On-board Trajectory Computation for Mars Atmospheric Entry Based on Parametric Sensitivity Analysis of Optimal Control Problems

Dissertation

submitted to the

Faculty of Mathematics and Computer Science of University Bremen

in partial fulfillment of the requirements for

the degree of doctor of engineering (Dr.-Ing.)

by Dipl.-Inform. David Seelbinder

August 22, 2017

1. reviewer: Prof. Dr. Christof Büskens
2. reviewer: Dr.-Ing. Stephan Theil

Abstract

This thesis develops a precision guidance algorithm for the entry of a small capsule into the atmosphere of Mars. The entry problem is treated as nonlinear optimal control problem and the thesis focuses on developing a suboptimal feedback law. Therefore parametric sensitivity analysis of optimal control problems is combined with dynamic programming. This approach enables a real-time capable, locally suboptimal feedback scheme.

The optimal control problem is initially considered in open loop fashion. To synthesize the feedback law, the optimal control problem is embedded into a family of neighboring problems, which are described by a parameter vector. The optimal solution for a nominal set of parameters is determined using direct optimization methods. In addition the directional derivatives (sensitivities) of the optimal solution with respect to the parameters are computed. Knowledge of the nominal solution and the sensitivities allows, under certain conditions, to apply Taylor series expansion to approximate the optimal solution for disturbed parameters almost instantly. Additional correction steps can be applied to improve the optimality of the solution and to eliminate errors in the constraints.

To transfer this strategy to the closed loop system, the computation of the sensitivities is performed with respect to different initial conditions. Determining the perturbation direction and interpolating between sensitivities of neighboring initial conditions allows the approximation of the extremal field in a neighborhood of the nominal trajectory. This constitutes a locally suboptimal feedback law.

The proposed strategy is applied to the atmospheric entry problem. The developed algorithm is part of the main control loop, i.e. optimal controls and trajectories are computed at a fixed rate, taking into account the current state and parameters. This approach is combined with a trajectory tracking controller based on the aerodynamic drag.

The performance and the strengths' and weaknesses of this two degree of freedom guidance system are analyzed using Monte Carlo simulation. Finally the real-time capability of the proposed algorithm is demonstrated in a flight representative processor-in-the-loop environment.

Key words: Real-time optimal control, parametric sensitivity analysis, entry guidance

Abstract

In dieser Arbeit wird ein Flugführungsalgorithmus für den Eintritt einer Kapsel in die Marsatmosphäre entwickelt. Die rahmengebenden Bedingungen werden als Optimalsteuerungsproblem aufgefasst und der Schwerpunkt der Arbeit liegt auf der Entwicklung eines suboptimalen Regelgesetzes für nicht lineare dynamische Systeme. Dazu wird die parametrische Sensitivitätsanalyse von Optimalsteuerungsproblemen kombiniert mit dynamischer Programmierung. Dieser Ansatz ermöglicht die Auswertung eines lokal suboptimalen Rückführgesetzes in Echtzeit.

Das Problem wird zunächst im offenen Regelkreis betrachtet. Zur Synthese des Regelgesetzes wird das Optimalsteuerungsproblem eingebettet in eine Familie benachbarter Probleme, die durch einen Parametersatz beschrieben werden. Die optimale Lösung wird durch Methoden der direkten Optimierung für einen Satz nomineller Parameter bestimmt und zusätzlich werden die direktionalen Ableitungen (Sensitivitäten) der optimalen Lösung gegenüber den Parametern berechnet. Die Kenntnis der nominellen Lösung sowie deren Ableitungen erlaubt unter gewissen Voraussetzungen die Anwendung der Taylorentwicklung zur schnellen Approximation der optimalen Lösung für gestörte Parameter. Zusätzlich können weitere Korrekturschritte ausgeführt werden, um die Approximation zu verbessern und Fehler in den Beschränkungen zu eliminieren.

Zur Übertragung dieser Strategie auf den geschlossenen Regelkreis werden die Sensitivitäten für unterschiedliche Anfangsbedingungen berechnet. Die Bestimmung der Störrichtung und die Interpolation zwischen Sensitivitätsableitungen benachbarter Anfangsbedingungen ermöglichen die Approximation des Extremalfeldes in einer Umgebung der nominellen Trajektorie, und damit eine suboptimale Regelung.

Die vorgeschlagene Strategie wird angewandt auf das Problem der Flugführung während des Eintritts in die Atmosphäre. Der Führungsalgorithmus ist dabei Teil des geschlossenen Regelkreises, so dass in einem regelmäßigen Takt optimale Steuerungen und Zustandstrajektorien für den aktuellen Zustand berechnet werden. Dieser Ansatz wird kombiniert mit einer Trajektorienfolgeregelung, basierend auf der Messung des aerodynamischen Strömungswiderstands.

Die Leistungsfähigkeit und die Stärken und Schwächen dieses Regelungssystems werden mit Hilfe einer Monte Carlo Simulation analysiert. Abschließend wird die Echtzeitfähigkeit des Algorithmus durch Laufzeittests auf einem für Weltraumexplorationsmissionen geeigneten Computersystem nachgewiesen.

Schlagworte: Echtzeitoptimalsteuerung, parametrische Sensitivitätsanalyse, Wiedereintritt, Flugführung

Contents

List of Abbreviations

Notation	Description
CoM	center of mass
DP	dynamic programming
EDL	entry, descent and landing
GNC	guidance, navigation and control
KKT	Karush-Kuhn-Tucker
LQR	linear quadratic regulator
MP	Minimum Principle
MPC	model predictive control
MPL	Mars Precision Lander
NLP	nonlinear program
OCP	optimal control problem
ODE	ordinary differential equation
PDE	partial differential equation
PMP	Pontryagin's Minimum (Maximum) Principle
PoO	Principle of Optimality
PS	parametric sensitivity
PSA	parametric sensitivity analysis
QP	quadratic program
RASTA	Reference Architecture System Test-Bed for Avionics
SDP	semidefinite program
SOCP	second order cone program
SQP	sequential quadratic programming
vLoD	vertical lift-over-drag

List of Symbols

Script Semantics

Symbols may have an overset, topscript and subscript which have the following semantics: Oversets mark special properties. Topscripts in round brackets denote the position of an element within it's parent structure. Topscripts in square brackets are iteration counters or derivative orders. Subscripts are used as descriptors (multiple subscripts are separated by commas). The following is a list of the most commonly used scripts throughout the thesis:

$\overset{\star}{(\cdot)}$		Optimality
$\tilde{(\cdot)}$		Approximation
$(\cdot)^{[k]}$		Iteration counter or Lie derivative order
$(\cdot)^{(i)} \in \mathbb{R}$,	$(\cdot) \in \mathbb{R}^n$	Ordered set/vector element at position i, $1 \leq i \leq n$
$(\cdot)^{(i,j)} \in \mathbb{R}$,	$(\cdot) \in \mathbb{R}^n \times \mathbb{R}^m$	Matrix element in row i, column j, $1 \leq i \leq n$, $1 \leq j \leq m$
$(\cdot)^{(i,:)} \in \mathbb{R}^m$,	$(\cdot) \in \mathbb{R}^n \times \mathbb{R}^m$	Row vector i, $1 \leq i \leq n$
$(\cdot)^{(:,j)} \in \mathbb{R}^n$,	$(\cdot) \in \mathbb{R}^n \times \mathbb{R}^m$	Column vector j, $1 \leq j \leq m$
$(\cdot)_0$		Nominal value
$(\cdot)_s$		Start/initial value
$(\cdot)_f$		Final/terminal value
$(\cdot)_a$		Belonging to the active set

Derivative Notation

Let $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_y} \to \mathbb{R}$ and $g : \mathbb{R}^{n_x} \times \mathbb{R}^{n_y} \to \mathbb{R}^n$ be sufficiently differentiable functions for $x \in \mathbb{R}^{n_x}$ and $y \in \mathbb{R}^{n_y}$. Furthermore let $V : \mathbb{R}^{n_x} \times \mathbb{R}^{n_x}$ be a differentiable vector field. The following notations are used to describe derivatives:

$$\nabla_x f(x,y) := \left(\frac{\partial f}{\partial x^{(1)}}(x,y), ..., \frac{\partial f}{\partial x^{(n_x)}}(x,y) \right)$$

$$\nabla_x f(x_0,y_0) := \left(\nabla_x f(x,y)\right)|_{x=x_0,\, y=y_0} \qquad \in \mathbb{R}^{n_x}$$

$$\nabla^2_{yx} f(x_0,y_0) := \begin{pmatrix} \nabla_x \left(\nabla_y f(x_0,y_0)\right)^{(1)} \\ \vdots \\ \nabla_x \left(\nabla_y f(x_0,y_0)\right)^{(n_y)} \end{pmatrix} \qquad \in \mathbb{R}^{n_y} \times \mathbb{R}^{n_x}$$

$$\nabla_x g(x_0,y_0) := \begin{pmatrix} \nabla_x g^{(1)}(x_0,y_0) \\ \vdots \\ \nabla_x g^{(n)}(x_0,y_0) \end{pmatrix} \qquad \in \mathbb{R}^{n} \times \mathbb{R}^{n_x}$$

$$\nabla^2_{yx} g(x_0,y_0) := \begin{pmatrix} \nabla_y(\nabla_x g^{(1)}(x_0,y_0))^{(1)} & \dots & \nabla_y(\nabla_x g^{(1)}(x_0,y_0))^{(n_x)} \\ \vdots & \ddots & \vdots \\ \nabla_y(\nabla_x g^{(n)}(x_0,y_0))^{(1)} & \dots & \nabla_y(\nabla_x g^{(n)}(x_0,y_0))^{(n_x)} \end{pmatrix} \in \mathbb{R}^{n} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_y}$$

$$L_V f(x,y) := \langle \nabla_x f(x,y), V(x) \rangle \quad \text{(Lie derivative)} \qquad \in \mathbb{R}$$

Planetary Atmospheric Flight

r	m	Distance from spacecraft to planetary center
h	m	Altitude
λ	rad	Longitude
φ	rad	Geocentric latitude
v	$\frac{\mathrm{m}}{\mathrm{s}}$	Speed
γ	rad	Flight path angle
χ	rad	Heading angle (measured positive from north to east)
q	Pa	Dynamic pressure
ρ	$\frac{\mathrm{kg}}{\mathrm{m}^3}$	Atmospheric or fluid density
ρ_0	$\frac{\mathrm{kg}}{\mathrm{m}^3}$	Atmospheric density at sea-level
ρ_{exp}	$\frac{\mathrm{kg}}{\mathrm{m}^3}$	Exponential atmosphere model
h_s	m	Scaled height
v_{atm}	$\frac{\mathrm{m}}{\mathrm{s}}$	Relative speed between atmosphere and object
v_{gnd}	$\frac{\mathrm{m}}{\mathrm{s}}$	Relative speed between ground and object
v_{wnd}	$\frac{\mathrm{m}}{\mathrm{s}}$	Relative speed between atmosphere and ground
F_L	N	Lift force
F_D	N	Drag force
L	$\frac{\mathrm{m}}{\mathrm{s}^2}$	Lift acceleration
D	$\frac{\mathrm{m}}{\mathrm{s}^2}$	Drag acceleration
C_L	[-]	Aerodynamic lift coefficient
C_D	[-]	Aerodynamic drag coefficient
S	m^2	Aerodynamic reference area
r_n	m	Effective nose radius
m	kg	Vehicle total wet mass
c	$\frac{\mathrm{m}}{\mathrm{s}}$	Local speed of sound
M	[-]	Mach number
α	rad	Angle of attack
β	rad	Angle of sideslip
σ	rad	Bank angle
M	kg	Mass of celestial body
G	$\frac{\mathrm{m}^3\,\mathrm{s}^2}{\mathrm{kg}}$	Gravitational constant
μ_M	$\frac{\mathrm{m}^3}{\mathrm{s}^2}$	Mars gravitational constant
r_M	m	Mars radius (spherical approximation)
Ω_M	$\frac{\mathrm{rad}}{\mathrm{s}}$	Mars rotation rate
E_{tot}	J	Total mechanical energy
E_{kin}	J	Kinetic energy
E_{pot}	J	Potential energy
E	J	Specific mechanical energy of the vehicle with respect to the planets surface
$\dot{Q}$	$\frac{\mathrm{W}}{\mathrm{m}^2}$	Convective heating at stagnation point (heat flux density)
k_p	$\sqrt{kg}/\mathrm{m}$	Sutton-Graves constant for the Martian atmosphere
n	[-]	Load factor
g_E	$\frac{\mathrm{m}}{\mathrm{s}^2}$	Earth gravitational acceleration at sea-level

Drag Tracking and Feedback Linearization

R_r	Reference (desired) range
D_r	Reference drag acceleration
$L_V^{[k]}C(x)$	k-th Lie derivative of a function C along a vector field V
r	Relative degree of the output
T	Coordinate transformation
u_{vLoD}	Vertikal lift-over-drag ratio (as determined by the bank angle)
u_{com}	Commanded vertikal lift-over-drag ratio
ν	Outer loop control (input to feedback linearized system)
K_P	Proportional gain
K_D	Derivative gain
K_I	Integral gain
ω_n	Undamped natural frequency
ζ	damping ratio

Optimal Control Processes

n_x	$\in$	$\mathbb{N}$	Dimension of the state vector function
n_u	$\in$	$\mathbb{N}$	Dimension of the control vector function
n_c	$\in$	$\mathbb{N}$	Dimension of the path constraint vector function
n_p	$\in$	$\mathbb{N}$	Dimension of the perturbation vector
t	$\in$	$[t_s; t_f]$	Independent variable, called time
t_s	$\in$	$\mathbb{R}$	Start time
t_f	$\in$	$\mathbb{R}$	Final time
t_d	$:=$	$t_f - t_s$	Process duration
L^∞			Lebesque Space, see Appendix B
$W^{1,\infty}$			Sobolev Space, see Appendix B
$\mathbb{U}(t)$	$\subset$	$\mathbb{R}^{n_u}$	Admissible control vectors at a fixed time t
$\mathbb{X}(t)$	$\subset$	$\mathbb{R}^{n_x}$	Admissible state vectors at a fixed time t
$u(t)$	$\in$	$\mathbb{U}(t)$	Control vector at time t
$x(t)$	$\in$	$\mathbb{X}(t)$	State vector at time t
$\mu(t)$	$\in$	$\mathbb{R}^{n_x}$	Adjoint vector at time t
f			System dynamic
c			Mixed state and control constraints
J			Cost or objective functional
m			Meyer term of the objective functional
l			Lagrange term of the objective functional
$\mathbb{S}_f$	$\subset$	$\mathbb{R}^{n_x}$	Target set, smooth manifold
ψ_s			Initial condition
n_{r_s}	$\in$	$\mathbb{N}$	Dimension of the initial condition ψ_s
ψ_f			Final condition
n_{r_f}	$\in$	$\mathbb{N}$	Dimension of the final condition ψ_f
τ	$\in$	$[0; 1]$	Normalized time
V			Optimal attainable value function
$\dot{V}$			Optimal value function time derivative
$\nabla_{x_s} V$			Gradient of V with respect to the initial state x_s

$\nabla^2_{x_s x_s} V$			Hessian of V with respect to the initial state x_s
H			Hamiltonian function
$\nabla_x H$			Gradient of H with respect to the state x
μ_{abn}	$\in$	$\mathbb{R}$	Abnormal multiplier
κ			Control as function of the state and adjoint

Nonlinear Optimization

n_z	$\in$	$\mathbb{N}$	Dimension of the optimization vector
n_p	$\in$	$\mathbb{N}$	Dimension of the perturbation vector
n_g	$\in$	$\mathbb{N}$	Number of constraints
n_a	$\in$	$\mathbb{N}$	Number of active constraints
z	$\in$	$\mathbb{R}^{n_z}$	Vector of primal optimization variables
f			Objective function
g			Constraint function
L			Lagrange function
μ	$\in$	$\mathbb{R}^{n_g}$	Vector of Lagrange multipliers
$a(z)$			Set of active constraints indices
g_a			Vector function of the active constraints
μ_a	$\in$	$\mathbb{R}^{n_a}$	Vector of active Lagrange multipliers

Discrete Optimal Control Processes and Direct Transcription

h	$\in$	$\mathbb{R}$	Integration step size
T_u	$\subset$	$[0;1]$	Set of discretization points of the control functions
T_x	$\subset$	$[0;1]$	Set of discretization points of the state functions
T_c	$\subset$	$[0;1]$	Set of discretization points of the path constraint functions
T_s	$:=$	$T_x \backslash \{1\}$	Shooting nodes
$\tau^{(i)}$	$\in$	T_u	Discrete point of the normalized time τ
l_u	$\in$	$\mathbb{N}$	Length of the control grid T_u
l_x	$\in$	$\mathbb{N}$	Length of the state grid T_x
l_c	$\in$	$\mathbb{N}$	Length of the constraint grid T_c
$u(\tau^{(i)})$	$\in$	$\mathbb{R}^{n_u}$	Control vector at $\tau^{(i)}$
$x(\tau^{(i)})$	$\in$	$\mathbb{R}^{n_x}$	State vector at $\tau^{(i)}$
$\tilde{x}(\tau^{(i)})$	$\in$	$\mathbb{R}^{n_x}$	State vector at $\tau^{(i)}$ obtained by integration of the control function
S^k			Shooting interval $[\tau^{(k)};\tau^{(k+1)}]$, $\tau^{(k)} \in T_x$
l_{S^k}	$\in$	$\mathbb{N}$	Number of control grid points in shooting interval S^k
d	$\in$	$\mathbb{R}$	Continuity defect of a state trajectory
D	$\in$	$\mathbb{R}^{(l_x-1)n_x}$	All continuity defects of all state trajectories
n_d	$\in$	$\mathbb{N}$	Number of state defect constraints
U	$\in$	$\mathbb{R}^{l_u} \times \mathbb{R}^{n_u}$	Control vectors at all $\tau^{(i)} \in T_u$
X	$\in$	$\mathbb{R}^{l_x} \times \mathbb{R}^{n_x}$	State vectors at all $\tau^{(i)} \in T_x$
$\tilde{X}_{S_k}$	$\in$	$\mathbb{R}^{l_{S^k}} \times \mathbb{R}^{n_x}$	State traj. from integration on shooting interval S^k
$\tilde{x}^{(i,j)}_{S_k}$	$\in$	$\tilde{X}_{S_k}$	Value of state func. j after $i-1$ integration steps in S^k
$\tilde{x}^{(i,:)}_{S_k}$	$\in$	$\tilde{X}_{S_k}$	State vector after $i-1$ integration steps in S^k

C	$\in$	$\mathbb{R}^{l_c n_c}$	Vector of the discretized path constraints
Ψ	$\in$	$\mathbb{R}^{n_{r_s}+n_{r_f}}$	Vector of the boundary conditions ψ_s and ψ_f
A	$\in$	$\mathbb{R}^{n_z} \times \mathbb{R}^{n_z}$	Diagonal matrix of the NLP variable scale factors
B	$\in$	$\mathbb{R}^{n_z} \times \mathbb{R}^{n_z}$	Diagonal matrix of the NLP constraint scale factors
$a_x^{(1)},\ldots,a_x^{(n_x)}$	$\in$	$\mathbb{R}$	Scale factors of the discretized state equations
$a_u^{(1)},\ldots,a_u^{(n_u)}$	$\in$	$\mathbb{R}$	Scale factors of the discretized control equations
$b_c^{(1)},\ldots,b_c^{(n_c)}$	$\in$	$\mathbb{R}$	Scale factors of the disc. path constraint equations
α	$\in$	$\mathbb{R}$	Scale factor of the objective function

Parametric Optimal Control Processes and Real-Time Optimal Control Approximation

p	$\in$	$\mathbb{R}^{n_p}$	Parameter vector
p_0	$\in$	$\mathbb{R}^{n_p}$	Nominal value of the parameter p
n_p	$\in$	$\mathbb{N}$	Dimension of the parameter vector p
q	$\in$	$\mathbb{R}^{n_g}$	Parameter vector that linearly enters the NLP constraint function
q_0	$\in$	$\mathbb{R}^{n_g}$	Nominal value of the parameter q
Δp	$:=$	$p - p_0$	Perturbation of the parameter p
Δq	$:=$	$q - q_0$	Perturbation of the parameter q
$\overset{\star}{z}$	$\in$	$\mathbb{R}^{n_z}$	Vector of optimal primal variables
$\overset{\star}{\mu}$	$\in$	$\mathbb{R}^{n_g}$	Vector of optimal dual variables
$\overset{\star}{z}_0 = \overset{\star}{z}(p_0)$	$\in$	$\mathbb{R}^{n_z}$	Optimal primal variables at the nominal parameter value
$\overset{\star}{\mu}_a(p_0)$	$\in$	$\mathbb{R}^{n_a}$	Optimal dual variables of active constraints at the nominal parameter value
$\overset{\star}{z}(p)$	$\in$	$\mathbb{R}^{n_z}$	Optimal primal variables at the disturbed parameter value
$\tilde{z}(p)$	$\in$	$\mathbb{R}^{n_z}$	Approximation of the optimal primal variables at the disturbed parameter value
$\tilde{u}(p)$	$\subset$	$\tilde{z}(p)$	Approximation of the optimal control sequence at the disturbed parameter value
$\tilde{x}(p)$	$\subset$	$\tilde{z}(p)$	Approximation of the optimal state trajectory at the disturbed parameter value
$\mathcal{N}_0(p_0)$	$\subset$	$\mathbb{R}^{n_p}$	Sensitivity neighborhood of p_0
$\tilde{p}^{(m,j)}$	$\in$	$\mathbb{R}$	Approximative value of $p^{(j)}$ at which $g^{(m)}$ switches from active to inactive or vice versa
$I^{(m,j)}$			Approximation of the interval around $p_0^{(j)}$ in which constraint $g^{(m)}$ does not switch from active to inactive or vice versa
$A^{(j)}(p_0)$			Approximation of the interval in which the active set remains unchanged for a perturbation in $p_0^{(j)}$
$\mathcal{A}(p_0)$	$\subset$	$\mathbb{R}^{n_p}$	Approximation of the neighborhood in which the active set remains unchanged for a perturbation in p_0
$\underset{\max}{\Delta p^{(j)}}$	$\in$	$\mathbb{R}$	Approximation of the maximal absolute deviation of $p^{(j)}$ from $p_0^{(j)}$ without causing a change of the active set

Closed Loop Near-Optimal Feedback in the Neighborhood of a Nominal Trajectory

$\boldsymbol{t}$	$\in$	$[0;\, \boldsymbol{t}_d]$	Closed loop time, defined as zero at the beginning of the process.
$\boldsymbol{t}_d$	$\in$	$\mathbb{R}^+$	Closed loop duration and terminal time (not known a priori)
$\boldsymbol{x}(\boldsymbol{t})$	$\subset$	$\mathbb{R}^{n_x}$	Closed loop state trajectory (not known a priori)
$\boldsymbol{t}^{(i)}$	$\in$	$[0;\, \boldsymbol{t}_d]$	Discrete closed loop time point
T_{cl}	$\subset$	$[0;\, \boldsymbol{t}_d]$	Discrete points $\boldsymbol{t}^{(1)}$,...,$\boldsymbol{t}^{(l_T)}$ of closed loop time
$\bar{p}(\boldsymbol{t}^{(i)})$	$\in$	$\mathbb{R}^{n_{\bar{p}}}$	Parameter vector at $\boldsymbol{t}^{(i)}$, excluding the closed loop state
$p(\boldsymbol{t}^{(i)})$	$\in$	$\mathbb{R}^{n_p}$	Parameter vector at $\boldsymbol{t}^{(i)}$, including the closed loop state
t_i	$\in$	$[0; t_{i,go}]$	Open loop time of $\mathrm{OCP}_{(5.45)}(p^{(i)})$
$t_{i,go}$	$\in$	$\mathbb{R}$	Duration of $\mathrm{OCP}_{(5.45)}(p^{(i)})$
τ_i	$\in$	$[0;\, 1]$	Normalized open loop time of $\mathrm{OCP}_{(5.45)}(p^{(i)})$
T_{nom}	$\in$	$[0;\, 1]$	Discrete normalized time points $\boldsymbol{\tau}^{(1)}$,...,$\boldsymbol{\tau}^{(l_p)}$
P_{nom}	$\subset$	$\mathbb{R}^{n_p}$	Parameters values at T_{nom}
$p_0^{(r)}$	$\in$	P_{nom}	Closest nominal parameter value
$\mathcal{N}^{(i)}$	$\subset$	$\mathbb{R}^{n_p}$	Convergence neighborhood of the RTS algorithm around $p_0(\tau^{(i)})$
$\mathcal{B}^{(i)}$	$\subset$	$\mathcal{U}^{(i)}$	Ball around $p_0(\tau^{(i)})$
$\mathcal{C}$	$\subset$	$\mathbb{R}^{n_p}$	Controllable space

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Motivation

More than ever before modern science relies on machines to accomplish incredible feats that humans could not achieve on their own. This is especially relevant for space exploration: Autonomous machines enable the exploration of distant celestial objects that are inaccessible to humans. Modern space missions rely on autonomous on-board computer programs to fulfill increasingly complex and challenging mission requirements. A prime example is the exploration of Mars: The Mars landers Viking (1976), Pathfinder (1996), MER (2003) and Phoenix (2008) were uncontrolled and had estimated landing ellipses of hundreds of kilometers diameter. The Mars Science Laboratory (2012) was the first mission to perform a guided and controlled entry, bringing down the landing ellipse to about ten kilometers. For a future human exploration of Mars a landing accuracy in the range of hundreds of meters is required. Many research fields must collaborate and contribute towards achieving this goal.

