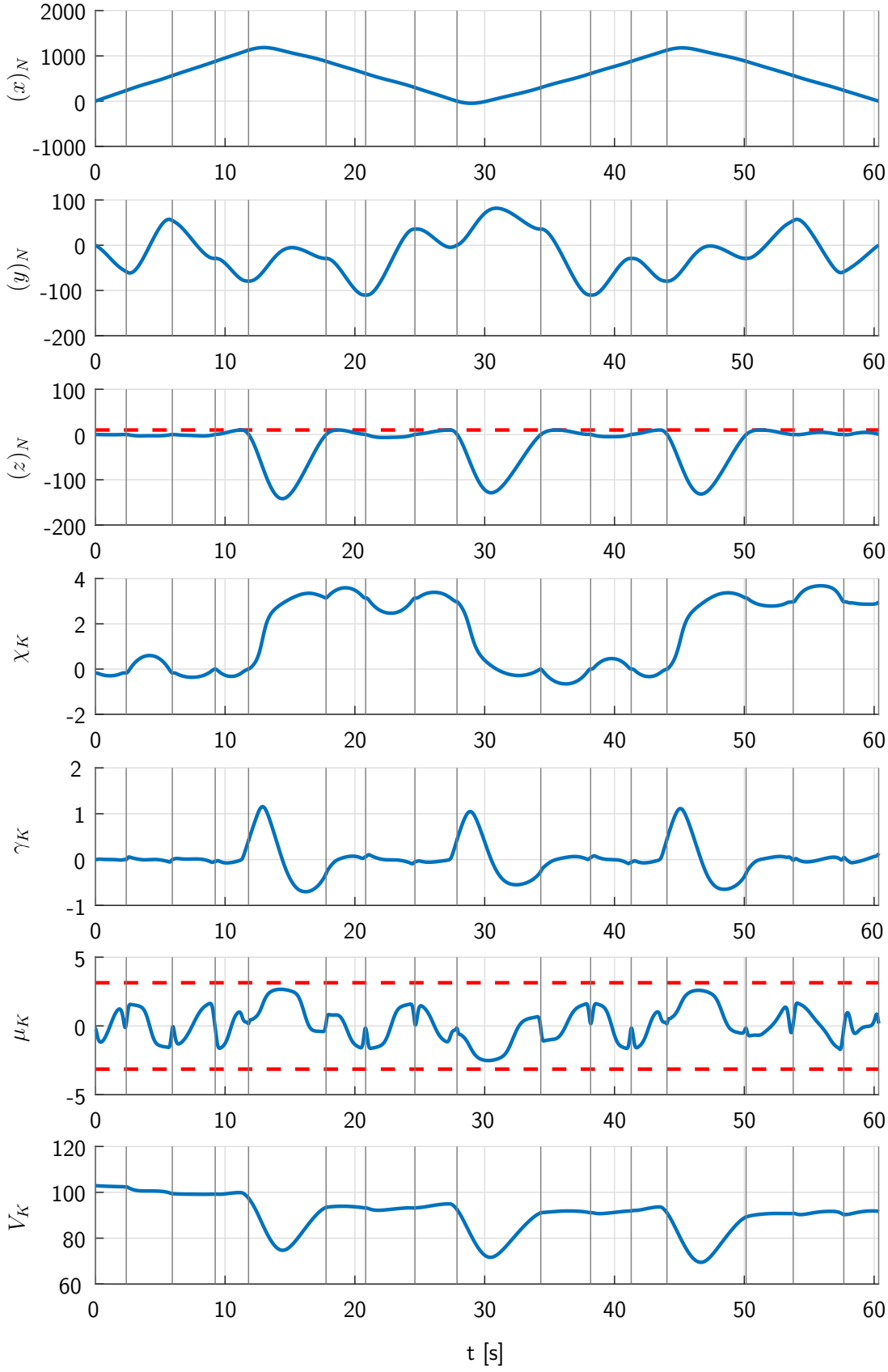

Figure 11.29: *Optimal state histories for the combined simulation model.*

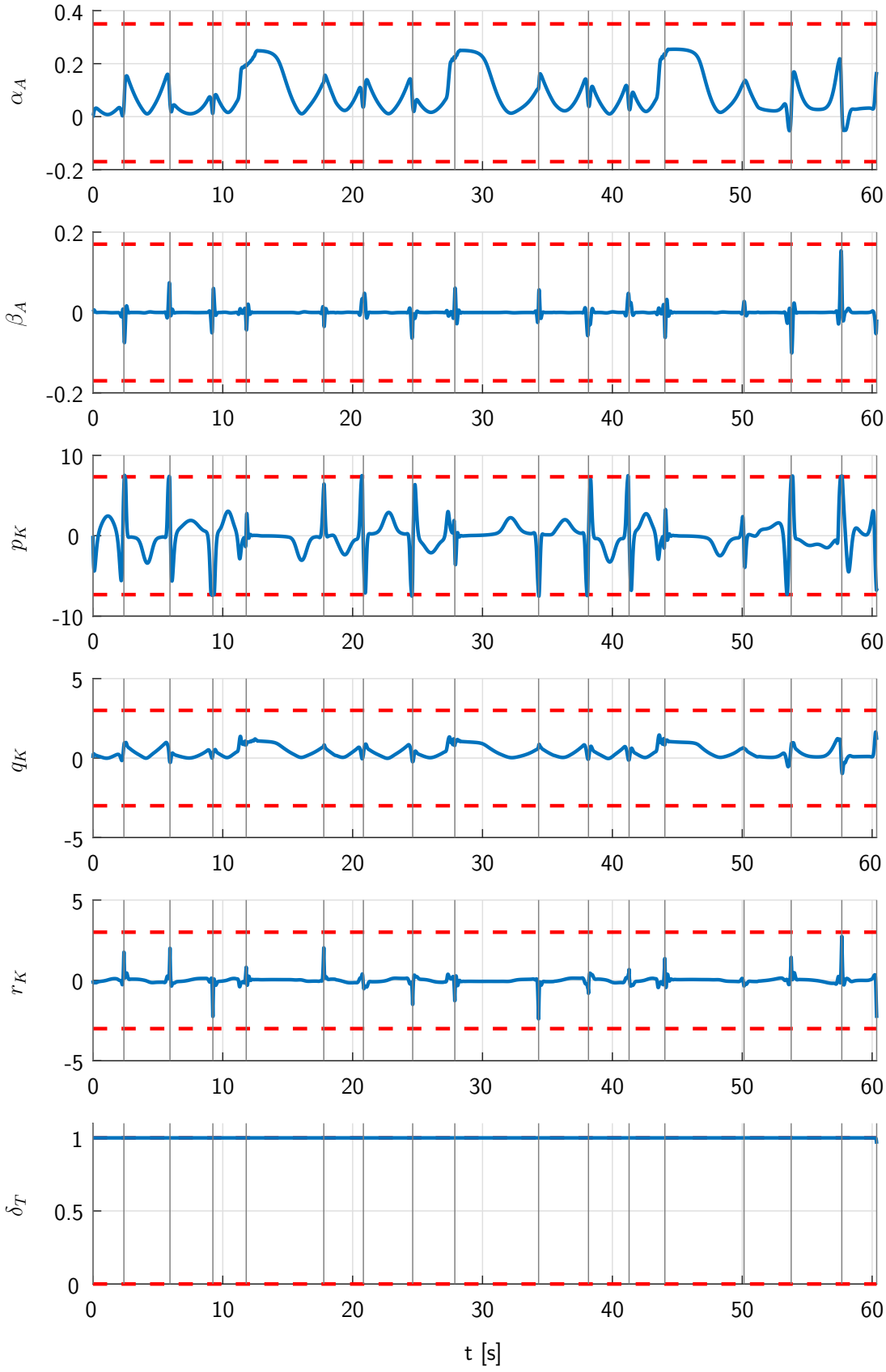

Figure 11.30: *Optimal state histories for the combined simulation model.*

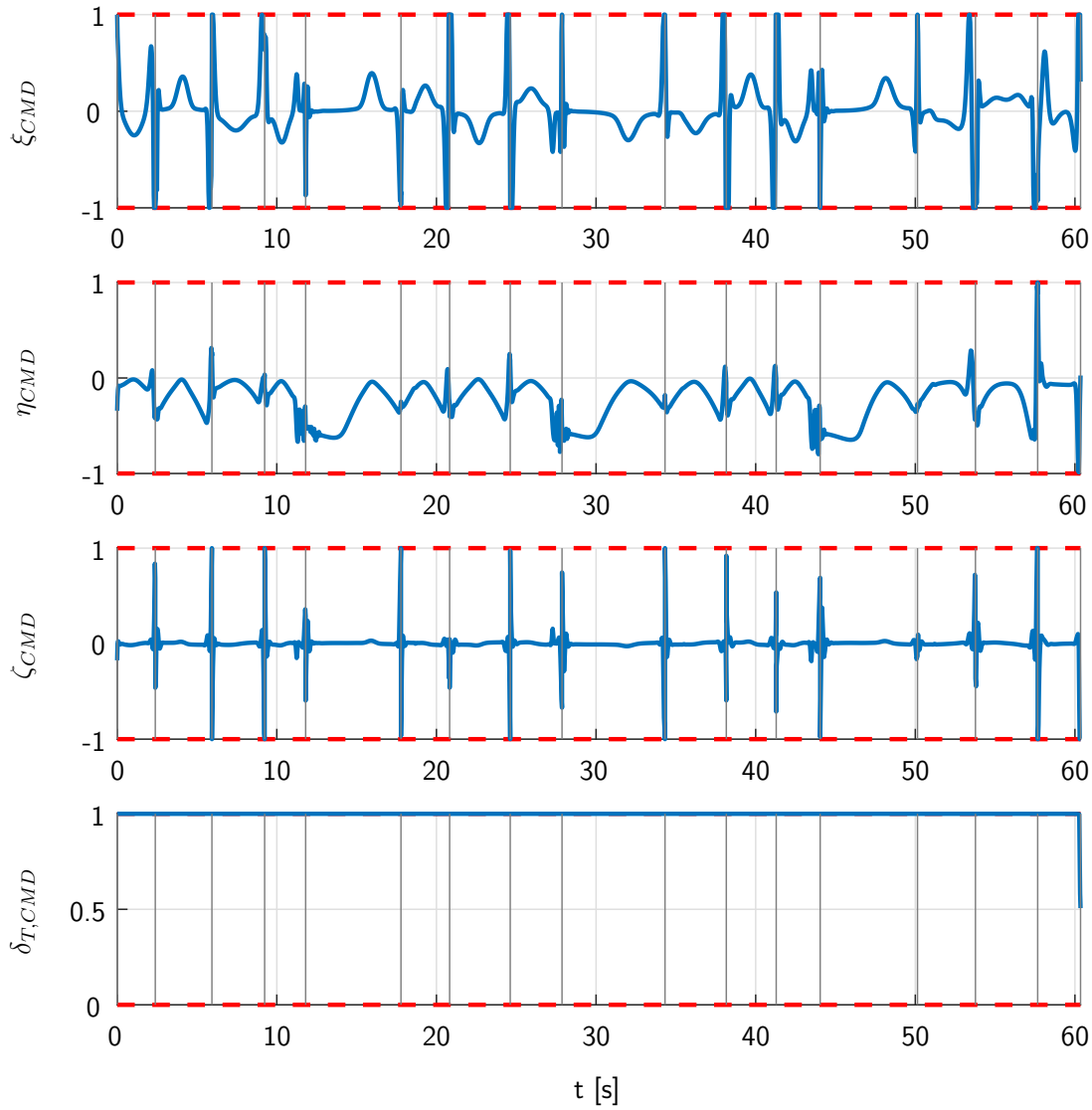

Figure 11.31: *Optimal control histories for the combined simulation model.*

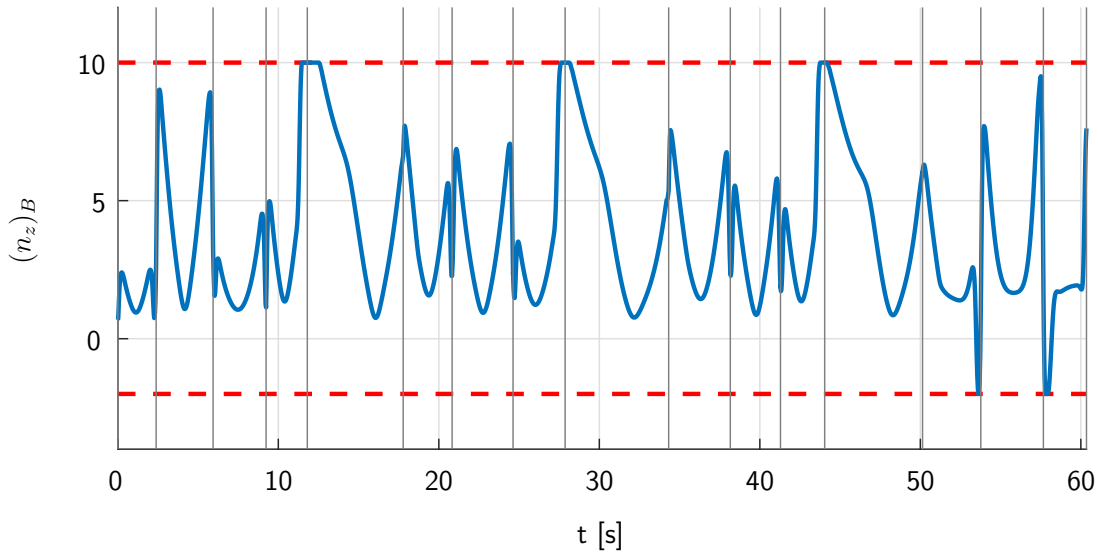

Figure 11.32: *Constrained load factor for the combined simulation model.*

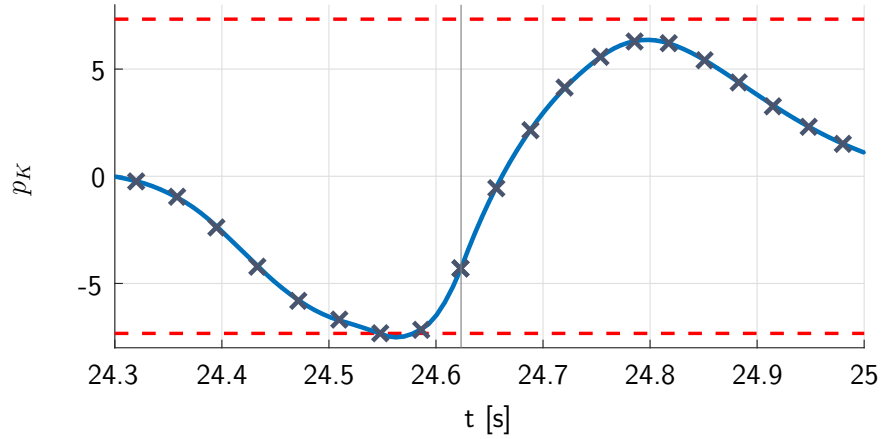

Figure 11.33: *Detail of the roll rate history for the combined simulation model.*

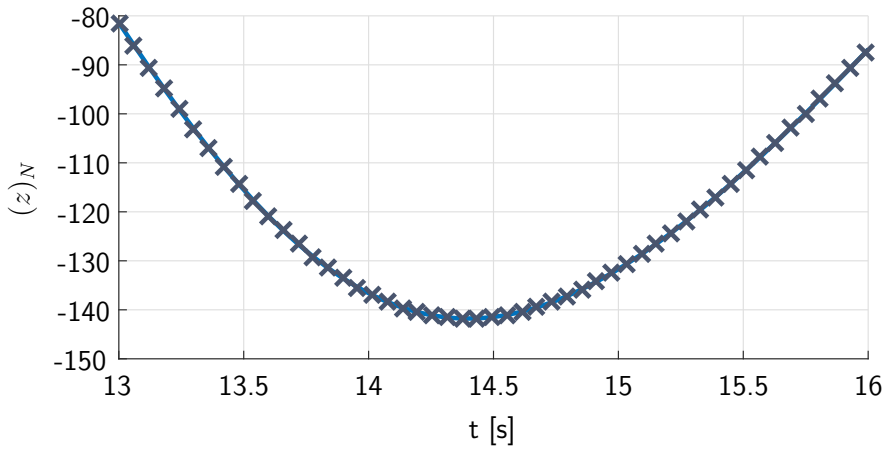

Figure 11.34: *Detail of the z-position history for the combined simulation model.*

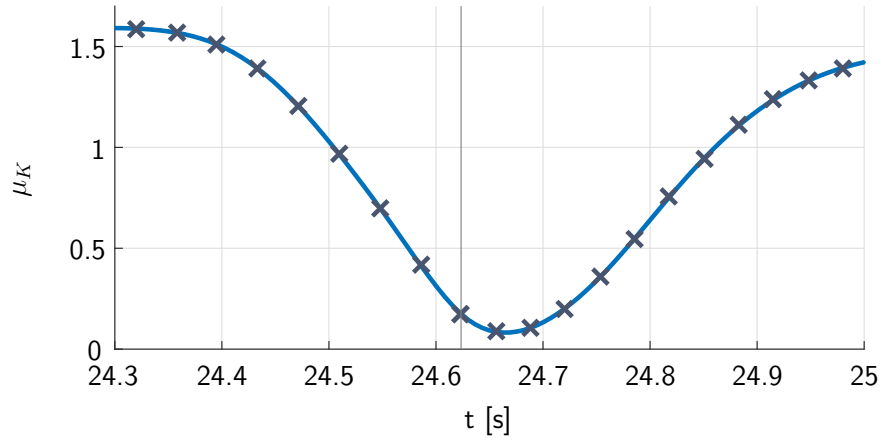

Figure 11.35: *Detail of the bank angle history for the combined simulation model.*

size will be increased by the additional control parameters without increasing the number of constraints. Anyway, the calculation of the gradient matrix will change in this case, requiring further implementation efforts. In cases where some controls are influencing the fast states while others are mainly changing the slow ones, one may want to use different control grids.

11.5 Comparison of the Results for the Different Models

In this section the results from sections 11.2–11.4 are compared to each other and further discussed. Figure 11.36 shows all spatial trajectories created above. No large deviations can be seen in the trajectories, illustrating that the results match closely.

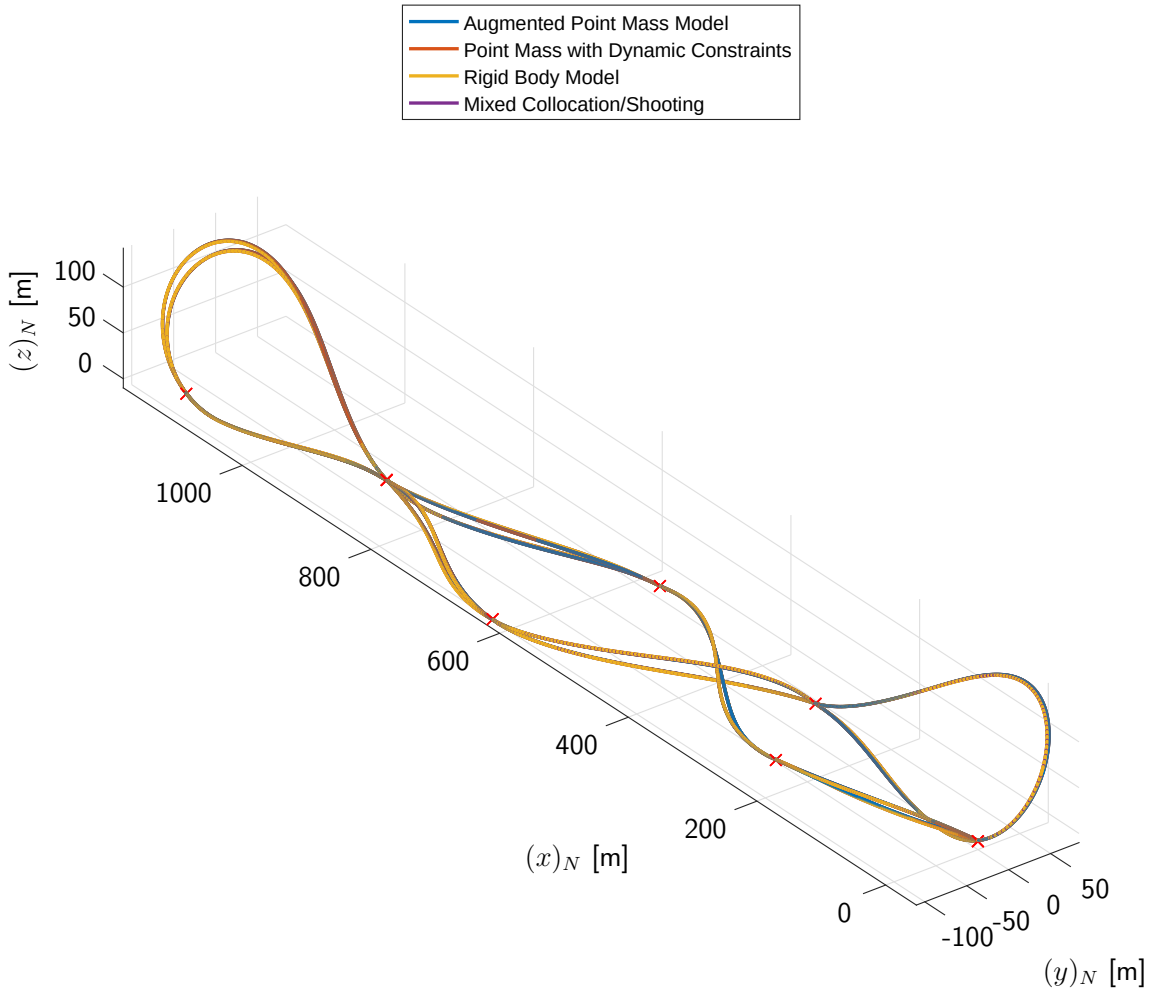

Figure 11.36: *Comparison of optimal race trajectories for the different models.*

In figure 11.37 the situation becomes even more clear. Therein, the *Euclidean* difference between the spatial solution of the highest fidelity solution and those using the other simulation models are visualized. The difference has been calculated based on the relative time progress along the results as the different models lead to different minimal race times. The abscissa of the plot is based on the time of the solution for the rigid body simulation model. As expected, no position differences can be seen when passing the race gates. Moreover, the overall maximum position difference that appears in the results is $4.22m$, which is approximately half the wing span of the aircraft.