A particular important task falls to the on-board guidance and control system, which must adapt the flight path to state perturbations and unpredictable environmental effects, like e.g. atmospheric density changes, wind or dust. Often it is desired that the trajectory is optimal with respect to a performance criterion, e.g. fuel consumption or heat load, while at the same time the trajectory must satisfy possibly counteractive constraints. This can be formally described by an *optimal control problem.*

The solution of optimal control problems is based on the mathematical modeling of dynamic systems through differential equations. The solution of a general differential equation system is commonly obtained numerically with the help of computer programs. This is subject to the fields of *transcription methods* and *nonlinear optimization.* Solving a nonlinear optimization problem is a computationally demanding task, that is often not straightforward and is therefore usually supervised by humans. Space missions to date rely on precomputed reference trajectories and tracking controllers to compensate perturbations and to steer the vehicle back to the nominal state.

One approach to improve the performance of such a control system is to recompute the reference trajectory during flight, taking into account the perturbed system state. In space missions this approach is especially challenging because the computational power of the on-board computer system is very low compared to commercially available desktop computers and it might not be possible at all to solve the associated optimal control problem in the available amount of time.

This motivates the search for fast, alternative methods for the optimal control of nonlinear dynamic systems. One approach is to investigate the changes of the solution of an optimal control problem with respect to small changes in model parameters. This is the subject of *parametric sensitivity analysis*. So called *parametric sensitivity differentials* can be understood as the susceptibility of the optimal solution to perturbations in the system parameters. Under certain conditions the knowledge of the sensitivities enables a real-time adaption of the reference trajectory and the control sequence through a Taylor series expansion. At the same time questions are opened whether this approach can be used to synthesize an optimal closed loop control law.

1.2 Thesis Goals

The goal of this thesis is the prototyping of a Mars entry guidance and control system that uses an in-flight re-computation of the reference trajectory to achieve a precise landing. A driving requirement is the computational feasibility of the guidance and control algorithm on space qualified (or equivalent) computing hardware. The target system environment is the ESA Reference Architecture System Test-Bed for Avionics (RASTA).

This thesis seeks to connect the practical engineering challenges of atmospheric entry guidance to recent theoretical mathematical methods from nonlinear optimization and optimal control. The atmospheric entry problem is formulated as parametric optimal control process, which is led back to a parametric nonlinear program via direct transcription. To facilitate a computationally feasible trajectory adaption, we focus on transferring and expanding recent results in sensitivity theory, which enable a trajectory adaption without requiring expensive derivative computations.

A main objective is to derive a closed loop feedback law for the entry dynamics to enable a repeated, clock based, re-computation of the optimal control function and the optimal state trajectory.

The feasibility and performance of the proposed guidance and control algorithm shall be assessed in the frame of the Mars Precision Lander (MPL) study of the European Space Agency. A numerical and stochastic analysis of the scenario shall be performed, and the feasibility of the guidance and control algorithm shall be demonstrated on the RASTA.

1.3 Thesis Structure

In the remainder of Chapter 1 a short introduction into the basics of a guidance, navigation and control (GNC) framework is given. The related terminology and the main tasks of the GNC software modules are explained, addressing readers who are not familiar with avionics. Furthermore a high level overview of the atmospheric entry sequence is given.

In chapter 2 the atmospheric entry dynamics and the Martian environment are modeled mathematically. The state vector and the flight dynamic equations are introduced as well as the aero- and thermodynamic constraints. The principle of bank angle control is illustrated, focusing on small, low lift entry capsules.

Chapter 3 reviews some classical entry guidance and control concepts. We focus in particular on the drag dynamics and discuss earlier related work. We incorporate the collected ideas in the synthesis of a drag control law based on the principle of feedback linearization, as firstly suggested by Mease et al. At the end of the chapter we summarize the control related challenges of atmospheric entry.

Chapter 4 shifts the focus to optimal control and nonlinear optimization. In preparation for the following chapters the most important definitions and theoretical foundations of dynamic and static optimization are stated. We discuss the direct transcription of infinite dimensional optimal control processes into finite dimensional nonlinear programs and introduce the notation to which we refer during the next chapters.

In Chapter 5 we explore parametric sensitivity analysis of nonlinear programs with the goal to achieve real-time optimal control. We apply a real-time capable solution approximation algorithm suggested by Büskens to an instructional example. We investigate the influence of different discretization schemes on the solution approximation with the goal to generalize the findings in order to identify the most advantageous formulation for the atmospheric entry problem and we discuss the limitations of the chosen approach. The central result of this chapter is the synthesis of a sub-optimal closed loop feedback law for nonlinear dynamic systems.

In Chapter 6 we formulate the ESA Mars Precision Lander scenario as parametric optimal control problem. The nominal solution and its parametric sensitivities are analyzed and we apply the proposed feedback law to the entry problem. We compare the achieved region of controllability with an approximation of the reachable set. This leads to the combination of the feedback law with the drag tracking controller developed in Chapter 3 into a two degree-of-freedom guidance system. The combined guidance system is numerically tested and the performance is stochastically analyzed in a Monte Carlo campaign. Finally, it is demonstrated that the novel guidance and control algorithm is real-time capable on the RASTA through a LEON2 processor-in-the-loop test.

In Chapter 7 the performed work is summarized and conclusions are stated.

1.4 Guidance, Navigation and Control

Spacecraft on-board software can be complex[1] and have many operational modes. The part of the on-board software that is responsible for controlling a spacecraft is the guidance, navigation and control system. According to the European Cooperation for Space Standardization (ECSS) these terms can be defined as follows:

Guidance is the determination of the desired state and the establishment of the nominal state trajectory.

Navigation is the determination of the state.

Control is the manipulation of forces to steer the current state towards the desired state.

For control design the flight dynamics are commonly separated in the translational motion and the rotational motion. Translational and rotational motion often happen on different time scales and the separation considerably simplifies the analysis. As a result the control loop of flight systems is cascaded into a guidance loop and an attitude control loop. The data flow of such a control system is sketched in Figure 1.1. The left part shows a modularized

1 The on-board software of the Mars Science Laboratory has about 1.3 million lines of code [Cox10].

schematic of the on-board software, the right part is the dynamic system which is to be controlled. The attitude control loop tracks the guidance command. The guidance function is concerned with the translational motion, while the rotational dynamics are handled by the attitude control system.

This thesis synthesizes, implements and validates a guidance algorithm for Mars atmospheric entry, i.e. a solution of the translational equations of motion is determined. The context within the overall entry, descent and landing sequence, is described in the next section.

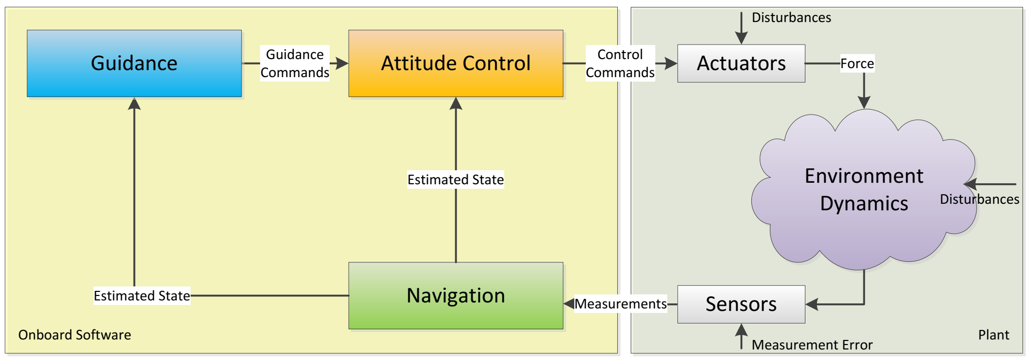


Figure 1.1: Guidance, navigation and control loop

1.5 Mars Entry, Descent and Landing

Unpowered atmospheric entry relies on the reduction of the spacecraft's speed through the drag generated inside the atmosphere. The gravity on Mars is approximately 62% of Earth gravity, but the Mars atmosphere only has a density of about 1% of the Earth's atmosphere.

The high initial entry speed causes the gases in front of the vehicle to be compressed into a shock wave. The compression is so rapid that it causes an immense amount of radiant heat. Normal materials melt under such conditions, therefore atmospheric entry systems require a dedicated heat shield and thermal protection system.

The thin atmosphere of Mars is barely enough to reduce the vehicle's speed sufficiently. Mars entry vehicles tend to decelerate at low altitude and may never reach the subsonic terminal velocity of Earth aerodynamic vehicles. A critical factor is the atmospheric density, which varies with solar irradiation and the seasons across a Martian year, because it depends on the temperature. Another influential factor is the amount of dust in the atmosphere, which depends upon local weather conditions. The uncertainty in the Martian atmosphere is challenging, because it requires a control design that does not depend on the correctness of a nominal atmosphere profile. In addition the entry trajectory must respect the aerodynamic and thermodynamic limits of the vehicle.

1.5.1 Atmospheric Entry of the Mars Science Laboratory

This section illustrates the entry, descent and landing (EDL) sequence of the Mars Science Laboratory as described in [Pra08]. The sequence starts with the separation of the aeroshell from the cruise stage. Control weights are jettisoned to create an offset of the center of mass (CoM) from the longitudinal symmetry axis. The offset of the CoM allows to control the direction of the lift vector by rotation around the velocity axis. This is explained in detail in Section 2.3.

The atmospheric entry starts at the border of the sensible atmosphere. The guided entry and maneuvering phase lasts until the opening of the parachute. The entry guidance is activated as soon as the sensed drag acceleration is above a certain threshold. During the entry phase the guidance system relies solely on inertial navigation. The navigation system computes a real-time state estimate based on an inertial measurement unit. It is particularly important that the state at the beginning of the entry is known accurately, as the dynamic system is not observable during that phase. The aeroshell first encounters peak heating followed by peak deceleration and up to 99% percent of the vehicles kinetic energy are dissipated through heat into the atmosphere. The guided entry phase lasts for about 4 minutes, during which the majority of downrange is covered.

Shortly before the parachute deployment the CoM offset is eliminated through the jettisoning of additional control weights. The supersonic parachute will reduce the speed from about 450 m/s to 100 m/s. Meanwhile the heat shield is jettisoned and the vehicle begins collecting radar and imagery data of the landing zone.

The powered descent stage begins with the separation of the back shell and a short free fall of the powered descent vehicle. At about 20 meters above ground at a vertical velocity of 0.75 m/s the skycrane maneuver is initiated and eventually the rover is lowered to the surface. After the touchdown of the rover, the powered descent vehicle flies to a safe distance and the EDL sequence is completed.

1.5.2 EDL Requirements for Future Mars Missions

The primary goals for EDL systems of future Mars missions are to

1. increase the landed mass and to
2. improve the landing accuracy.

For robotic exploration an increase in landed mass will result into a reduction of the cost per kilogram of payload mass. Landed mass can also be traded to access a landing zone at a higher elevation level. The increased accuracy is a requirement to access specific scientifically interesting areas, e.g. valleys, rims or rough terrain. Furthermore these goals are stepping stones to enable a human exploration of Mars in the future. Braun and Manning [Bra06] anticipate that the “[. . .] human exploration of Mars call[s] for the landing of 40-80 metric tons surface elements at scientifically interesting locations within close proximity (10’s of meters) of pre-positioned robotic assets. These plans require a simultaneous two order of magnitude increase in landed mass capability, four order of magnitude increase in landed accuracy, and an entry, descent and landing operations sequence that may need to be completed in a lower density (higher surface elevation) environment. This is a tall order that will require the space qualification of new EDL approaches and technologies.”

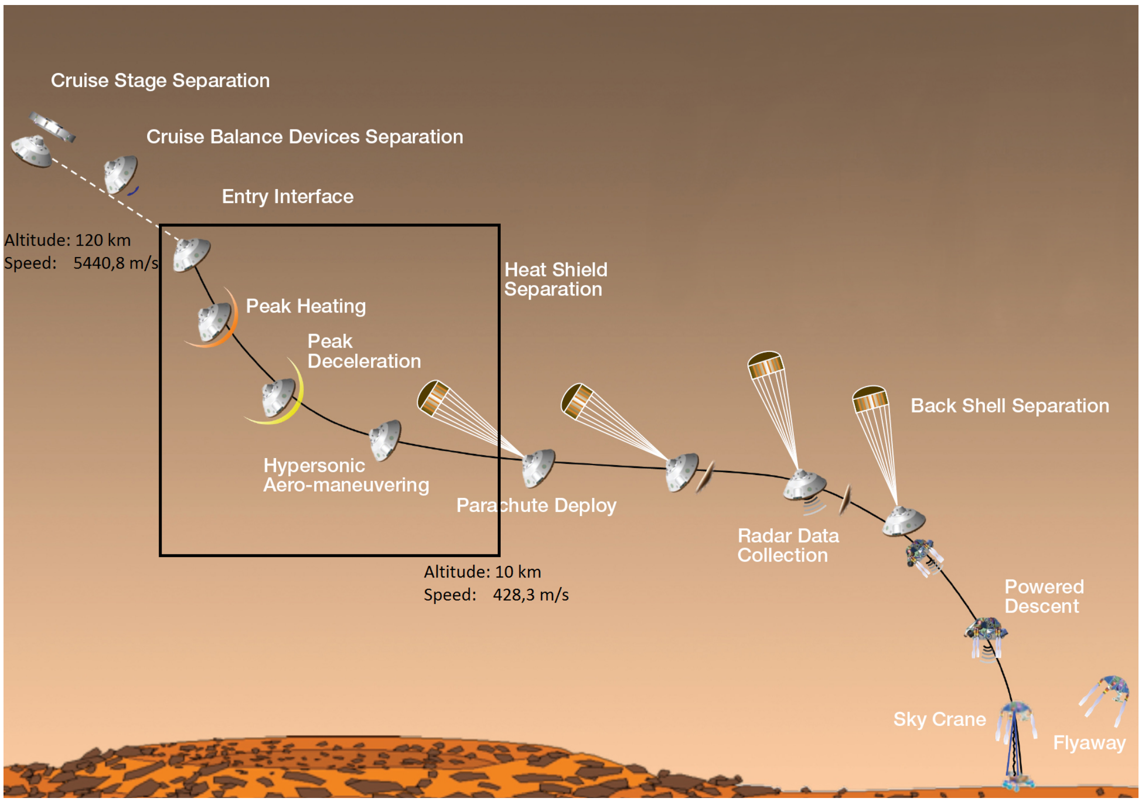


Figure 1.2: MSL entry, descent and landing sequence.
Image credit: NASA, MSL press release 2012 (modified)

CHAPTER 2

A Mathematical Model for Planetary Atmospheric Flight

The mathematical solution of control problems requires a mathematical model of the environment and the dynamics. A model should describe the real world as accurately as necessary while being as simplistic as possible. This chapter introduces the models and dynamics which are used in the derivation of the guidance and control laws. At some points we mention improvements or additions to the models that are required to create a more accurate simulation environment for the purpose of testing the control algorithms.

2.1 Aerodynamic Force

The most influential factors in planetary atmospheric flight are the atmosphere itself and the gravity of the planet. To describe the influence of the atmosphere we must understand the aerodynamic force that acts on an object exposed to an airflow. To this end we consider the dynamic pressure q, which measures the kinetic energy per unit volume of particles. According to [Vin80] the dynamic pressure depends on the atmospheric density ρ and the relative speed v_{atm} between atmosphere and object.

$$q = \frac{1}{2}\rho v_{\text{atm}}^2 \tag{2.1}$$

Dynamic pressure is measured in newton per square meter or pascal, i.e. $\frac{\text{N}}{\text{m}^2} = \text{Pa}$. The magnitude of the aerodynamic force on an object moving through the atmosphere is proportional to the dynamic pressure. The force component in direction of the flow is called drag F_D and the force component perpendicular to the flow is called lift F_L. The deflection properties of surfaces with complex shape are described using drag and lift coefficients C_D and C_L. The drag and lift forces are

$$F_D = qSC_D = \frac{1}{2}\rho v_{\text{atm}}^2 SC_D, \tag{2.2a}$$

$$F_L = qSC_L = \frac{1}{2}\rho v_{\text{atm}}^2 SC_L, \tag{2.2b}$$

where S is the reference size of the surface exposed to the flow.

If the atmosphere is at rest and if there is no wind, then the velocity with respect to the atmosphere v_{atm} is equal to the velocity v_{gnd} with which the vehicle moves with respect to the ground. The vector relationship between atmosphere-velocity and ground-velocity is given by

$$v_{\text{atm}} = v_{\text{gnd}} - v_{\text{wnd}}, \tag{2.3}$$

where v_{wnd} is the velocity vector of the atmosphere with respect to the ground, i.e. the wind. In the model for guidance and control we assume $v_{\text{wnd}} = 0$ such that

$$v_{\text{atm}} = v_{\text{gnd}} = v. \tag{2.4}$$

2.2 Atmosphere Model

The aerodynamic forces depend on the atmospheric density ρ. The density is a function of the atmospheric pressure, the temperature and the composition of the atmosphere. From these properties the density can be obtained using the ideal gas law, but commonly this complicated relation is simplified by mapping density against altitude h. The most common and simple approximation is the exponential atmosphere

$$\rho_{\exp}(h) = \rho_0 e^{-\frac{h}{h_s}}, \tag{2.5}$$

where ρ_0 is the density at a reference level (e.g. sea level) and h_s is the so called scale height, the vertical distance at which the density decreases by a factor of $\frac{1}{e}$. But the exponential model is not very accurate.

A more accurate approach is to numerically interpolate the density between sufficiently many discrete altitude values. The density-altitude relation is obtained from the European Mars Climate Database [Lew99]. The density is interpolated using cubic splines.

Another important atmospheric property is the local speed of sound c, which mainly depends on pressure, density and thus temperature. This relationship is likewise simplified by using an altitude profile. The function $c(h)$ is obtained analogously by cubic spline interpolation.

The local speed of sound allows the computation of the Mach number M as a function of velocity and altitude.

$$M = \frac{v}{c(h)} \tag{2.6}$$

With the Mach number speeds can be compared, taking into account differences in pressure, density and temperature, via the normalization through the speed of sound. The Mach number is used to determine the aerodynamic coefficients of the entry vehicle, as explained in the next section.

2.3 Entry Vehicle Model

The entry vehicle is a small capsule with a diameter of 2.8 meters and a total wet mass of 1050 kilograms. The capsule is trimmed at a fixed angle of attack α. At the trim angle the heat shield nose is tilted below the velocity vector to generate an upwards directed lift force, as shown in Figure 2.1. The vertical component of the lift vector can be controlled by rotating, or *banking*, the capsule around the velocity axis. This is facilitated by multiple small thrusters mounted on the aft section. The control of the vertical lift vector affects the sink velocity and thus how fast the vehicle descents into the denser regions of the atmosphere, which in turn controls the drag. Drag controls the longitudinal range, or *downrange*, of the flight. This is explained in detail in Chapter 3.1.

Decoupling the translational and rotational motion of the vehicle allows for some simplifications concerning the attitude of the entry vehicle for the purpose of guidance design, as for example detailed in [Hir09]. The important simplifying assumptions are:

1. There is no side slip, thus the velocity vector always lies in the vehicle's vertical plane of symmetry.
2. The vehicle is statically and dynamically stable. The pitching motion is neglected and the angle of attack is treated as constant.

The aerodynamic coefficients C_D and C_L are functions of the Mach number and the angle of attack. Because the angle of attack is assumed to be constant, this leaves C_D and C_L as a function of the Mach number only. The aerodynamic coefficients used in this thesis are provided by ESA from the aerodynamic coefficient database of the Mars Precision Lander study. Analogously to the density profile the coefficients are obtained at discrete Mach numbers and cubic spline interpolation is used to obtain the functions $C_D(M)$ and $C_L(M)$.

The average lift over drag ratio of the entry capsule at hyper- and supersonic speeds at the trim angle of attack is about 0.2. This lift ratio is very small in comparison to other lifting bodies. As a result the control reserve before going into saturation, i.e. full lift-up or lift-down, is small, which is a great challenge for the guidance and control design. The capsule reference parameters are summarized in Table 2.1.

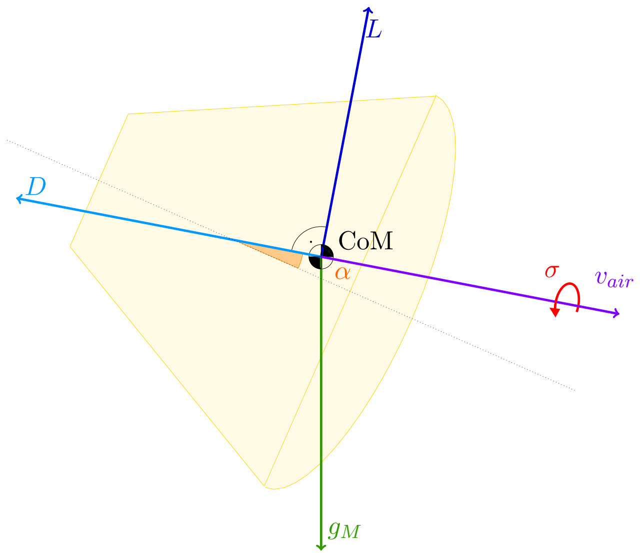


Figure 2.1: Lateral view of the entry capsule and the aerodynamic lift and drag forces.

Table 2.1: Capsule reference parameters

Symbol	**Property**	**Value**
m	Mass	1050 kg
d	Diameter	2.8 m
r_n	Effective nose radius	0.7 m
S	Aerodynamic reference area	$6.1575\,\mathrm{m}^2$

2.4 Gravity Model

The planet Mars is modeled as a sphere with radius r_M. We assume a uniform mass distribution, which allows to model the gravity as a central field around a point having the mass of the planet. The gravitational acceleration towards the planetary center is according to Newton's law

$$g(r) = \frac{\mu_M}{r^2}, \tag{2.7}$$

where μ_M is the gravitational constant multiplied by the mass of Mars. The radial distance r of the spacecraft from the planetary center is the sum of the planetary radius r_M and the altitude h.

$$r = r_M + h \tag{2.8}$$

The relevant reference values are summarized in Table 2.2. They result into a gravitational acceleration of $3.72\,\frac{\mathrm{m}}{\mathrm{s}^2}$ at the surface, which is about 38 % of the gravity acceleration on Earth.

A more accurate approximation of the gravitational acceleration can be obtained by modeling the nonuniform mass distribution. This requires to take into account the so called zonal harmonic coefficients. These correction terms depend on the geodetic latitude. The influence of these coefficients is very small compared to the aerodynamic forces and the uncertainty in the aerodynamic model. Hence this more accurate model is only used for simulation and not for the derivation of the control laws, because the minor accuracy gain does not justify the increase in complexity.

Table 2.2: Mars reference parameters

Symbol	Property	Value
r_M	Mars radius	3 393 940 m
μ_M	Mars gravity constant	$4.282\,828\,29 \cdot 10^{13}\,\frac{\mathrm{m}^3}{\mathrm{s}^2}$
Ω_M	Mars rotation rate	$7.088\,218 \cdot 10^{-5}\,\frac{\mathrm{rad}}{\mathrm{s}}$

2.5 Translational Equations of Motion

In this section the motion of a spacecraft over a rotating, spherical planet is described. We assume the atmosphere rotates uniformly with the planet, hence there is no wind. The state of the spacecraft is defined with respect to a set of reference coordinate frames. Their definitions are given in Appendix A.

The planet centered inertial frame (PCI) is considered inertial for the purpose of vector derivatives. The position of the spacecraft is expressed with respect to the planet centered, planet fixed (PCPF) coordinate system, which rotates about the z_{PCI} axis with the angular velocity Ω_M. The position is defined using the spherical coordinates r, λ, φ as described in Table 2.3.

The trajectory axis (TA) frame is used to express the flight-path velocity vector (relative to the ground) with respect to the north-east-down (NED) frame. The trajectory axis frame relates the local vertical plane (spanned by z_{NED} and the velocity vector v) to the NED frame by the flight-path inclination angle γ and the heading angle χ. For the definition of the spacecraft state we prefer to use the altitude h instead of the radial distance r, their simple relation is given by (2.8). The state vector for translational motion at time t is thus

$$x(t) = (h, \lambda, \varphi, v, \gamma, \chi). \tag{2.9}$$

Table 2.3: Definition of the state variables

Symbol	Property	Description	Unit	Domain
h	Altitude	over the spherical planet approximation.	m	$\mathbb{R}$
λ	Longitude	is measured positive from the zero meridian towards east.	rad	$[-\pi, \pi]$
φ	Geocentric latitude	is measured positive from the equatorial plane towards north.	rad	$[-\frac{\pi}{2}, \frac{\pi}{2}]$
v	Velocity	with respect to the ground.	$\frac{\text{m}}{\text{s}}$	$\mathbb{R}_0^+$
γ	Flight path angle	is measured from the local horizontal plane towards the velocity vector; positive above the horizontal plane.	rad	$[-\frac{\pi}{2}, \frac{\pi}{2}]$
χ	Heading angle (azimuth)	is measured from X_{ned} towards the projection of the velocity vector on the horizontal plane; positive towards the east.	rad	$[-\pi, \pi]$

2.5.1 Time Domain

The spatial relation of the states is shown in Figure 2.2. According to trigonometric relationships the velocity vector $\vec{v}$ can be expressed in the NED system as

$$\vec{v}_{NED} = \begin{pmatrix} v\cos\gamma\cos\chi \\ v\cos\gamma\sin\chi \\ v\sin\gamma \end{pmatrix} \tag{2.10}$$

and the position vector is

$$\vec{r}_{NED} = \begin{pmatrix} 0 \\ 0 \\ -r \end{pmatrix}. \tag{2.11}$$

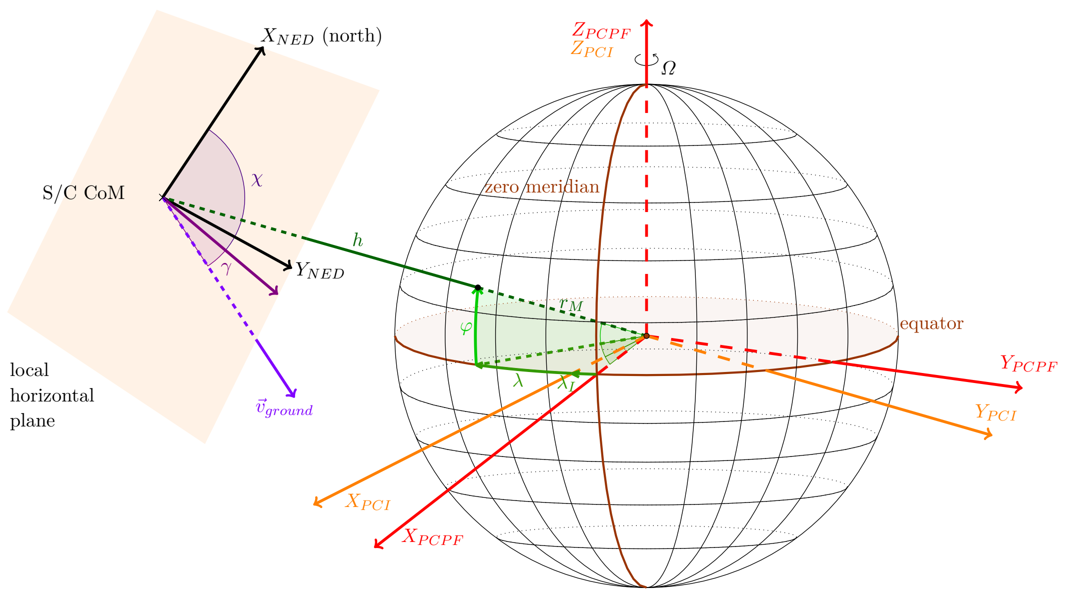


Figure 2.2: Spatial relation of the state variables

The time derivative of the position vector in the NED system can be obtained by taking into account the rotation of the NED system with respect to the PCI system. According to [Wei10, p. 98] the angular velocity of the NED system relative to the PCI system is

$$\Omega_{NED}^{PCI} = \begin{pmatrix} \dot{\lambda}\cos\varphi \\ -\dot{\varphi} \\ -\dot{\lambda}\sin\varphi \end{pmatrix}. \tag{2.12}$$

and the time derivative of the position vector in a rotating system is

$$\left(\frac{d}{\mathrm{d}t}\right)_{PCI} \vec{r}_{PCI} = \left(\frac{d}{\mathrm{d}t}\right)_{NED} \vec{r}_{NED} + \Omega_{NED}^{PCI} \times \vec{r}_{NED}. \tag{2.13}$$

This gives

$$\left(\frac{d}{\mathrm{d}t}\right)_{PCI} \vec{r}_{PCI} = \begin{pmatrix} 0 \\ 0 \\ \dot{r} \end{pmatrix}_{NED} + \begin{pmatrix} \dot{\lambda}\cos\varphi \\ -\dot{\varphi} \\ -\dot{\lambda}\sin\varphi \end{pmatrix}_{NED} \times \begin{pmatrix} 0 \\ 0 \\ -r \end{pmatrix}_{NED} = \begin{pmatrix} \dot{\varphi}\, r \\ \dot{\lambda}\, r\, \cos\varphi \\ \dot{r} \end{pmatrix}_{NED} \tag{2.14}$$

Equating (2.14) with (2.10) leads to

$$\dot{h} = v\sin\gamma \tag{2.15a}$$

$$\dot{\lambda} = v\frac{\cos\gamma\sin\chi}{r\cos\varphi} \tag{2.15b}$$

$$\dot{\varphi} = v\frac{\cos\gamma\cos\chi}{r}. \tag{2.15c}$$

This are the time domain kinematic equations.

The dynamic relationships between v, γ and χ are obtained based on the force equation, Newton's second law of motion. Newton's law is only valid in an inertial frame, thus the forces are calculated in the PCI system.

$$m\vec{a}_{PCI} = m\frac{\mathrm{d}^2\vec{r}_{PCI}}{\mathrm{d}t^2} = \vec{D}_{PCI} + \vec{L}_{PCI} + m\vec{g}_{PCI} \tag{2.16}$$

Here the previously introduced models take effect. The dynamic equations are obtained by transforming (2.16) to the NED system and solving for $\frac{\mathrm{d}v}{\mathrm{d}t}$, $\frac{\mathrm{d}\gamma}{\mathrm{d}t}$ and $\frac{\mathrm{d}\chi}{\mathrm{d}t}$. For a detailed derivation we refer to [Wei10, Chap. 6]. The time domain dynamic equations are

$$\begin{aligned}
\dot{v} = &-D - g\sin\gamma && \text{(2.17a)}\\
&+ \Omega_M^2 r\cos\varphi\left(\sin\gamma\cos\varphi - \sin\varphi\cos\gamma\cos\chi\right),\\
\dot{\gamma} = &\frac{L}{v}\cos\sigma + \left(\frac{v}{r} - \frac{g}{v}\right)\cos\gamma && \text{(2.17b)}\\
&+ 2\Omega_M 2\cos\varphi\sin\chi + \Omega_M^2\frac{r}{v}\cos\varphi\left(\cos\gamma\cos\varphi - \sin\gamma\sin\varphi\cos\chi\right),\\
\dot{\chi} = &\frac{L\sin\sigma}{v\cos\gamma} + \frac{v}{r}\cos\gamma\sin\chi\tan\varphi && \text{(2.17c)}\\
&- 2\Omega_M\left(\tan\gamma\cos\varphi\cos\chi - \sin\varphi\right) + \Omega_M^2\frac{r}{v\cos\gamma}\sin\varphi\cos\varphi\sin\chi,
\end{aligned}$$

where $L = \frac{F_L}{m}$ and $D = \frac{F_D}{m}$ are the lift and drag acceleration. The terms containing the factor Ω_M are due to the rotation of the planet. Thus by setting $\Omega_M := 0$ the dynamic equations for a spherical non-rotating planet are obtained.

2.5.2 Energy Domain

It can be advantageous for control design, in particular in the context of trajectory tracking, to eliminate time from the equations of motion by replacing it with a different independent variable. If no temporal context must be satisfied, time tracking imposes an unnecessary constraint. A less restrictive tracking strategy can be obtained using an energy measure which is directly based on the state variables.

Definition 2.18 (Specific Total Mechanical Energy)

The total mechanical energy is the sum of kinetic and potential energy.

$$E_{\text{tot}} = E_{\text{kin}} + E_{\text{pot}} = \frac{1}{2}mv^2 + \int \frac{m}{r^2}GM\ \mathrm{d}t = \frac{1}{2}mv^2 - \frac{m}{r}GM + c \tag{2.19}$$

G is the gravitational constant and M is the planet mass. Energy per unit mass is called specific energy. If the potential energy is normalized to zero at the planets surface, the specific total mechanical energy is

$$E = \frac{v^2}{2} - \left(\frac{GM}{r_p + h} - \frac{GM}{r_p}\right), \tag{2.20}$$

where r_p is the planet radius. We refer to E abbreviating as 'energy'.

The time derivative of energy is found by differentiating v and h and using the Newtonian gravity law $g = GM/(r_p + h)^2$.

$$\begin{aligned}\dot{E} &= v\dot{v} - \left(\frac{-GM}{(r_p + h)^2}\dot{h}\right) \\ &= v\dot{v} + g\dot{h} \\ &= -vD + \Omega_M^2 r \cos\varphi\,(\sin\gamma\cos\varphi - \sin\varphi\cos\gamma\cos\chi) \\ &= -vD + \Phi\end{aligned} \tag{2.21}$$

For a non rotating planet we have $\Omega_M = \Phi = 0$. The derivative then simplifies to $\dot{E} = -vD$. In this case $E(t)$ is strongly monotonically decreasing, because during entry flight $v(t) > 0$ and $D(t) > 0$. This makes energy a suitable candidate for replacing time as the independent variable. Note that for a rotating planet model $E(t)$ is not necessarily monotonically decreasing.

For the substitution of time with energy the equations of motion are transformed according the the chain rule

$$\frac{d(\cdot)}{\mathrm{d}E} = \frac{d(\cdot)}{\mathrm{d}t}\frac{\mathrm{d}t}{\mathrm{d}E}. \tag{2.22}$$

The kinematic and dynamic equations with respect to energy are

$$h' = -\frac{1}{D}\sin\gamma \tag{2.23a}$$

$$\lambda' = -\frac{1}{D}\frac{\cos\gamma\sin\chi}{r\cos\varphi} \tag{2.24a}$$

$$\varphi' = -\frac{1}{D}\frac{\cos\gamma\cos\chi}{r} \tag{2.25a}$$

$$v' = \frac{D + g\sin\gamma}{Dv} \tag{2.25b}$$

$$\gamma' = -\frac{L}{D}\frac{\cos\sigma}{v^2} - \left(\frac{v}{r} - \frac{g}{v}\right)\frac{\cos\gamma}{Dv} \tag{2.25c}$$

$$\chi' = -\frac{L}{D}\frac{\sin\sigma}{v^2\cos\gamma} - \frac{1}{Dr}\cos\gamma\sin\chi\tan\varphi. \tag{2.25d}$$

A major difficulty in control design for entry dynamics is that the time domain equations of motion are strongly nonlinear around the peak deceleration, i.e. when the energy derivative $\dot{E}$ is steep. The substitution of time with energy is a nonlinear state transformation that corresponds to a stretching of the dynamics when the energy rate is high, and to a compression when the energy rate is low. This works against the strongest nonlinearity in the system, which can be exploited for analysis and control design, as we will see later.

2.6 Path Constraints

The ability of the entry vehicle to withstand heat is determined by the thermal protection system. The system's heat tolerance is defined in terms of the heat flux $\dot{Q}$. An approximation of the heat flux is given by the Sutton-Graves-Equation, based on the effective nose radius r_n of the heat shield and a parameter k_p depending on the composition of the atmosphere. For the Martian atmosphere we have $k_p = 1.9027 \cdot 10^{-4}\,\sqrt{\mathrm{kg}}/\mathrm{m}$, comparatively for the Earth it would be $k_p = 1.75 \cdot 10^{-4}\,\sqrt{\mathrm{kg}}/\mathrm{m}$.

$$\dot{Q} = k_p \frac{\sqrt{\rho(h)}}{r_n} v^3 \tag{2.26}$$

The tolerance of the vehicle to structural load is defined in terms of dynamic pressure

$$d = \frac{1}{2}\rho(h)v^2 \tag{2.27}$$

and the load factor

$$n = \frac{\sqrt{D^2 + L^2}}{g_E}. \tag{2.28}$$

The limits for these values depend on numerous factors, but a lighter structure and heat shield will generally result into reduced tolerance. It is an obvious design goal for the entry trajectory to keep the heat flux, the dynamic pressure and the load factor low to allow for a lighter structure/heat shield to maximize the potential payload.

Let $\dot{Q}_{\max}$, $d_{\max}$ and $n_{\max}$ be the system's limit values. Note that all constraints are functions of only altitude and velocity (at a constant angle of attack). We are interested in determining the altitude and velocity profiles in which the constraints are at the limits. This is especially easy in the energy domain because the energy E constitutes a relationship between altitude and velocity. As an example consider the constraint $\dot{Q}_{\max}$. The pointwise (numerical) solution of the nonlinear equation system

$$E = \frac{v^2}{2} - \left(\frac{GM}{r_p + h} - \frac{GM}{r_p}\right), \quad E > 0,\, h > 0,\, v > 0, \tag{2.29a}$$

$$\dot{Q}_{\max} = k_p \frac{\sqrt{\rho(h)}}{r_n} v^3. \tag{2.29b}$$

for energy levels $E \in [E_0; E_f]$ determines the critical altitude profile $h_{\dot{Q}_{\max}}(E)$ and the critical velocity profile $v_{\dot{Q}_{\max}}(E)$. The critical values for the other constraints can be obtained analog. The combination of all critical profiles determines the feasible space (also called entry corridor) for the trajectory. Note that from the critical altitude and velocity profiles the critical drag profile can be obtained. This allows to represent all path constraints in the drag-energy domain. This is a major advantage for trajectory planning, because shaping a trajectory to respect the drag constraint is easier than to take into account multiple constraints on multiple states.

2.7 Aerodynamic Attitude and Wind

The aerodynamic forces depend on the relative air-flow, i.e. they must be calculated with respect to the atmosphere relative speed v_{atm}. But the equations of motion instead refer to the ground speed v_{gnd}. If the atmosphere above the ground is not at rest ($v_{\mathrm{gnd}} \neq v_{\mathrm{atm}}$), additional transformations and coordinate frames are required, as defined in Appendix A. The corresponding aerodynamic attitude angles are defined in Table 2.4 and their spatial relationships are shown in Figure 2.3.

For the derivation of the guidance laws this problem is simplified by assuming $v = v_{\mathrm{gnd}} = v_{\mathrm{atm}}$.

For the correct derivation of the influence of wind in a simulation environment, it is necessary to calculate the atmosphere relative velocity and the resulting aerodynamic forces, and then transform them back to ground speed related forces, for details we refer to [Moo94, Chap. 6.2].

Table 2.4: Aerodynamic attitude angles

Symbol	Property	Description	Unit	Domain
α	Angle of attack	is measured in the vehicle's vertical plane of symmetry from the the X_B-axis towards the projection of the air-path velocity vector v_{atm}.	rad	$[-\pi, \pi]$
β	Side slip angle	is measured from the vehicle's vertical plane of symmetry towards the air-path axis X_A.	rad	$[-\pi, \pi]$, $\beta := 0$
σ	Bank angle	is measured from Z_{NED} towards the vehicle's vertical plane of symmetry, when rotating around the air-path axis.	rad	$[-\pi, \pi]$

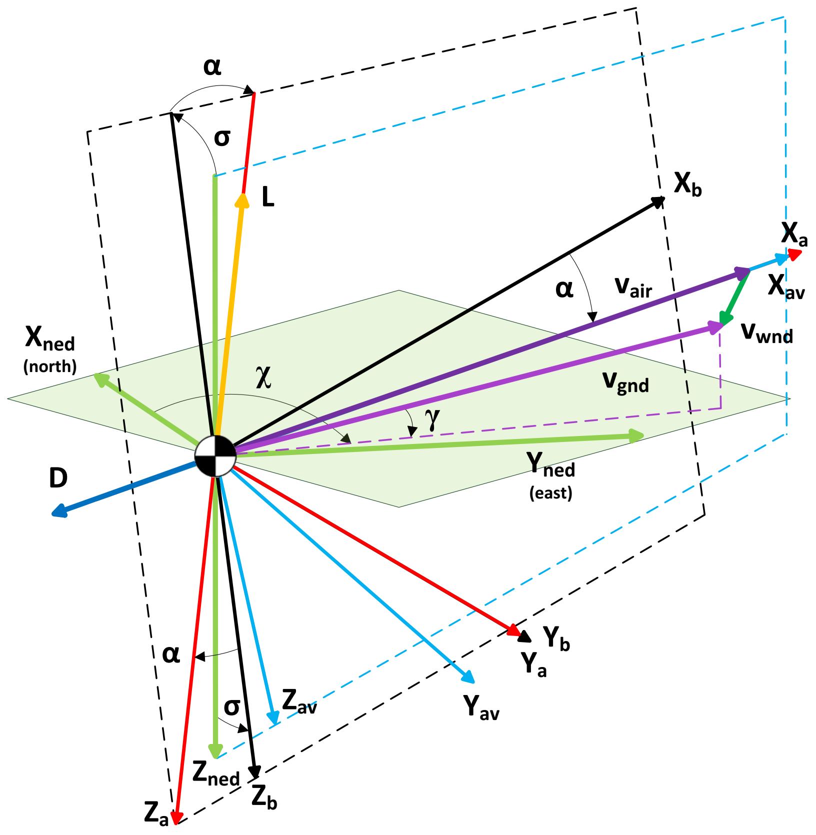


Figure 2.3: Air path angles assuming $\beta = 0$.

CHAPTER 3

Entry Guidance and Control Concepts

In this chapter some classical entry guidance and control concepts are studied. Tracking methods drive the error between a reference state and the actual state to zero, usually based on a closed form feedback control law. Prediction based methods use analytic or numeric strategies to predict the future trajectory online and then use a gradient based approach to find control commands that eliminate the error between the predicted trajectory and the desired target. The control laws of prediction based methods can often not be expressed in closed form, but rather as an algorithm. This relatively new field is also referred to as computational guidance. The combination of prediction based trajectory adaption and tracking leads to a cascaded two degree-of-freedom control design.

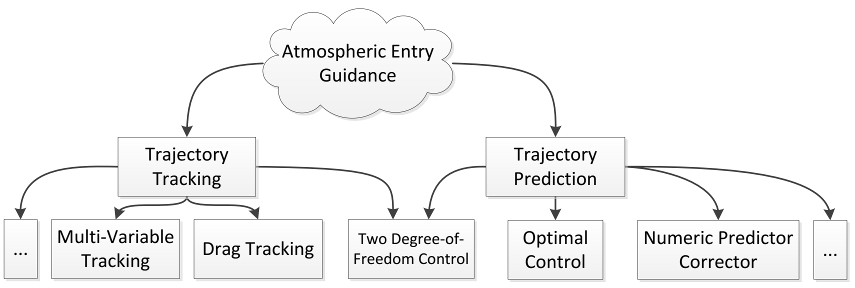


Figure 3.1: Selected approaches to entry guidance & control

Out of the methods shown in Figure 3.1 *drag tracking* has had the most operational use. Different variants of this technique have been used for Apollo, the Space Shuttle, and the Mars Science Laboratory. The main part of this chapter will therefore focus on drag tracking. To understand the intimacies of this method, we synthesize a drag feedback law in the next section, based on the work of Mease, Kremer, Bharadwaj, Tu and Saraf. In the second section we briefly introduce some alternative approaches. In the last section of the chapter we summarize major theoretical challenges of atmospheric entry guidance algorithms.

3.1 Drag Tracking

The aerodynamic forces can be manipulated to control the entry dynamics. The entry capsule's control system does not allow to influence drag directly (there are no flaps), but it allows to control the lift vector, respectively its vertical component, through bank angle modulation. Controlling lift allows a (delayed) control of drag, which in turn allows to control the range. This dependency chain is shown in Figure 3.2.