Besides, the results of the augmented point mass simulation model and the point mass simulation model with the dynamic control constraints do not differ much. Again, this was expected beforehand, as from a theoretical point of view they both use the same physical model. Anyway, a different numerical behavior was experienced in the solution process, also

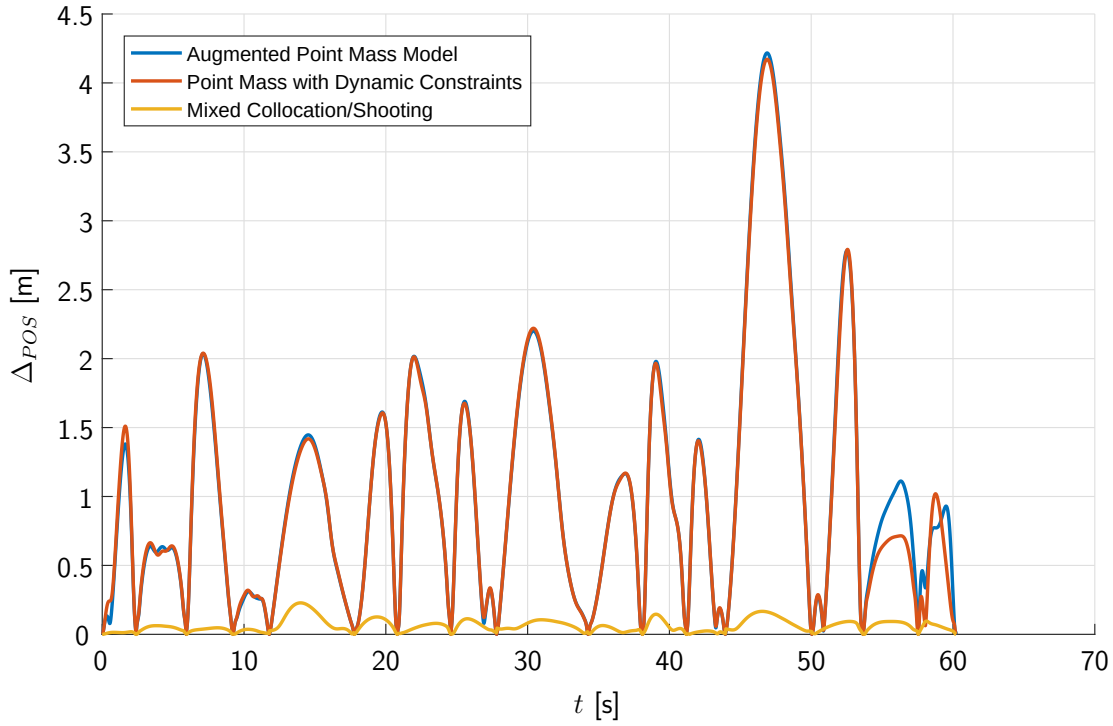

Figure 11.37: *Position difference with respect to the solution for the rigid body simulation model.*

leading to different achievable solution accuracies for the two formulations. Compared to the mixed shooting and collocation approach – which is also based on the rigid body simulation model – the trajectory differences between the pure rigid body simulation model and the point mass simulation models are larger.

The yellow line in figure 11.37 shows the difference between the mixed collocation and shooting approach and the pure collocation of the rigid body simulation model on the fine grid. The observed difference in the trajectory is within the range of some centimeters. Here, the reason is twofold:

- On the one hand, the discretization of the controls is coarser, reducing the theoretical "possibilities of the pilot", and
- on the other hand, different integration schemes are used for the fast dynamics in the two approaches, which influences the numerical solution of the differential equation.

Anyway, the differences in the results achieved are expected to be much lower than any possible errors due to model inaccuracies.

Table 11.9 gives a numeric overview of the different results. First of all, one can see that the optimality tolerance that can be achieved when using the augmented point mass simulation model is worse than that of the other models and methods. The reason for this is the pure numerical condition of the discretized optimization problem. More details on this can be found in chapter 6.

When comparing the results for the first two models listed in the table, it can be seen how the number of optimization parameters can be reduced by removing the linear state equations

Table 11.9: *Overview of the results using the different simulation models and discretization methods.*

	Optimality tolerance	Optimization parameters	Constraints	Iterations	Solution time	Cost function value
Augmented point mass model	$1 \cdot 10^{-5}$	96208	72152	13219	$10782s$	$60.2253s$
Point mass with control constraint	$5 \cdot 10^{-6}$	80176	72184	1927	$1730s$	$60.2304s$
Rigid body simulation model	$5 \cdot 10^{-6}$	136288	112227	883	$2290s$	$60.3437s$
Combined collocation and shooting	$5 \cdot 10^{-6}$	27488	22627	1355	$2144s$	$60.3512s$

and moving them to a dynamic control constraint. On the other side, the number of constraints hardly changes, as only a transformation of them takes place there. The number of iterations as well as the solution time show that this problem reformulation leads to a significant decrease in the required computational effort, due to a better conditioned problem, without significant changes in the cost function value.

Comparing the resulting cost values for the rigid body simulation model (in the last two rows of table 11.9) to that of the point mass simulation model, one can see how the additional constraints inherent in the rotational dynamics increase the minimum race time. The marginal difference in the race time between the two last rows of the table is attributed to the coarser control grid in the last case which reduces the theoretical "control authority of the pilot".

Of course, the number of optimization parameters and controls increases when adding the rotational dynamics to the simulation model. Interestingly, the number of iterations required to solve the optimal control problem with the rigid body simulation model is smaller than that of the point mass simulation model. Again, the reason seems to be the better numerical condition of the problem. On the other side, the solution time of the optimization is larger for the more precise model as the evaluation of the model dynamics is more complex. When using the combined discretization method, the number of optimization parameters as well as the number of constraints decreases strongly as the number of collocation grid points is also reduced. However, *IPOPT* takes more iterations to converge in the last case, even though convergence can be achieved in a shorter time.

For the sake of completeness of the comparison, another optimization has been performed for the point mass simulation model with the dynamic control constraints, using 101 grid points per phase. This way the results of the point mass model and the ones of the combined collocation and shooting can also be compared. Interestingly, the results for the point mass optimization were barely different from the ones for 501 discretization points, listed in table 11.9.

Summarizing, one of the most significant result of this thesis is, that there are cases where a high fidelity optimization using a rigid body simulation model converges better and faster than one using a pure point mass simulation model. This can clearly be seen from the comparison of the first and the third row of table 11.9.

11.6 Initial Guess Generation Using Extended Homotopy

An exemplary slalom part of an air race has been selected for demonstrating the extended homotopy principle based on sensitivities that is described in section 5.1.3. The problem is built up by three race gates, located and oriented as given in table 11.10. The orientation of *Gate 2* is chosen as the *homotopy parameter* using the values also given in the table.

Table 11.10: *Race gate constraints in the extended homotopy example.*

	Gate 1	**Gate 2**	**Gate 3**
$(x)_N$	$0m$	$200m$	$400m$
$(y)_N$	$0m$	$0m$	$0m$
$(z)_N$	$0m$	$0m$	$0m$
χ_K	0	$\left[\frac{\pi}{36}, \frac{\pi}{12}, \frac{\pi}{6}\right]$ $\frac{\pi}{4}$	0
γ_K	0	0	0
μ_K		$-\frac{\pi}{18} < \mu_K < \frac{\pi}{18}$	

The dynamic model used in the example is the point mass simulation model with dynamic control constraints from section 11.2.2. The general model parameters are taken from section 11.1. The states and the controls in each phase are discretized on a grid containing 151 nodes. The numerical optimization problems are solved using *IPOPT* [WB06]. In the example, the analytic *Hessian* is calculated and also handed over to *IPOPT* where it is used for the optimization.

As reference, the final problem was directly solved, starting from an initial guess that has been obtained by linear interpolation of the gate constraints. Figure 11.38 shows the optimal spatial trajectory through the race course.

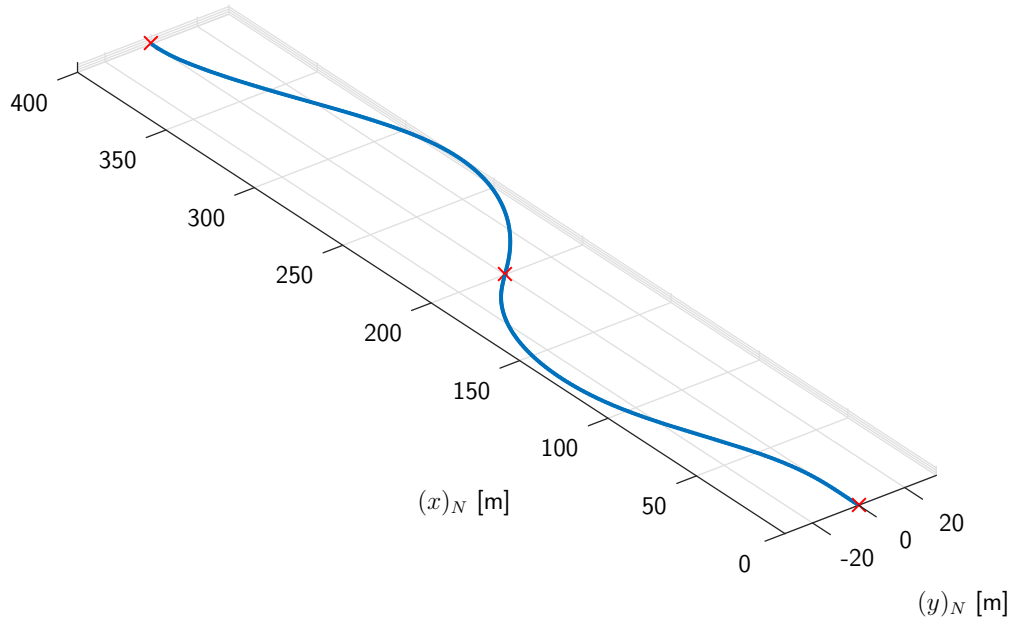

Figure 11.38: *Optimal spatial trajectory through the race course for the homotopy example.*

In figure 11.39 the optimal state histories are depicted. It can be seen that none of the pure state constraints becomes active and the aircraft is not required to exploit the allowed vertical range. Similar to the optimizations performed before, the aircraft is not able to maintain the initial velocity as the drag during the agile maneuvers cannot be compensated by the thrust installed on board the vehicle.

The optimal control histories resulting from the optimization can be seen in figure 11.40. Around the second race gate, the angle of sideslip reaches its maximum, as this allows a faster turn and a faster passing through the race gate while maintaining the required aircraft orientation. In the bank angle control a turn with maximum allowed bank rate can be seen around the second gate and between the first and the second gate. The thrust lever position is always at full throttle, in order to best compensate for the drag and reaching the highest possible velocity.

The computational times required to solve the optimal control problems to a feasibility tolerance of $tol_C = 10^{-6}$ and an optimality tolerance of $tol_J = 5 \cdot 10^{-6}$ directly, using the basic homotopy approach or based on the extended homotopy approach from section 5.1.3 are summed up in table 11.11. It can be seen how, in this example, a reduction in computational time by approximately 34% can be achieved by using the basic homotopy method while a reduction by approximately 38% is possible using the sensitivity based extended homotopy procedure. Additionally, it has to be mentioned that the evaluation of the sensitivity update has a noticeable influence on the runtime of the extended homotopy procedure and is not yet tuned to maximum efficiency in the code. Hence, further improvements in computational time may be possible here.

Table 11.11: *Solution times for the different initial guess approaches.*

Method	**Solution time**
Solve final problem directly	$199s$
Basic homotopy	$130s$
Sensitivity based homotopy	$123s$

11.6.1 Basic Homotopy Scheme

In this section the results using the basic homotopy scheme from section 5.1.1 are presented. In the problem considered here, the second race gate is stepwise rotated from an orientation of $\chi_{G2} = \frac{\pi}{36} = 5\,\mathrm{deg}$ to an orientation of $\chi_{G2} = \frac{\pi}{4} = 45\,\mathrm{deg}$. Figure 11.41 shows the spatial trajectories for the different orientations of the race gate. The intermediate solutions have been calculated to an optimality tolerance of $tol_J = 10^{-4}$ and a feasibility tolerance of $tol_C = 10^{-4}$, while the final problem has again been solved using $tol_J = 5 \cdot 10^{-6}$ and $tol_C = 10^{-6}$. Each solution for the optimization parameter vector of the intermediate problems is directly used as the initial guess for the next problem. The same holds for the multipliers that are supplied to *IPOPT*.

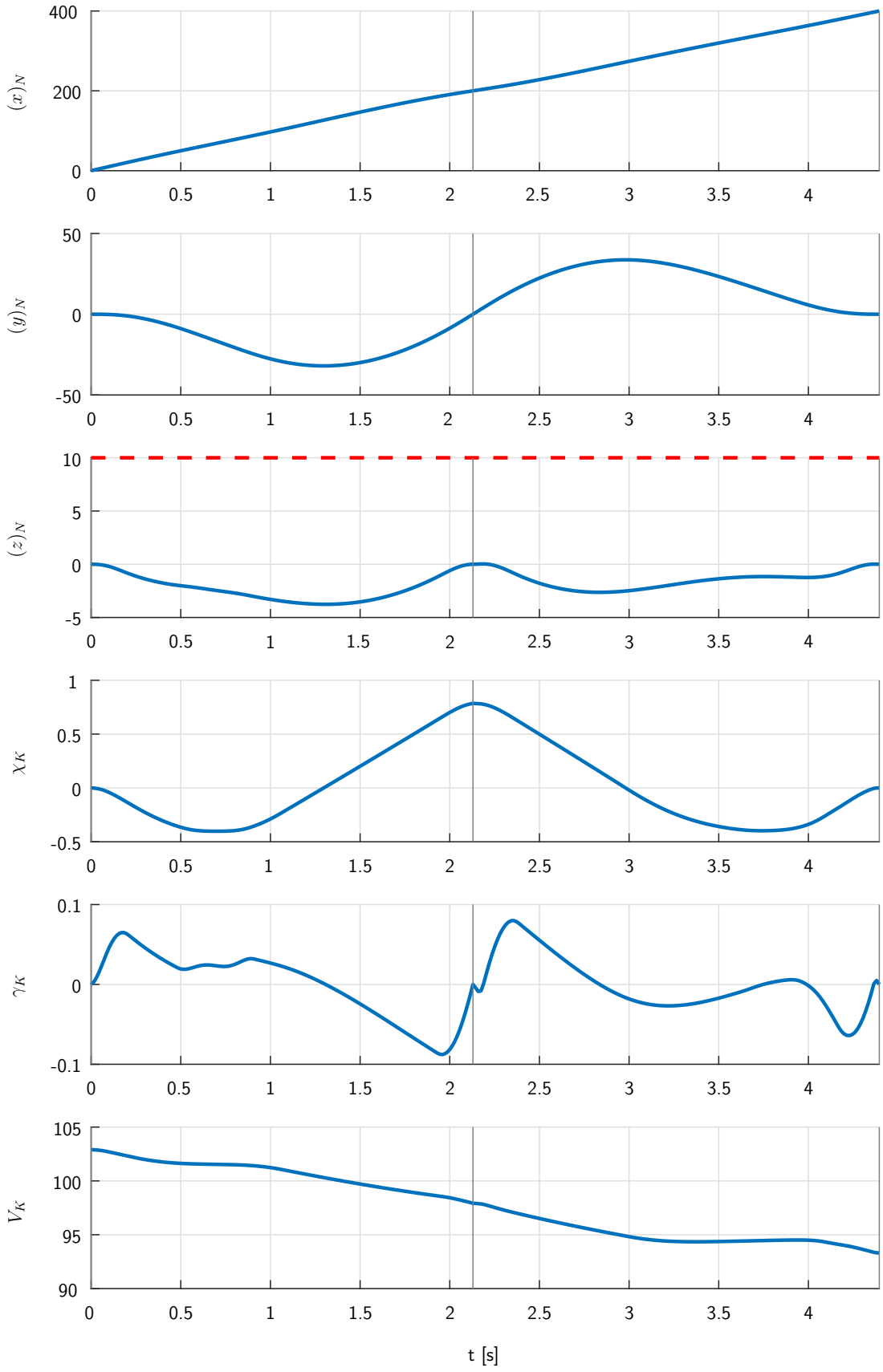

Figure 11.39: *Optimal state histories for the homotopy example.*

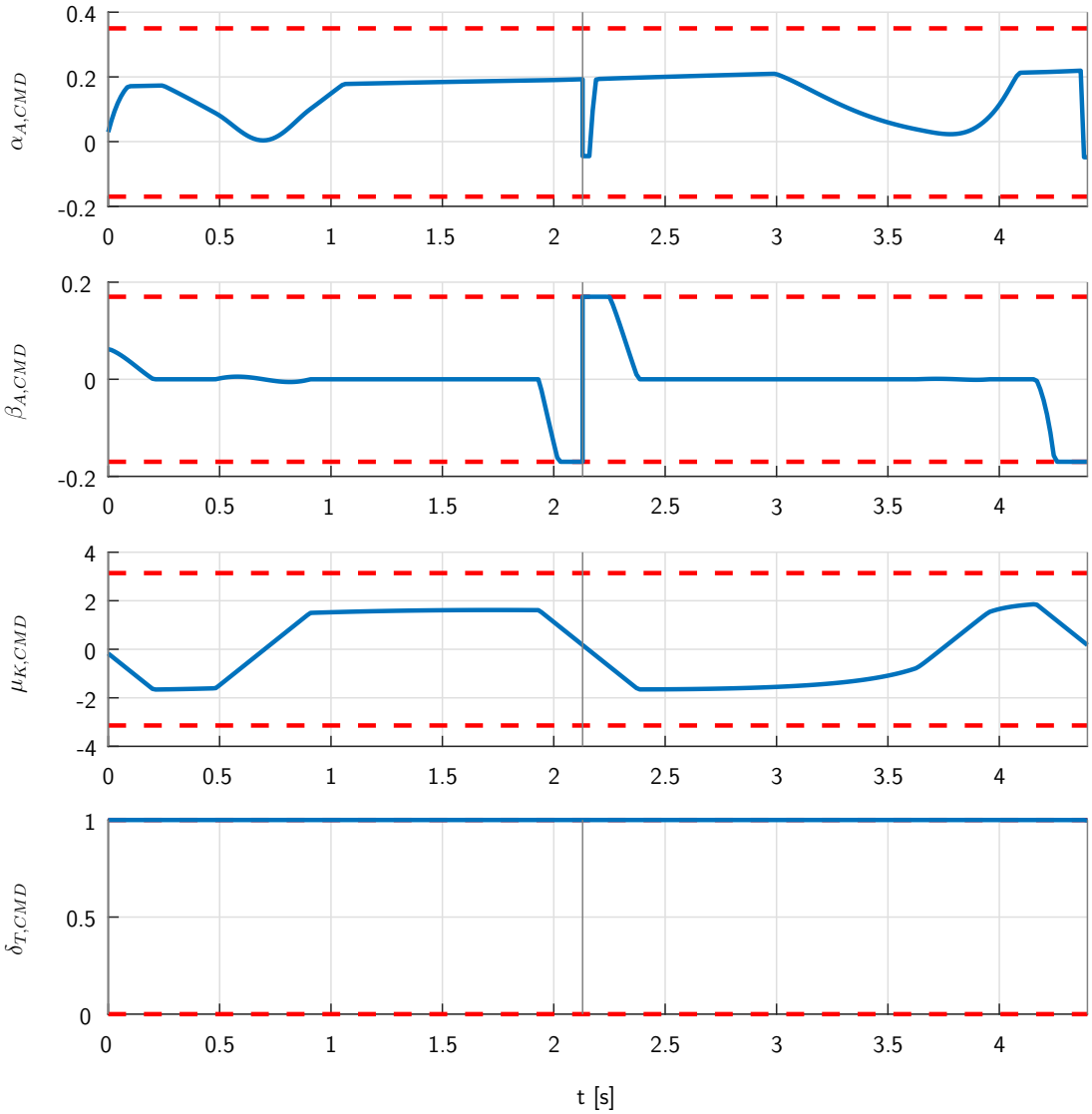

Figure 11.40: *Optimal control histories.*

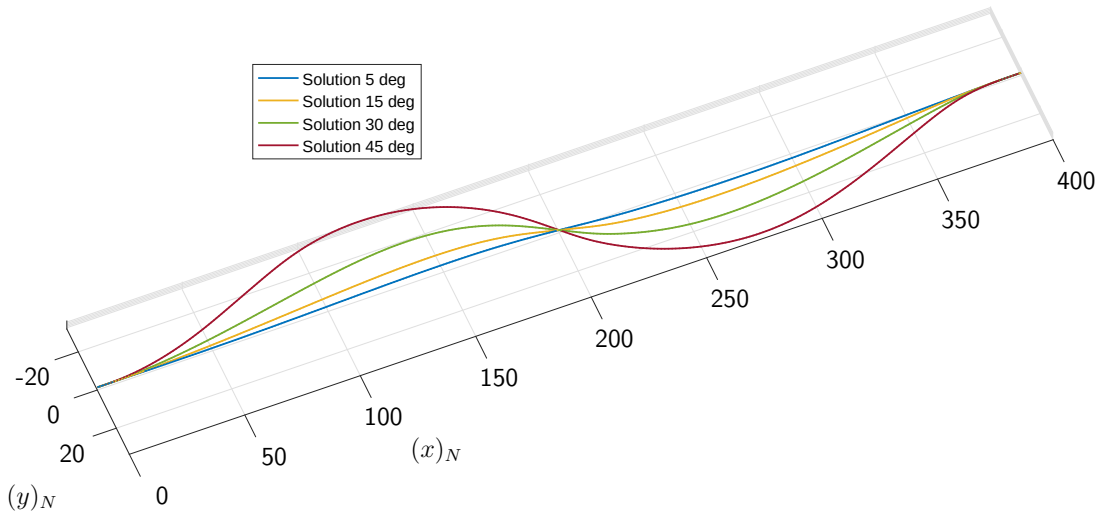

Figure 11.41: *Optimal spatial trajectories for the basic homotopy steps.*

11.6.2 Sensitivity Based Homotopy Scheme

Now, the results using the sensitivity based homotopy scheme from section 5.1.3 are presented. Similar to the procedure above, the second race gate has been turned in three steps from $\chi_{G2} = \frac{\pi}{36}$ to $\chi_{G2} = \frac{\pi}{4}$. The same intermediate and final tolerances have been used as with the basic homotopy scheme. Figure 11.42 shows the optimal spatial trajectories for the different problems as well as the initial guess trajectories calculated using equations (5.1)–(5.3).

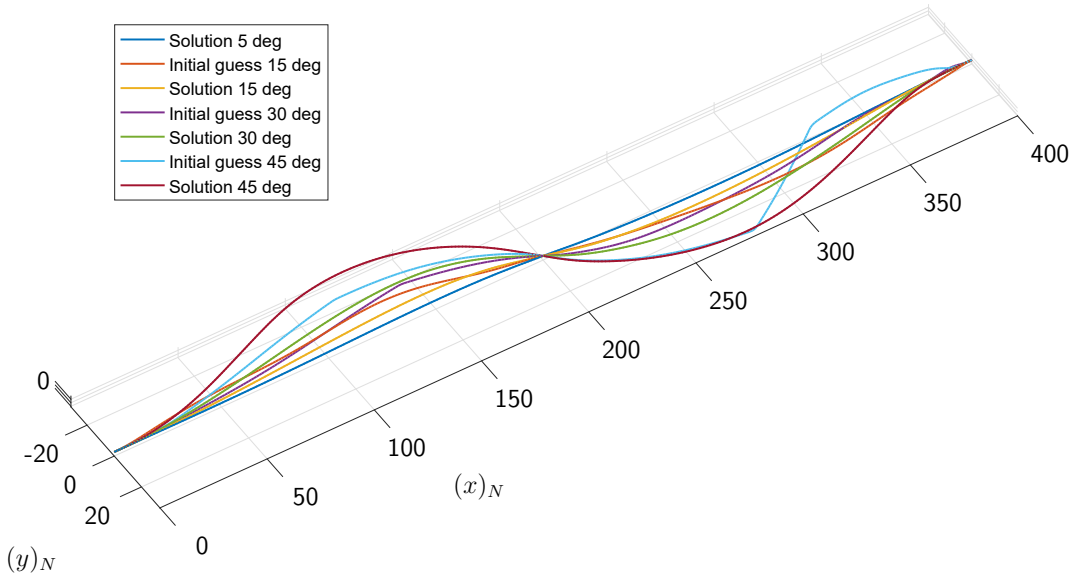

Figure 11.42: *Optimal spatial trajectories for the extended homotopy steps.*

The blue line represents the optimal trajectory for the first problem, with a course angle for the second race gate of $\chi_{G2} = \frac{\pi}{36}$. Based on this solution and the sensitivities, the first order update of the trajectory can be calculated. When using a gate angle of $\chi_{G2} = \frac{\pi}{12}$, the orange line in figure 11.42 results. Even though the figure only depicts the spatial states, during the process all states and all multipliers are updated. Now, the problem is solved again, resulting in the yellow line. The previously mentioned steps are repeated again in order to determine the purple and the green lines. When moving from $\chi_{G2} = \frac{\pi}{6}$ to $\chi_{G2} = \frac{\pi}{4}$ the update of the trajectory results in the light blue line that can be seen in the figure. Strangely, at this point the update seems to be worse than the previous solution. At this point the explanation for this result is not clear and needs further research.

11.7 Initial Guess Generation Based on RTS Smoothing

In order to demonstrate the idea presented in section 5.2, a short part of an air race track has been chosen as an illustrative example. The considered race course is defined by five race gates that are positioned and oriented as given in table 11.12. The point mass simulation model with the dynamic control constraints from section 11.2.1 is used in the simplified optimal control problem for determining the initial result. The full problem is then formulated based on the

rigid body simulation model from section 11.1. The same model – extended by zero dynamics for the controls – is also used for the RAUCH-TUNG-STRIEBEL (*RTS*) smoother.

Table 11.12: *Race gate constraints for the initial guess generation example using an RTS smoother.*

	Gate 1	**Gate 2**	**Gate 3**	**Gate 4**	**Gate 5**
$(x)_N$	$0m$	$300m$	$600m$	$600m$	$0m$
$(y)_N$	$0m$	$100m$	$0m$	$100m$	$100m$
$(z)_N$	$0m$	$0m$	$0m$	$0m$	$0m$
χ_K	0	0	0	π	$\frac{3}{2}\pi$
μ_K	$-\frac{\pi}{18} < \mu_K < \frac{\pi}{18}$				

Besides the initial and final boundary conditions defined in table 11.12, the initial velocity is bounded by $80\frac{m}{s} \leq V_K \leq 102.9\frac{m}{s}$. As the race course is defined by five race gates, the trajectory optimization problem consists of four phases, each discretized using 201 discretization nodes for the states and the controls, independent of the model. Due to the different lengths of the phases, a grid spacing of $0.0156s \leq \Delta t \leq 0.0370s$ results. In order to achieve smooth results, in this case, a small LAGRANGE cost function of the form

$$L = \frac{1}{2} \cdot \left(\xi_{CMD}^2 + \eta_{CMD}^2 + \zeta_{CMD}^2\right) \tag{11.21}$$

has been added in the examples using the rigid body simulation model.

In the optimization that has been performed for comparison as well as in the initial optimization using the point mass simulation model, the initial guess has been generated by linear interpolation between the initial and the final boundary conditions of each phase. In case no such conditions were present, the mean between the upper and lower boundaries for the respective value has been selected.

The model used in the *RTS* smoother is based on the rigid body simulation model, and has been extended by three dimensional zero dynamics in order to estimate the unknown control inputs based on the process noise, as described in section 5.2.2:

$$\begin{pmatrix} \dot{\hat{\xi}} \\ \dot{\hat{\eta}} \\ \dot{\hat{\zeta}} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{11.22}$$

Overall, a model containing 16 states and one control (the thrust lever position δ_T) results. The measurements for the filter are chosen to be the states and the controls of the point mass simulation model.

All variables appearing in the model of the *RTS* smoother can be found in table 11.13. The table lists in which groups the respective values appear, with $\hat{\mathbf{x}}$ representing the states of the smoother, $\hat{\mathbf{u}}$ its controls, and $\hat{\mathbf{y}}$ the measurements. In the next column, the orders of magnitude for the values are given that can either be derived from the solution of the point mass

optimization or from the constraints that apply along the trajectory. The last three columns of the table list the assumed stochastic properties of the noise processes associated to the signals. Therein, $\mathbf{Q}$ represents the process noise power spectral density, $\mathbf{R}$ the measurement noise power spectral density, and $\mathbf{P}_{\mathbf{xx},0}$ the initial state covariance. It can be seen that they have been estimated from the orders of magnitude as described in section 5.2.1 with some slight modifications. The process noise for the rotational states p_K, q_K, and r_K has been decreased, putting more weight on the system dynamics. On the other side, the measurement noise of the attitude states μ_K, α_A, and β_A has been increased, also tying the attitude more to the dynamics than to the measurements.

Table 11.13: *Variables appearing in the* RTS *smoother.* $\hat{\mathbf{x}}$ *represents the states of the smoother,* $\hat{\mathbf{u}}$ *the controls, and* $\hat{\mathbf{y}}$ *the measurements. The process power spectral density is given by* $\mathbf{Q}$*, the measurement noise power spectral density by* $\mathbf{R}$*, and the initial state covariance by* $\mathbf{P}_{\mathbf{xx},0}$.

Variable	Part of	Magnitude	$\mathbf{Q}$	$\mathbf{R}$	$\mathbf{P}_{\mathbf{xx},0}$
$(x)_N$	$\hat{\mathbf{x}}, \hat{\mathbf{y}}$	10^3	10^6	10	1
$(y)_N$	$\hat{\mathbf{x}}, \hat{\mathbf{y}}$	10^3	10^6	10	1
$(z)_N$	$\hat{\mathbf{x}}, \hat{\mathbf{y}}$	10^3	10^6	10	1
χ_K	$\hat{\mathbf{x}}, \hat{\mathbf{y}}$	1	10^3	10^{-2}	10^{-3}
γ_K	$\hat{\mathbf{x}}, \hat{\mathbf{y}}$	1	10^3	10^{-2}	10^{-3}
μ_K	$\hat{\mathbf{x}}, \hat{\mathbf{y}}$	1	10^3	1	1
V_K	$\hat{\mathbf{x}}, \hat{\mathbf{y}}$	10^2	10^5	1	10^2
α_A	$\hat{\mathbf{x}}, \hat{\mathbf{y}}$	1	10^3	1	1
β_A	$\hat{\mathbf{x}}, \hat{\mathbf{y}}$	1	10^3	1	1
p_K	$\hat{\mathbf{x}}$	10	1	–	10
q_K	$\hat{\mathbf{x}}$	5	0.5	–	5
r_K	$\hat{\mathbf{x}}$	5	0.5	–	5
δ_T	$\hat{\mathbf{x}}$	1	10^3	–	1
ξ	$\hat{\mathbf{x}}$	1	10^6	–	10^4
η	$\hat{\mathbf{x}}$	1	10^6	–	10^4
ζ	$\hat{\mathbf{x}}$	1	10^6	–	10^4
$\delta_{T,CMD}$	$\hat{\mathbf{u}}$	1	–	–	–

In this example, an initial guess for the LAGRANGE multipliers of all constraints appearing in the discretized optimal control problem, that already have been in the simplified problem has been specified. In order to do so, the resulting multipliers from the simplified problem have been resorted to the multiplier vector of the high fidelity problem. All remaining values have been initialized with zeros.

Figure 11.43 depicts the resulting spatial trajectory for the final rigid body optimization. Similar to the previous examples, the positions of the race gates are marked by red crosses.

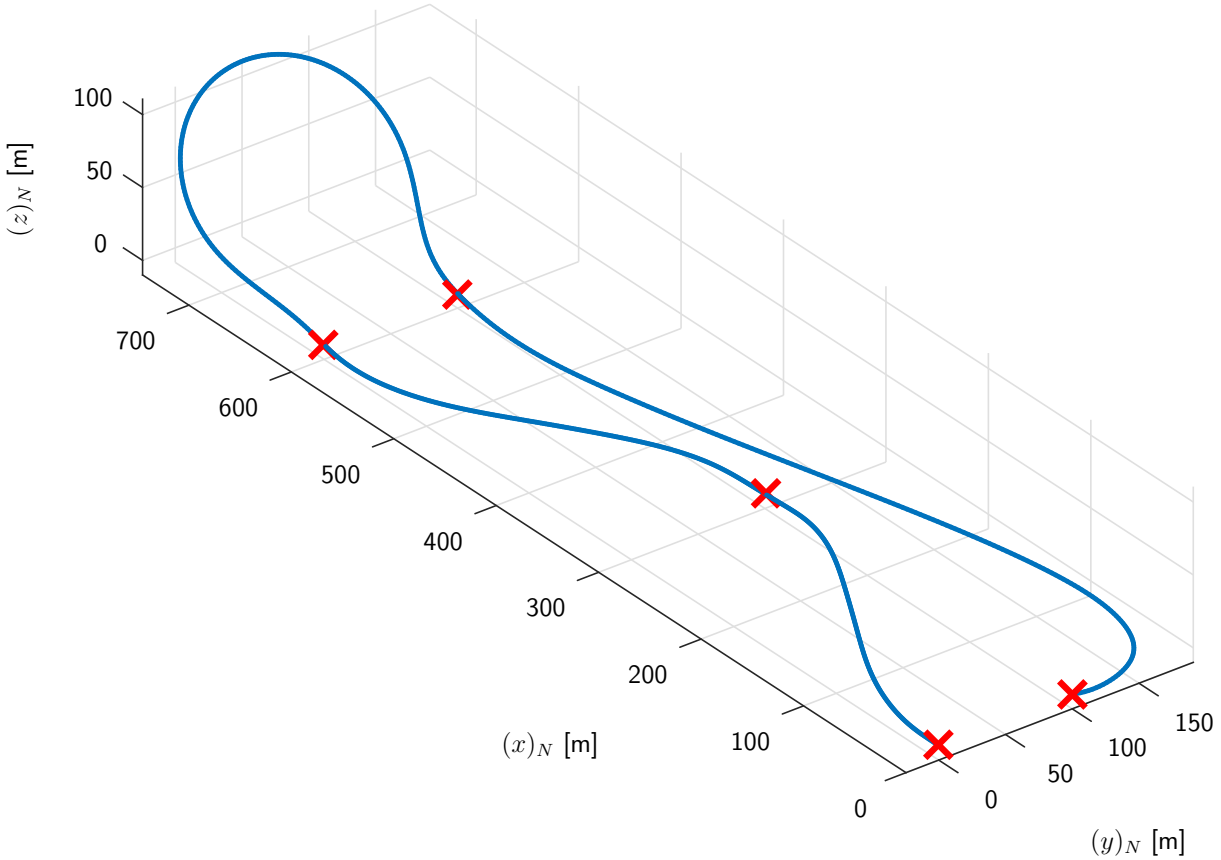

Figure 11.43: *Optimal spatial trajectory through the race course for the RTS smoother example.*

The plots in the figures 11.44 and 11.45 show the state and control histories that result from the different steps of the method. As the optimal simulation times for the point mass model and the rigid body simulation model slightly differ (see table 11.14), the histories are drawn with respect to relative time along the race track. This way, the states and controls at the same point along the trajectory can be directly compared. Where available, the yellow line represents the optimal histories for the point mass optimization, and the red line the one for the rigid body optimization (the final result). The blue line shows the states and the controls that result from the *RTS* smoother.

It can be seen that the smoother can follow the measured states very well – as expected due to the specification of the noise processes. Concerning the controls of the simplified model, the smoother is able to follow μ_K quite closely, while in α_A and especially in β_A slight deviations appear. When looking at the results for the rotational rates p_K, q_K, and r_K the reason becomes more visible: As the point mass simulation model does not feature any rotational dynamics (except the limit in $\dot{\mu}_K$), differences have to be expected there. In general, the results from the smoother are not as smooth as the optimal trajectory. The reason here is twofold: On the one hand, the underlying assumption that the control surfaces of the aircraft and the inaccuracies in the dynamics are driven by noise processes is not valid. On the other hand, the smoother tries to reconstruct the trajectory from the available information, while