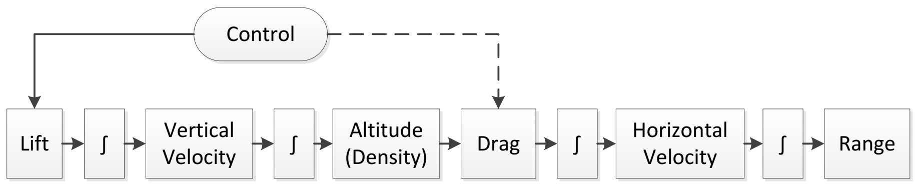


Figure 3.2: Relationship between the aerodynamic forces and range

The modulation of the bank angle however also alters the horizontal lift vector component and thus influences the lateral motion and the trajectory curvature. Drag tracking strategies solve the trajectory length problem and the trajectory curvature problem separately. In the following we are first concerned with the longitudinal motion of the vehicle, i.e. the range problem. The range R is given by

$$R = \int_{t_s}^{t_f} v \cos\gamma \, \mathrm{d}t. \tag{3.1}$$

With $\varOmega_M = 0$ it follows from (2.17a) that

$$\mathrm{d}t = \frac{\mathrm{d}v}{-D - g\sin\gamma} \tag{3.2}$$

and it follows from (2.21) that

$$\mathrm{d}t = \frac{\mathrm{d}E}{-vD}. \tag{3.3}$$

By substituting (3.2) into (3.1) the range can be written as integral over velocity. With the approximations $\cos\gamma \approx 1$ and $\sin\gamma \approx 0$ it follows that

$$R = \int_{t_s}^{t_f} v\cos\gamma \, \mathrm{d}t = \int_{v(t_s)}^{v(t_f)} \frac{v\cos\gamma}{-D - g\sin\gamma} \, \mathrm{d}v \approx \int_{v(t_s)}^{v(t_f)} \frac{v}{-D} \, \mathrm{d}v. \tag{3.4}$$

Likewise the range can be written as integral over energy

$$R = \int_{t_s}^{t_f} v\cos\gamma \, \mathrm{d}t = \int_{E(t_s)}^{E(t_f)} \frac{v\cos\gamma}{-vD} \, \mathrm{d}E \approx \int_{E(t_s)}^{E(t_f)} \frac{1}{-D} \, \mathrm{d}E. \tag{3.5}$$

The desired range R_r can thus be specified using a reference drag acceleration vs. energy profile such that

$$\int_{E(t_s)}^{E(t_f)} \frac{1}{-D_r} \, \mathrm{d}E = R_r. \tag{3.6}$$

Perfect tracking of D_r consequentially achieves the desired range.

The above mentioned authors (amongst others) propose tracking of D_r using feedback linearization. We follow this approach and synthesize a downrange drag tracking law in the next sections.

3.1.1 Feedback Linearization

This section is an introduction to feedback linearization, oriented at [Isi95]. The core idea is to find a transformation of a nonlinear control affine system into an equivalent linear system through a change of variables and a suitable control input. Although feedback linearization can be applied to a wider category of systems, for the application of drag tracking it is sufficient to consider a single-input single-output system.

Definition 3.7 (Single-Input Single-Output System (affine, time invariant))
Let $x(t) \in \mathbb{R}^{n_x}$ be the state at time t and $u(t) \in \mathbb{R}$ the control at time t. Let $A : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$ and $B : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$ be smooth vector fields and $C : \mathbb{R}^{n_x} \to \mathbb{R}$ a smooth function. The control influences the state according to a control affine system of first order ordinary differential equations (ODEs).

$$\begin{aligned}\dot{x}(t) &= A(x(t)) + B(x(t))\, u(t)\\ y(t) &= C(x(t))\end{aligned}$$

$\dot{x}$ is called system dynamic and C is called output function.

The objective is to find the relationship between the control input $u(t)$ and the output $y(t)$ along a trajectory $x(t)$ resulting from the vector field $V(x) = A(x) + B(x)u$. To this end the output function is derived with respect to time until the control explicitly enters the derivative. The first time derivative of the output function is

$$\begin{aligned}\dot{y} &= \frac{\mathrm{d}C(x)}{\mathrm{d}t} && (3.8a)\\ &= \frac{\partial C(x)}{\partial x_1}\dot{x}_1 + \ldots + \frac{\partial C(x)}{\partial x_n}\dot{x}_{n_x} && (3.8b)\\ &= \langle \nabla C(x), \dot{x}\rangle && (3.8c)\\ &= \langle \nabla C(x), A(x)\rangle + \langle \nabla C(x), B(x)\rangle u && (3.8d)\end{aligned}$$

Equation (3.8) describes the application of the vector field V on the output function C, which can be interpreted as the directional derivative of C along V, or equivalently the rate of change of C measured by an observer who is moved along V. This relationship is expressed by the Lie derivative.

Definition 3.9 (Lie Derivative)
The Lie derivative of a scalar function $C : \mathbb{R}^{n_x} \to \mathbb{R}$ along a vector field $V : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$ at a point x is defined as

$$L_V C(x) := \langle \nabla C(x), V(x)\rangle$$

where $\langle\cdot\rangle$ is the scalar product. If C is differentiated k times along V the notation $L_V^{[k]}C(x)$ is used.

Using the Lie derivative, the time derivative of the output function can be written as

$$\dot{y} = L_A C(x) + L_B C(x) u. \tag{3.10}$$

For technical systems it usually holds that

$$L_B C(x) = 0 \quad \Longrightarrow \quad \dot{y} = L_A C(x). \tag{3.11}$$

Finding the effect of the control u then requires higher order time derivatives of the output. For the case $L_B C(x) = 0$ the second time derivative is

$$\begin{aligned}
\ddot{y} &= \frac{\mathrm{d} L_A C(x)}{\mathrm{d}t} && (3.12a)\\
&= \frac{\partial L_A C(x)}{\partial x} \dot{x} && (3.12b)\\
&= L_A L_A C(x) + L_B L_A C(x) u && (3.12c)
\end{aligned}$$

Should it hold repeatedly that $L_B C(x) = 0$ this leads to the generalized scheme

$$\begin{aligned}
y &= C(x) && (3.13a)\\
\dot{y} &= L_A C(x) && (3.13b)\\
\ddot{y} &= L_A^{[2]} C(x) && (3.13c)\\
&\vdots && (3.13d)\\
y^{(r-1)} &= L_A^{[r-1]} C(x) && (3.13e)\\
y^{(r)} &= L_A^{[r]} C(x) + L_B L_A^{[r-1]} C(x) u && (3.13f)
\end{aligned}$$

where $L_B L_A^{[i]} C(x) = 0$, for $i = 0,...,r-2$. The quantity r is called relative degree.

Definition 3.14 (Relative Degree)
System (3.7) is said to have relative degree r at a point x_1 if

1. $L_B L_A^{[k]} C(x) = 0$ *for all x in a neighborhood of x_1, for $k = 0,...,r-2$*
2. $L_B L_A^{[r-1]} C(x) \neq 0$

We focus on the case where the relative degree is equal to the system order. The system can then be transformed to a new set of coordinates $z^{(1)},...,z^{(r)}$.

$$\begin{pmatrix} z^{(1)} \\ z^{(2)} \\ \vdots \\ z^{(r)} \end{pmatrix} = \begin{pmatrix} y \\ \dot{y} \\ \vdots \\ y^{[r-1]} \end{pmatrix} = \begin{pmatrix} C(x) \\ L_A C(x) \\ \vdots \\ L_A^{[r-1]} C(x) \end{pmatrix} = T(x) \tag{3.15}$$

Definition 3.16 (Diffeomorphism)
If a function $T : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$ is continuously differentiable and if the inverse function $T(x)^{-1}$ exists and is also continuously differentiable such that

$$T^{-1}(T(x)) = x,$$

then T is called Diffeomorphism.

Assuming all functions are sufficiently often continuously differentiable such that $T(x)$ in (3.15) is a diffeomorphism, then system (3.7) can be uniquely transformed to a system representation expressed in the z coordinates. Smooth trajectories in the original coordinate system have smooth, unique representations in the z coordinates.

$$\begin{pmatrix} \dot{z}^{(1)} \\ \vdots \\ \dot{z}^{(r-1)} \\ \dot{z}^{(r)} \end{pmatrix} = \begin{pmatrix} z^{(2)} \\ \vdots \\ z^{(r)} \\ L_A^{[r]}C(x) \end{pmatrix} + \begin{pmatrix} 0 \\ \vdots \\ 0 \\ L_B L_A^{[r-1]}C(x) \end{pmatrix} u \tag{3.17}$$

If the control u is chosen as the feedback law

$$u(x) = \frac{1}{L_B L_A^{[r-1]}C(x)} \left(-L_A^{[r]}C(x) + \nu\right) \tag{3.18}$$

system (3.17) is linearized to the form

$$\begin{pmatrix} \dot{z}^{(1)} \\ \vdots \\ \dot{z}^{(r-1)} \\ \dot{z}^{(r)} \end{pmatrix} = \begin{pmatrix} z^{(2)} \\ \vdots \\ z^{(r)} \\ \nu \end{pmatrix} \tag{3.19a}$$

$$y = z^{(1)} \tag{3.19b}$$

The linearized system is a cascade of r integrators and the original output $C = z^{(1)}$ is a linear function of the new control ν which can be chosen in an outer loop using linear control theory.

In case the relative degree of the output is less than the system order, feedback linearization can still be feasible. Similarly a diffeomorsphism can be used to transform the system to more advantageous coordinates. But the transformed system will have states which do not influence the output and which are consequentially unobservable. These unobservable states are called internal or zero dynamics. Although the internal dynamics are independent from the system output and the linearized part of the dynamics, they are still relevant for the overall system dynamics. The overall system is stable if and only if the internal and the linearized dynamics are both stable.

In case of the drag tracking problem considered in the next section the system is of full relative degree, hence a treatment of internal dynamics is not required.

3.1.2 Drag Tracking in the Energy Domain

Energy can be a more advantageous independent variable for re-entry problems than time, because the elimination of time removes the strict relationship between altitude and velocity. This opens a new degree of freedom if the problem is time invariant and the final time is free. Energy is a function of the system state, thus an energy based tracking law adapts to disturbances in the states by shifting the energy tracking point. Furthermore energy allows for a better approximation of the range integral in case of a steep flight path angle. To capitalize on these advantages the tracking law in the energy domain is derived.

The longitudinal dynamics can be separated from the translational dynamics by assuming a non rotating planet. The longitudinal dynamics in the energy domain (cf. (2.23)) are

$$h' = -\frac{1}{D}\sin\gamma, \tag{3.20a}$$
$$v' = \frac{D + g\sin\gamma}{Dv}, \tag{3.20b}$$
$$\gamma' = -\frac{L}{D}\frac{\cos\sigma}{v^2} - \left(\frac{v}{r} - \frac{g}{v}\right)\frac{\cos\gamma}{Dv}. \tag{3.20c}$$

Drag acceleration is considered as the system output

$$y = D = \frac{1}{2}v^2\rho C_D\frac{S}{m}. \tag{3.21}$$

An additional advantage of the energy representation is that the order of the system is reduced from three to two. E is considered an additional input. The equations for h' and v' are not independent [Bha98], because with respect to energy they have the differential relation

$$\frac{\mathrm{d}E}{\mathrm{d}E} = 1 = vv' + gh'. \tag{3.22}$$

A minimal representation of the system needs to retain only either h' or v'. Arbitrarily it is chosen to retain h'. If E and h are known the velocity can be computed by solving (2.20) for v.

As control for system (3.20) we define the vertical lift-over-drag (vLoD) ratio.

$$u_{\text{vLoD}} := \frac{L}{D}\cos(\sigma) \tag{3.23}$$

This has the advantage that principally not only the bank angle but also a modulation of the angle of attack could be used to achieve the commanded vLoD.

The starting point for the application of feedback linearization is the minimal system representation

$$\begin{pmatrix} h' \\ \gamma' \end{pmatrix} = \begin{pmatrix} -\frac{\sin(\gamma)}{D} \\ \left[\frac{g}{Dv^2} - \frac{1}{(h+r_M)D}\right]\cos(\gamma) \end{pmatrix} + \begin{pmatrix} 0 \\ -\frac{1}{v^2} \end{pmatrix} u_{\text{vLoD}} \tag{3.24a}$$
$$= A(x) + B(x)\; u_{\text{vLoD}} \tag{3.24b}$$

with drag as system output. Following the methodology of feedback linearization the first step is to derive the drag output equation with respect to energy until the control appears explicitly. In the following M is the Mach number as defined in (2.6) and $c(h)$ is the local speed of sound as a function of altitude. Under the assumption that the gravitational acceleration g is constant, the first and second order energy derivatives of drag are:

$$
\begin{aligned}
y \ &= D = \frac{1}{2}\rho(h)v^2 C_D(M)\frac{S}{m} && (3.25a)\\
y' \ &= D' = D\left(\frac{1}{\rho}\frac{\mathrm{d}\rho}{\mathrm{d}h}h' + \frac{2}{v}v' + \frac{1}{C_D}\frac{\mathrm{d}C_D}{\mathrm{d}M}M'\right) && (3.25b)\\
y'' &= D'' = D\left(\frac{1}{\rho}\frac{\mathrm{d}\rho}{\mathrm{d}h}h' + \frac{2}{v}v' + \frac{1}{C_D}\frac{\mathrm{d}C_D}{\mathrm{d}M}M'\right)^2 && (3.25c)\\
&\quad + D\left[\frac{1}{\rho}\frac{\mathrm{d}\rho}{\mathrm{d}h}h'' + \left(\frac{1}{\rho}\frac{\mathrm{d}^2\rho}{\mathrm{d}h^2} - \frac{1}{\rho^2}\left(\frac{\mathrm{d}\rho}{\mathrm{d}h}\right)^2\right)(h')^2 + \frac{2}{v}v'' - \frac{2}{v^2}(v')^2\right]\\
&\quad + D\left[\frac{1}{C_D}\frac{\mathrm{d}C_D}{\mathrm{d}M}M'' + \left(\frac{1}{C_D}\frac{\mathrm{d}^2C_D}{\mathrm{d}M^2} - \frac{1}{C_D^2}\left(\frac{\mathrm{d}C_D}{\mathrm{d}M}\right)^2\right)(M')^2\right].
\end{aligned}
$$

In the following we calculate all derivatives appearing in D''. The energy derivatives of Mach are

$$M' = \frac{v'c - vh'\frac{\mathrm{d}c}{\mathrm{d}h}}{c^2} \tag{3.26}$$

$$M'' = \frac{v''}{c} - \frac{v\frac{\mathrm{d}c}{\mathrm{d}h}(h')^2}{c^2} - \frac{2v'\frac{\mathrm{d}c}{\mathrm{d}h}h'}{c^2} + \frac{2\frac{\mathrm{d}^2c}{\mathrm{d}h^2}v(h')^2}{c^3} \tag{3.27}$$

The derivatives $\frac{\mathrm{d}\rho}{\mathrm{d}h}$, $\frac{\mathrm{d}^2\rho}{\mathrm{d}h^2}$, $\frac{\mathrm{d}C_D}{\mathrm{d}M}$, $\frac{\mathrm{d}^2C_D}{\mathrm{d}M^2}$, $\frac{\mathrm{d}c}{\mathrm{d}h}$, $\frac{\mathrm{d}^2c}{\mathrm{d}h^2}$ are given as part of the model. The second energy derivatives of altitude and velocity are

$$h'' = \frac{\sin(\gamma)D'}{D^2} - \frac{D\cos(\gamma)y'}{D^2} \tag{3.28}$$

$$v'' = \frac{gv\left(D\cos(\gamma)y' - \sin(\gamma)D'\right)}{D^2v^2} - \frac{Dv'\left(D + g\sin(\gamma)\right)}{D^2v^2} \tag{3.29}$$

It is now obvious that the first time the control u_{vLoD} enters the output's derivatives explicitly is in D'' through the term γ' (cf. (3.21)). γ' is contained in h'' and v''. The relative degree of the drag output is therefore two. The next step is to insert the calculated derivatives in D'' and factor out all appearances of the control u_{vLoD} and write D'' as

$$D'' = L_A^{[2]}C(x) + L_B L_A^{[1]}C(x)u_{\mathrm{vLoD}} = a + bu_{\mathrm{vLoD}} \tag{3.30}$$

The factoring of u_{vLoD} is tedious because the terms a and b are very large (see Appendix D). Inserting the linearizing feedback control law

$$u_{\mathrm{vLoD}} = \frac{1}{b}\left(-a + D_r'' + \nu\right) \tag{3.31}$$

leads to the system

$$
\begin{aligned}
z_1' &= D' && (3.32a)\\
z_2' &= D'' = a + bu_{\mathrm{vLoD}} = \nu && (3.32b)\\
y = D \ &= z_1. && (3.32c)
\end{aligned}
$$

The dynamic system between input ν and output y is linear, time invariant, and in double-integrator form.

The remaining task is to ensure that the output tracks the reference drag profile D_r. Let D_r be at least two times continuously differentiable with respect to energy and let $\mathit{\Delta} D = D - D_r$ and $\mathit{\Delta} D' = D' - D'_r$. For tracking the reference profile a controller of the form

$$\nu = -K_P \mathit{\Delta} D - K_D \mathit{\Delta} D' - K_I \int \mathit{\Delta} D \,\mathrm{d}E \tag{3.33}$$

is used. Combining (3.31) and (3.33) yields the control law

$$u_{\mathrm{com}} = \frac{1}{b}\left(-a + D''_r - K_P \mathit{\Delta} D - K_D \mathit{\Delta} D' - K_I \int \mathit{\Delta} D \,\mathrm{d}E\right) \tag{3.34}$$

resulting into the closed loop system

$$z'_1 = z_2 \tag{3.35a}$$

$$z'_2 = D''_r - K_P \mathit{\Delta} D - K_D \mathit{\Delta} D' - K_I \int \mathit{\Delta} D \,\mathrm{d}E \tag{3.35b}$$

With appropriate positive gains K_P, K_D and K_I it is obvious that the energy varying equilibrium of the closed loop system (3.35) is at $[D_r,\, D'_r]^\intercal$. Consequently the closed loop tracking error dynamics (3.36) with error states $e_1 = \mathit{\Delta} D$ and $e_2 = \mathit{\Delta} D'$ have a unique equilibrium at $e = [e_1, e_2]^\intercal = [0,\, 0]^\intercal$.

$$e'_1 = e_2 \tag{3.36a}$$

$$\begin{aligned} e'_2 &= D'' - D''_r \qquad (3.36b)\\ &= D''_r - K_P e_1 - K_D e_2 - K_I \int e_1 \,\mathrm{d}E - D''_r \\ &= -K_P \mathit{\Delta} D - K_D \mathit{\Delta} D' - K_I \int \mathit{\Delta} D \,\mathrm{d}E \end{aligned}$$

3.1.3 Induced Time Domain Gain Scheduling

The synthesis of the tracking law based on energy has the convenient consequence of working with a system of full relative degree. Another advantage of this approach is pointed out by Tu [Tu00] through analyzing the relationship between the gains in the time and energy domain. To make this relationship obvious the integral part of the control action is neglected, that is $K_I := 0$, and perfect cancellation of the nonlinearities is assumed. Then the energy domain closed loop error dynamics are

$$\mathit{\Delta} D'' + K_{D,E} \mathit{\Delta} D' + K_{P,E} \mathit{\Delta} D = 0 \tag{3.37a}$$

and the time domain closed loop error dynamics are

$$\mathit{\Delta} \ddot{D} + K_{D,t} \mathit{\Delta} \dot{D} + K_{P,t} \mathit{\Delta} D = 0. \tag{3.38}$$

The derivatives of the tracking error in the energy domain are

$$\Delta D' = \frac{\Delta \dot{D}}{\dot{E}} \tag{3.39a}$$
$$\Delta D'' = \frac{\Delta \ddot{D}\dot{E} - \frac{\Delta \dot{D}\ddot{E}}{\dot{E}}}{\dot{E}^2} = \frac{\Delta \ddot{D}\dot{E} - \Delta D'\ddot{E}}{\dot{E}^2}. \tag{3.39b}$$

If (3.39) are substituted in (3.37a) and rearranged into the form of (3.38) one obtains the corresponding gains in the time domain

$$K_{P,t} = K_{P,E}\dot{E}^2 \tag{3.40a}$$
$$K_{D,t} = K_{D,E}\dot{E} - \frac{\ddot{E}}{\dot{E}}. \tag{3.40b}$$

Obviously if the controller is designed in the energy domain with constant gains $K_{P,E}$, $K_{D,E}$ this corresponds to a scheduling of the time domain gains. $K_{P,t}$ increases when the energy derivative is high, meaning the controller is more aggressive in regions of high dynamic change. Because $\ddot{E}/\dot{E} \approx 0$ the derivative gain $K_{D,t}$ is likewise predominantly scheduled on the energy derivative.

3.1.4 Cross Range Control and Bank Angle Command Generation

The commanded vLoD (3.31) is realized using the bank angle $\sigma \in [-\pi,\pi]$. The bank angle magnitude $|\sigma| \in [0,\pi]$ is obtained by solving (3.23) for $|\sigma|$.

$$|\sigma| = \arccos\left(\frac{D}{L}u_{\text{com}}\right) \tag{3.41}$$

The bank angle sign is left to control the crossrange. The crossrange control is based on a deadband defined on the heading error. If the heading error exceeds the deadband threshold, a rapid change of the bank angle to the opposite sign is commanded. This maneuver is called bank reversal. Obviously bank angles of opposite sign result in equal vertical lift, but the lateral lift component is in opposite direction. Through the decoupling of downrange and crossrange logic, the bank reversal maneuver is neglected in the derivation of the tracking law and is thus treated as a perturbation in the aerodynamic forces.

This control strategy has singularities at full lift up or lift down. If

$$\left|\frac{D}{L}u_{\text{com}}\right| > 1 \tag{3.42}$$

the system is not able to produce the required lift to match the desired vLoD. The bank angle is saturated and the ability to control the crossrange is lost.

$$\frac{D}{L}u_{\text{com}} > \quad 1 \quad \Longrightarrow \quad |\sigma| = 0 \tag{3.43a}$$
$$\frac{D}{L}u_{\text{com}} < \; -1 \quad \Longrightarrow \quad |\sigma| = \pi \tag{3.43b}$$

The loss of crossrange control can be avoided by limiting the bank angle magnitude to an interval $[0+s, \pi-s]$, $0 < s < \frac{\pi}{2}$. But as soon as the required vLoD cannot be matched, the tracking of the reference drag profile is not guaranteed. This situation should be avoided by carefully budgeting the available control authority. This concludes the synthesis of the drag tracking control law.

3.1.5 Drag Tracking Conclusion

Drag tracking is relatively robust against perturbations in the aerodynamic coefficients and the atmospheric model. The drag control law evaluates the model locally to determine the linearizing control. The stabilizing PID controller does only depend on the model indirectly via the gain scheduling, otherwise the PID feedback component is based only on the direct physical measurement of the drag acceleration by the IMU. The incurred drag error is accumulated and taken into account using integral control. In most state feedback methods and for trajectory prediction the entire atmosphere model must be taken into account at some point which makes these methods more susceptible to errors in the aerodynamic coefficients and the atmospheric density.

The representation of the reference trajectory as drag-energy profile has the advantage that the path constraints can be completely represented in the energy domain (cf. Section 2.6). Although the feedback controller does not take into account these constraints directly, it is possible to include them in the planning of the drag profile. The unified constraint representation enables the use of comparatively simple geometric methods to shape the drag profile online.

The decoupling of the downrange and crossrange is instrumental to the feedback design. But during the bank angle reversals the longitudinal tracking is suspended, and the tracking error increases, which can lead to a reduced targeting accuracy, if there is not enough time to correct the introduced error after the maneuver.

3.2 Other Approaches to Entry Guidance and Control

In the following the central ideas of some alternative entry guidance concepts are introduced. The purpose of these brief reviews is to understand the main advantages and disadvantages of each method. Our conclusions are stated in Section 3.3.