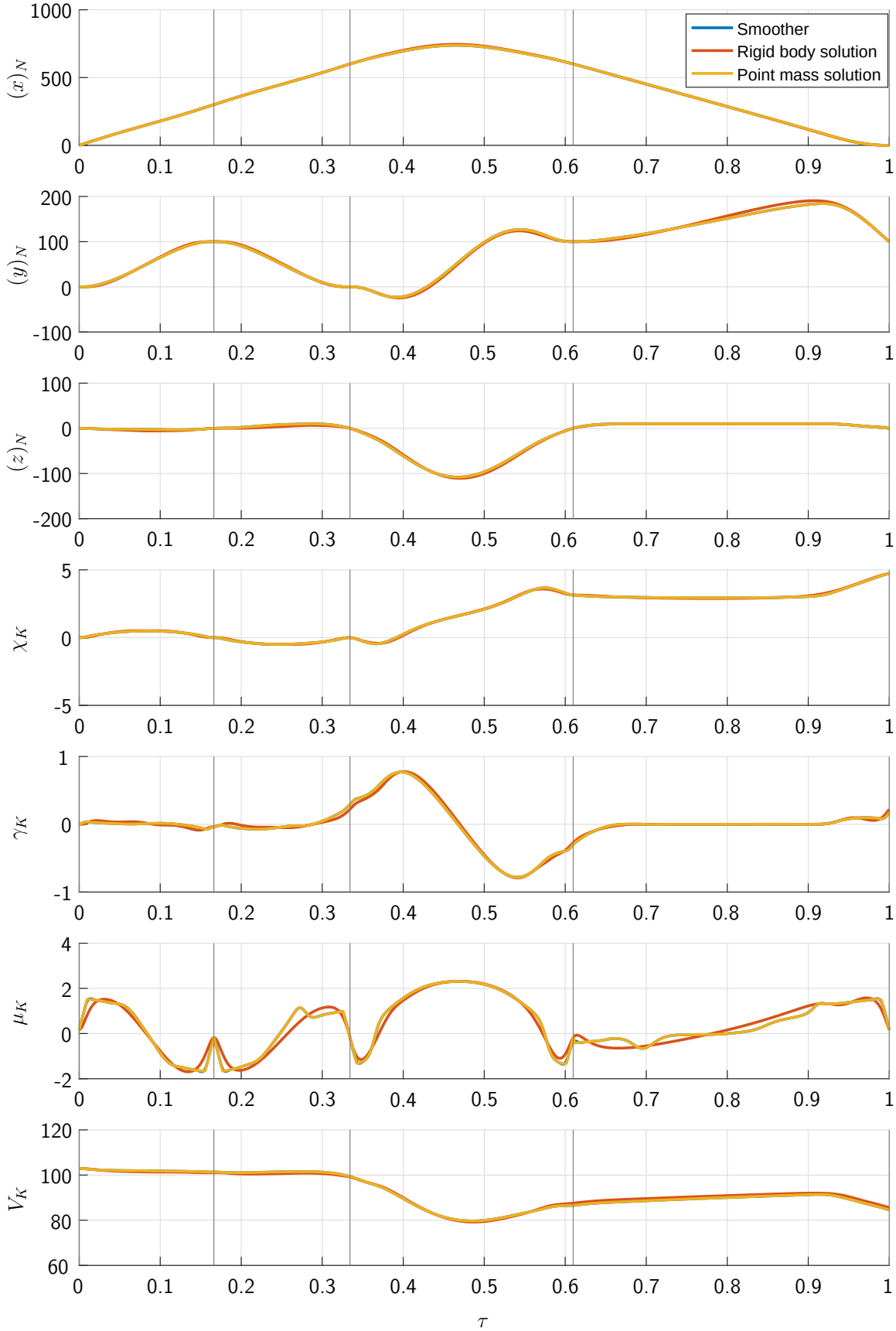

Figure 11.44: *Optimal state histories for the RTS smoother example.*

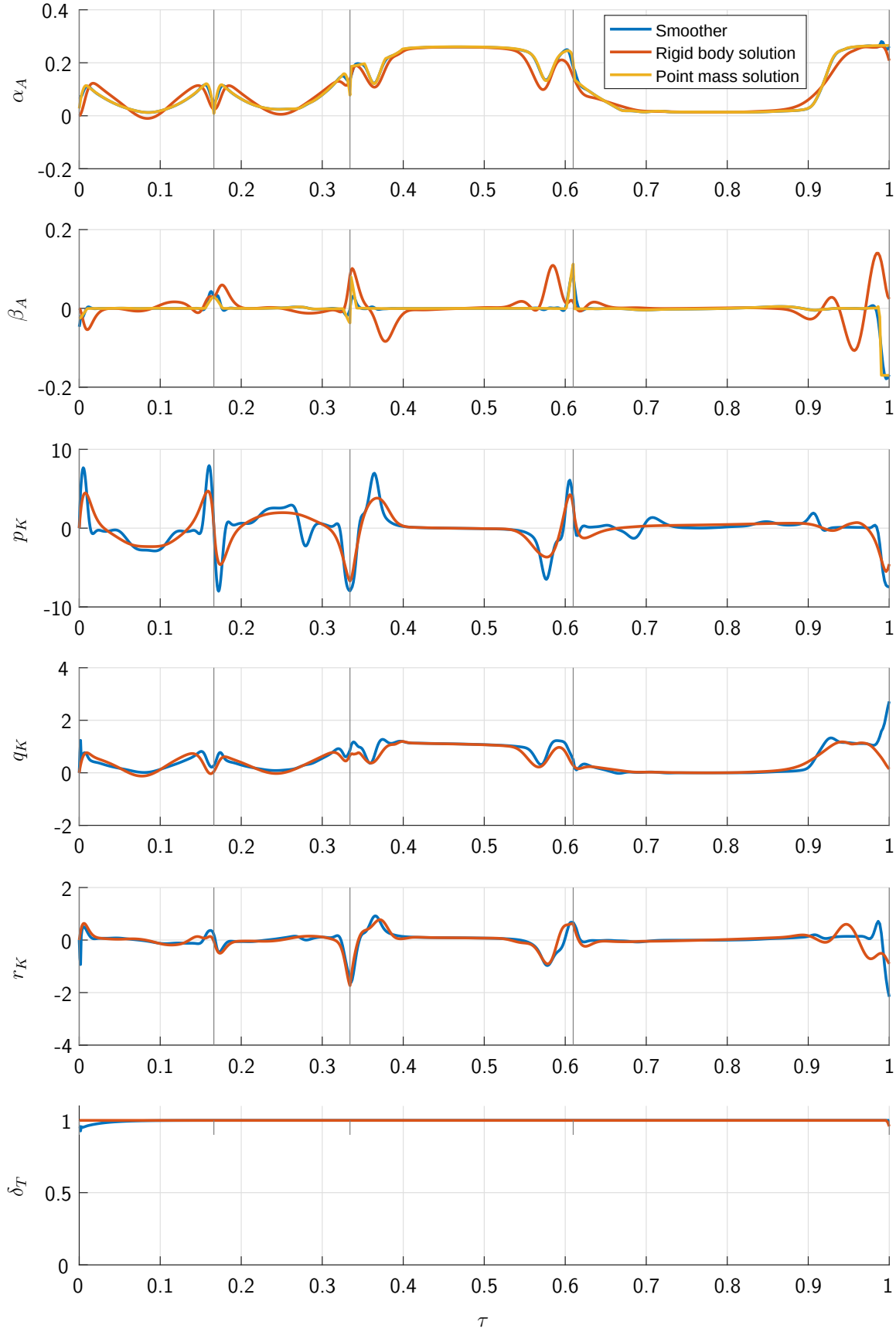

Figure 11.45: *Optimal state histories for the RTS smoother example.*

the optimization has another goal, namely to find a race track that minimizes the race time. As the thrust lever is in principle not touched throughout the race, no differences between the estimated values and the real values can be seen.

Figure 11.46 compares the controls estimated by the smoother to these calculated in the high fidelity optimization. In general, good accordance between the estimated values and the optimal values can be seen. (One has to keep in mind that the model that has been used to determine the data that is used in the smoother does not feature any rotational dynamics.) Again, the optimal history is slightly more steady than the estimated one as the basis for the estimation of the control surface deflections is pure *Gaussian* white noise. As the thrust lever dynamics is not used in the example, no meaningful conclusions can be drawn for the corresponding history.

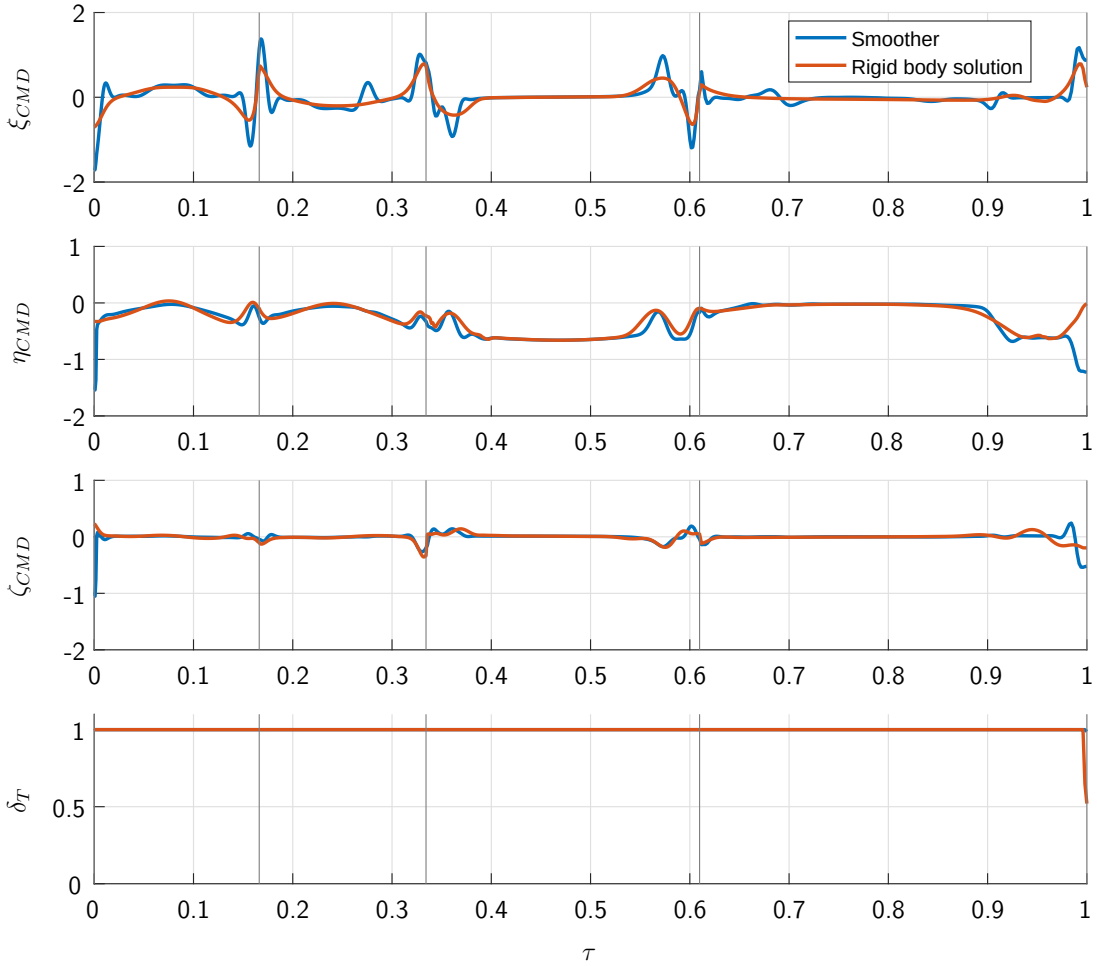

Figure 11.46: *Optimal control histories for the RTS smoother example.*

In table 11.14, the results from this example are summarized, showing that the runtime and the number of required iterations for the high fidelity rigid body optimization can be significantly improved when using a good initial guess calculated based on the result from the simplified model. Anyway, when summing up the overall calculation time that is required for the process described here, no runtime improvement can be achieved in this example. However, in other cases where the solution of the high fidelity problem may even be impossible as no

convergence can be achieved, the process described may be helpful for improving solution robustness or solution speed.

Table 11.14: *Results for the RTS smoother example.*

Grid	Iterations	Runtime	Cost function value
Point Mass Model	1196	$186s$	$18.90s$
Rigid Body Model (direct)	679	$262s$	$19.10s$
Rigid Body Model (after smoothing)	479	$147s$	$19.10s$

The idea illustrated here has already been introduced in [BFH12], using a *Gauss Pseudospectral Discretization Method* and *SNOPT* [GMS02] for solving the numerical optimization problem. The example presented there was slightly different from the one used here, both regarding the models and the race track, and resulted in an overall improvement of optimization runtime. It turned out in several tests that *SNOPT* – which cannot be used for solving extremely large scale discretized problems like the one presented here – is more sensitive with respect to the initial guess, possibly explaining that overall runtime improvements could be achieved there.

Depending on the application and the required optimality of a trajectory, it may even be feasible to only perform an optimization using the simplified model and then use the method presented here to create an approximation for the final trajectory. Of course, feasibility with respect to the constraints and the dynamics has to be ensured in these cases by other means. Further research is required here.

11.8 Control Grid Refinement

In this section, an air race trajectory optimization problem comprising four segments, bounded by five race gates, is presented that has been solved using the control grid refinement method from section 9.3 and 9.4. In the example the same model as in section 11.6 is used but a different race track is considered. Table 11.15 lists the race gate constraints that have been taken from the first part of a real air race track. Besides, the initial velocity has been bounded by $80\frac{m}{s} \leq V_K \leq 102.9\frac{m}{s}$. The problem has been solved on a state grid containing 101 state nodes per phase, resulting in a total of 404 grid points. As a trapezoidal collocation scheme is used for the discretization of the problem, the control grid nodes have to coincide with the state nodes, resulting in a maximum of 404 control grid points. Figure 11.47 shows the optimal spatial trajectory, while in figure 11.48 the state trajectories are depicted. The vertical, gray lines represent the times when a race gate is passed.

The optimal control histories that have been calculated during the optimization can be seen in figure 11.49. In the first plot therein, the nodes of the different grids are shown. From bottom to top, the nodes of the regular grids with 84, 204, and 404 nodes can be seen.

Table 11.15: *Race gate constraints for the control grid refinement example.*

	Gate 1	Gate 2	Gate 3	Gate 4	Gate 5
$(x)_N$	$0m$	$237.6m$	$565.0m$	$880.0m$	$1127.1m$
$(y)_N$	$0m$	$-58.5m$	$54.4m$	$-29.3m$	$-79.5m$
$(z)_N$	$0m$	$0m$	$0m$	$0m$	$0m$
χ_K	$\frac{\pi}{18}$	-0.1747	-0.1728	0	0
γ_K	0	0	0	0	0
μ_K	$-\frac{\pi}{18} < \gamma_K < \frac{\pi}{18}$				

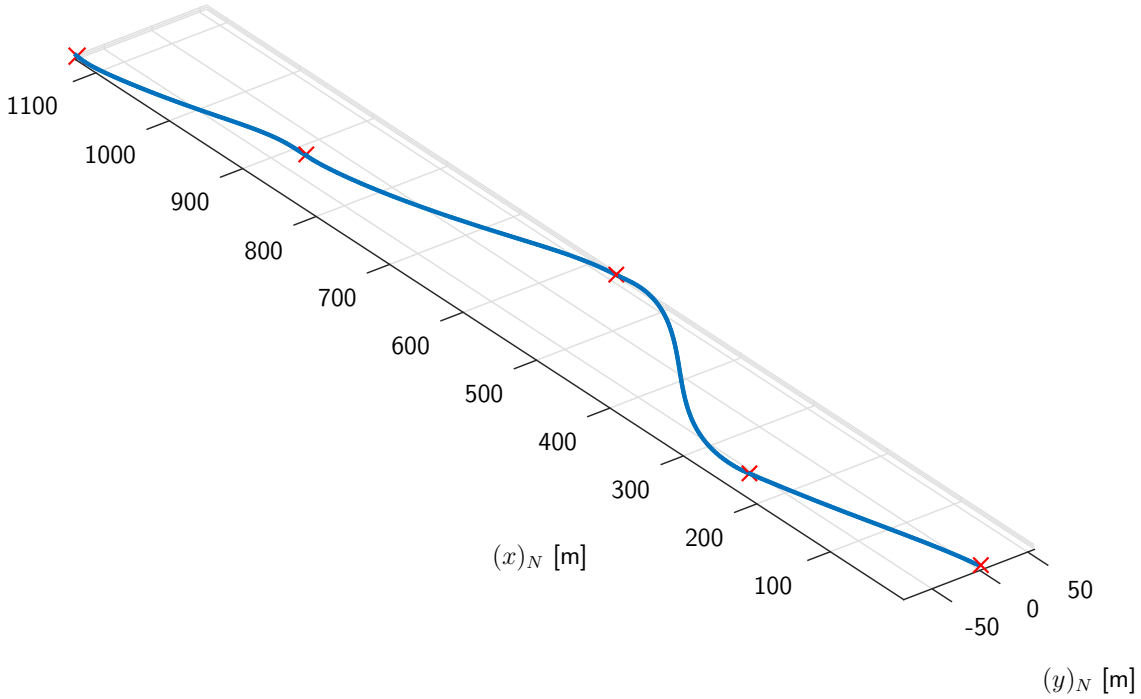

Figure 11.47: *Optimal spatial trajectory through the race course.*

The topmost line represents the refined grid that has been calculated based on the solution discretized on the regular grid with 84 nodes.

In figure 11.50 a detail of the control histories around the second last race gate is plotted. Again, the grid points are visualized in the top plot. It becomes visible how the refinement algorithm places nodes in the area where control action is required for fulfilling the race gate constraints. Moreover, especially the control histories in $\alpha_{A,CMD}$ and $\mu_{A,CMD}$ show how the coarser, regular grids with 84 and 204 nodes are not dense enough to represent the optimal control history accurately. On the other side, the refined grid, which features many control points in this critical area, matches the finest possible grid (with 404 nodes) closely.

The dynamic control constraint – that is required in the model formulation used here – can be seen in figure 11.51 for the refined grid. Again, the coarse grid in between the race gates and the fine grid at the race gates become visible in the plot. Similar to the control histories, the regions in between two race gates are not critical by means of the constraints,

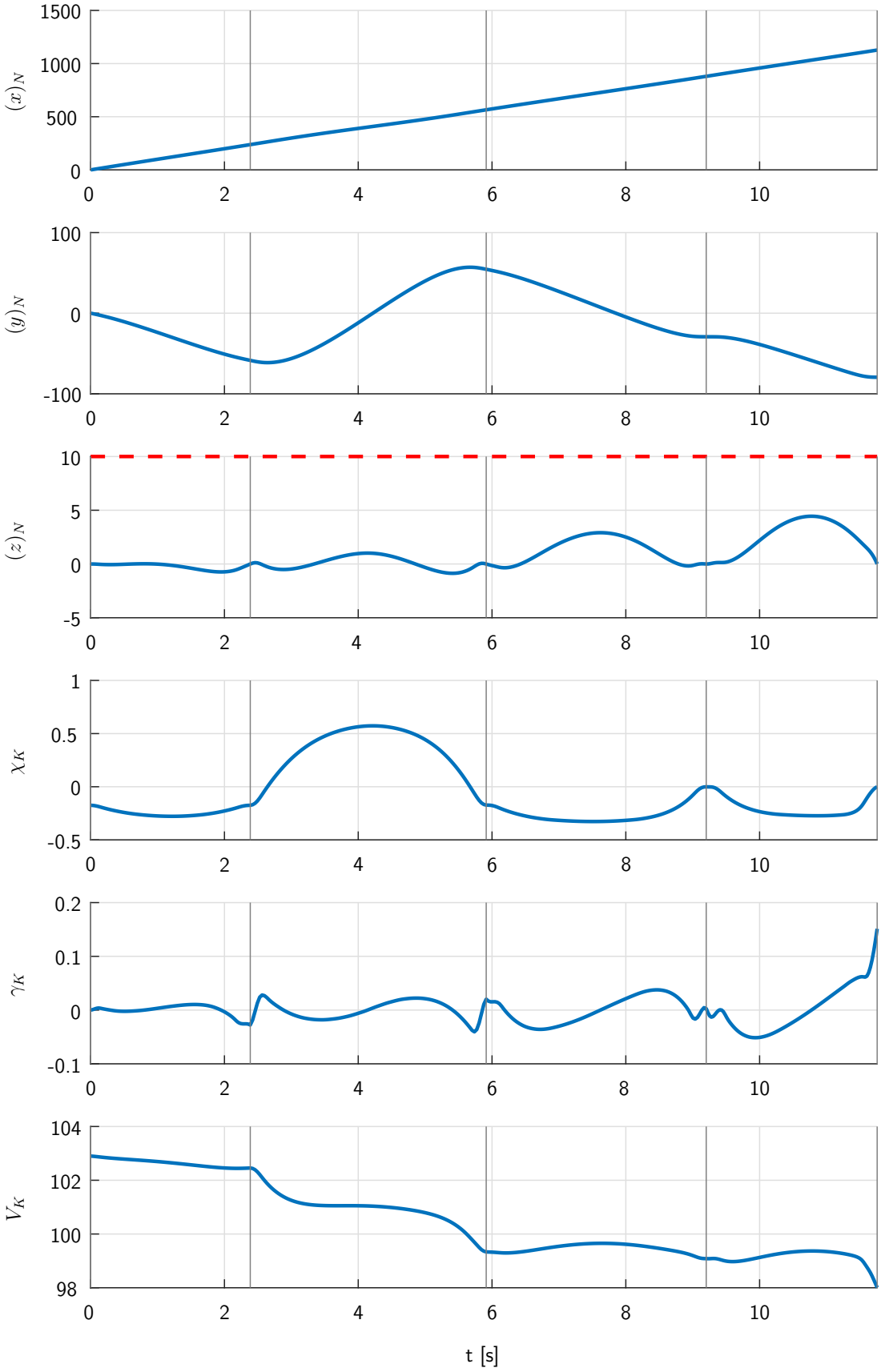

Figure 11.48: *Optimal state histories for the grid refinement example.*

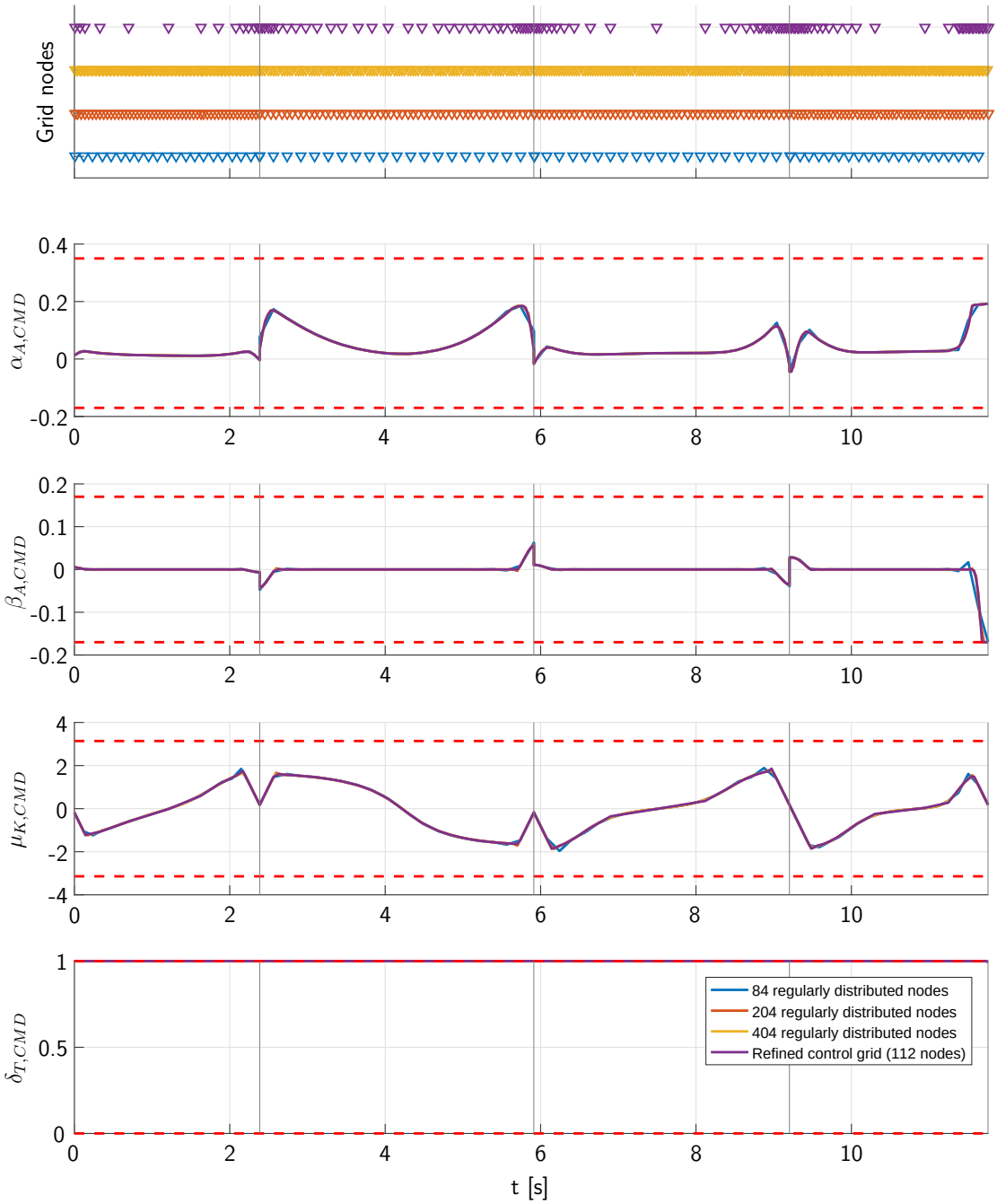

Figure 11.49: *Optimal control histories for the different control grids in the grid refinement example.*

not requiring a finer discretization there. Consequently, the dynamic constraint can also be evaluated on the refined grid with a high accuracy, where required.

In figure 11.52, the *density base function* $b(t)$ (yellow), the *density function* $(b(t))^e$ (orange, the exponent was found to be $e = 0.925$ in the example), and the resulting *distribution function* $F(t)$ (green) over the real simulation time t can be seen. More details on the functions can be found in section 9.2 and in equations (9.4) and (9.8). In the example, the density and the distribution function have been interpolated onto a finer grid in order to achieve a better numerical accuracy. As a consequence, not all grid points resulting from the refinement process

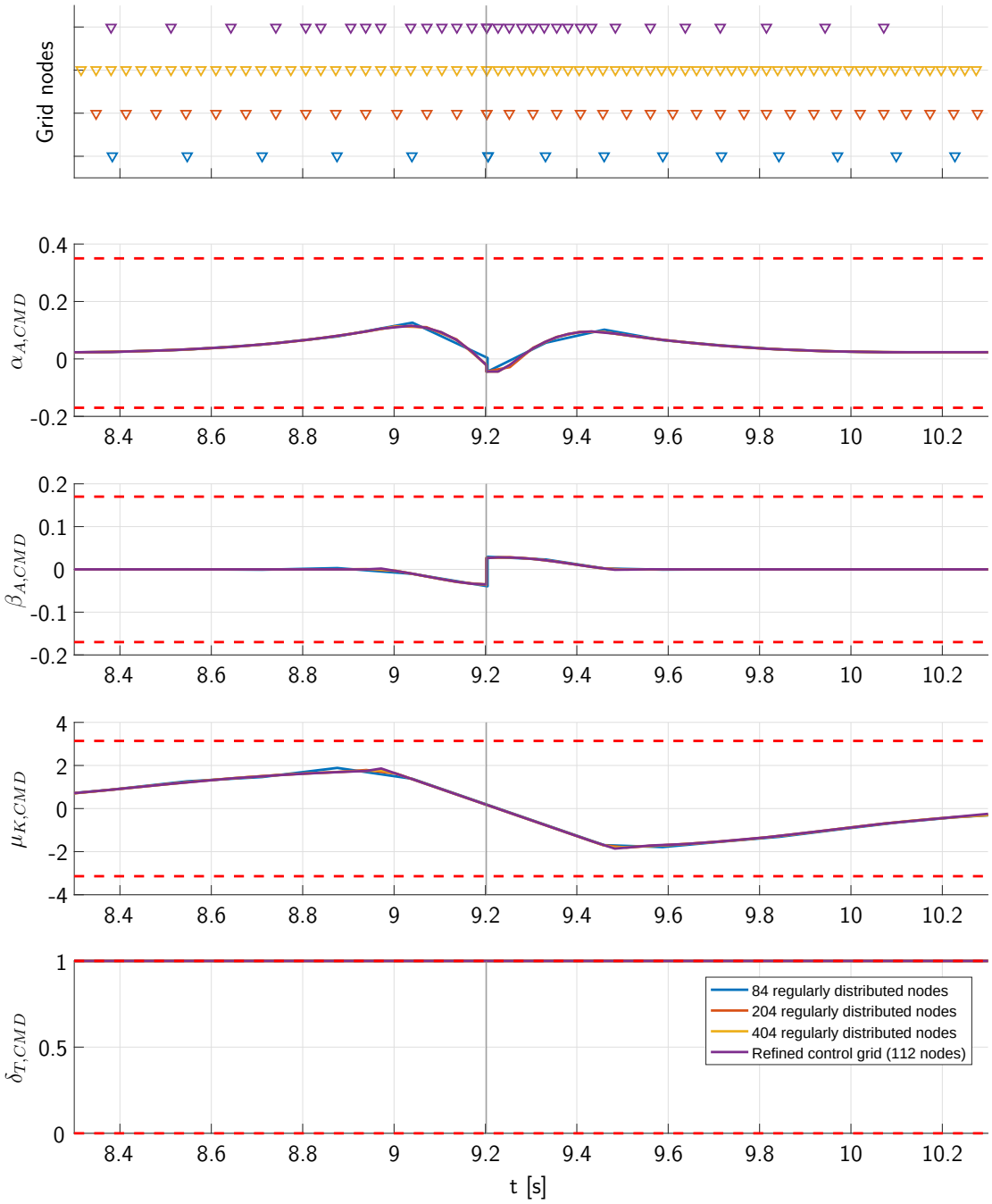

Figure 11.50: *Detail of the optimal control histories for the different control grids in the grid refinement example.*

can be used in the optimal control problem, as they may be located in between two state grid points. Hence, the diagram at the top of figure 11.52 shows the theoretically resulting grid, while the bottom diagram shows the result after matching it to the state grid nodes. As expected, a high density function results in a dense grid.

The minimum allowed grid distance in the refinement algorithm has been chosen to be equal to the state grid spacing while the maximum was set to be $\Delta\tau_{max} = 0.8s$, resulting in a grid containing 112 nodes. Anyway, due to the required rounding of N to the next integer, the maximum grid spacing in the result is reduced to $\Delta\tau_{max} = 0.64$.

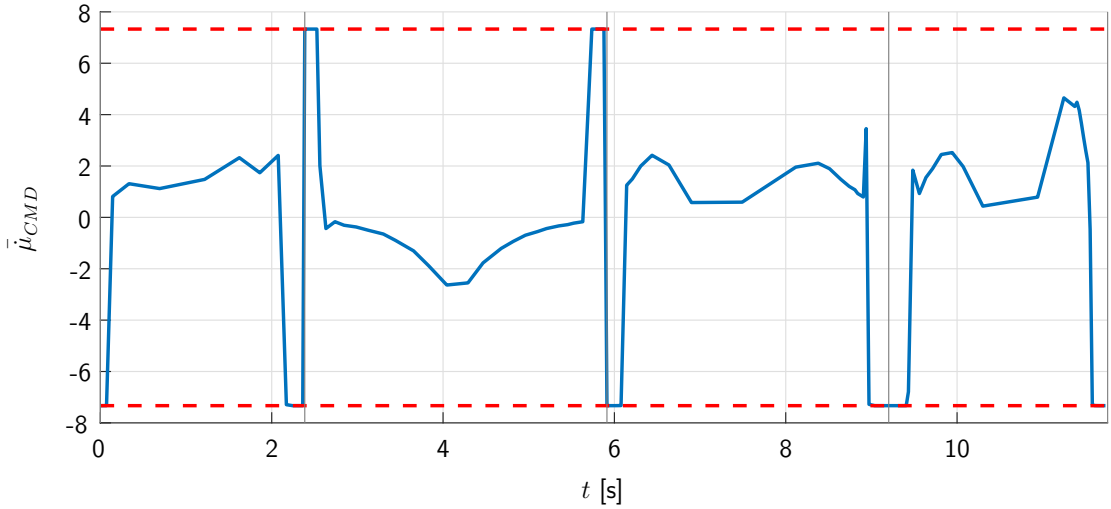

Figure 11.51: *Dynamic control constraint for the grid refinement example.*

The different achievable minimum cost function values for the different control grids are listed in table 11.16. It can be seen how the reachable cost function value – which may be seen as an indicator for the real optimality of the solution – drops with an increasing number of grid points as the authority of the controls also increases. The reachable cost function value for the refined grid is slightly larger than the one for the full grid containing 404 nodes, but still lower than the result for a control grid with 204 nodes. In the example presented here, no convergence could be achieved for a regular grid with less than 84 control nodes, as the maneuvers required to meet the constraints while passing the race gates cannot be performed with less control grid points.

Table 11.16: *Results for the grid refinement example.*

Grid	Nodes	Cost function value
Regularly distributed grid	84	$11.75971s$
Regularly distributed grid	204	$11.75779s$
Regularly distributed grid	404	$11.75753s$
Refined grid	112	$11.75757s$

In the paper [Bit+13a] an earlier version of the algorithm used here and the example shown here has been presented, using a multiple shooting approach and *SNOPT* [GMS02] for the optimizations. Therein, the results of the presented method have also been compared to the classical *DENMRA* approach, showing superior performance.

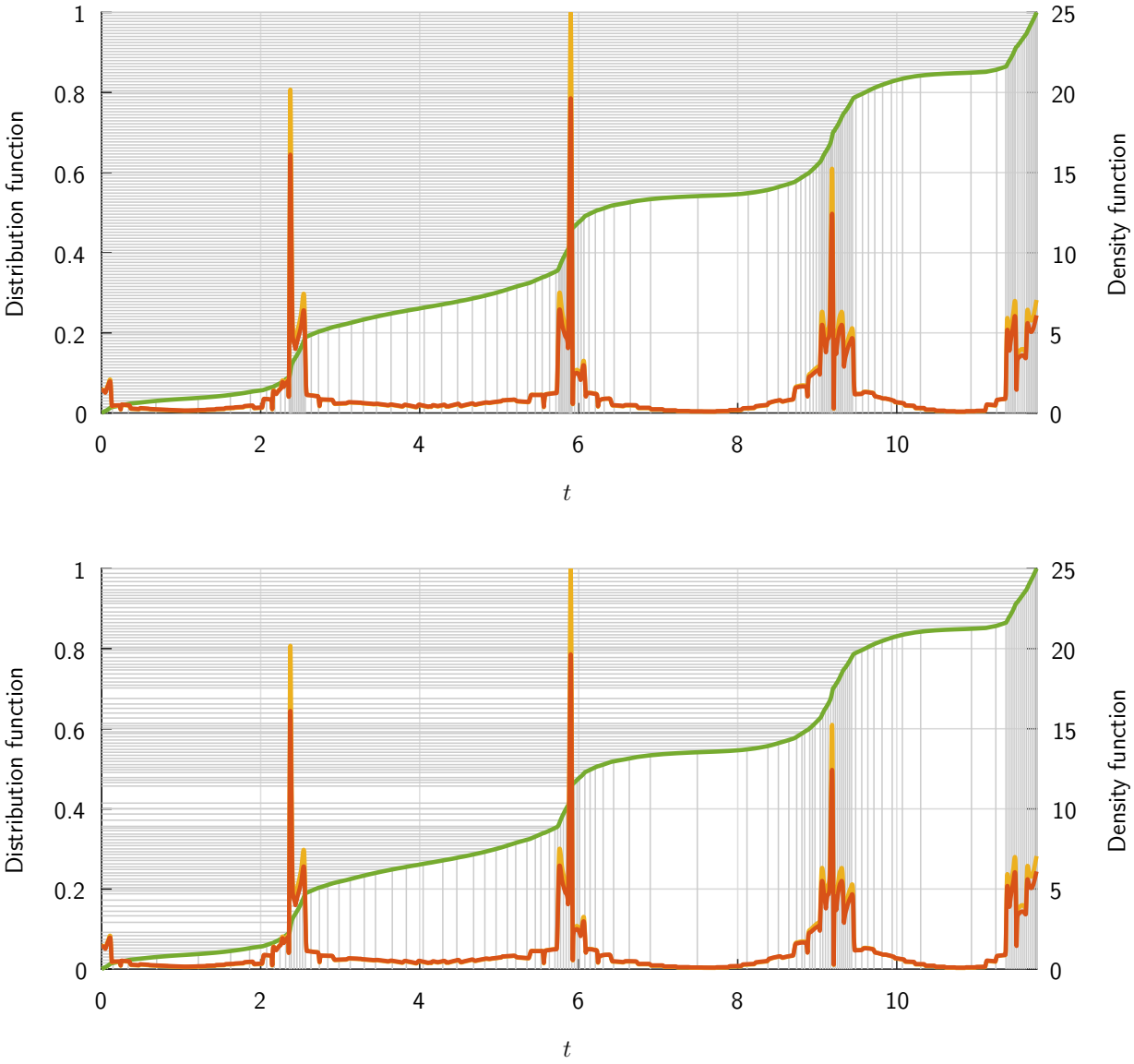

Figure 11.52: *Distribution and density function for the grid refinement example. Green line: distribution function F, yellow line: density base function b, orange line: density function b^e. The top diagram shows the directly calculated grid based on the interpolation (134 nodes) while the bottom graph shows the grid matched to the state grid (112 nodes).*

Chapter 12

Summary and Perspective

In the work at hand, different approaches that facilitate the solution of large scale, complex, and especially realistic optimal control problems are presented. This requires on the one hand adaptations of the implementation of the mathematical theory to the applications, and on the other hand the formulation of the problems to be especially tailored for optimal control. In the following, the presented approaches are recapitulated and extended by some outlooks on potential future work.

After an introduction in chapter 1, chapter 2 describes the aircraft simulation models used in the aircraft trajectory optimization examples that are presented in chapters 10 and 11. Motivated by the physical causal chain, the dynamics of the movement of an aircraft in three dimensional space, based on a point mass model and a rigid body model, is presented. The difference between the two model formulations, also resulting from the inherent timescales, has a strong impact on the properties of the solution process of respective optimal control problems. The effects are analyzed in more detail in chapters 6, 7, and 8. Moreover, ideas for extensions of the dynamic model, like actuator dynamics, are presented. Anyway, they are not fully incorporated in the exemplary models used for the optimization tasks, leaving room for future work.

The introduction to optimization and optimal control theory in chapter 3 gives a short, yet comprehensive overview of the relationships between numerical optimization and discretized optimal control. Whenever shooting methods are used for the discretization and are combined with a gradient based optimization algorithm, sensitivity equations are one possibility for the analytic calculation of the required gradient information. In section 3.2.4 it is shown that the analytic differentiation of a general RUNGE-KUTTA integration scheme leads to the same results as the numeric integration of the sensitivity equations, if the gradient of the state equation is evaluated at the same intermediate stages as the state equation itself. The relations between the LAGRANGE multipliers of the numerical optimization and the costates of the optimal control problem, that are derived in section 3.2.5, enable a thorough analysis of the results of optimal control problems. The general formulation derived is further simplified and applied to a EULER forward and a trapezoidal collocation scheme. However, a direct application to other schemes needs to be done in further research. The examples in sections

11.2 and 11.3 show results for the aforementioned analysis. The last section of chapter 3 focuses on multi criteria optimization and optimal control, where the example in section 10.4 demonstrates how a fair distribution of the cost created by a conflict resolution within an airspace sector can be achieved.

FALCON.m, the optimal control tool for *MATLAB* that is described in chapter 4 allows for the solution of many different optimal control problems without the need to re-implement the underlying discretization code over and over again. The software architecture developed in this work enables the easy use of many different combinations of parameters, cost and constraint functions in a large variety of optimal control problems.

In chapter 5, methods for the generation of initial guesses are considered that improve the convergence robustness of discretized optimal control problems. After a short repetition of the basic idea of *homotopy*, a specially tailored *homotopy* approach for optimal control problems containing multiple systems is presented. The example in section 10.2 applies this method to an approach optimization at *Tokyo International Airport*, incorporating 18 aircraft. In the next part of section 5.1, the sensitivity information, calculated based on the theory from section 3.1.6, is used to enhance the *homotopy* idea for continuously varying parameters. The results of the example in section 11.6 show a slight decrease in computational time for the solution of an exemplary air race trajectory optimization problem when using the enhanced *homotopy* scheme. However, at some of the *homotopy* steps performed, the sensitivity update does not lead to an improvement. Further research is required in order to be able to make more precise predictions about the influence of the method on the solution process. In section 5.2, an extended *RTS* smoother is used to calculate an approximation of the optimal trajectory for a rigid body simulation model based on the optimal state and control histories for a point mass simulation model. In the example in section 11.7, the solution time for the high fidelity optimal control problem is reduced by approximately 44%. However, in this example, no reduction in *overall* solution time can be achieved as the calculation of the optimal trajectory for the simplified model also consumes time. Anyway, the trajectory resulting from the smoother shows very good accordance with the final optimal trajectory for the rigid body simulation model. In some applications this may enable the use of the state and control histories resulting from the smoother directly and without any high fidelity optimization. This way, high fidelity realtime applications may be more feasible. Again, further research may show the applicability of this idea to real problems.

In chapter 6, the different model fidelities of an aircraft simulation model and some constraints are analyzed from an optimal control point of view. The analysis shows that for the particular models presented before, in some cases the high fidelity model shows preferable properties compared to the low fidelity model. However, based on the book of BETTS [Bet09], a model reformulation can be found that is at least partially removing the inherent drawbacks. The optimal control problems solved in sections 11.2 and 11.3 illustrate that the solution time of an optimal control problem based on a (not obviously) malformed but simple simulation model may be significantly larger than that of a problem based on a far more complex simula-

tion model. In the example for the simple point mass model, the solution time can be reduced by 84% when using the suggested model reformulation. The chapter on aircraft modeling for optimal control is concluded by a novel method of smooth data representation using freely selected base functions that are blended by a hyperbolic tangent. The method is used in the examples of chapter 11.