3.2.1 Multi-Variable Tracking

Roenneke [Roe93] suggests the tracking of reference profiles for multiple states using a gain scheduled, time dependent Riccati controller. Let $r_r(t), v_r(t), \gamma_r(t),\ t \in [t_s;\ t_f]$ be the reference state profiles and $\sigma_r(t)$ the reference bank angle command. The equations of motion are linearized numerically at the reference state at times $t^{(k)} \in [t_s;\ t_f]$, $k = 1, ..., N$. The linearization is of the form

$$\delta\dot{x} = A(t^{(k)})\delta x + B(t^{(k)})\delta\sigma. \tag{3.44}$$

The state deviation is $\delta x = (r - r_r,\ v - v_r,\ \gamma - \gamma_r)^\intercal$ and the bank angle deviation is $\delta\sigma = \sigma - \sigma_r$. At each time $t^{(k)}$ a control law of the form

$$\delta\sigma = -F(t^{(k)})\delta x \tag{3.45}$$

with

$$F(t^{(k)}) = \frac{1}{q_0} B^\intercal(t^{(k)}) P(t^{(k)}) \tag{3.46}$$

is obtained. The matrix $P(t^{(k)})$ is the solution of the continuous time algebraic Riccati equation

$$0 = Q - PB\frac{1}{q_0}B^\intercal P + PA + A^\intercal P. \tag{3.47}$$

q_0 and $Q = \operatorname{diag}(q_1, q_2, q_3)$ are the weighting coefficients of the infinite horizon quadratic performance index

$$P = \lim_{t_f \to \infty} \int_{t_s}^{t_f} \left(q_1 \delta r^2 + q_2 \delta v^2 + q_3 \delta y^2 + q_0 \delta \sigma^2\right) \, \mathrm{d}t. \tag{3.48}$$

The gain matrix $F(t)$ is interpolated between the linearization points.

Roenneke shows that this control law can handle state perturbations of up to 10 %, lift-to-drag ratio perturbation of 4 % and atmospheric density variations of up to 50 %.

3.2.2 Numeric Predictor-Corrector Guidance

A numeric predictor uses numerical integration of the equations of motion to predict the future trajectory during flight. Based on the prediction a correction of the control function is performed to eliminate the predicted final state error. As an example we consider the correction method suggested by Spreng [Spr11].

In this approach the reference control sequence is specified by piecewise constant segments of the vertical lift vector $u_r(E)$, $E \in [E_s;\, E_f]$. The energy points $E_1, ..., E_n$ of bank reversals are an additional input. At a fixed energy point $\bar{E}$ let $L_{cur} = u_r(\bar{E})$ denote the reference vertical lift, and let $E_{nxt} \in \{E_1, ..., E_n\}$ with $E_{nxt} < \bar{E}$ be the next reference bank reversal point.

The predictor function performs three trajectory predictions: The first one uses the reference lift L_{cur} and reference bank reversal point E_{nxt}. Let the achieved downrange and crossrange of this prediction be $\mathcal{D}(L_{cur}, E_{nxt})$ and $\mathcal{C}(L_{cur}, E_{nxt})$. For the second prediction a small change δL of the vertical lift in the current segment is considered. Let $\mathcal{D}(L_{cur} + \delta L, E_{nxt})$ and $\mathcal{C}(L_{cur} + \delta L, E_{nxt})$ be the resulting downrange and crossrange. For the third prediction a small change δE of the next bank reversal point is applied. Let the resulting downrange and crossrange be $\mathcal{D}(L_{cur}, E_{nxt} + \delta E)$ and $\mathcal{C}(L_{cur}, E_{nxt} + \delta E)$.

The partial derivatives of downrange and crossrange with respect to changes in the vertical lift and the point of bank reversal can be approximated as

$$\frac{\partial \mathcal{D}}{\partial L} \approx \frac{\mathcal{D}(L_{cur} + \delta L, E_{nxt}) - \mathcal{D}(L_{cur})}{\delta L}, \tag{3.49a}$$
$$\frac{\partial \mathcal{D}}{\partial E} \approx \frac{\mathcal{D}(L_{cur}, E_{nxt} + \delta E) - \mathcal{D}(L_{cur})}{\delta E}, \tag{3.49b}$$
$$\frac{\partial \mathcal{C}}{\partial L} \approx \frac{\mathcal{C}(L_{cur} + \delta L, E_{nxt}) - \mathcal{C}(L_{cur})}{\delta L}, \tag{3.49c}$$
$$\frac{\partial \mathcal{C}}{\partial E} \approx \frac{\mathcal{C}(L_{cur}, E_{nxt} + \delta E) - \mathcal{C}(L_{cur})}{\delta E}. \tag{3.49d}$$

The adjustments $\varDelta L$ and $\varDelta E$ required to eliminate the downrange and crossrange errors $\varDelta \mathcal{D}$ and $\varDelta \mathcal{C}$ can be approximated by solving

$$\begin{pmatrix} \varDelta \mathcal{D} \\ \varDelta \mathcal{C} \end{pmatrix} = \begin{pmatrix} \frac{\partial \mathcal{D}}{\partial L} & \frac{\partial \mathcal{D}}{\partial E} \\ \frac{\partial \mathcal{C}}{\partial L} & \frac{\partial \mathcal{C}}{\partial E} \end{pmatrix} \begin{pmatrix} \varDelta L \\ \varDelta E \end{pmatrix}. \tag{3.50}$$

For simplicity assume the matrix inversion is feasible, then we obtain

$$\begin{pmatrix} \varDelta L \\ \varDelta E \end{pmatrix} = \begin{pmatrix} \frac{\partial \mathcal{D}}{\partial L} & \frac{\partial \mathcal{D}}{\partial E} \\ \frac{\partial \mathcal{C}}{\partial L} & \frac{\partial \mathcal{C}}{\partial E} \end{pmatrix}^{-1} \begin{pmatrix} \varDelta \mathcal{D} \\ \varDelta \mathcal{C} \end{pmatrix}. \tag{3.51}$$

In subsequent guidance calls the adapted control profile is used as the new reference.

As previously discussed the prediction of the future trajectory, based on the reference aerodynamic model, is not accurate in a disturbed aerodynamic environment. To improve the trajectory prediction the lift and drag accelerations of the model are compared to the actual accelerations in each guidance cycle and the aerodynamic model is iteratively adjusted to reduce the error.

The numeric predictor-corrector approach can be seen as sensitivity analysis of the control variables L_{cur}, E_{nxt}. Equation (3.51) can be written as a first order Taylor expansion around the reference control.

$$\begin{pmatrix} L \\ E \end{pmatrix}_{new} = \begin{pmatrix} L_{cur} \\ E_{nxt} \end{pmatrix} + \begin{pmatrix} \frac{\partial \mathcal{D}}{\partial L} & \frac{\partial \mathcal{D}}{\partial E} \\ \frac{\partial \mathcal{C}}{\partial L} & \frac{\partial \mathcal{C}}{\partial E} \end{pmatrix}^{-1} \begin{pmatrix} \varDelta \mathcal{D} \\ \varDelta \mathcal{C} \end{pmatrix} \tag{3.52}$$

The inverted matrix contains the sensitivity of the control variables with respect to the downrange and crossrange.

We will come back to this idea in Chapter 5 where we perform a Taylor expansion of the necessary optimality conditions of a nonlinear program.

3.2.3 Moving Horizon Control

Moving horizon control repeatedly solves an optimal control process[1] on a finite time horizon to achieve a sub-optimal closed loop control for an infinite time horizon. Moving horizon control is also called model predictive control (MPC). In MPC theory we differentiate between the process time $t \in [t_s; t_f]$ and the MPC time $\tau \in [0; T]$, as shown in Figure 3.3. T is called *prediction horizon*. At the current time t_k the MPC time is $\tau_s := 0$. Starting

1 The term *optimal control process* is formally defined in the subsequent Chapter 4.

from the current state $x_k = x(t_k)$ the trajectory of the process is predicted on the prediction horizon T. The control $\tilde{u}_k(\tau)$, $\tau \in [0; T]$ is determined such, that the objective function of the optimal control process in minimized over T.

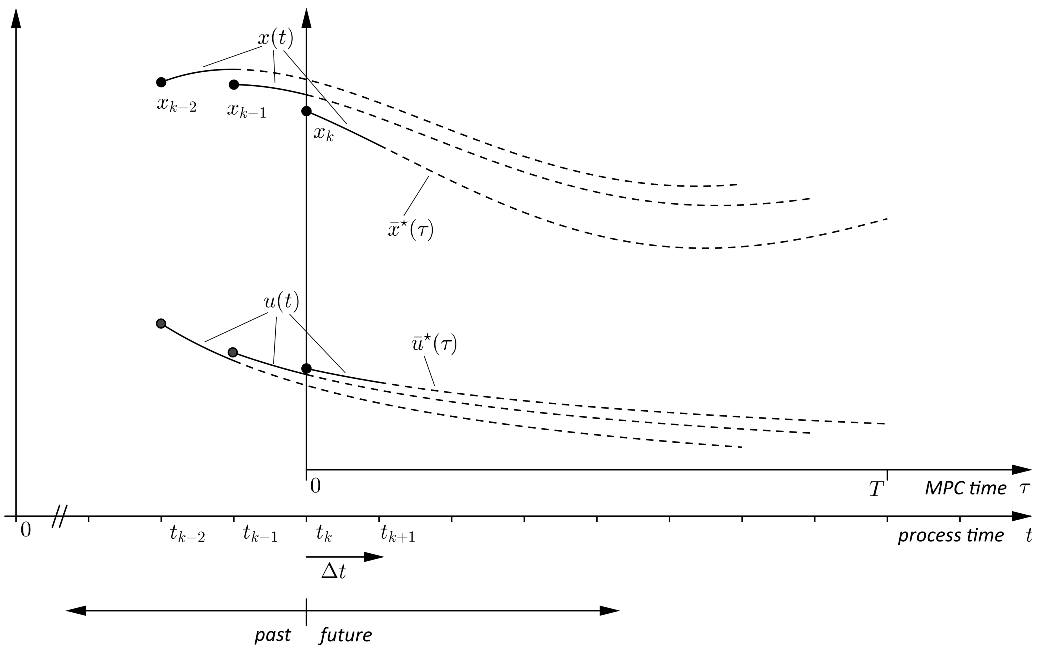


Figure 3.3: Model predictive control scheme

At $t_{k+1} = t_k + \Delta t$ the horizon is shifted and the optimal control process is solved again. This repeats at a fixed rate. MPC is thus a clock based sequence of sub-optimal open loop solutions. If the time horizon is sufficiently long, i.e. $t_s + T \geq t_f$, the control $\tilde{u}_k(\tau)$ is optimal for the entire process.

An ideal MPC assumes that the computation of the control is instant, thus $\tilde{u}_k(\tau)$ is implemented on the interval $[t_k; t_{k+1}]$, until at t_{k+1} the new control $\tilde{u}_{k+1}$ becomes available. A major difficulty for actual implementations is the lag introduced by the computation time. In each step the solution of a nonconvex optimal control process requires the solution of a nonlinear optimization problem. Methods that exploit the similarity of the subsequently solved optimization problems are investigated by e.g. Diehl [Die02] to achieve a decisive speed-up of the required computations.

Bollino [Bol06] demonstrates the unique advantage of MPC to take into account the constraints on the control and the state during closed loop control. In MPC the control command is directly available from the solution of the optimization problem without requiring a trajectory tracking controller. In particular nonlinear MPC methods therefore have the potential to overcome the separation between translational and rotational dynamics through formulation of the optimal control process based on the joint dynamcis. The computational requirements and the complexity of solving optimal control problems are the major reasons that so far hindered MPC from becoming a predominantly used control strategy.

Nonlinear MPC has also been investigated in the frame of this project as master thesis subject of Schomakers [Sch14]. Schomakers investigated the stability and performance of an NMPC controller with terminal costs depending on the prediction horizon. While for sufficiently long horizons the NMPC method proved to be successful in the compensation of strong state errors, Schomakers showed that errors in the aerodynamic model are much more difficult to compensate. The strong dependency on the accuracy of the model and the lack of a direct counterpart to integral control motivate the combination of MPC with an adaptive model, e.g. through using a Kalman filter to estimate model parameters online. MPC in combination with an adaptive model is arguably the preferable choice for the control of any dynamic system, if sufficient computational power is available.

3.3 Challenges of Atmospheric Entry Guidance

Atmospheric entry guidance is a very challenging problem for multiple reasons:

The first one is the combination of the strong nonlinearity of the equations of motion with the large uncertainty in the atmosphere model. Prediction based methods are required to achieve the landing accuracy required for future missions. However prediction is based on the integration of the equations of motion and is thus susceptible to an erroneous model. The ability to update the model online is a key factor to achieve a precise landing. The availability and accuracy of sensor data is thus of great importance, because it is the only possibility to draw conclusions concerning model discrepancies. During the atmospheric entry phase so far only IMU measurements are available. This makes the in-flight adaption of parametrized models problematic, because discerning the influence of individual parameters from the measurements can be difficult and must heavily rely on a priori knowledge. In-flight dynamic pressure measurements[1] could improve this situation.

A second challenge is the handling of constraints on the control and the state. The entry capsule has a low lift-to-drag ratio and bank angle rotation only provides a limited control effect, such that saturation of the control is likely. Only MPC methods are able to take constraints into account during the control phase. Otherwise a guarantee to satisfy the state constraints requires additional guidance modes based on dedicated, boundary tracking controllers, which severely complicates the G&C design.

A partial solution to that problem is the representation of the path constraints in the drag-energy domain. A less desirable alternative is checking the path constraints only during the design of the reference trajectory and include sufficient margin, such that it is unlikely that the constraints become active during flight.

Lastly it would be an advantage to manage downrange and crossrange at the same time to eliminate the error induced by neglecting the bank reversals and to pro-actively prevent problems related to the singularities at full lift-up or lift-down. Unified management of downrange and crossrange however prevents the decoupling of the equations of motion which is the basis for classical drag tracking.

The first choice to solving these problems is MPC. If the computational power does not permit the use of MPC, an alternative approach are guidance systems that combine online trajectory adaption and trajectory tracking. The trajectory is only updated at a low rate to satisfy the computational constraints and the control commands are determined by a

1 An example is the Mars Entry Atmospheric Data System (MEADS) that was used by the MSL [Kar09].

tracking controller that tracks the newest available trajectory at a high rate. The guidance is split up into two loops, one slow trajectory computation loop and one fast tracking loop. This is also referred to as predictive tracking or more generally as *two degree-of-freedom control*. This approach combines the strength of prediction based correction, with the low computational requirement of tracking. Predictive tracking was for instance implemented in the space shuttle guidance (Harpold and Graves [Har79]) and numerous other authors suggest this general design ([Lu98][Tu00]).

This concludes the overview of entry guidance and control concepts. We come back to these preliminary considerations in Chapter 6.

CHAPTER 4

Preliminaries of Optimal Control and Optimization Theory

This chapter contains definitions and theorems of different approaches to optimization and optimal control theory. Together with Appendix Spaces and Norms this chapter provides a minimal formal framework for the thesis. The remarks are based on different sources which are stated individually within the sections.

4.1 Dynamic Optimization

Dynamic or infinite dimensional optimization is concerned with the optimization of dynamic control systems over an independent variable which is generally called *time* although it may have another meaning. The optimization is carried out over a linear vector space of functions equipped with a norm as a distance measure. The space of admissible functions depends on the analysis method. There are three distinct approaches to dynamic optimization: *Calculus of Variations*, *Dynamic Programming* and *Optimal Control*. In the following the standard definition of an optimal control problem (OCP) is given and the fundamental theorems of *Dynamic Programming* and *Optimal Control Theory* are stated.

4.1.1 Formulation of Optimal Control Processes

An optimal control process describes the performance of a (constrained) control system over *time* according to a cost functional. In the following the building blocks of an optimal control process are defined.

Let $x(t) \in \mathbb{R}^{n_x}$ be the *state* at time t and $u(t) \in \mathbb{R}^{n_u}$ the *control* at time t. The time t is defined over an interval $t \in [t_s, t_f]$. Without loss of generality we assume the start time t_s is fixed, while the final time t_f may be fixed or free. The control influences the state according to a system of first order ordinary differential equations (ODEs).

Definition 4.1 (System dynamic (nonlinear, time variant))
Let $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times [t_s, t_f] \to \mathbb{R}^{n_x}$ be continuous and continuously differentiable w.r.t. all arguments. The ordinary differential equation

$$\dot{x} = f(x(t), u(t), t), \quad t \in [t_s, t_f]$$

is called system dynamic.

Definition 4.2 (Solution of the system)
Let $x : [t_s,t_f] \to \mathbb{R}^{n_x}$ be continuous and piecewise continuously differentiable and let $u : [t_s,t_f] \to \mathbb{R}^{n_u}$ be piecewise continuous. If it holds for all continuous arcs of $u(t)$ that

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

then the pair (x,u) is called solution to the differential equation (4.2) and $x(t)$ is called trajectory corresponding to $u(t)$.

The solution of the dynamics can be subject to boundary conditions and mixed state and control constraints:

Definition 4.3 (Boundary conditions)
The boundary conditions on the initial state $x(t_s)$ and the final state $x(t_f)$ can be given by continuously differentiable functions $\psi_s : \mathbb{R}^{n_x} \to \mathbb{R}^{n_{r_s}}$ and $\psi_f : \mathbb{R}^{n_x} \to \mathbb{R}^{n_{r_f}}$

$$\psi_s(x(t_s)) = 0 \qquad \textit{(initial condition)} \tag{4.3a}$$
$$\psi_f(x(t_f)) = 0 \qquad \textit{(final condition)} \tag{4.3b}$$

In many applications the boundary conditions take the form

$$\psi_s(x(t_s)) = x(t_s) - x_s, \tag{4.3c}$$
$$\psi_f(x(t_f)) = x(t_f) - x_f. \tag{4.3d}$$

This is referred to as standard boundary conditions.

Definition 4.4 (State and control constraints)
With a continuously differentiable function $c : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times [t_s,t_f] \to \mathbb{R}^{n_c}$ mixed state and control constraints can be given with

$$c(x(t),u(t),t) \leq 0, \quad \text{for all } t \in [t_s,t_f]. \tag{4.4a}$$

If a constraint $c^{(j)}$, $1 \leq j \leq n_c$ is not depending on the control, $c^{(j)}$ is called pure state constraint.

$$c^{(j)}(x(t),t) \leq 0. \tag{4.4b}$$

Analogously if constraint $c^{(j)}$ is not depending on the state, $c^{(j)}$ is called pure control constraint.

$$c^{(j)}(u(t),t) \leq 0. \tag{4.4c}$$

Definition 4.5 (Control set)
The pure control constraints and the mixed state and control constraints may restrict the control $u(t)$ to a nonempty, convex and closed set $\mathbb{U}(t) \subset \mathbb{R}^{n_u}$

$$u(t) \in \mathbb{U}(t), \quad \text{for all } t \in [t_s,t_f].$$

$\mathbb{U}(t)$ is called control set. A control $u(t) \in \mathbb{U}(t)$ is called admissible.

A trajectory $x(t)$ is called feasible for the time t_f if it fulfills all above conditions for all

$t \in [t_s,t_f]$. The performance of a solution is measured using a scalar cost functional J comprised of the Meyer (or terminal) cost m and the Lagrange (or running) cost l.

Definition 4.6 (Cost functional)
Let the functions $m : \mathbb{R}^{n_x} \times [t_s,t_f] \to \mathbb{R}$ and $l : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times [t_s,t_f] \to \mathbb{R}$ be continuously differentiable w.r.t to their arguments. A function of the form

$$J(x(t),u(t),t) = m(x(t_f),t_f) + \int_{t_s}^{t_f} l(x(t),u(t),t)\,\mathrm{d}t$$

is called cost or objective functional. Depending on the form of the cost functional the OCP is called

- *Bolza problem if $m \not\equiv 0$, $l \not\equiv 0$*
- *Mayer problem if $m \not\equiv 0$, $l \equiv 0$*
- *Lagrange problem if $m \equiv 0$, $l \not\equiv 0$.*

An optimal control process can be written in compact form as so called optimal control problem (OCP).

Problem 4.7 (Standard Optimal Control Problem)
The infinite dimensional optimization problem

$$\begin{aligned}
\min_{x,u,t_f} J(x(t),u(t),t) &= m(x(t_f),t_f) + \int_{t_s}^{t_f} l(x(t),u(t),t)\,\mathrm{d}t && (4.7a)\\
\text{s.t.} \qquad \dot{x} &= f(x(t),u(t),t) && (4.7b)\\
\psi_s(x(t_s)) &= 0 && (4.7c)\\
\psi_f(x(t_f)) &= 0 && (4.7d)\\
c(x(t),u(t),t) &\le 0 && (4.7e)
\end{aligned}$$

is called standard optimal control problem OCP$_{(4.7)}$. If the problem does not depend on time explicitly it is called autonomous.

The goal of the minimization task is to find an optimal control function $\overset{\star}{u}(t) \in L^\infty$ and a corresponding state trajectory $\overset{\star}{x} \in W^{1,\infty}$ for which $\varepsilon > 0$ exists such that

$$J(\overset{\star}{x},\overset{\star}{u}) \le J(x,u)$$

for all admissible (x,u) with $\|x - \overset{\star}{x}\|_\infty < \varepsilon$. The pair $(\overset{\star}{x},\overset{\star}{u})$ is called strong local minimum of OCP$_{(4.7)}$ (cf. Appendix Strong and Weak Minima). If the ε neighborhood includes all admissible state trajectories the minimum is called *global*.

To solve OCP$_{(4.7)}$ with the help of a computer program the problem needs to be discretized into a finite dimensional programming problem, because obviously the computer only has a limited amount of memory. The problem can be discretized either before or after the evaluation of optimality conditions. This leads to the so called *direct* and *indirect* solution methods. This thesis focuses on the direct solution approach detailed in Section 4.4. A conceptual overview of an indirect approach is given in Section 4.1.4.

4.1.2 Problem Transformations

Solving an OCP is often based on transforming or reformulating the standard problem. Some important problem transformations are illustrated in the following.