The exploitation of problem sparsity is of utmost importance in the examples presented here, because the numerical optimization algorithms used to solve the resulting optimization problems are all tailored to sparse formulations. Chapter 7 gives an overview of the different components of an optimal control problem that influence the sparsity of the *Jacobian* and the *Hessian*. Besides local effects originating from the model dynamics, the control discretization, and the path constraints, more global effects of the discretization schemes are discussed. When considering problems incorporating multiple dynamic systems, sparsity becomes even more relevant. In the example in section 10.2, the sparsity ratio of the *Jacobian* of the discretized problem is 99.9938%, this means more vividly that in average only 62 out of $1,000,000$ possible entries of the *Jacobian* are non-zero. Besides, in all other examples, problem sparsity is also exploited automatically by the algorithms included in *FALCON.m*.

In chapter 8, a novel transcription method for optimal control problems is introduced that combines multiple shooting and collocation. The method is used in an example in section 11.4 for a rigid body simulation model. Here, the model dynamic equations are divided into fast and slow varying groups. The fast dynamics are integrated using multiple shooting, while the slow group is discretized by a direct collocation method on the grid of the multiple shooting nodes. In the example, the dimensions of the optimization parameter vector and the constraint vector are reduced by approximately 80%. The solution time is decreased by roughly 7% without a significant deterioration of the solution accuracy. The large decrease in problem size is mainly related to the smaller number of required grid nodes, as the fast and the slow dynamics are not discretized on the same grid anymore.

An enhanced control grid refinement method is presented in chapter 9. After a short introduction of commonly used approaches, one that is based on the concept of density functions, is adapted to the physical properties of a point mass simulation model. Besides, an extension is presented that uses a small optimization in order to determine the ideal number of control grid points from a user defined maximum and minimum grid spacing. The example in section 11.8 demonstrates that the control grid refinement allows for a reduction of control nodes from 404 to 84 without a significant decrease in solution accuracy. As expected, the resulting numerical optimization problem shrinks in size, allowing for a faster and more robust solution. It this point, the adaptation of the method to other models leaves room for further research.

Chapter 10 collects all examples that are related to air traffic management. First, the simultaneous optimization of approach trajectories for 18 aircraft in the vicinity of *Tokyo International Airport* is considered, where a combination of fuel and time minimal trajectories is determined. Next, the calculation of trajectories for civil airliners that are refueled during flight in order to reduce overall fuel consumption is presented, allowing for a reduction in fuel burn of

up to 14% when redesigning the carrier aircraft for the refueling mission. Otherwise, savings of up to 7% are still possible in the simulations. The last example in the chapter focuses on the aforementioned multi criteria optimization considering fairness and shows approximations for the different PARETO optimal points of the problems.

All examples related to air race trajectories are comprised in chapter 11. Here, the influence of the different model fidelities is investigated, before the combined shooting and collocation discretization is applied. In section 11.6, the initial guess generation based on the extended *homotopy* is presented, while in the next section the initial guess generation based on the *RTS* smoother is demonstrated. The final example shows the physically motivated control grid refinement. The main results of these examples are the methodological achievements, illustrated mainly by means of overall feasibility, reduced convergence time, or smaller number of iterations. More details on the particular optimizations performed in the examples have already been recapitulated in relation to the respective theoretical contributions of this work. Moreover, one of the main results derived from the examples and their analysis is that there are cases where a high fidelity optimization converges better and faster than one using a more simplified model, if the model formulation is not chosen appropriately.

Overall, the methods presented in this work improve the solution process for large scale, complex, and real world optimal control problems. However, none of the methods currently available – may it be presented here or elsewhere – does work *always* and for *every problem*. In general, a larger set of tools available to the user improves the chances of solving a particular optimal control problem. From this perspective, the contributions of this thesis can be seen as additions to and extensions of the *global toolset* for large and complex optimal control problems.

Appendix A

Coordinate Frames

The coordinate frames used here, can e.g. be found in [Fis11, Hol+15].

A.1 Earth-Centered Inertial (ECI), Index I

Index	I
Purpose	EUCLIDIAN frame, inertial system, system to apply NEWTON's law
Origin	Center of mass of the earth
Translation	Around the sun, with the solar system
Rotation	None
x-axis	In the equatorial plane, pointing to the vernal equinox
y-axis	In the equatorial plane, forming a right hand system with the x and z-axis
z-axis	Earth's rotational axis

A.2 Earth-Centered Earth Fixed Frame (ECEF), Index E

Index	E
Purpose	Frame for navigation and positioning
Origin	Center of mass of the earth
Translation	Moves with the *ECI* Frame
Rotation	Earth rotation around z-axis with earth's rotational rate $\boldsymbol{\omega}^{IE}$, i.e. approximately $\frac{2\pi}{24h}$
x-axis	In the equatorial plane, pointing through the *Greenwich* meridian
y-axis	In the equatorial plane to form a right-hand system with the x and z-axis
z-axis	Rotation axis of earth, identical to z-axis of the *ECI* system

A.3 North-East-Down Frame (NED), Index O

Index	O
Purpose	Frame for describing the aircraft's orientation, rotation, and velocity
Origin	Aircraft's reference point R
Translation	Moves with the aircraft's reference point
Rotation	Rotates with the transport rate of the earth $\boldsymbol{\omega}^{EO}$ in order to keep the *NED* alignment
x-axis	Parallel to the local geoid surface, pointing to the geographic north pole
y-axis	Parallel to the local geoid surface, pointing to the east in order to form a right hand system with the x and z-axis
z-axis	Pointing downwards, perpendicular to the local geoid surface

A.4 Navigational Frame, Index N

Index	N
Purpose	Local navigational frame, derived from the *NED* frame, but fixed to the earth's surface, used for local positioning
Origin	Point on the earth's surface
Translation	Moves with the point on the earth's surface
Rotation	Rotation of the earth
x-axis	Parallel to the local geoid surface, pointing to the geographic north pole
y-axis	Parallel to the local geoid surface, pointing to the east in order to form a right hand system with the x and z-axis
z-axis	Pointing downwards, perpendicular to the local geoid surface

A.5 Aerodynamic Frame, Index A

Index	A
Purpose	Frame for notating aerodynamic quantities
Origin	Aircraft's reference point R
Translation	Moves with the aircraft's reference point
Rotation	Rotates with the direction of the airflow
x-axis	Aligned with aerodynamic velocity (with respect to the E-Frame, pointing into the direction of the aerodynamic velocity)
y-axis	Pointing to the right to form a right hand system with the x and z-axis
z-axis	Pointing downwards in the symmetry plane of the aircraft, perpendicular to the x-axis

A.6 Rotated Aerodynamic Frame, Index $\bar{A}$

Index	$\bar{A}$
Purpose	Intermediate notation frame
Origin	Aircraft's reference point R
Translation	Moves with the aircraft's reference point
Rotation	Rotates with the direction of the airflow
x-axis	Aligned with aerodynamic velocity, pointing into the direction of the aerodynamic velocity
y-axis	Pointing to the right to form a right hand system with the x and z-axis
z-axis	z-axis of the aerodynamic frame A rotated counterclockwise by the aerodynamic flight-path bank angle μ_A

A.7 Kinematic Frame, Index K

Index	K
Purpose	Frame for notating the flight path
Origin	Aircraft's reference point R
Translation	Moves with the aircraft's reference point
Rotation	Rotates with direction of kinematic aircraft motion (w.r.t. the E-Frame)
x-axis	Aligned with the kinematic velocity, pointing into the direction of the kinematic velocity
y-axis	Pointing to the right in order to form a right hand system with the x and z-axis
z-axis	Pointing downwards, parallel to the projection of the local surface normal of the *WGS-84* ellipsoid into a plane perpendicular to the x-axis

A.8 Rotated Kinematic Frame, Index $\bar{K}$

Index	$\bar{K}$
Purpose	Intermediate notation frame
Origin	Aircraft's reference point R
Translation	Moves with the aircraft's reference point
Rotation	Rotates with direction of kinematic aircraft motion
x-axis	Aligned with the kinematic velocity, pointing into the direction of the kinematic velocity
y-axis	Pointing to the right in order to form a right hand system with the x and z-axis
z-axis	z-axis of the Kinematic Frame K rotated clockwise by the kinematic flight-path bank angle μ_K

A.9 Body Fixed Frame, Index B

Index	B
Purpose	Frame for notating frames and moments
Origin	Aircraft's reference point R
Translation	Moves with the aircraft's reference point
Rotation	Rotates with the aircraft
x-axis	Pointing towards the nose of the aircraft, in the aircraft's symmetry plane
y-axis	Pointing towards the right wing to form a right hand system with the x and z-axis
z-axis	Pointing downwards in the symmetry plane of the aircraft

Appendix B

Scientific Publications

During the period at the Institute of Flight System Dynamics at TUM, the following scientific publications have been published by this author. Besides, the author contributed to the notes for the lecture *Aircraft Trajectory Optimization* [Hol15] and to the notes for the lab course *Optimal Control* [Hol+14].

- M. Bittner, M. Rieck, B. Grüter, and F. Holzapfel. *"Optimal Approach Trajectories for Multiple Aircraft Considering Disturbances and Configuration Changes."* In: *30th International Congress of the International Council of the Aeronautical Sciences, ICAS.* Daejeon, South Korea. 2016.
- B. Grüter, M. Bittner, M. Rieck, J. Diepolder, and F. Holzapfel. *"Optimal Sequencing in ATM Combining Genetic Algorithms and Gradient Based Methods to a Bilevel Approach."* In: *30th International Congress of the International Council of the Aeronautical Sciences, ICAS.* Daejeon, South Korea. 2016.
- C. Göttlicher, M. Gnoth, M. Bittner, and F. Holzapfel. *"Aircraft Parameter Estimation Using Optimal Control Methods."* In: *AIAA Atmospheric Flight Mechanics Conference.* San Diego, California, USA. 2016.
- M. Bittner, M. Rieck, B. Grüter, and F. Holzapfel. *"Optimal Conflict Free Approach Trajectories for Multiple Aircraft."* In: *ENRI Int. Workshop on ATM/CNS, EIWAC.* Tokyo, Japan. 2015.
- B. Grüter, M. Bittner, M. Rieck, F. Holzapfel, and A. Harada. *"Allocation, Sequencing and Trajectory for Aircraft Using Superimposed Navigation and Optimal Control."* In: *ENRI Int. Workshop on ATM/CNS, EIWAC.* Tokyo, Japan. 2015.
- M. Rieck, M. Bittner, B. Grüter, and F. Holzapfel. *"Generation of dynamic models with automatically generated derivatives for ATM optimal control in MATLAB."* In: *ENRI Int. Workshop on ATM/CNS, EIWAC.* Tokyo, Japan. 2015.
- M. Bittner, M. Buhl, B. Lohmann. *"Zustandsschätzung nach Kalman auf einem bewegten Horizont variabler Länge."* In: *Methoden und Anwendungen der Regelungstechnik. Erlangen-Münchener Workshops 2013 und 2014.* Ed. by B. Lohmann, G. Roppe-

necker. 1. Aufl. Berichte aus der Steuerungs- und Regelungstechnik. Herzogenrath: Shaker, 2015. ISBN: 978-3-8440-3797-5.

- M. Rieck, M. Richter, M. Bittner and F. Holzapfel, *"Optimal trajectories for RPAS with discrete controls and discrete constraints."* In: *2014 IEEE International Conference on Aerospace Electronics and Remote Sensing Technology.* Yogyakarta, Indonesia. 2014. pp. 34–38. ISBN: 978-1-4799-6187-0.
- M. Rieck, M. Richter, M. Bittner, and F. Holzapfel. *"Generation of Initial Guesses for Optimal Control Problems with Mixed Integer Dependent Constraints."* In: *29th International Congress of the International Council of the Aeronautical Sciences, ICAS.* St. Petersburg, Russia. 2014.
- M. Richter, M. Bittner, M. Rieck, and F. Holzapfel. *"A Non-Cooperative Bi-Level Optimal Control Problem Formulation for Noise Minimal Departure Trajectories."* In: *29th International Congress of the International Council of the Aeronautical Sciences, ICAS.* St. Petersburg, Russia. 2014.
- M. Richter, M. Bittner, M. Rieck, and F. Holzapfel. *"A Realistic Flight Path Parameterization for Calculation of Noise Minimal Trajectories using Bi-level Optimal Control."* In: *AIAA GNC and Co- located Conferences.* National Harbor, Maryland, USA. 2014.
- M. Richter, M. Hochstrasser, M. Bittner, L.Walter, and F. Holzapfel. *"Application of MINLP Techniques to Conflict Resolution of Multiple Aircraft."* In: *AIAA GNC and Co-located Conferences.* National Harbor, Maryland, USA. 2014.
- F. Fisch, M. Bittner and F. Holzapfel, *"Optimal Scheduling of Fuel-Minimal Approach Trajectories."* In: *Journal of Aerospace Operations.* 2014.
- M. Bittner, B. Fleischmann, M. Richter, and F. Holzapfel. *"Optimization of ATM Scenarios Considering Overall and Single Costs."* In: *6th International Conference on Research in Air Transportation, ICRAT.* Istanbul, Turkey. 2014.
- M. Richter, M. Bittner, M. Rieck, and F. Holzapfel. *"Berechnung lärmminimaler An- und Abflugrouten."* In: *Matlab Expo 2013.* Munich, Germany. 2013.
- M. Richter, M. Bittner, and F. Holzapfel. *"Noise Minimal Approaches on Parallel Runways."* In: *4th International Air Transport and Operations Symposium, ATOS.* Toulouse, France. 2013.
- M. Bittner, C. Hornfeck, F. Fisch, M. Schwarze, and F. Holzapfel. *"Optimization of Mid to Long Range Flights Considering Air-to-Air Refueling for Fuel Savings."* In: *5th European Conference for Aeronautics and Space Sciences, EUCASS 2013.* Munich, Germany. 2013.
- M. Bittner, P. Bruhs, M. Richter, F. Holzapfel, *"An Automatic Mesh Refinement Method for Aircraft Trajectory Optimization Problems."* In: *AIAA GNC and Co-Located Conferences.* Boston, Massachusetts, USA. 2013.

- F. Fisch, M. Bittner, and F. Holzapfel. *"Optimal Scheduling of Fuel-Minimal Approach Trajectories."* In: *3rd International Air Transport and Operations Symposium, ATOS.* Delft, the Netherlands. 2012.
- M. Bittner, F. Fisch and F. Holzapfel, *"A Multi-model Gauss Pseudospectral Optimization Method for Aircraft Trajectories."* In: *AIAA Atmospheric Flight Mechanics Conference.* Minneapolis, Minnesota, USA. 2012.

Bibliography

[Air15] Airbus S.A.S. *A380, Aircraft Characteristics - Airport and Maintenance Planning*. Version Mar 30/05, Dec 01/15. Airbus S.A.S., Customer Services, 2015. URL: http://www.airbus.com/fileadmin/media_gallery/files/tech_data/AC/Airbus-AC-A380-Jan16.pdf.

[Alb13] S. Albrecht. "Modeling and numerical solution of inverse optimal control problems for the analysis of human motions." Dissertation. München: Technische Universität München, 2013.

[And11] J. D. Anderson. *Fundamentals of Aerodynamics*. Fifth edition. McGraw-Hill Series in Aeronautical and Aerospace Engineering. New York: McGraw-Hill, 2011. ISBN: 978-0-07-339810-5.

[Arc+08] J. K. Archibald, J. C. Hill, N. A. Jepsen, W. C. Stirling, and R. L. Frost. "A Satisficing Approach to Aircraft Conflict Resolution." In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* Vol. 38. No. 4 (2008), pp. 510–521.

[Arm66] L. Armijo. "Minimization of functions having Lipschitz continuous first partial derivatives." In: *Pacific Journal of Mathematics* Vol. 16. No. 1 (1966), pp. 1–3.

[Art] Artelys. *Artelys Knitro. Artelys Optimization Solution*. URL: http://www.artelys.com/en/optimization-tools/knitro (visited on 12/16/2015).

[Ast16] Astos Solutions GmbH. *GESOP - Astos Solutions*. 2016. URL: https://www.astos.de/products/gesop (visited on 04/21/2016).

[BAL11] R. Brockhaus, W. Alles, and R. Luckner. *Flugregelung*. 3., neu bearb. Aufl. Heidelberg, Dordrecht, London, New York: Springer, 2011. ISBN: 978-3-642-01442-0.

[BE03] J. T. Betts and S. O. Erb. "Optimal Low Thrust Trajectories to the Moon." In: *SIAM Journal on Applied Dynamical Systems* Vol. 2. No. 2 (2003), pp. 144–170. ISSN: 1536-0040.

[BE12] F. A. Bukhari and A. El-Gohary. "Optimal control of a production-maintenance system with deteriorating items." In: *Journal of King Saud University - Science* Vol. 24. No. 4 (2012), pp. 351–357. ISSN: 10183647.

[Bec11] V. M. Becerra. *PSOPT Optimal Control Solver. User Manual, Release 3*. 2011. URL: https://psopt.googlecode.com/files/PSOPT_Manual_R3.pdf (visited on 04/21/2016).

[Ben10] J. Z. Ben-Asher. *Optimal Control Theory with Aerospace Applications*. AIAA Education Series. Reston, VA, USA: American Institute of Aeronautics and Astronautics, 2010. ISBN: 978-1600867323.

[Bet09] J. T. Betts. *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. Second edition. Advances in Design and Control. Philadelphia: SIAM, Society for Industrial and Applied Mathematics, 2009.

[BFH12] M. Bittner, F. Fisch, and F. Holzapfel. "A Multi-Model Gauss Pseudospectral Optimization Method for Aircraft Trajectories." In: *AIAA Atmospheric Flight Mechanics Conference*. (Minneapolis, Minnesota, USA, Aug. 13–16, 2012). AIAA, American Institute of Aeronautics and Astronautics. 2012.

[BH75] A. E. Bryson and Y.-C. Ho. *Applied Optimal Control. Optimization, Estimation, and Control*. Revised Printing. Washington and New York: Hemisphere, 1975. ISBN: 978-0891162285.

[Bie10] L. T. Biegler. *Nonlinear Programming*. SIAM, Society for Industrial and Applied Mathematics, 2010. ISBN: 978-0-89871-702-0.

[Bin+00] T. Binder, A. Cruse, C. A. Cruz Villar, and W. Marquardt. "Dynamic Optimization Using a Wavelet Based Adaptive Control Vector Parameterization Strategy." In: *Computers and Chemical Engineering* Vol. 24 (2000), pp. 1201–1207.

[Bit+13a] M. Bittner, P. Bruhs, M. Richter, and F. Holzapfel. "An Automatic Mesh Refinement Method for Aircraft Trajectory Optimization Problems." In: *Guidance, Navigation, and Control and Co-located Conferences*. (Boston, Massachusetts, Aug. 19–22, 2013). AIAA, American Institute of Aeronautics and Astronautics. 2013.

[Bit+13b] M. Bittner, C. Hornfeck, F. Fisch, M. Schwarze, and F. Holzapfel. "Optimization of Mid to Long Range Flights Considering Air-to-Air Refueling for Fuel Savings." In: *EUCASS 2013, 5th European Conference for Aeronautics and Space Sciences*. (Munich, Germany, July 1–5, 2013). 2013.

[Bit+14] M. Bittner, B. Fleischmann, M. Richter, and F. Holzapfel. "Optimization of ATM Scenarios Considering Overall and Single Costs." In: *6th International Conference on Research in Air Transportation (ICRAT)*. (Istanbul, Turkey, May 26–30, 2014). 2014.

[Bit+15] M. Bittner, M. Rieck, B. Grüter, and F. Holzapfel. "Optimal Conflict Free Approach Trajectories for Multiple Aircraft." In: *The 4th ENRI International Workshop on ATM/CNS (EIWAC2015)*. (Tokyo, Japan, Nov. 17–19, 2015). ENRI. 2015.

[Bit+16] M. Bittner, M. Rieck, B. Grüter, and F. Holzapfel. "Optimal Approach Trajectories for Multiple Aircraft Considering Disturbances and Configuration Changes." In: *30th Congress of the International Council of the Aeronautical Sciences (ICAS)*. (Daejeon, Korea, Sept. 25–30, 2016). The International Council of the Aeronautical Sciences. 2016.

[BLM12] C. L. Bottasso, F. Luraghi, and G. Maisano. "Efficient rotorcraft trajectory optimization using comprehensive models by improved shooting methods." In: *Aerospace Science and Technology* Vol. 23. No. 1 (2012), pp. 34–42. ISSN: 1270-9638.