Mayer Transformation

A Bolza problem can be transformed into a Mayer problem by defining a new state variable:

$$x^{(n_x+1)} := \int_{t_s}^{t_f} l(x(s),u(s),s)\,\mathrm{d}s, \quad t \in [t_s,t_f] \tag{4.8}$$

Then $x^{(n_x+1)}$ satisfies the initial value problem

$$\dot{x}^{(n_x+1)} = l(x(t),u(t),t), \quad t \in [t_s,t_f], \quad x^{(n_x+1)}(t_s) = 0. \tag{4.9}$$

With the augmented state vector $\bar{x} := \begin{pmatrix} x(t) \\ x^{(n_x+1)}(t) \end{pmatrix}$ and

$$\bar{f}(\bar{x},u,t) := \begin{pmatrix} f(x,u,t) \\ l(x,u,t) \end{pmatrix}, \qquad \bar{c}(\bar{x}(t),u(t),t) := c(x(t),u(t),t), \tag{4.10}$$

$$\bar{\psi}_s(\bar{x}) := \begin{pmatrix} \psi_s \\ x^{(n_x+1)}(t_s) \end{pmatrix}, \qquad \bar{\psi}_f(\bar{x}(t_f)) := \psi_f(x(t_f)) \tag{4.11}$$

the equivalent Mayer problem is obtained.

$$\min_{\bar{x},u,t_f} \quad \bar{J}(\bar{x}(t_f),t_f) = m(x(t_f),t_f) + x^{(n_x+1)}(t_f) \tag{4.12a}$$

$$\text{s.t.} \quad \dot{\bar{x}} = \bar{f}(\bar{x}(t),u(t),t) \tag{4.12b}$$

$$\bar{\psi}_s(\bar{x}(t_s)) = 0 \tag{4.12c}$$

$$\bar{\psi}_f(\bar{x}(t_f)) = 0 \tag{4.12d}$$

$$\bar{c}(\bar{x}(t),u(t),t) \le 0 \tag{4.12e}$$

Lagrange Transformation

A Mayer problem can be transformed into a Lagrange problem by extension of the domain of $m(x(t_f),t_f)$. Define a continuously differentiable function

$$\bar{m}(x(t),t_f) := \int_{t_s}^{t_f} \frac{\partial m}{\partial t} + \frac{\partial m}{\partial x} f(x,u,t)\ \mathrm{d}t, \quad \bar{m}(x(t_s),t_s) = 0. \tag{4.13}$$

The extended function $\bar{m}$ is likely only defined at points (x,t) where the process can terminate. The cost functional of the equivalent Lagrange problem is then

$$J(x(t),u(t),t) = \bar{m}(x(t),t_f). \tag{4.14}$$

Fixed Final Time Transformation

A process with free final time can be transformed in a process with fixed final time by definition of a new normalized time variable $\tau \in [0,1]$:

$$\tau := \frac{t - t_s}{t_f - t_s}, \quad t \in [t_s;\, t_f] \tag{4.15}$$

The state and control w.r.t. the transformed time are

$$\tilde{x}(\tau) := x(t_s + (t_f - t_s)\tau) = x(t), \quad \tilde{u}(\tau) := u(t_s + (t_f - t_s)\tau) = u(t) \tag{4.16}$$

which leads to a transformed dynamic system

$$\frac{\mathrm{d}\tilde{x}}{\mathrm{d}\tau} = \frac{dx}{\mathrm{d}t}\frac{\mathrm{d}t}{\mathrm{d}\tau} = (t_f - t_s)\ f(\tilde{x}(\tau),\tilde{u}(\tau),(t_f - t_s)\tau) \tag{4.17}$$

and a transformed cost functional

$$\tilde{J}(\tilde{x},\tilde{u},\tau) = m(\tilde{x}(1),1) + \int_0^1 (t_f - t_s)\ l(\tilde{x}(\tau),\tilde{u}(\tau),(t_f - t_s)\tau)\ \mathrm{d}\tau. \tag{4.18}$$

Defining the extended state vector $\bar{x}(\tau) := \begin{pmatrix} \tilde{x}(\tau) \\ t_f - t_s \end{pmatrix}$, the control $\bar{u}(\tau) := \tilde{u}(\tau)$, the functions

$$\bar{m}(\bar{x}(1),1) := m(\tilde{x}(1),1) \tag{4.19}$$

$$\bar{l}(\bar{x}(\tau),\bar{u}(\tau),\tau) := (t_f - t_s)\ l(\tilde{x}(\tau),\tilde{u}(\tau),(t_f - t_s)\tau) \tag{4.20}$$

$$\bar{c}(\bar{x}(\tau),\bar{u}(\tau),\tau) := c(\tilde{x}(\tau),\tilde{u}(\tau),(t_f - t_s)\tau) \tag{4.21}$$

and the dynamics

$$\bar{f}(\bar{x}(\tau),\bar{u}(\tau),\tau) := \begin{pmatrix} (t_f - t_s)\ f(\tilde{x}(\tau),\tilde{u}(\tau),(t_f - t_s)\tau) \\ 0 \end{pmatrix}, \quad \begin{matrix} \bar{\psi}_s(\bar{x}(0)) := \psi_s(\tilde{x}(0)), \\ \bar{\psi}_f(\bar{x}(1)) := \psi_f(\tilde{x}(1)), \end{matrix} \tag{4.22}$$

yields an equivalent OCP with fixed final time $t_f = 1$.

$$\min_{\bar{u}} \quad \bar{J}(\bar{x}(\tau),\bar{u}(\tau)) = \bar{m}(\bar{x}(1),1) + \int_0^1 \bar{l}(\bar{x},\bar{u},\tau)\ \mathrm{d}\tau \tag{4.23a}$$

$$\text{s.t.} \quad \frac{\mathrm{d}\bar{x}(\tau)}{\mathrm{d}\tau} = \bar{f}(\bar{x}(\tau),\bar{u}(\tau),\tau) \tag{4.23b}$$

$$\bar{\psi}_s(\bar{x}(0)) = \psi_s(\tilde{x}(0)) \tag{4.23c}$$

$$\bar{\psi}_f(\bar{x}(1)) = \psi_f(\tilde{x}(1)) \tag{4.23d}$$

$$\bar{c}(\bar{x}(\tau),\bar{u}(\tau),\tau) \leq 0 \tag{4.23e}$$

Autonomy Transformation

A non-autonomous problem can be transformed into an autonomous problem by defining the extended state vector $\bar{x}(t) := \begin{pmatrix} x(t) \\ t \end{pmatrix}$ and the functions

$$\bar{f}(\bar{x},u) := \begin{pmatrix} f(x,u,\bar{x}^{(n_x+1)}) \\ 1 \end{pmatrix}, \qquad \bar{m}(\bar{x},t_f) := m(x,t_f), \tag{4.24}$$

$$\bar{\psi}_0(\bar{x}) := \begin{pmatrix} \psi(x)_0 \\ t_s \end{pmatrix}, \qquad \bar{l}(\bar{x},u) := l(x,u,\bar{x}^{(n_x+1)}), \tag{4.25}$$

$$\bar{\psi}_1(\bar{x}) := \begin{pmatrix} \psi(x)_1 \\ \bar{x}^{(n_x+1)} \end{pmatrix}, \qquad \bar{c}(\bar{x},u) := c(x,u,\bar{x}^{(n_x+1)}). \tag{4.26}$$

The autonomous problem formulaton is obtained by replacing the corresponding functions in $\text{OCP}_{(4.7)}$.

4.1.3 Dynamic Programming in Continuous Time

The core idea of dynamic programming (DP) is that optimization can often be thought of as optimization in stages, based on a separable cost functional. The choice at any stage is a trade-off between being greedy, i.e. minimizing the cost at the current stage, and foreseeing the consequences of a greedy choice for the costs incurred at future stages. The optimal strategy minimizes the sum of the cost at the current stage plus the minimum of costs which occur at subsequent stages. Dynamic Programming was conceived in the 1940ties. A central figure in its development was Richard Bellman.

It was found that the idea of DP is not limited to discrete decision processes but that it can also be applied to optimal control problems based on ordinary differential equations. This leads to the Hamilton-Jacobi-Bellman (HJB) nonlinear partial differential equation (PDE). Although the guidance law developed within this thesis is not derived using the HJB equation, Dynamic Programming is an insightful approach to closed loop optimal control. In this frame we illustrate the dependency of the closed loop optimal control on the initial condition, and the relationship between neighboring initial conditions on an optimal trajectory. Finally we arrive at the definition of an optimal feedback law, which is the central point of interest for the following chapters.

One way to derive the HJB equation is to investigate the cost functional with respect to its explicit dependence on the initial condition. The cost incurred from a from a specific initial condition (t_s,x_s) to an admissible target is called the *cost-to-go*. Naturally one is interested in the *optimal* cost-to-go. But instead of minimizing the cost functional for a single initial condition, DP considers a family of problems for all initial conditions $x_s(t_s) \in \mathbb{R}^{n_x}$, $t_s \in [\bar{t}_s,t_f]$:

$$\inf_u J(t_s,x_s,x,u,t) = m(x(t_f)) + \int_{t_s}^{t_f} l(u(t),x(t),t)\,\mathrm{d}t \tag{4.27}$$

DP derives the dynamic relationship between problems with neighboring initial conditions which in some cases ultimately allows to solve all of them[1]. The dynamic relationship between problems of neighboring initial conditions is the so called Principle of Optimality (PoO).

Theorem 4.28 (Principle of Optimality)
Let $\overset{\star}{u} : [t_s,t_f] \to \mathbb{R}^{n_u}$ be an optimal control with corresponding optimal trajectory $\overset{\star}{x} : [t_s,t_f] \to \mathbb{R}^{n_x}$ for OCP$_{(4.7)}$. Then for any $t_1 \in [t_s,t_f]$ the restriction of the optimal control to $[t_1,t_f]$, $\overset{\star}{u}|_{[t_1,t_f]}$, is optimal for $\min_u J(t_1,\overset{\star}{x}(t_1),x,u,t)$ and the corresponding optimal trajectory is $\overset{\star}{x}|_{[t_1,t_f]}$.

Proof: The cost functional can be written as

$$J(t_s,x_s,u) = \int_{t_s}^{t_1} l(u(\tau),x(\tau),\tau)\,\mathrm{d}\tau + J(t_1,\overset{\star}{x}(t_1),\overset{\star}{u}|_{[t_1,t_f]}) \tag{4.29a}$$

Assume that $\overset{\star}{u}|_{[t_1,t_f]}$ is not optimal over the interval $[t_1,t_f]$ when the initial point is $x(t_1) = \overset{\star}{x}(t_1)$. Then there would exist an admissible control function $\bar{u}(t)$ defined on $[t_1,t_f]$ such that

$$J(t_1,\overset{\star}{x}(t_1),\bar{u}) < J(t_1,\overset{\star}{x}(t_1),\overset{\star}{u}|_{[t_1,t_f]}). \tag{4.29b}$$

But then the admissible control

$$u(t) = \begin{cases} \overset{\star}{u}(t) & t \in [t_s,t_1) \\ \bar{u}(t) & t \in [t_1,t_f] \end{cases} \tag{4.29c}$$

has the cost

$$J(t_s,x_s,u) = \int_{t_s}^{t_1} l(\overset{\star}{u}(\tau),\overset{\star}{x}(\tau),\tau)\,\mathrm{d}\tau + J(t_1,\overset{\star}{x}(t_1),\bar{u}) \tag{4.29d}$$

$$< \int_{t_s}^{t_1} l(\overset{\star}{u}(\tau),\overset{\star}{x}(\tau),\tau)\,\mathrm{d}\tau + J(t_1,\overset{\star}{x}(t_1),\overset{\star}{u}|_{[t_1,t_f]}) = \overset{\star}{J}(t_s,x_s,\overset{\star}{u}) \tag{4.29e}$$

which is a contradiction to the optimality of $\overset{\star}{u}$. Hence $\overset{\star}{u}|_{[t_1,t_f]}$ must be optimal for $[t_1,t_f]$, yielding the feasible optimal trajectory $\overset{\star}{x}|_{[t_1,t_f]}$. □

Written in infinitesimal form the PoO leads to the Hamilton-Jacobi-Bellman PDE. To derive the HJB equation we formulate the optimal cost-to-go as the *value function*:

$$V(t_s,x_s) = \inf_u J(t_s,x_s,x,u,t) \tag{4.30}$$

The value function is scalar and returns the optimal cost for all possible initial conditions. The PoO holds for all problems with separable cost, notwithstanding problems with control or state constraints. But in case of active inequality constraints the value function is usually not differentiable which is problematic for mathematical treatment. For the following derivation of the HJB equation we thus assume that V is continuously differentiable with respect to both arguments.

1 Note that in case of terminally and/or path- constraint problems an admissible optimal control might not exist, hence the infimum is used instead of a minimum.

According to the PoO for $\varDelta t > 0$:

$$V(t_s,x_s) = \inf_{u_{[t,t+\varDelta t]}} \left\{ \int_t^{t+\varDelta t} l(x(\tau),u(\tau),\tau)\mathrm{d}\tau + V\left(t+\varDelta t, x\left(t+\varDelta t, u_{[t,t+\varDelta t]}\right)\right) \right\} \tag{4.31}$$

For small $\varDelta t$ the integral term in (4.31) can be approximated assuming the quantity $l(x(\tau),u(\tau),\tau)$ is constant over the interval.

$$\int_t^{t+\varDelta t} l(x(\tau),u(\tau),\tau)\mathrm{d}\tau \approx l(x(t),u(t),t)\varDelta t \tag{4.32}$$

Small changes $\varDelta t, \varDelta x$ in the initial conditions of the value function can be approximated using a Taylor expansion. We note the Taylor expansion of first order and abbreviate higher order terms (HOT):

$$V(t_s+\varDelta t, x_s+\varDelta x) = V(t_s,x_s) + \dot{V}(t_s,x_s)\varDelta t + \langle \nabla_{x_s} V(t_s,x_s), \varDelta x\rangle + \mathrm{HOT} \tag{4.33}$$

where $\dot{V}$ is the time derivative, $\nabla_{x_s} V$ is the gradient w.r.t. x_s and $\langle \cdot,\cdot \rangle$ is the scalar product. Combining (4.32) and (4.33) with the PoO (4.31) and canceling $V(t_s,x_s)$ on both sides yields

$$0 \approx \inf_{u_{[t,t+\varDelta t]}} \left\{ l(x(t),u(t),t)\varDelta t + \dot{V}(t_s,x_s)\varDelta t + \langle \nabla_{x_s} V(t_s,x_s), \varDelta x\rangle + \mathrm{HOT} \right\}. \tag{4.34}$$

Divide by $\varDelta t$

$$0 \approx \inf_{u_{[t,t+\varDelta t]}} \left\{ l(x(t),u(t),t) + \dot{V}(t_s,x_s) + \left\langle \nabla_{x_s} V(t_s,x_s), \frac{\varDelta x}{\varDelta t} \right\rangle + \mathrm{HOT} \right\} \tag{4.35}$$

and take the limits $\varDelta t \to 0$, $\varDelta x \to 0$. The higher order terms go to zero, $\dot{V}(t_s,x_s)$ is independent of u and can hence be pulled out of the infimum, then one obtains the HJB partial differential equation

$$-\dot{V}(t_s,x_s) = \inf_u \left\{ l(x(t),u(t),t) + \langle \nabla_{x_s} V(t_s,x_s), f(x(t),u(t),t)\rangle \right\} \tag{4.36}$$

If the dynamics and the Lagrange term are time invariant and the final time is free the value function $V(x_s)$ only depends on the inital state and the HJB simplifies.

$$0 = \inf_u \left\{ l(x(t),u(t)) + \langle \nabla_{x_s} V(x_s), f(x(t),u(t))\rangle \right\} \tag{4.37}$$

Note that this is still a PDE unless $\dim(x) = 1$ for which it becomes an ODE.

The HJB equation (4.36) defines an optimal control problem in feedback or closed loop form through a backward recursion in time in which the last decision is made first. This is in contrast to an open loop formulation where $\overset{\star}{u}(t)$ is determined for all $t \in [t_s,t_f]$ at t_s. If $V(t_s,x_s)$ has continuous partial derivatives then $V(t_s,x_s)$ is a solution of the HJB equation and it can be shown that the minimization of $V(t_s,x_s)$ by a control $\overset{\star}{u}(t)$ is a sufficient condition for optimality. For a comprehensive discussion of this object we refer to Bertsekas [Ber07] and Sontag [Son98].

In DP the instantaneous optimal control is perceived as a function of the instantaneous time and state. This is a powerful approach in cases where the optimization problem in

(4.36) has an obtainable unique solution as a function of the instantaneous data (t_s,x_s). The knowledge of $V(t_s,x_s)$ then enables an optimal global feedback law.

Definition 4.38 (Feedback Law)
A continuous function $\Theta : I \times \mathbb{X} \to \mathbb{U}, I = [t_s,t_f]$ is called an admissible feedback law on I if for all $t_0 \in I$ and $x_0 \in \mathbb{X} \subseteq \mathbb{R}^{n_x}$ there exists a unique trajectory $x(u(t),t)$ on $[t_0,t_f]$ with

$$x(t_0) = x_0, \tag{4.38a}$$
$$u(t) = \Theta(t,x), \quad \text{for all } t \in [t_0,t_f]. \tag{4.38b}$$

In that case u is called the closed loop control starting at (t_0,x_0). An admissible feedback law is called optimal if $u(t) = \Theta(t,x)$ is a solution to (4.36).

If $V(t_s,x_s)$ would be known then the optimal feedback control is the point wise solution of

$$\overset{\star}{u}(t) = \arg\min_u \{l(x(t),u(t),t) + \langle \nabla_{x_s} V(t_s,x_s), f(x(t),u(t),t)\rangle\} \tag{4.39}$$

A well known example for optimal feedback is the linear quadratic regulator (LQR) for unconstrained, linear, time invariant systems with quadratic cost.

However for more complex problems no closed form solution of the HJB equation is known. Obtaining the value function analytically is exceptionally difficult and a numerical approach requires a discretization of the state space which is infeasible for almost all practically relevant problems, due to the *curse of dimensionality*. Furthermore the approach via the HJB equation is based on the assumption that the value function is continuously differentiable, which does not necessarily hold, even for simple problems, as demonstrated by Vinter [Vin10]. There have been efforts to extend the theory of DP using notions from nonsmooth analysis, so called viscosity solutions, this is for example discussed in Bressan [Bre07].

Finding the optimal closed loop feedback control (4.39) for more complex systems than the LQR case thus requires the solution of an optimal control problem at every time instant.

4.1.4 Optimal Control

Optimal control has a rich history dating back to 1697 when Bernoulli published the solution to his brachystochrone problem[1]. A major achievement was the publication of *Pontryagin's Minimum (Maximum) Principle* (PMP) in 1962 which was conceived in competition with Bellman's *Principle of Optimality* during the Cold War era[2]. The major evolution of PMP from the calculus of variations is the distinction of control and state variables and the new possibility to consider control constraints. The naming ambiguity results from whether the optimization is a minimization or maximization problem and the related sign convention for the Hamilton function.

Previously we pointed out that for most problems it is exceptionally hard if not impossible, to solve the HJB equation and to find the closed loop optimal control. Sometimes however

1 Sussmann, Hector J. et al.: *300 Years Of Optimal Control: From The Brachystochrone To The Maximum Principle.* 1997

2 Pesch, Hans Josef et al.: 'The Cold War and the Maximum Principle of Optimal Control'. *Documenta Mathematica* (2012), vol.: pp. 331–344

it is not necessary to know $\nabla_{x_s} V(x_s)$ at all values $x_s \in \mathbb{R}^{n_x}$, but only for a single fixed initial point $\bar{x}_s$. Then it is sufficient to know $\nabla_{x_s} V$ at only one value of x for each t, that is $\nabla_{x_s} V(\overset{\star}{x}(t))$ with $\overset{\star}{x}(t_s) = \bar{x}_s$. This is the core idea of the *Minimum Principle* (MP). The MP provides a strategy to actually obtaining the optimal solution for a single initial condition. Yet the guidance law developed in this thesis is not derived from the MP directly. However understanding the connection between the optimality conditions for a single initial condition (MP), and the optimality conditions for the closed loop solution (HJB equation) is fundamental for closed loop optimal control.

We want to illustrating a general idea rather than the details of the method, thus we consider the MP without state constraints and with free final time as stated by Athans [Ath06].

Theorem 4.40 (Minimum Principle (autonomous system, free time))
Given the problem

$$\begin{aligned}
\min_{u} \quad & J(u) = \int_{t_s}^{t_f} l(u(t),x(t))\,\mathrm{d}t && (4.40a)\\
\text{s.t.} \quad & \dot{x} = f(x,u)\\
& x(t_s) = x_s\\
& x(t_f) \in \mathbb{S}_f \subseteq \mathbb{R}^{n_x}\\
& u \in \mathbb{U} \subseteq \mathbb{R}^{n_u}
\end{aligned}$$

where l is continuous and f is continuously differentiable in x. The target set $\mathbb{S}_f$ is a smooth k-manifold in $\mathbb{R}^{n_x}$. Let $\overset{\star}{u}(t)$ be an admissible control which transfers x_s to $\overset{\star}{x}(\overset{\star}{t}_f) \in \mathbb{S}_f$ at free final time $\overset{\star}{t}_f$. Let $\overset{\star}{x}(t)$ be the corresponding state trajectory. In order for $\overset{\star}{u}(t)$ to be optimal it is necessary that there exists a function $\overset{\star}{\mu}(t)$ and a constant $\overset{\star}{\mu}_{\mathrm{abn}} \leq 0$, satisfying $(\overset{\star}{\mu}(t),\overset{\star}{\mu}_{\mathrm{abn}}) \neq (0,0), t \in [t_s,\overset{\star}{t}_f]$ such that

1. *$\overset{\star}{u}(t)$ and $\overset{\star}{\mu}(t)$ satisfy the canonical equations*

$$\begin{aligned}
\dot{\overset{\star}{x}}(t) &= \frac{\partial H}{\partial \mu}(\overset{\star}{x},\overset{\star}{u},\overset{\star}{\mu},\overset{\star}{\mu}_{\mathrm{abn}}) && \textit{state equation}\\
\dot{\overset{\star}{\mu}}(t) &= -\frac{\partial H}{\partial x}(\overset{\star}{x},\overset{\star}{u},\overset{\star}{\mu},\overset{\star}{\mu}_{\mathrm{abn}}) && \textit{adjoint equation}
\end{aligned}$$

   with the boundary conditions $\overset{\star}{x}(t_s) = x_s$, $\overset{\star}{x}(\overset{\star}{t}_f) \in \mathbb{S}_f$ where the Hamiltonian function H is defined as

$$H(x,u,\mu,\mu_{\mathrm{abn}}) := \langle \mu, f(x,u)\rangle + \mu_{\mathrm{abn}} l(x,u), \qquad \textit{for all } u \in \mathbb{U} \textit{ and for all } t \in [t_s,\overset{\star}{t}_f].$$

2. *The control $\overset{\star}{u}(t)$ is a global minimum of $H(\overset{\star}{x}(t),\cdot,\overset{\star}{\mu}(t),\overset{\star}{\mu}_{\mathrm{abn}})$.*

$$H(\overset{\star}{x}(t),\overset{\star}{u}(t),\overset{\star}{\mu}(t),\overset{\star}{\mu}_{\mathrm{abn}}) \leq H(\overset{\star}{x}(t),u,\overset{\star}{\mu}(t),\overset{\star}{\mu}_{abn}), \qquad \textit{for all } u \in \mathbb{U} \textit{ and for all } t \in [t_s,\overset{\star}{t}_f].$$

3. *The Hamiltonian is equivalent zero along the optimal path.*

$$H(\overset{\star}{x}(t),\overset{\star}{u}(t),\overset{\star}{\mu}(t),\overset{\star}{\mu}_{\mathrm{abn}}) \equiv 0, \qquad \textit{for all } t \in [t_s,\overset{\star}{t}_f].$$

4. *If the final state $\overset{\star}{x}(\overset{\star}{t}_f)$ is not restricted to a fixed endpoint x_f but is optimized in a*

target set $\mathbb{S}_f$ then the vector $\overset{\star}{\mu}(\overset{\star}{t}_f)$ is normal (transversal) to $\mathbb{S}_f$ at $\overset{\star}{x}(\overset{\star}{t}_f)$.

Remark 4.41 (Boundary Conditions)
Concerning (4.40.4): The canonical system consists of $2n_x$ ODEs and $2n_x$ boundary conditions defined by $\overset{\star}{x}(t_s) = x_s$ and $\overset{\star}{x}(\overset{\star}{t}_f) = x_f$. This situation changes when considering a variable endpoint. For $\overset{\star}{x}(\overset{\star}{t}_f) \in \mathbb{S}_f$, with $\mathbb{S}_f$ being a k-dimensional surface, conditions (4.40.1-3) hold as for a fixed endpoint. But for the $n_x - k$ free dimensions the final boundary conditions are replaced by the so called transversality condition

$$\langle \overset{\star}{\mu}(\overset{\star}{t}_f),\, d \rangle = 0 \tag{4.42}$$

for $d \in T_{\overset{\star}{x}(\overset{\star}{t}_f)}\mathbb{S}_f$ which is the tangent space to $\mathbb{S}_f$ at $\overset{\star}{x}(\overset{\star}{t}_f)$. Note that if $\mathbb{S}_f = \mathbb{R}^{n_x}$ then it can be shown that $\overset{\star}{\mu}(\overset{\star}{t}_f) = 0$, whereas when $\mathbb{S}_f = \{x_f\}$ then $\overset{\star}{\mu}(\overset{\star}{t}_f)$ is free.

Remark 4.43 (Abnormal Case)
In well posed problems it holds that $\overset{\star}{\mu}_{\text{abn}} > 0$ [Ath06](Chapter 5.13). Then the vector (μ_{abn},μ) can be normalized to $(1,\frac{\mu}{\mu_{\text{abn}}})$ while theorem (4.40) holds. Hence it is implicitly assumed that $\overset{\star}{\mu}_{\text{abn}} = 1$. The case $\overset{\star}{\mu}_{\text{abn}} = 0$ is called abnormal and is not considered.

The MP yields a system of highly nonlinear ODEs for the optimal trajectory and the corresponding adjoint vector which must be solved with appropriate boundary conditions. A general approach to solve an OCP using the MP is described in [Bre07](Chapter 6.2):

1. The Hamilton function $H(x,u,\mu)$ is determined.
2. The minimization problem $\overset{\star}{u}(t) = \arg\min_{u\in\mathbb{U}} H(x,u,\mu)$ is solved and the optimal control is expressed as a function of the state and the adjoint $u = \kappa(x,\mu)$. This requires determining the general structure of the control function and its switching points. If the dynamics $f(x,u)$ or the Lagrange term $l(x,u)$ depend nonlinear on u this can be a difficult task which can require a numerical approximation of $\kappa(x,\mu)$.
3. The control u is eliminated in the canonical equations by substituting $\kappa(x,\mu)$. This leads to the boundary value problem

$$\dot{x} = f(x,\kappa(x,\mu)) \tag{4.44a}$$
$$\dot{\mu} = -\nabla_x H(x,\kappa(x,\mu),\mu) \tag{4.44b}$$

   The boundary conditions are identical to the boundary conditions of the canonical system (4.41). Depending on the terminal conditions and/or a fixed final time transversality conditions may apply. The boundary value problem (4.44) is solved numerically for $\overset{\star}{x}(t)$, $\overset{\star}{\mu}(t)$.
4. The optimal control can then be computed as $\overset{\star}{u}(t) = \kappa(\overset{\star}{x}(t),\overset{\star}{\mu}(t))$.

The MP is a necessary condition for optimality, hence a solution of the canonical system does not imply optimality. Second order sufficient conditions are required to verify an optimal solution. In practice though it is often assumed that the canonical system has a unique solution, which must hence be optimal, such that the evaluation of second order conditions is not required. For a comprehensive discussion, the inclusion of state constraints and proofs of the MP we refer to Athans [Ath06] and Bressan [Bre07].

4.1.5 The Connection between DP and PMP

The HJB equation and the boundary conditions for OCP$_{(4.41)}$ are

$$\begin{aligned} 0 &= \inf_u \left\{ l(x,u) + \langle \nabla_{x_s} V(x_s), f(x,u) \rangle \right\}, \qquad (4.45)\\ V(x(t_f)) &= 0,\\ x(t_f) &\in \mathbb{S}_f. \end{aligned}$$

Because OCP$_{(4.41)}$ is autonomous the optimal closed loop control is according to (4.39)

$$\overset{\star}{u}_{\text{HJB}}(t; x_s) = \arg\min_u \left\{ l(x,u) + \langle \nabla_{x_s} V(x_s), f(x,u) \rangle \right\}, \quad x_s \in \mathbb{R}^{n_x} \tag{4.46}$$

To find the optimal control for a single fixed initial point $\bar{x}_s$ it is not necessary to know $\nabla_{x_s} V(x_s)$ at all values $x_s \in \mathbb{R}^{n_x}$. It is sufficient to know $\nabla_{x_s} V$ at only one value of x for each t, that is $\nabla_{x_s} V(\overset{\star}{x}(t))$ with $\overset{\star}{x}(t_s) = \bar{x}_s$.

The MP provides a strategy to actually obtaining $\nabla_{x_s} V(\overset{\star}{x}(t))$ by formulating the necessary conditions that a triple $\{\overset{\star}{u}, \overset{\star}{x}, \nabla_{x_s} V(\overset{\star}{x})\}$ must fulfill along an optimal trajectory. By restriction to a single initial condition the HJB equation can be converted into an ordinary differential equation by substituting

$$\overset{\star}{\mu}(t) = \nabla_{x_s} V(\overset{\star}{x}(t)) \tag{4.47}$$

into (4.45) which now only depends on time. This leads to the minimization of the Hamiltonian and to condition (4.41) of the MP

$$0 = \min_u \left\{ l(x,u) + \langle \overset{\star}{\mu}, f(x,u) \rangle \right\} = \min_u H(x,u,\overset{\star}{\mu}) = H(\overset{\star}{x}, \overset{\star}{u}_{\text{MP}}, \overset{\star}{\mu}) \tag{4.48}$$

The open loop optimal control

$$\overset{\star}{u}_{\text{MP}}(t) = \min_u H(x,u,\overset{\star}{\mu}) \tag{4.49}$$

depends on the unknown initial adjoint state $\mu_s = \overset{\star}{\mu}(t_s)$ which is obtained by solving the adjoint equation backwards in time through the canonical equation system (4.40.1). The final condition for $\overset{\star}{\mu}(\overset{\star}{t}_f)$ is given by the transversality condition (4.40.4). Note that the less restricted the final state $\overset{\star}{x}(\overset{\star}{t}_f)$ is, the more restricted is the adjoint state $\overset{\star}{\mu}(\overset{\star}{t}_f)$. The adjoint equation can be derived from the HJB equation under the assumption that $V(t_s,x_s)$ is two times continuously differentiable:

$$\dot{\overset{\star}{\mu}}(t) = \frac{\mathrm{d}\nabla_{x_s} V(\overset{\star}{x}(t))}{\mathrm{d}t} = \dot{\overset{\star}{x}}{}^{\intercal} \nabla^2_{x_s x_s} V(\overset{\star}{x}) = f(\overset{\star}{x},\overset{\star}{u})^{\intercal} \nabla^2_{x_s x_s} V(\overset{\star}{x}) \tag{4.50}$$

Differentiating (4.45) w.r.t. x at the optimal solution yields

$$\begin{aligned} 0 &= l_x(\overset{\star}{x},\overset{\star}{u}) + f(\overset{\star}{x},\overset{\star}{u})^{\intercal} \nabla^2_{x_s x_s} V(\overset{\star}{x}) + \nabla_{x_s} V(\overset{\star}{x}) f_x(\overset{\star}{x},\overset{\star}{u}) \qquad (4.51)\\ -\nabla_x H(\overset{\star}{x},\overset{\star}{u},\overset{\star}{\mu}) &= f(\overset{\star}{x},\overset{\star}{u})^{\intercal} \nabla^2_{x_s x_s} V(\overset{\star}{x}) \end{aligned}$$

which combined with (4.50) results into the adjoint equation. The state equation in (4.40.1) is a restatement of the relation between the adjoint- and the state variables in the Hamilton function which can be easily verified. For checking the boundary and transversality conditions we refer to Bertsekas [Ber07] and Bressan [Bre07].

The conclusion is that for problems with a two times continuously differentiable value function the minimum principle can be derived from the HJB equation. The central piece of this connection is that the adjoint variables are the gradient of the value function along the optimal trajectory.

4.2 Static Optimization

Static optimization refers to the minimization of a scalar objective function $f : \mathbb{R}^{n_z} \to \mathbb{R}$ which depends upon a finite dimensional vector of optimization variables $z \in \mathbb{R}^{n_z}$. The optimization can be subject to a finite number of constraint functions $g^{(m)} : \mathbb{R}^{n_z} \to \mathbb{R}$, $1 \leq m \leq n_g$. We only consider functions f and g that are at least one time continuously differentiable. The standard problem of continuous, nonlinear optimization is:

Problem 4.52 (Nonlinear Program)
Let the functions $f : \mathbb{R}^{n_z} \to \mathbb{R}$ and $g : \mathbb{R}^{n_z} \to \mathbb{R}^{n_g}$ be continuously differentiable. The problem

$$\min_{z \in \mathbb{R}^{n_z}} \quad f(z) \tag{4.52a}$$
$$\text{s.t.} \quad g^{(m)}(z) = 0 \quad m = 1,\ldots,n_{eq} \tag{4.52b}$$
$$g^{(m)}(z) \leq 0 \quad m = n_{eq}+1,\ldots,n_g \tag{4.52c}$$

is called standard nonlinear program (NLP).

In contrast to an OCP an NLP can be solved by a computer program, because it is finite dimensional and thus suitable for numeric solution methods. In the following we introduce the terminology that is required to characterize an NLP and it's optimal solution.

Definition 4.53 (Feasibility)
A vector z is said to be feasible for NLP$_{(4.52)}$ if z satisfies the constraints (4.52b) and (4.52c).

Definition 4.54 (Active Constraint)
A constraint $g^{(m)}$, $m = 1,\ldots,n_g$ is called active at a feasible z if $g^{(m)}(z) = 0$.

Definition 4.55 (Sets of Active Indices)
The set of active indices is

$$a(z) := \{m \in 1,\ldots,n_{eq}\} \cup \{m \in n_{eq}+1,\ldots,n_g \mid g^{(m)}(z) = 0\} \tag{4.56}$$

Definition 4.57 (Set of Active Constraints)
The set of active constraints is

$$g_a(z) := \{g^{(m)}(z),\, m \in a(z)\}$$

Definition 4.58 (Regularity)
A feasible vector z is called regular if the gradients $\nabla g_a(z)$ are linearly independent.

Definition 4.59 (Normality)
A feasible vector z is called normal if the gradients $\nabla g(z)$ are linearly independent.

Definition 4.60 (Minimum)
The function $f : D \to \mathbb{R}$ with domain $D \subset \mathbb{R}^{n_z}$ is said to have a local minimum at $\overset{\star}{z} \in D$ if $\varepsilon > 0$ exists such that for all $z \in D$ satisfying $\|z - \overset{\star}{z}\| < 0$ it holds that

$$f(\overset{\star}{z}) \leq f(z).$$

where $\|\cdot\|$ is a norm on $\mathbb{R}^{n_z}$. The minimum is called strict if the inequality is strict for $z \neq \overset{\star}{z}$. The minimum is called global if it holds for all $z \in D$.

4.2.1 Local Optimality Conditions

In the following the necessary and sufficient conditions for a local optimum of NLPs is stated according to Geiger and Kanzow [Gei02].

Definition 4.61 (Lagrange Function)
The function $L : \mathbb{R}^{n_z} \times \mathbb{R}^{n_g} \to \mathbb{R}$

$$L(z,\mu) = f(z) + \mu^{\intercal} g(z)$$

with multipliers $\mu \in \mathbb{R}^{n_g}$ is called Lagrange function of NLP$_{(4.52)}$ and the components of μ are called Lagrange multipliers.

Theorem 4.62 (First Order Necessary Conditions)
Let f, g be continuously differentiable and let $\overset{\star}{z}$ be regular. Then there exist unique Lagrange multipliers $\overset{\star}{\mu}$ such that

$$\begin{aligned}
\nabla_z L(\overset{\star}{z},\overset{\star}{\mu}) &= 0 && && \textit{Optimality} && (4.62a)\\
g^{(m)}(\overset{\star}{z}) &= 0, && m = 1,\ldots,n_{eq}, && \textit{Primal Feasibility (EQ)} && (4.62b)\\
g^{(m)}(\overset{\star}{z}) &\leq 0, && m = n_{eq}+1,\ldots,n_g, && \textit{Primal Feasibility (IEQ)} && (4.62c)\\
\overset{\star}{\mu}^{(m)} &\geq 0, && m = n_{eq}+1,\ldots,n_g, && \textit{Dual Feasibility} && (4.62d)\\
\overset{\star}{\mu}^{(m)} g^{(m)}(\overset{\star}{z}) &= 0, && m = n_{eq}+1,\ldots,n_g. && \textit{Complementarity} && (4.62e)
\end{aligned}$$

These conditions are known as Karush-Kuhn-Tucker (KKT) conditions. A satisfying point $(\overset{\star}{z},\overset{\star}{\mu})$ is called KKT point. The KKT conditions can be written in compact matrix form using the set of active indices:

$$K(\overset{\star}{z},\overset{\star}{\mu}_a) = \begin{pmatrix} \nabla_z L(\overset{\star}{z},\overset{\star}{\mu}_a) \\ g_a(\overset{\star}{z}) \end{pmatrix} = \begin{pmatrix} \nabla_z f(\overset{\star}{z}) + \overset{\star}{\mu}_a^{\intercal} \nabla_z g_a(\overset{\star}{z}) \\ g_a(\overset{\star}{z}) \end{pmatrix} = 0 \tag{4.63}$$

Necessary and sufficient optimality conditions of second order are:

Theorem 4.64 (Second Order Necessary Conditions)
If the functions f, g are twice continuously differentiable a necessary condition of second order for a KKT point $(\overset{\star}{z},\overset{\star}{\mu})$ to be a local minimum is

$$v^{\intercal} \nabla^2_{zz} L(\overset{\star}{z},\overset{\star}{\mu}) v \geq 0, \quad v \in \mathbb{R}^{n_z}$$

$\nabla^2_{zz} L(\overset{\star}{z},\overset{\star}{\mu})$ is the Hessian matrix of the Lagrange function.

Theorem 4.65 (Second Order Sufficient Conditions)
If the functions f,g are twice continuously differentiable a sufficient condition of second order for a KKT-point $(\overset{\star}{z},\overset{\star}{\mu})$ to be a local minimum is

$$v^\intercal \nabla^2_{zz} L(\overset{\star}{z},\overset{\star}{\mu}) v > 0, \quad v \in \mathbb{R}^{n_z} \backslash \{0\}$$

4.2.2 Solution of Nonlinear Programs with Sequential Quadratic Programming

An often used method for solving NLPs is sequential quadratic programming (SQP). In sequential quadratic programming NLP$_{(4.52)}$ is locally approximated by a convex subproblem which can be solved more easily and reliably. The solution of the subproblem is used as search direction for the original problem. The convex subproblems are usually obtained by a quadratic approximation of the Lagrange function and a linearization of the constraints as examplary shown below.

Problem 4.66 (Linearly Constrained Quadratic Approximation Problem)

$$\begin{aligned} \min_{d} \quad & \frac{1}{2} d^{[k]\intercal} \nabla^2_{zz} L(z^{[k]},\mu^{[k]}) d^{[k]} + \nabla_z f(z^{[k]})^\intercal d^{[k]} && && (4.66a) \\ \text{s.t.} \quad & g^{(m)}(z^{[k]}) + \nabla_z g^{(m)}(z^{[k]})^\intercal d^{[k]} = 0 && m = 1,\ldots,n_{eq} && (4.66b) \\ & g^{(m)}(z^{[k]}) + \nabla_z g^{(m)}(z^{[k]})^\intercal d^{[k]} \leq 0 && m = n_{eq}+1,\ldots,n_g && (4.66c) \end{aligned}$$

k is the iteration counter and $d^{[k]} = z - z^{[k]}$ are the optimization variables. If $(z^{[k]},\mu^{[k]})$ is in sufficiently close neighborhood to a KKT point of NLP$_{(4.52)}$ and all required functions are sufficiently differentiable then problem (4.66) is a local quadratic approximation of NLP$_{(4.52)}$ around $(z^{[k]},\mu^{[k]})$.

In each iteration problem (4.66) is solved using appropriate quadratic programming algorithms, for example the interior point method. The optimal solution $\overset{\star}{d}{}^{[k]}$ is used as search direction:

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

The step size $\alpha \in \mathbb{R}$ is determined using a line search method, like for example the *Armijo*-rule. The solution $\overset{\star}{d}{}^{[k]}$ is a descent direction for $L(z^{[k]},\mu^{[k]})$ if the matrix $H^{[k]} = \nabla^2_{zz} L(z^{[k]},\mu^{[k]})$ is positive definite. In praxis $H^{[k]}$ is often approximated because the exact second derivative evaluation is computationally very expensive.

The top level SQP algorithm can be summarized as follows:

1. Set $k := 0$, choose $z^{[k]} \in \mathbb{R}^{n_z}$, $\mu^{[k]} \in \mathbb{R}^{n_g}$ and $H^{[k]} \in \mathbb{R}^{n_z} \times \mathbb{R}^{n_z}$ symmetric.
2. If $(z^{[k]},\mu^{[k]})$ is a KKT-point of NLP$_{(4.52)}$: stop.
3. Compute solution $\overset{\star}{d}{}^{[k]}$ of subproblem (4.66) and obtain the associated multipliers $\mu^{[k+1]}$.
4. Determine the step size α
5. Update the estimate $z^{[k+1]} = z^{[k]} + \alpha \overset{\star}{d}{}^{[k]}$, choose $H^{[k+1]} \in \mathbb{R}^{n_z} \times \mathbb{R}^{n_z}$ symmetric, set $k := k + 1$ and go to 2.

More detailed information can for example be found in [Ber99] and [Gei02].

4.2.3 Relationship Between the Lagrange Multipliers and the Optimal Value

In Section 4.1.5 we have seen that the adjoint variables are the gradient of the value function on the optimal trajectory. In static optimization the Lagrange multipliers have a similar relationship with the optimal value. To make that relation obvious we consider an optimization problem with a scalar equality constraint $g(z) : \mathbb{R}^{n_z} \to \mathbb{R}$.

$$\min_{z} \quad f(z) \tag{4.67a}$$
$$\text{s.t.} \quad g(z) = 0 \tag{4.67b}$$

At a strict minimum $(\overset{\star}{z},\overset{\star}{\mu})$ it holds according to the KKT conditions that

$$\nabla_z L(\overset{\star}{z},\overset{\star}{\mu}) = \nabla_z f(\overset{\star}{z}) + \nabla_z \overset{\star}{\mu} g(\overset{\star}{z}) = 0. \tag{4.68}$$

Consider a parametrization of the constraint using a parameter $q \in \mathbb{R}$. For a nominal value $q := q_0 = 0$ we obtain the original solution $\overset{\star}{z}(q_0)$.

$$g(z) = q_0 \quad \Longrightarrow \quad g(\overset{\star}{z}(q_0),q_0) = g(\overset{\star}{z}(q_0)) - q_0 = 0 \tag{4.69}$$

If the solution of $\text{NLP}_{(4.67)}$ also exists in a neighborhood $\mathcal{N}$ around q_0, then for $q \in \mathcal{N}$ it holds that

$$\frac{\mathrm{d}g}{\mathrm{d}q} = \frac{\partial g}{\partial q} + \nabla_z g(\overset{\star}{z}(q),q)^\intercal \frac{\mathrm{d}\overset{\star}{z}}{\mathrm{d}q} = 0. \tag{4.70}$$

With $\frac{\partial g}{\partial q} = -1$ it follows that

$$\nabla_z g(\overset{\star}{z}(q),q)^\intercal \frac{\mathrm{d}\overset{\star}{z}}{\mathrm{d}q} = 1. \tag{4.71}$$

We can now obtain the directional total derivative of the objective function with respect to the perturbation q, in a neighborhood around the nominal value q_0:

$$\frac{\mathrm{d}f}{\mathrm{d}q}(q_0) = \nabla_z f(\overset{\star}{z})^\intercal \frac{\mathrm{d}z}{\mathrm{d}q}(q_0) \overset{(4.68)}{=} -\overset{\star}{\mu}\nabla_z g(\overset{\star}{z})^\intercal \frac{\mathrm{d}z}{\mathrm{d}q}(q_0) \overset{(4.71)}{=} -\overset{\star}{\mu} \tag{4.72}$$

Obviously the Lagrange multiplier is the total derivative (or sensitivity) of the objective function w.r.t. a perturbation in the constraint function. If $\overset{\star}{\mu} \gg 0$ it indicates that the constraint has a strong negative impact on the objective function and that a relaxation of the constraint could improve the optimal value. This result can be generalized to multiple constraints as well as to active inequality constraints. This leads to parametric sensitivity analysis, to which we come back in Chapter 5. .

4.3 Indirect and Direct Optimization Methods

In the previous sections we have illustrated the difference between the open and the closed loop solution of an optimal control process and we have stated optimality conditions for dynamic and static optimization problems. The closed loop solution of an optimal control process requires the solution of the HJB PDE, which is seldom possible. The open loop solution for each individual initial condition is the solution of an infinite dimensional optimal control problem. There are two approaches to actually obtaining the solution of an optimal control problem, which are both based on the discretization of the OCP into a finite dimensional, static optimization problem:

The first approach uses the necessary optimality conditions of Pontryagin's Minimum Principle. The OCP is transformed into a boundary value problem by formulating the optimal control as a function of the state and the adjoint state. The boundary value problem is then discretized and transformed into a static optimization problem, which can be solved numerically. Accordingly this approach is sometimes phrased as *'optimize then discretize'* [Bet10] and commonly referred to as *indirect* solution strategy.

The second approach discretizes the OCP directly, without evaluating the PMP optimality conditions. This likewise results into a static optimization problem. The difference is that this approach solely relies on the discrete optimality conditions (i.e. KKT) to solve the underlying optimal control problem. This is commonly known as *direct* optimization approach.

As far as a generalization is possible, *direct* optimization has the advantage of not requiring detailed apriori knowledge of the control switching structure and the adjoint variables, thus requiring a less accurate optimization start value. In turn *indirect* methods provide better insight in the solution structure and may achieve a higher solution accuracy.

In the remainder of this thesis we solely focus on *direct* optimization.

4.4 Direct Transcription Using the Shooting Method

This section describes a *direct* optimization approach for solving an OCP using the multiple shooting method [Boc84]. The OCP is discretized using a numeric integration method. The control variables (and depending on the method also the state variables) at the discretization points become the decision variables of a static optimization problem. The decision variables and constraints are ordered, such that the discrete OCP can be written as an NLP. The NLP is then solved using an appropriate numeric method, e.g. SQP. The process of transforming an OCP into an NLP is commonly referred to as *transcription*.

In the following it is assumed (without loss of generality) that the OCP is in autonomous, normalized form. If an OCP is not autonomous, it can be transformed into the autonomous form as described in Section 4.1.2. The independent variable $t \in [t_s,t_f]$ of the autonomous problem is then scaled to the *normalized time* $\tau \in [0,1]$ using the final time transformation as in Section 4.1.2. The process runtime is specified using the variable $t_d = t_f - t_s$. If either the initial time t_s or the final time t_f are free, the process duration t_d is free. If t_s and t_f are fixed, t_d is fixed. The standard OCP 4.7 can then be written as:

Problem 4.73 (Autonomous Normalized Optimal Control Problem)

$$\min_{x,u,t_d} J(x(\tau),u(\tau),t_d) = m(x(1),t_d) + t_d \int_0^1 l(x(\tau),u(\tau))\,\mathrm{d}\tau \tag{4.73a}$$

$$\text{s.t.} \qquad \dot{x} = t_d f(x(\tau),u(\tau)) \tag{4.73b}$$

$$\psi_s(x(0)) = 0 \tag{4.73c}$$

$$\psi_f(x(1)) = 0 \tag{4.73d}$$

$$c(x(\tau),u(\tau)) \leq 0 \tag{4.73e}$$

4.4.1 Discretization

$\mathrm{OCP}_{(4.73)}$ is discretized using a numerical integration scheme. In the following we exemplary use the explicit Euler method. It is advantageous to base the numerical implementation on the normalized formulation $\mathrm{OCP}_{(4.73)}$, because this allows to use the same implementation for solving fixed and free terminal time problems. For the discretization of the system dynamics $f(x,u)$ the control function $u(\tau)$ is thus discretized on a grid of the normalized time $\tau \in [0;\, 1]$. Select $l_u \geq 2$ support nodes $\tau^{(i)} \in [0;\, 1]$, $1 \leq i \leq l_u$ with

$$0 = \tau^{(1)} \leq \ldots \leq \tau^{(l_u)} = 1. \tag{4.74}$$

This defines the integration step sizes

$$h^{(i)} = \tau^{(i+1)} - \tau^{(i)}, \quad i \in [1, \ldots, l_u - 1] \tag{4.75}$$

and the control grid

$$T_u := \left\{\tau^{(1)}, \ldots, \tau^{(l_u)}\right\}. \tag{4.76}$$

The choice of the integration step sizes and the integration method has a strong impact on the size of the resulting NLP and the accuracy of the obtained solution.