[BM09] C. L. Bottasso and G. Maisano. "Efficient Rotorcraft Trajectory Optimization using Comprehensive Vehicle Models by Improved Shooting Methods." In: *35th European Rotorcraft Forum Proceedings*. (Hamburg, Germany, Sept. 22–25, 2009). Ed. by I. Lopez and P. Brandt. Red Hook, NY: Curran Associates, 2009. ISBN: 9781615678747.

[BMP91] R. Bulirsch, F. Montrone, and H. J. Pesch. "Abort landing in the presence of windshear as a minimax optimal control problem, part 2: Multiple shooting and homotopy." In: *Journal of Optimization Theory and Applications* Vol. 70. No. 2 (1991), pp. 223–254.

[BMS10] C. L. Bottasso, G. Maisano, and F. Scorlcelletti. "Trajectory Optimization Procedures for Rotorcraft Vehicles, Their Software Implementation, and Applicability to Models of Increasing Complexity." In: *Journal of the American Helicopter Society* Vol. 55. No. 032010 (2010).

[BMT08] F. Bonnans, P. Martinon, and E. Trélat. "Singular Arcs in the Generalized Goddard's Problem." In: *Journal of Optimization Theory and Applications* Vol. 139. No. 2 (2008), pp. 439–461.

[Bra68] D. H. Brandin. "Mathematics of continuous system simulations." In: *AFIPS '68 (Fall, part I) Proceedings of the December 9-11, 1968, fall joint computer conference, part I*. (San Francisco, California, Dec. 9–11, 1968). New York, New York, USA: ACM Press, 1968, pp. 345–352.

[Büs96] C. Büskens. *Lösung Optimaler Steuerprozesse, Lösung adjungierter Variablen, Automatische Gitteranpassung. Anleitung zur Benutzung der Fortran Bibliothek NUDOCCS, Version 8.04*. Münster: Westfälsiche Wilhelms-Universität, 1996. URL: http://wwwmath.uni-muenster.de/num/Arbeitsgruppen/ag_maurer/

Lehrveranstaltungen/NUDOCCCS/Nudocccsanleitung.pdf (visited on 04/21/2016).

[Büs98] C. Büskens. "Optimierungsmethoden und Sensitivitätsanalyse für optimale Steuerprozesse mit Steuer- und Zustands-Beschränkungen." Fachbereich Mathematik und Informatik. Dissertation. Münster: Westfälische Wilhelms-Universität, 1998.

[BW13] C. Büskens and D. Wassel. "The ESA NLP Solver WORHP." In: *Modeling and Optimization in Space Engineering*. Ed. by G. Fasano and J. D. Pintér. Vol. 73. New York: Springer, 2013, pp. 85–110. ISBN: 978-1-4614-4468-8.

[CB07] J. L. Christensen and W. L. Brogan. "Modelling and optimal control of a production process." In: *International Journal of Systems Science* Vol. 1. No. 3 (2007), pp. 247–255. ISSN: 0020-7721.

[CG09] M. Ciarcià and C. Grillo. "Collision Avoidance Trajectory for an Ekranoplan." In: *AIAA Atmospheric Flight Mechanics Conference*. (Chicago, Illinois, Aug. 10–13, 2009). AIAA, American Institute of Aeronautics and Astronautics. 2009.

[CHL10] G. Chaloulos, P. Hokayem, and J. Lygeros. "Distributed hierarchical MPC for conflict resolution in air traffic control." In: *2010 American Control Conference (ACC2010)*. (Baltimore, Maryland, USA, June 30–July 2, 2010). 2010, pp. 3945–3950.

[CJ12] J. L. Crassidis and J. L. Junkins. *Optimal Estimation of Dynamic Systems*. 2nd ed. Vol. 24. Chapman & Hall/CRC applied mathematics & nonlinear science. Boca Raton, FL: CRC Press, 2012. ISBN: 9781439839850.

[CL98] B. A. Conway and K. M. Larson. "Collocation Versus Differential Inclusion in Direct Optimization." In: *Journal of Guidance, Control, and Dynamics* Vol. 21 (1998), pp. 780–785. ISSN: 0731-5090.

[Com+11] G. P. Compo et al. "The Twentieth Century Reanalysis Project." In: *Quarterly Journal of the Royal Meteorological Society* Vol. 137. No. 654 (2011), pp. 1–28. ISSN: 1477-870X.

[CRS11] Y. Cao, S. Rathinam, and D. Sun. "A Rescheduling Method for Conflict-free Continuous Descent Approach." In: *AIAA Guidance, Navigation, and Control Conference*. (Portland, Oregon, USA, Aug. 8–11, 2011). AIAA, American Institute of Aeronautics and Astronautics. 2011.

[DC08a] P. N. Desai and B. A. Conway. "Six-Degree-of-Freedom Trajectory Optimization Using a Two-Timescale Collocation Architecture." In: *Journal of Guidance, Control, and Dynamics* Vol. 31. No. 5 (2008), pp. 1308–1315. ISSN: 0731-5090.

[DC08b] P. N. Desai and B. A. Conway. "Two-Timescale Discretization Scheme for Collocation." In: *Journal of Guidance, Control, and Dynamics* Vol. 31. No. 5 (2008), pp. 1316–1322. ISSN: 0731-5090.

[Den01] N. Dennis. "Developments of Hubbing at European Airports." In: *Air & Space Europe* Vol. 3. No. 1-2 (2001), pp. 51–55. ISSN: 12900958.

[Deu79] Deutsches Institut für Normung, ed. *Normatmosphäre*. DIN ISO. Version 1979-12. Berlin: Beuth Verlag GmbH, 1979.

[Die+16] J. Diepolder et al. "Flight Control Law Clearance Using Worst-Case Inputs." In: *30th Congress of the International Council of the Aeronautical Sciences (ICAS)*. (Daejeon, Korea, Sept. 25–30, 2016). The International Council of the Aeronautical Sciences. 2016.

[DP06] D. DuBois and G. C. Paynter. ""Fuel Flow Method2" for Estimating Aircraft Emissions." In: *Non-Conference Specific Technical Papers - 2006*. (). Warrendale, PA, United States: SAE International, 2006.

[DP80] J. R. Dormand and P. J. Prince. "A family of embedded Runge-Kutta formulae." In: *Journal of Computational and Applied Mathematics* Vol. 6. No. 1 (1980), pp. 19–26.

[Dub57] L. E. Dubins. "On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents." In: *American Journal of Mathematics* Vol. 79. No. 3 (1957), pp. 497–516. ISSN: 00029327.

[EC92] P. J. Enright and B. A. Conway. "Discrete Approximations to Optimal Trajectories Using Direct Transcription and Nonlinear Programming." In: *Journal of Guidance, Control, and Dynamics* Vol. 15 (1992), pp. 994–1002. ISSN: 0731-5090.

[EE15] European Union and EUROCONTROL. *European ATM Master Plan. The roadmap for delivering high performing aviation for Europe*. Executive View. Edition 2015. Luxembourg: Publications Office, 2015. ISBN: 978-92-9216-034-0.

[Eli16] Elissar Global. *DIDO leading optimal control software*. 2016. URL: http://www.elissarglobal.com/academic/products/ (visited on 04/21/2016).

[ER96] B. Etkin and L. D. Reid. *Dynamics of Flight. Stability and Control*. Third Edition. New York: Wiley, 1996. ISBN: 0-471-03418-5.

[EUR10] EUROCONTROL. *Long-Term Forecasts. Flight Movements 2010 - 2030*. 2010.

[EUR12] EUROCONTROL Experimental Centre. *User Manual for the Base of Aircraft Data (BADA) Family 4*. Ed. by European Organisation for the Safety of Air Navigation. 2012.

[EUR14] EUROCONTROL. *User Manual for the Base of Aircraft Data (BADA) Revision 3.12*. EUROCONTROL Experimental Centre, 2014.

[FB11] R. Fernandes de Oliveira and C. Büskens. *On-Board Trajectory Optimization of RNAV Departure and Arrival Procedures Concerning Emissions and Population Annoyance*. SAE Technical Paper 2011-01-2595, 2011.

[FB12] R. Fernandes de Oliveira and C. Büskens. "Benefits of Optimal Flight Planning on Noise and Emissions Abatement at the Frankfurt Airport." In: *AIAA Guidance, Navigation, and Control Conference*. (Minneapolis, Minnesota, Aug. 13–16, 2012). AIAA, American Institute of Aeronautics and Astronautics. 2012. ISBN: 978-1-60086-938-9.

[FBH12] F. Fisch, M. Bittner, and F. Holzapfel. "Optimal Scheduling of Fuel-Minimal Approach Trajectories." In: *Air Transport and Operations. Proceedings of the third International Air Transport and Operations Symposium 2012*. (Delft, the Netherlands, June 12–20, 2012). Ed. by R. Curran and L. Fischer. Amsterdam and Washington, D.C.: IOS Press, 2012. ISBN: 978-1-61499-119-9.

[FBH13] F. Fisch, M. Bittner, and F. Holzapfel. "Optimal Scheduling of Fuel-Minimal Approach Trajectories." In: *Journal of Aerospace Operations* Vol. 2. No. 2–4 (2013), pp. 145–160.

[Feh69] E. Fehlberg. *Low-order classical Runge-Kutta formulas with stepsize control and their application to some heat transfer problems*. Washington, United States: NASA Marshall Space Flight Center, Huntsville, AL, United States, 1969. URL: http://ntrs.nasa.gov/search.jsp?R=19690021375 (visited on 07/26/2016).

[Fia76] A. V. Fiacco. "Sensitivity Analysis for Nonlinear Programming using Penalty Methods." In: *Mathematical Programming* Vol. 10 (1976), pp. 287–311. ISSN: 0025-5610.

[Fis+08] F. Fisch, H. Pfifer, M. Weingartner, F. Holzapfel, G. Sachs, and S. Myschik. "Airframe and Trajectory Pursuit Modeling for Simulation Assisted Air Race Planning." In: *AIAA Modeling and Simulation Technologies Conference and Exhibit*. (Honolulu, Hawaii, USA, Aug. 18–21, 2008). AIAA, American Institute of Aeronautics and Astronautics. 2008.

[Fis+09] F. Fisch, J. Lenz, F. Holzapfel, and G. Sachs. "Trajectory Optimization Applied to Air Races." In: *AIAA Atmospheric Flight Mechanics Conference*. (Chicago, Illinois, Aug. 10–13, 2009). AIAA, American Institute of Aeronautics and Astronautics. 2009.

[Fis11] F. Fisch. *Development of a Framework for the Solution of High-Fidelity Trajectory Optimization Problems and Bilevel Optimal Control Problems*. 1. Aufl. München: Verlag Dr. Hut, 2011. ISBN: 978-3-86853-851-9.

[FLH12] F. Fisch, J. Lenz, and F. Holzapfel. "Aircraft Configuration Settings within the Optimization of Approach Trajectories." In: *28th International Congress Of The Aeronautical Sciences*. (Brisbane, Australia, Sept. 23–28, 2012). 2012.

[FP11] R. Fernandes de Oliveira and G. Puyou. "On the use of optimization for flight control laws clearance. A practical approach." In: *IFAC Proceedings Volumes* Vol. 44. No. 1 (2011), pp. 9881–9886. ISSN: 14746670.

[FR08] F. Fahroo and I. M. Ross. "Advances in Pseudospectral Methods for Optimal Control." In: *AIAA Guidance, Navigation and Control Conference and Exhibit*. (Honolulu, Hawaii, USA, Aug. 18–21, 2008). AIAA, American Institute of Aeronautics and Astronautics. 2008.

[Fra+01] E. Frazzoli, Z.-H. Mao, J.-H. Oh, and E. Feron. "Resolution of Conflicts Involving Many Aircraft via Semidefinite Programming." In: *Journal of Guidance, Control, and Dynamics* Vol. 24. No. 1 (2001), pp. 79–86. ISSN: 0731-5090.

[Fra08] C. Francolin. *GPOPS - File Exchange - MATLAB Central*. 2008. URL: http://www.mathworks.com/matlabcentral/fileexchange/21729-gpops (visited on 04/20/2016).

[Ger08] M. Gerdts. *Optimierung. Vorlesungsskript*. Edgbaston, Birmingham: Management Mathematics Group, School of Mathematics, University of Birmingham, 2008. URL: https://www.unibw.de/lrt1/gerdts/lehre/optimierung.pdf (visited on 12/08/2015).

[Ger09] M. Gerdts. *Optimale Steuerung. Vorlesungsskript der Universität Würzburg 2009/2010*. Würzburg: Institut für Mathematik, Universität Würzburg, 2009. URL: https://www.unibw.de/lrt1/gerdts/lehre/optimale_steuerung.pdf (visited on 12/08/2015).

[Ger12] M. Gerdts. *Optimal Control of ODEs and DAEs*. De Gruyter Textbook. Berlin and Boston: De Gruyter, 2012. ISBN: 978-3-11-024995-8.

[Ger13] M. Gerdts. *User's Guide - OCPID-DAE1. Optimal Control and Parameter Identification with Differential-Algebraic Equations of Index 1*. Institut für Mathematik und Rechneranwendung, Universität der Bundeswehr München, 2013. URL: https://www.unibw.de/lrt1/gerdts/lehre/praktikum-optimale-steuerung/ocpiddae1.pdf (visited on 04/21/2016).

[GH16] C. Göttlicher and F. Holzapfel. "Flight Path Reconstruction for an Unmanned Aerial Vehicle Using Low-Cost Sensors." In: *30th Congress of the International Council of the Aeronautical Sciences (ICAS)*. (Daejeon, Korea, Sept. 25–30, 2016). The International Council of the Aeronautical Sciences. 2016.

[GKR01] M. Günther, A. Kværnø, and P. Rentrop. "Multirate Partitioned Runge-Kutta Methods." In: *BIT Numerical Mathematics* Vol. 41. No. 3 (2001), pp. 504–514. ISSN: 00063835.

[GL11] M. Gerdts and F. Lempio. *Mathematische Optimierungsverfahren des Operations Research*. De Gruyter Studium. Berlin and Boston: De Gruyter, 2011. ISBN: 3110249987.

[GMS02] P. E. Gill, W. Murray, and M. A. Saunders. "SNOPT. An SQP Algorithm for Large-Scale Constrained Optimization." In: *SIAM Journal of Optimization* Vol. 12. No. 4 (2002), pp. 979–1006.

[Gol91] D. Goldberg. "What every computer scientist should know about floating-point arithmetic." In: *ACM Computing Surveys* Vol. 23. No. 1 (1991), pp. 5–48. ISSN: 03600300.

[Gre02] J. E. Green. "Greener by Design – the Technology Challenge." In: *Aeronautical Journal* Vol. 1056. No. 106 (2002), pp. 57–103.

[GW84] C. W. Gear and D. R. Wells. "Multirate linear multistep methods." In: *BIT Numerical Mathematics* Vol. 24. No. 4 (1984), pp. 484–502. ISSN: 00063835.

[Hag00] W. W. Hager. "Runge-Kutta methods in optimal control and the transformed adjoint system." In: *Numerische Mathematik* Vol. 87. No. 2 (2000), pp. 247–282. ISSN: 0029-599X.

[Hah07] A. Hahn. "Staging Airliner Service." In: *7th AIAA ATIO Conf, 2nd CEIAT Int'l Conf on Innov & Integr in Aero Sciences,17th LTA Systems Tech Conf; followed by 2nd TEOS Forum*. (Belfast, Northern Ireland, Sept. 18–20, 2007). AIAA, American Institute of Aeronautics and Astronautics. 2007. (Visited on 02/04/2013).

[HC96] A. L. Herman and B. A. Conway. "Direct optimization using collocation based on high-order Gauss-Lobatto quadrature rules." In: *Journal of Guidance, Control, and Dynamics* Vol. 19. No. 3 (1996), pp. 592–599. ISSN: 0731-5090.

[HC98] A. L. Herman and B. A. Conway. "Optimal, Low-Thrust, Earth-Moon Orbit Transfer." In: *Journal of Guidance, Control, and Dynamics* Vol. 21. No. 1 (1998), pp. 141–147. ISSN: 0731-5090.

[HG16] F. Holzapfel and C. Göttlicher. *Flight System Identification. Lecture Notes*. München: Lehrstuhl für Flugsystemdynamik, Technische Universität München, 2016.

[Höc14] L. Höcht. "Advances in Stability Analysis for Model Reference Adaptive Control Systems and Application to Unmanned Aerial Systems." Lehrstuhl für Flugsystemdynamik. Dissertation. München: Technische Universität München, 2014.

[Hol+14] F. Holzapfel, M. Rieck, M. Bittner, M. Richter, and J. Diepolder. *Practical Course on Optimal Control. Lecture Notes*. München: Lehrstuhl für Flugsystemdynamik, Technische Universität München, 2014.

[Hol+15] F. Holzapfel, M. Gerdts, F. Fisch, and M. Bittner. *Aircraft Trajectory Optimization. Lecture Notes*. München: Lehrstuhl für Flugsystemdynamik, Technische Universität München, 2015.

[Hol04] F. Holzapfel. "Nichtlineare adaptive Regelung eines unbemannten Fluggerätes." Lehrstuhl für Flugmechanik und Flugregelung. Dissertation. München: Technische Universität München, 2004.

[Hol15] F. Holzapfel. *Flight System Dynamic II. Lecture Notes*. München: Lehrstuhl für Flugsystemdynamik, Technische Universität München, 2015.

[HPS02] J. Hu, M. Pradini, and S. Sastry. "Three Dimensional Optimal Coordinated Maneuvers for Aircraft Conflict Avoidance." In: *Journal of Guidance, Control, and Dynamics* Vol. 25. No. 5 (2002), pp. 888–900. ISSN: 0731-5090.

[Hug93] J. Hugger. "The theory of density representation of finite element meshes. Examples of density operators with quadrilateral elements in the mapped domain." In: *Computer Methods in Applied Mechanics and Engineering* Vol. 109. No. 1 (1993), pp. 17–39.

[ICA01] ICAO. *Procedures for Air Navigation Services, Air Traffic Management. Doc 4444*. Version Fourteenth Edition. International Civil Aviation Organization, 2001.

[ICA02] ICAO. *Manual of the ICAO standard atmosphere. Extended to 80 kilometres (262 500 feet)*. Version Third Edition. Montreal: International Civil Aviation Organization, 2002.

[ICA16] ICAO. *ICAO Aircraft Engine Emissions Databank*. International Civil Aviation Organization. 2016. URL: https://easa.europa.eu/document-library/icao-aircraft-engine-emissions-databank#2 (visited on 03/03/2016).

[ICA95] ICAO. *ICAO Engine Exhaust Emissions Databank*. Montreal, Quebec: International Civil Aviation Organization, 1995.

[JT08] S. Jain and P. Tsiotras. "Trajectory Optimization Using Multiresolution Techniques." In: *Journal of Guidance, Control, and Dynamics* Vol. 31. No. 5 (2008), pp. 1424–1436. ISSN: 0731-5090.

[Kim+07] B. Y. Kim et al. "System for assessing Aviation's Global Emissions (SAGE), Part 1. Model description and inventory results." In: *Transportation Research Part D: Transport and Environment* Vol. 12. No. 5 (2007), pp. 325–346. ISSN: 13619209.

[Kir70] D. E. Kirk. *Optimal Control Theory. An Introduction*. Prentice-Hall Networks Series. Englewood Cliffs, N.J.: Prentice-Hall, 1970. ISBN: 978-0-486-43484-1.

[KM03] J.-H. R. Kim and H. Maurer. "Sensitivity Analysis of Optimal Control Problems with Bang-Bang Controls." In: *42nd IEEE Conference on Decision and Control*. (Maui, Hawaii, USA, Dec. 9–12, 2003). Piscataway, N.J.: IEEE, 2003, pp. 3281–3286. ISBN: 0-7803-7924-1.

[KM13] C. Y. Kaya and H. Maurer. "A numerical method for nonconvex multi-objective optimal control problems." In: *Computational Optimization and Applications* Vol. 57. No. 3 (2013), pp. 685–702. ISSN: 0926-6003 1573-2894.

[KT51] H. W. Kuhn and A. W. Tucker. "Nonlinear Programming." In: *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*. Berkeley, Calif.: University of California Press, 1951.

[Lee+07] J. J. Lee, I. A. Waitz, B. Y. Kim, G. G. Fleming, L. Maurice, and C. A. Holsclaw. "System for assessing Aviation's Global Emissions (SAGE), Part 2. Uncertainty assessment." In: *Transportation Research Part D: Transport and Environment* Vol. 12. No. 6 (2007), pp. 381–395. ISSN: 13619209.

[Lei81] G. Leitmann. *The calculus of variations and optimal control. An introduction*. Mathematical Concepts and Methods in Science and Engineering. New York: Plenum Press, 1981. ISBN: 0306407078.

[Len15] J. Lenz. "Optimisation of Periodic Flight Trajectories." Lehrstuhl für Flugsystemdynamik. München: Technische Universität München, 2015.

[Lin05] J. G. Lin. "On min-norm and min-max methods of multi-objective optimization." In: *Mathematical Programming* Vol. 103. No. 1 (2005), pp. 1–33. ISSN: 0025-5610.

[Lof80] Loftin, L. K., Jr. *Subsonic aircraft: Evolution and the matching of size to performance*. NASA Langley Research Center and Hampton, VA, United States, 1980. URL: http://ntrs.nasa.gov/search.jsp?R=19800020744.

[LZZ12] S. Li, R. Zhao, and Q. Zhang. "Optimization method for solving bang-bang and singular control problems." In: *Journal of Control Theory and Applications* Vol. 10. No. 4 (2012), pp. 559–564. ISSN: 1672-6340.

[Mat] T. MathWorks. *fmincon - Documentation. Find minimum of constrained nonlinear multivariable function*. URL: http://de.mathworks.com/help/optim/ug/fmincon.html (visited on 12/16/2015).

[Mat+15] H. Matsuda, A. Harada, T. Kozuka, and Y. Miyazawa. "Arrival Time Assignment by Dynamic Programming Optimization." In: *The 4th ENRI International Workshop on ATM/CNS (EIWAC2015)*. (Tokyo, Japan, Nov. 17–19, 2015). ENRI. 2015.

[Mie01] K. Miettinen. "Some Methods for Nonlinear Multi-objective Optimization." In: *Evolutionary Multi-Criterion Optimization*. Ed. by E. Zitzler, L. Thiele, K. Deb, C. Coello Coello, and D. Corne. Vol. 1993. Lecture Notes in Computer Science. Berlin and Heidelberg: Springer, 2001, pp. 1–20. ISBN: 978-3-540-41745-3.

[Min15] Ministry of Land, Infrastructure, Transport and Tourism. *Data Extracted from the Air Navigation Services Department's Database Becomes Available*. Air Navigation Services Department, Civil Aviation Bureau, MLIT. 2015. URL: http://www.mlit.go.jp/common/001082594.pdf (visited on 06/10/2016).

[MMR12] K. Mohan, Michael A. Patterson, and A. V. Rao. "Optimal Trajectory and Control Generation for Landing of Multiple Aircraft in the Presence of Obstacles." In: *28th International Congress Of The Aeronautical Sciences*. (Brisbane, Australia, Sept. 23–28, 2012). 2012.

[MSS99] P. K. Menon, G. D. Sweriduk, and B. Sridhar. "Optimal Strategies for Free-Flight Air Traffic Conflict Resolution." In: *Journal of Guidance, Control, and Dynamics* Vol. 22. No. 2 (1999), pp. 202–211. ISSN: 0731-5090.

[MTF14] H. Maurer, T. Tarnopolskaya, and N. Fulton. "Computation of bang-bang and singular controls in collision avoidance." In: *Journal of Industrial and Management Optimization* Vol. 10. No. 2 (2014), pp. 443–460. ISSN: 1547-5816.

[MU10] G. V. Milovanović and Z. Udovičić. "Calculation of coefficients of a cardinal B-spline." In: *Applied Mathematics Letters* Vol. 23. No. 11 (2010), pp. 1346–1350. ISSN: 08939659.

[Nan06a] R. K. Nangia. "Efficiency Parameters for Modern Commercial Aircraft." In: *The Aeronautical Journal* Vol. August 2006. No. 110 (2006).

[Nan06b] R. K. Nangia. "Operations and Aircraft Design Towards Greener Civil Aviation Using Air-to-Air Refueling." In: *The Aeronautical Journal* Vol. November 2006. No. 110 (2006).

[Nan08] R. K. Nangia. "Highly Efficient and Greener Civil Aviation - Organising a Step Jump towards ACARE Goals. An Opportunity for the Present & a Vision for Future." In: *Royal Aeronautical Society Conference. Aerospace 2008, The Way Forward*. (London, Apr. 22–24, 2008). Royal Aeronautical Society. 2008.

[Nat00] National Imagery and Mapping Agency, NIMA. *Department of Defense World Geodetic System 1984. Its Definition and Relationships with Local Geodetic Systems*. Technical Report 8350.2. Version Third Edition, incl. Amendment 1. St. Louis, MO: National Imagery and Mapping Agency, NIMA, 2000.

[NE13] Network Manager nominated by the European Commission and EUROCONTROL. *EUROCONTROL Seven-Year Forecast September 2013. Flight Movements and Service Units 2013 - 2019*. 2013.

[OS83] S. Osher and R. Sanders. "Numerical approximations to nonlinear conservation laws with locally varying time and space grids." In: *Mathematics of Computation* Vol. 41. No. 164 (1983), p. 321. ISSN: 0025-5718.

[OWH15] A. Okubo, S. Watanabe, and M. Harada. "Three-Dimensional Trajectory Optimization Considering Arrival Time at the Merging Point." In: *The 4th ENRI International Workshop on ATM/CNS (EIWAC2015)*. (Tokyo, Japan, Nov. 17–19, 2015). ENRI. 2015.

[Pes96] H. J. Pesch. "A Practical Guide To The Solution Of Real-Life Optimal Control Problems." In: *Control and Cybernetics* Vol. 23. No. 1 (1996). ISSN: 0324-8569.

[PKH11] S. H. Pourtakdoust, M. Kiani, and A. Hassanpour. "Optimal trajectory planning for flight through microburst wind shears." In: *Aerospace Science and Technology* Vol. 15. No. 7 (2011), pp. 567–576. ISSN: 1270-9638.

[Pol97] E. Polak. *Optimization. Algorithms and Consistent Approximations*. Vol. 124. Applied Mathematical Sciences. New York, NY: Springer, 1997. ISBN: 1461206634.

[PP12] H. J. Pesch and M. Plail. "The Cold War and the Maximum Principle of Optimal Control." In: *Optimization stories. 21st International Symposium on Mathematical Programming, Berlin, August 19 – 24, 2012*. Ed. by M. Grötschel. Vol. Extra Volume. Documenta Mathematica Extra Volume. Bielefeld, 2012. ISBN: 978-3-936609-58-5.

[PR14] 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* Vol. 41. No. 1 (2014). ISSN: 0098-3500.

[Rag+04] A. U. Raghunathan, V. Gopal, D. Subramanian, L. T. Biegler, and T. Samad. "Dynamic Optimization Strategies for Three-Dimensional Conflict Resolution of Multiple Aircraft." In: *Journal of Guidance, Control, and Dynamics* Vol. 27. No. 4 (2004), pp. 586–594. ISSN: 0731-5090.

[Rao+10] A. V. Rao et al. "Algorithm 902: GPOPS, A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using the Gauss Pseudospectral Method." In: *ACM Transactions on Mathematical Software* Vol. 37. No. 2 (2010), pp. 1–39. ISSN: 0098-3500.

[Ray12] D. P. Raymer. *Aircraft design. A conceptual approach*. 5th ed. AIAA Education Series. Reston, VA: American Institute of Aeronautics and Astronautics, 2012. ISBN: 1600869114.

[RBH13] M. Richter, M. Bittner, and F. Holzapfel. "Noise Minimal Approaches on Parallel Runways." In: *Proceedings of the Fourth International Air Transport and Operations Symposium (ATOS)*. (Toulouse, France, July 8–12, 2013). 2013.

[RE10] P. E. Rutquist and M. M. Edvall. *PROPT - Matlab Optimal Control Software. - One of a kind, lightning fast solutions to your optimal control problems!* Tomlab Optimization Inc., 2010. URL: http://tomopt.com/docs/TOMLAB_PROPT.pdf (visited on 04/21/2016).

[Ren88] R. J. Renka. "Multivariate Interpolation of Large Sets of Scattered Data." In: *ACM Transactions on Mathematical Software* Vol. 14. No. 2 (1988), pp. 139–148. ISSN: 0098-3500.

[RH02] A. Richards and J. P. How. "Aircraft trajectory planning with collision avoidance using mixed integer linear programming." In: *Proceedings of the 2002 American Control Conference*. (Anchorage, Alaska, USA, May 8–10, 2002). Piscataway, NJ: IEEE, 2002, pp. 1936–1941. ISBN: 0-7803-7298-0.

[Ric+13] M. Richter, M. Bittner, M. Rieck, and F. Holzapfel. "Berechnung lärmminimaler An- und Abflugrouten." In: *Matlab EXPO 2013*. (Munich, Germany, July 2, 2013). The MathWorks. 2013.

[Ric+14a] M. Richter, M. Bittner, M. Rieck, and F. Holzapfel. "A Non-Cooperative Bi-Level Optimal Control Problem Formulation for Noise Minimal Departure Trajectories." In: *29th Congress of the International Council of the Aeronautical Sciences (ICAS)*. (St. Petersburg, Russia, Sept. 7–12, 2014). The International Council of the Aeronautical Sciences. 2014.

[Ric+14b] M. Richter, M. Bittner, M. Rieck, and F. Holzapfel. "A Realistic Flight Path Parameterization for Calculation of Noise Minimal Trajectories using Bi-level Optimal Control." In: *AIAA Guidance, Navigation, and Control Conference*. (National Harbor, Maryland, USA, Jan. 13–17, 2014). AIAA, American Institute of Aeronautics and Astronautics. 2014.

[Ric+14c] M. Richter, M. Hochstrasser, M. Bittner, L. Walter, and F. Holzapfel. "Application of MINLP Techniques to Conflict Resolution of Multiple Aircraft." In: *AIAA Guidance, Navigation, and Control Conference*. (National Harbor, Maryland, USA, Jan. 13–17, 2014). AIAA, American Institute of Aeronautics and Astronautics. 2014.

[Rie+14] M. Rieck, M. Richter, M. Bittner, and F. Holzapfel. "Generation of Initial Guesses for Optimal Control Problems with Mixed Integer Dependent Constraints." In: *29th Congress of the International Council of the Aeronautical Sciences (ICAS)*. (St. Petersburg, Russia, Sept. 7–12, 2014). The International Council of the Aeronautical Sciences. 2014.

[Rie+15] M. Rieck, M. Bittner, B. Grüter, and F. Holzapfel. "Generation of Dynamic Models with Automatically Generated Derivatives for ATM Optimal Control in MATLAB." In: *The 4th ENRI International Workshop on ATM/CNS (EIWAC2015)*. (Tokyo, Japan, Nov. 17–19, 2015). ENRI. 2015.

[Rie+16] M. Rieck, M. Bittner, B. Grüter, and J. Diepolder. *FALCON.m User Guide*. Institute of Flight System Dynamics, Technical University of Munich, 2016. URL: www.falcon-m.com.

[Ros04] J. Roskam. *Airplane Design*. 8 Volumes vols. Lawrence, Kansas: DARcorporation, 2004. ISBN: 1884885241.

[Ros79] J. Roskam. *Airplane Flight Dynamics and Automatic Flight Controls*. Ottawa, Canada: Roskam Aviation and Engineering Corporation, 1979.

[RRH13] M. Rieck, M. Richter, and F. Holzapfel. "Discrete Control Dependent Constraints in Multiple Shooting Optimal Control Problems." In: *Guidance, Navigation, and Control and Co-located Conferences*. (Boston, Massachusetts, Aug. 19–22, 2013). AIAA, American Institute of Aeronautics and Astronautics. 2013.

[RTC05] RTCA. *DO-258A Interoperability Requirements for ATS Applications Using ARINC 622 Data Communications*. Radio Technical Commission for Aeronautics, 2005.

[Sch+05] M. Schlegel, K. Stockmann, T. Binder, and W. Marquardt. "Dynamic optimization using adaptive control vector parameterization." In: *Computers & Chemical Engineering* Vol. 29. No. 8 (2005), pp. 1731–1751. ISSN: 0098-1354.

[Sch96] A. L. Schwartz. "Theory and implementation of numerical methods based on Runge-Kutta integration for solving optimal control problems." Electrical Engineering and Computer Sciences. Doctoral Thesis. University of California at Berkeley, 1996.

[She68] D. Shepard. "A Two-dimensional Interpolation Function for Irregularly-spaced Data." In: *ACM '68 Proceedings of the 1968 23rd ACM National Conference*. (Aug. 27–29, 1968). New York, NY, USA: ACM, 1968, pp. 517–524.

[SHF04] T. Schouwenaars, J. How, and E. Feron. "Decentralized Cooperative Trajectory Planning of Multiple Aircraft with Hard Safety Guarantees." In: *AIAA Guidance, Navigation, and Control Conference and Exhibit*. (Providence, Rhode Island, Aug. 16–19, 2004). AIAA, American Institute of Aeronautics and Astronautics. 2004. ISBN: 978-1-62410-073-4.

[Sim06] D. Simon. *Optimal State Estimation. Kalman, H Infinity, and Nonlinear Approaches*. Hoboken, N.J.: Wiley-Interscience, 2006. ISBN: 978-0-470-04533-6.

[SL03] B. L. Stevens and F. L. Lewis. *Aircraft control and simulation*. 2nd ed. Hoboken, N.J.: J. Wiley, 2003. ISBN: 9780471371458.

[SM06a] M. Schlegel and W. Marquardt. "Adaptive Switching Structure Detection for the Solution of Dynamic Optimization Problems." In: *Industrial & Engineering Chemistry Research* Vol. 45. No. 24 (2006), pp. 8083–8094. ISSN: 0888-5885.

[SM06b] M. Schlegel and W. Marquardt. "Detection and exploitation of the control switching structure in the solution of dynamic optimization problems." In: *Journal of Process Control* Vol. 16. No. 3 (2006), pp. 275–290. ISSN: 09591524.

[Sol+12] M. Soler, M. Kamgarpour, C. Tomlin, and E. Staffetti. "Multiphase mixed-integer optimal control framework for aircraft conflict avoidance." In: *2012 IEEE 51st Annual Conference on Decision and Control (CDC)*. (Maui, HI, USA). 2012, pp. 1740–1745.

[Sol+16] M. Soler, M. Kamgarpour, J. Lloret, and J. Lygeros. "A Hybrid Optimal Control Approach to Fuel-Efficient Aircraft Conflict Avoidance." In: *IEEE Transactions on Intelligent Transportation Systems* Vol. 17. No. 7 (2016), pp. 1826–1838. ISSN: 1524-9050.

[Str93] O. von Stryk. "Numerical Solution of Optimal Control Problems by Direct Collocation." In: *Optimal Control. Calculus of Variations, Optimal Control Theory and Numerical Methods*. Ed. by R. Bulirsch, A. Miele, J. Stoer, and K. Well. Vol. 111. International Series of Numerical Mathematics 111. Basel: Birkhäuser, 1993, pp. 129–143. ISBN: 978-3-0348-7541-7.

[Str94] O. von Stryk. "Numerische Lösung Optimaler Steuerungsprobleme: Diskretisierung, Parameteroptimierung und Berechnung adjungierter Variablen." Fakultät für Mathematik. Dissertation. München: Technische Universität München, 1994.

[The16] The MathWorks. *Comparison of Handle and Value Classes*. 2016. URL: http://de.mathworks.com/help/matlab/matlab_oop/comparing-handle-and-value-classes.html?refresh=true (visited on 04/28/2016).

[TKS13] C. Tsotskas, T. Kipouros, and M. Savill. "Biobjective Optimisation of Preliminary Aircraft Trajectories." In: *Evolutionary Multi-Criterion Optimization*. Ed. by R. C. Purshouse, P. J. Fleming, C. M. Fonseca, S. Greco, and J. Shaw. Vol. 7811. Lecture Notes in Computer Science. Berlin and Heidelberg: Springer, 2013, pp. 741–755. ISBN: 978-3-642-37139-4.

[Tom11] Tomlab Optimization Inc. *PROPT - Matlab Optimal Control Software (DAE, ODE)*. 2011. URL: http://tomdyn.com/propt_key_features.html (visited on 04/21/2016).

[Tor+15] D. Toratani, D. Delahaye, S. Ueno, and T. Higuchi. "Merging Optimization Method with Multiple Entry Points for Extended Terminal Maneuvering Area." In: *The 4th ENRI International Workshop on ATM/CNS (EIWAC2015)*. (Tokyo, Japan, Nov. 17–19, 2015). ENRI. 2015.

[Tor06] D. F. M. Torres. "On the Constancy of the Pontryagin Hamiltonian for Autonomous Problems." In: *International Journal of Applied Mathematics & Statistics (IJAMAS)* Vol. 4. No. J06 (2006), pp. 23–24.

[Tor82] E. Torenbeek. *Synthesis of Subsonic Airplane Design*. Dordrecht: Springer Netherlands, 1982. ISBN: 978-90-481-8273-2.

[Uni16] University of Wisconsin. *SOCS - NEOS*. 2016. URL: http://www.neos-guide.org/content/socs (visited on 04/21/2016).

[Upr13] S. R. Upreti. *Optimal control for chemical engineers*. Boca Raton, FL: CRC Press, 2013. ISBN: 978-1-4398-3895-2.

[Vel+09] A. E. Vela, S. Solak, E. Feron, K. Feigh, W. Singhose, and J. P. Clarke. "A fuel optimal and reduced controller workload optimization model for conflict resolution." In: *IEEE/AIAA 28th Digital Avionics Systems Conference, 2009 (DASC '09)*. (Orlando, Florida, Oct. 25–29, 2009). IEEE/AIAA. 2009.

[Vis07] H. G. Visser. "Environmentally-Optimized Air Traffic Conflict Resolution." In: *7th AIAA Aviation Technology, Integration and Operations Conference*. (Belfast, Northern Ireland). AIAA, American Institute of Aeronautics and Astronautics. 2007. ISBN: 978-1-62410-014-7.

[WA85] W. E. Wiesel and S. Alfano. "Optimal many-revolution orbit transfer." In: *Journal of Guidance, Control, and Dynamics* Vol. 8. No. 1 (1985), pp. 155–157. ISSN: 0731-5090.

[WB06] A. Wächter and L. T. Biegler. "On the Implementation of a Primal-Dual Interior Point Filter Line Search Algorithm for Large-Scale Nonlinear Programming." In: *Mathematical Programming* Vol. 106. No. 1 (2006), pp. 25–57. ISSN: 0025-5610.

[Wei14] H. Weirather. "Flugrennsimulation gegen ein zeitoptimiertes Flugzeug." Institute of Flight System Dynamics. Bachelor Thesis. Garching: Technische Universität München, 2014.

[Wic+15] N. K. Wickramasinghe, M. Brown, S. Fukushima, and Y. Fukuda. "Optimization-Based Performance Assessment on 4D-Trajectory Based Operations with Track Data." In: *The 4th ENRI International Workshop on ATM/CNS (EIWAC2015)*. (Tokyo, Japan, Nov. 17–19, 2015). ENRI. 2015.

[Wol69] P. Wolfe. "Convergence Conditions for Ascent Methods." In: *SIAM Review* Vol. 11. No. 2 (1969), pp. 226–235.

[Wol71] P. Wolfe. "Convergence Conditions for Ascent Methods. II: Some Corrections." In: *SIAM Review* Vol. 13. No. 2 (1971), pp. 185–188.

[Yok16] N. Yokoyama. "Inference of Flight Intent Based on Inverse Optimal Control." In: *30th Congress of the International Council of the Aeronautical Sciences (ICAS)*. (Daejeon, Korea, Sept. 25–30, 2016). The International Council of the Aeronautical Sciences. 2016.

[Zip07] P. H. Zipfel. *Modeling and simulation of aerospace vehicle dynamics*. 2nd ed. AIAA Education Series. Reston, Va.: American Institute of Aeronautics and Astronautics, 2007. ISBN: 978-1-56347-875-8.

[ZS97] Y. Zhao and R. Schultz. "Deterministic Resolution of two aircraft conflict in free flight." In: *Guidance, Navigation, and Control Conference*. (New Orleans,LA,USA, Aug. 11–13, 1997). AIAA, American Institute of Aeronautics and Astronautics. 1997.

[ZT11] Y. Zhao and P. Tsiotras. "Density Functions for Mesh Refinement in Numerical Optimal Control." In: *Journal of Guidance, Control, and Dynamics* Vol. 34. No. 1 (2011), pp. 271–277. ISSN: 0731-5090.