We denote $u^{(i)} = u(\tau^{(i)})$ as an approximation of the control vector at normalized time $\tau^{(i)} \in T_u$. Numerically integrating the dynamics from the initial state $x(0)$ using the controls $u^{(1)}, \ldots, u^{(l_u-1)}$ yields the integrated state trajectory $x^{(2)}, \ldots, x^{(l_u)}$ with respect to normalized time. Relation (4.16) can be used to obtain an approximation of the trajectory with respect to the original time.

The objective function is approximated with the trapezoidal rule. Finally $\mathrm{OCP}_{(4.73)}$ can be written in discretized form:

Problem 4.77 (Discretized Optimal Control Problem (Euler, Trapezoidal Quadrature))

$$\min_{x,u,t_d} \; f(x,u,t_d) = m(x(1),t_d) + \frac{t_d}{2}\sum_{i=1}^{l_u-1} h^{(i)}\left[l(x^{(i)},u^{(i)}) + l(x^{(i+1)},u^{(i+1)})\right] \tag{4.77a}$$

$$\text{s.t.} \quad x^{(i+1)} = x^{(i)} + h^{(i)} t_d f(x^{(i)},u^{(i)}), \quad i \in [1, \ldots, l_u - 1] \tag{4.77b}$$

$$\psi_s(x(0)) = 0 \tag{4.77c}$$

$$\psi_f(x(1)) = 0 \tag{4.77d}$$

$$c(x^{(i)},u^{(i)}) \leq 0, \quad i \in [1, \ldots, l_u]. \tag{4.77e}$$

Single Shooting

The idea of single shooting is to transform $\mathrm{OCP}_{(4.77)}$ into a finite dimensional problem by considering principally the control variables at the chosen discretization points as decision variables

$$z_{\mathrm{SS}} = \left(u^{(1)}, \ldots, u^{(l_u-1)}\right). \tag{4.78}$$

If the process duration t_d is free, or if there are free dimensions in the initial or final state, the corresponding variables in $x(0), x(1)$ can be added as additional decision variables.

In OCP$_{(4.77)}$ the state at $\tau^{(i)}$, $i \in [1, ..., l_u]$ is obtained recursively based on the discrete dynamics (4.77b), i.e. state is determined by integration, starting from $x(0)$, using the control vectors $u(0), ..., u(\tau^{(i-1)})$. We denote the integrated state as

$$\tilde{x}(\tau^{(i)}) = x(x(0), u(0), ..., u(\tau^{(i-1)})). \tag{4.79}$$

The trajectory defined by the integrated state is referred to as *shooting trajectory* (cf. Figure 4.1a). The boundary condition to reach the terminal state $x(1)$ can now be written as shooting defect constraint

$$D = \tilde{x}(1) - x(1) = 0. \tag{4.80}$$

To complete the NLP formulation the path constraints (4.77e) are checked at a subset of T_u, using the integrated state $\tilde{x}(\tau^{(i)})$ as approximation of $x^{(i)}$, meaning $c(u^{(i)}, \tilde{x}^{(i)})$ is evaluated for some $i \in [1, ..., l_u]$.

Multiple Shooting

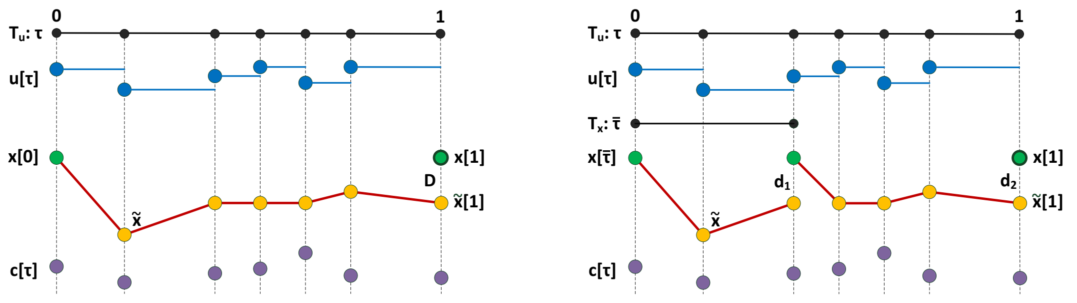
Multiple shooting breaks the OCP into segments and performs *single shooting* on each segment individually. Multiple shooting discretizes the state equations directly, while at the same time adding additional defect constraints to enforce the continuity of the state trajectory at the discretization points.

Let the state trajectory $x(\tau)$ be discretized on the grid

$$T_x \subseteq T_u, \quad \{0; 1\} \in T_x. \tag{4.81}$$

$T_s := T_x \backslash \{1\}$ are called shooting nodes.

Let k, $1 \leq k \leq l_x - 1$ be the index of the shooting nodes. The intervals $S^{(k)} = [\tau^{(k)}; \tau^{(k+1)}]$, $\tau^{(k)} \in T_s$, are called shooting intervals. Note that $\tau^{(k)} \in T_s$ is also in T_u. The index i for which $\tau^{(k)} = \tau^{(i)} \in T_u$ depends on the distribution of the integration steps over the shooting



(a) (b)

Figure 4.1: Single shooting (a) and multiple shooting (b) using Euler integration of the state equations and zero order hold of the control.

intervals (cf. Figure 4.1b).

The initial state for the integration in interval $S^{(k)}$ is $x(\tau^{(k)})$. Let the endpoint of the integration chain in interval $S^{(k)}$ be $\tilde{x}[\tau^{(k+1)}]$. To assure the continuity of the state trajectory the defect constraints D are added:

$$D = \begin{pmatrix} d^{(1)} \\ \vdots \\ d^{(l_x-1)} \end{pmatrix} = \begin{pmatrix} \tilde{x}(\tau^{(2)}) - x(\tau^{(2)}) \\ \vdots \\ \tilde{x}(1) - x(1) \end{pmatrix} = 0. \tag{4.82}$$

Multiple shooting breaks the dependency of the state on the control prior to the previous shooting node. Depending on the length of the shooting intervals this leads to a large sparse NLP. For the case $T_x := T_u$ all shooting intervals have length one, which is referred to as *full discretization* (cf. Figure 4.2b). Full discretization obviously results into the highest possible sparsity.

The decision variables for multiple shooting are the control vectors on the grid T_u and the state vectors on the grid T_x.

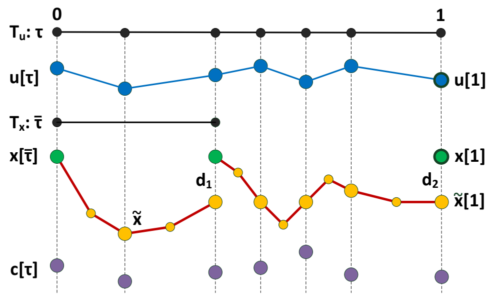
$$z_{\text{MS}} = \left(u^{(1)},\ldots,u^{(l_u-1)},x^{(1)},\ldots,x^{(l_x)}\right). \tag{4.83}$$

If the process duration t_d is free, it is an additional decision variable.

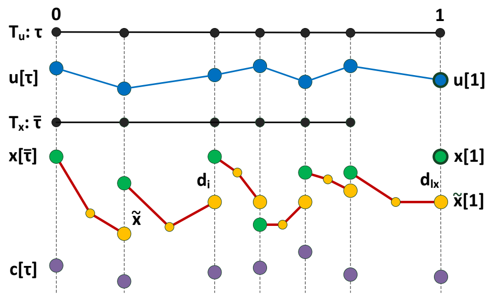
Remark 4.84 (Interpolation of the Control Function)
For higher order integration methods the control function must be evaluated at intermediate points of the grid T_u. To obtain the control variables at the intermediate points the control function is commonly interpolated using constant, linear or cubic spline interpolation. In case of an explicit one-step method, e.g. Runge-Kutta-4, the state at the inner interval points can be computed by numerical forward-evaluation of the inner integration steps (cf. Figure 4.2). Another approach is to make the intermediate points part of the discretization grid. The advantageous and disadvantageous of different combinations of integration-, discretization- and control interpolation methods are for example discussed in [Bet10].

Remark 4.85 (Approximation of the Location of Corners and Points of Discontinuity)
With a fixed discretization grid, the exact location of corners and points of discontinuity remains unknown. The solution of the transcribed NLP is inaccurate, because it is smeared over these critical points. If the structure of the continuously differentiable arcs is known, the solution accuracy can be improved by determining the exact location of the critical points. The problem must therefore be divided into multiple phases, corresponding to the continuously differentiable arcs. The phase boundaries are linked together by defect constraints, analogously to the shooting defects.

An optimization of the phase lengths can be used to determine the location of critical points exactly, as at an optimum the phase transition points must match the critical points. This approach however requires a very good optimization start value. It can be practical to first solve the problem, neglecting the critical points, and then to use the solution as start value for a multi-phase formulation.



(a)



(b)

Figure 4.2: Multiple shooting (a) and full discretization (b) transcription using linear control interpolation. The dynamic equations are integrated using a higher order one-step method, e.g. Runge-Kutta (RK4), requiring the evaluation of the state and control at an intermediate interval point.

4.4.2 Formulation as Nonlinear Program

By grouping the optimization variables and constraints a discretized OCP can be written in form of an NLP.

$$\begin{aligned} \min_{z} \quad & f(z) \\ \text{s.t.} \quad & g^{(m)}(z) = 0 \quad m = 1,\ldots,n_{eq} \\ & g^{(m)}(z) \leq 0 \quad m = n_{eq}+1,\ldots,n_g \end{aligned}$$

While the NLP objective function can be directly obtained from the discretized OCP, the discretized constraints must be assembled into joined the function $g(z)$.

The human readability of an NLP and the observability of the temporal context of the underlying optimal control problem heavily depend on the ordering of the decision variables and constraints. An exemplary ordering that allows a blockwise extraction of the discretized state and control functions is detailed in Appendix E.

After the transcription into an NLP an OCP can be treated numerically. NLPs are predominately solved using Newton or Quasi-Newton methods, where in each iteration either QP or an interior point method is used to determine a feasible descent direction. This concludes the preliminary considerations on optimization theory.

Remark 4.86 (NLP Scaling)
The convergence rate of Newton or Quasi-Newton methods depends on the condition of the KKT matrix (4.63)*. In practice scaling techniques to improving the condition of the KKT matrix play an important role: Even if theoretically an optimal solution is attainable, numerical reasons prevent the solution of badly conditioned problems. An automatic scaling technique for discretized OCPs based on Büskens [Büs98] and Betts [Bet10] is illustrated in Appendix F.*

CHAPTER 5

Real-Time Approximation of Optimal Controls Using Parametric Sensitivities

For closed loop optimal control an OCP has to be solved at each time instant to take into account the current initial state. In a clocked control system the time required to solve the associated optimization problem is the limiting factor concerning the frequency of the control loop. It is thus a central goal of current research to speed-up the solution of OCPs arising during closed loop control.

The complexity of OCPs is not determined by linearity, but by convexity and the consideration of inequality constraints. Less complex problems allow for more specialized solution algorithms which require less computational effort and thus enable a faster solution, which is of greatest importance for the control of fast paced dynamics. Closed loop optimal control technologies can be categorized according to the classes of static optimization problems to which they can be applied (Figure 5.1).

Linear control systems without constraints and with a quadratic objective[1] can be transcribed into a quadratic program (QP) without inequality constraints. For this special

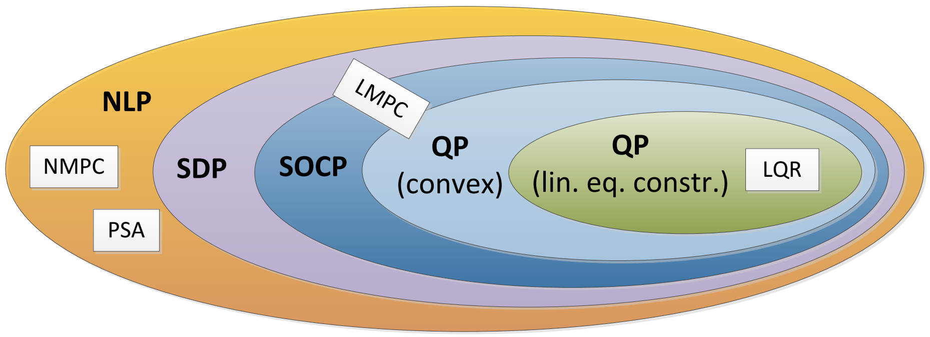


Figure 5.1: Classes of optimization problem and closed loop optimal control strategies

1 More specifically a weighted sum of quadratic control input and quadratic state error.

case a solution of the HJB equation is known, having the form of a linear, time invariant state feedback law which is known as the Ricatti control law, or linear quadratic regulator (LQR).

If inequality constraints on the control or state functions must be taken into account a closed form optimal feedback law, like LQR, is not possible, because the associated optimization problem must be solved iteratively, e.g. by using Newton's method.

An important distinction is whether or not the optimization problem is convex, because for convex problems finding the global optimum can be guaranteed with a polynomial bound on the required computation time. The most general convex optimization problem formulation is a semidefinite program (SDP) (based on linear matrix inequalities). A popular subclass of SDPs are second order cone programs (SOCPs), which restrict the optimum to lie in cone. SOCPs include QPs with convex constraints as special cases. Because of the guaranteed upper runtime bound convex optimization problems are well suited for clocked receding horizon control. Based on the underlying linear dynamic system this approach is referred to as linear model predictive control (LMPC), instead of more accurately convex MPC.

The open loop solution for a nonconvex OCP can be obtained by solving an NLP. Likewise receding horizon control can be based on NLPs to achieve (sub-) optimal closed loop control, which is commonly referred to as nonlinear model predictive control (NMPC).

A popular approach to reduce the computational effort for closed loop optimal control is to cast the associated optimization problem into a lower complexity class, which is especially attractive if one is able to go from a nonconvex to a convex formulation. Determining such a reformulation without or with acceptable simplifications can be very difficult though.

This thesis investigates a different approach to reduce the computational effort for closed loop control of a nonconvex optimal control process. We avoid online optimization altogether by approximating neighboring optimal solutions based on Taylor expansion using linearizions of the first order necessary optimality conditions of multiple optimization problems which arise during closed loop execution. In this chapter we review recent related results in the field of parametric sensitivity analysis (PSA) of NLPs and then extend the theory to a closed loop control law for nonlinear dynamic systems.

5.1 Parametric Optimal Problem and Parametric Sensitivity

A parametric OCP is the extension of a single OCP into a family of OCPs described by a parameter vector.

Problem 5.1 (Parametric, Autonomous, Normalized Optimal Control Problem)
Based on the definitions from Section 4.1.1 a family of optimal control problems is defined by introducing a parameter vector $p \in \mathbb{R}^{n_p}$.

$$\min_{x,u,t_d} J(x(\tau),u(\tau),t_d,p) = m(x(1),t_d,p) + t_d \int_0^1 l(x(\tau),u(\tau),p)\,\mathrm{d}\tau \tag{5.1a}$$

$$\text{s.t.} \qquad \dot{x} = t_d f(x(\tau),u(\tau),p) \tag{5.1b}$$

$$\psi_s(x(0),p) = 0 \tag{5.1c}$$

$$\psi_f(x(1),p) = 0 \tag{5.1d}$$

$$c(x(\tau),u(\tau),p) \le 0 \tag{5.1e}$$

This problem is denoted $OCP_{(5.1)}(p)$*. For a nominal parameter value* $p := p_0$ *the problem* $OCP_{(5.1)}(p_0)$ *has the nominal, optimal solution* $\overset{\star}{x}(\tau,p_0)$*,* $\overset{\star}{u}(\tau,p_0)$*,* $\overset{\star}{t}_d(p_0)$*.*

Parametric sensitivity analysis determines the change of the nominal solution with respect to a small change in the nominal parameter.

Let all functions in $\mathrm{OCP}_{(5.1)}(p)$ be sufficiently continuously differentiable with respect to their arguments such that the nominal solution is locally a continuously differentiable function of the parameter vector p almost everywhere. Consider the perturbations

$$\varDelta p_k = (0, ..., \varDelta p^{(k)}, ..., 0)^\intercal, \quad 1 \le k \le n_p. \tag{5.2}$$

Let $p_k = p_0 + \varDelta p_k$ and let $\overset{\star}{u}^{(j)}(\tau,p_0)$, $1 \le j \le n_u$ be the nominal optimal control functions of $\mathrm{OCP}_{(5.1)}(p_0)$ with the corresponding optimal state trajectories $\overset{\star}{x}^{(j)}(\tau,p_0)$, $1 \le j \le n_x$ and the optimal duration $\overset{\star}{t}_d(p_0)$. The total directional derivatives

$$\frac{\mathrm{d}\overset{\star}{u}^{(j)}}{\mathrm{d}p^{(k)}}(\tau,p_0) = \lim_{\varDelta p^{(k)} \to 0} \frac{\overset{\star}{u}^{(j)}(\tau,p_0 + \varDelta p^{(k)}) - \overset{\star}{u}^{(j)}(\tau,p_0)}{\varDelta p^{(k)}}, \quad 1 \le j \le n_u,\ 1 \le k \le n_p \tag{5.3a}$$

$$\frac{\mathrm{d}\overset{\star}{x}^{(j)}}{\mathrm{d}p^{(k)}}(\tau,p_0) = \lim_{\varDelta p^{(k)} \to 0} \frac{\overset{\star}{x}^{(j)}(\tau,p_0 + \varDelta p^{(k)}) - \overset{\star}{x}^{(j)}(\tau,p_0)}{\varDelta p^{(k)}}, \quad 1 \le j \le n_x,\ 1 \le k \le n_p \tag{5.3b}$$

$$\frac{\mathrm{d}\overset{\star}{t}_d}{\mathrm{d}p^{(k)}}(p_0) = \lim_{\varDelta p^{(k)} \to 0} \frac{\overset{\star}{t}_d(p_0 + \varDelta p^{(k)}) - \overset{\star}{t}_d(p_0)}{\varDelta p^{(k)}}, \quad 1 \le k \le n_p. \tag{5.3c}$$

are called parametric sensitivities (PS) of the optimal solution in direction $\varDelta p_k$.

The above calculation however is numerically problematic and inefficient (n_p additional optimal control problems would have to be solved to determine the effects of the perturbations on the optimum). A numerically stable and efficient computation method is illustrated in the following section.

A necessary condition for parametric sensitivity analysis (PSA) is the differentiability of the nominal solution with respect to the parameters, which must hold at least locally around the nominal parameter value. Maurer and Pesch [Mau94][Mau95] provide a theoretical basis for PSA by proving the solution differentiability for parametric OCPs under mixed state and control constraints. Parts of the theory are subject to ongoing research, namely a proof for OCPs with pure state constraints is an open problem.

Depending on how the nominal solution of an OCP was obtained, PSA can be based on either indirect or direct solution methods. Both approaches linearize necessary conditions of optimality around the nominal parameter value. This work focuses on the direct approach based on a linearization of the KKT conditions of NLPs.

5.2 Parametric Sensitivity Analysis of Nonlinear Programs

The PSA of NLPs was first investigated by Fiacco [Fia83]. Later Büskens [Büs98] and Maurer [Mau09] investigate the sensitivity of discretized OCPs in the frame of NLPs using SQP methods [Büs00].

The sensitivities obtained under KKT conditions are interpreted as approximative solutions to (5.3). A prerequisite for this approximation is the convergence of the finite dimensional NLP to the optimal control solution obtained using PMP. This convergence has been proven for different classes of OCPs. Malanowski et al. [Mal97] derive convergence for problems with mixed control and state constraints using an explicit Euler discretization, Dontchev and Hager [Don01] prove the convergence of the explicit Euler discretization for simultaneous control and state constraints. Hager [Hag00] proves convergence for control constrained problems using Runge-Kutta methods.

To directly obtain parametric sensitivities for $\mathrm{OCP}_{(5.1)}(p)$ the problem is discretized and transcribed into a parametric NLP analogously to the process described in Chapter 4.4, while additionally carrying the parameter p.

Problem 5.4 (Parametric Nonlinear Program)
Let $z \in \mathbb{R}^{n_z}$, $p \in \mathcal{P} \subset \mathbb{R}^{n_p}$ and let $f : \mathbb{R}^{n_z} \times \mathbb{R}^{n_p} \to \mathbb{R}$ and $g : \mathbb{R}^{n_z} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_g}$. The family of nonlinear programming problems defined by a parameter vector p

$$\min_{z\in\mathbb{R}^{n_z}} \quad f(z,p) \tag{5.4a}$$
$$\text{s.t.} \quad g^{(m)}(z,p) = 0 \quad m = 1,\ldots,n_{eq} \tag{5.4b}$$
$$g^{(m)}(z,p) = 0 \quad m = n_{eq}+1,\ldots,n_g \tag{5.4c}$$

is called $\mathrm{NLP}_{(5.4)}(p)$. For a nominal parameter $p := p_0$ the nominal solution is $\overset{\star}{z}_0 = \overset{\star}{z}(p_0)$ with corresponding Lagrange multipliers $\overset{\star}{\mu}_0 = \overset{\star}{\mu}(p_0)$.

We follow the work of Büskens [Büs02] to introduce a fast solution approximation method for parametric NLPs, which is the basis of our investigation.

Note that inactive constraints have no influence on the evaluation of necessary or sufficient conditions, because for the Lagrangian multipliers at an optimal solution it holds that $\overset{\star}{\mu}^{(m)} = 0$, $m \notin a(z)$, cf. (4.55). If the nominal solution and the active indices are known, the considerations for post optimal sensitivity analysis can be reduced to the active constraints g_a and the associated multipliers μ_a. The Lagrangian function of the reduced problem, expanded with the parameter p, is thus

$$L(z,\mu_a,p) = f(z,p) + (\mu_a)^\intercal g_a(z,p). \tag{5.5}$$

Consider a fixed nominal parameter $p_0 \in \mathcal{P}$ and the corresponding nominal problem $\mathrm{NLP}_{(5.4)}(p_0)$. If the nominal solution $\overset{\star}{z}(p_0)$ is regular and the sufficient conditions (4.65) hold, then the nominal solution and the associated Lagrange multipliers are locally differentiable functions of the parameter p:

Theorem 5.6 (Sensitivity)
Let f and g be two times continuously differentiable w.r.t. to z and let the gradients $\nabla_z f$ and $\nabla_z g$ and the function g be one time continuously differentiable with respect to p. Furthermore let $\overset{\star}{z}$ and $\overset{\star}{\mu}$ be regular and fulfill the sufficient conditions of optimality (4.65) for $\mathrm{NLP}_{(5.4)}(p_0)$. Then there exists a neighborhood $\mathcal{N}_0(p_0)$ and continuously differentiable functions $z : \mathcal{N}_0(p_0) \to \mathbb{R}^{n_z}$ and $\mu : \mathcal{N}_0(p_0) \to \mathbb{R}^{n_g}$ such that

1. *$z(p_0) = \overset{\star}{z}$, $\mu(p_0) = \overset{\star}{\mu}$,*
2. *the set of active indices does not change,*

$$a(z(p),p) \equiv a(z(p_0),p_0), \quad \text{for all } p \in \mathcal{N}_0(p_0), \tag{5.6a}$$

3. *the gradients in $\nabla_z g_a(z(p),p)$ are linearly independent*

$$rank(\nabla_z g_a(z(p),p)) = |a(z(p),p)|, \quad \textit{for all } p \in \mathcal{N}_0(p_0), \tag{5.6b}$$

4. *$\overset{\star}{z}(p)$ and $\overset{\star}{\mu}(p)$ are regular and fulfill the sufficient conditions of optimality (4.65) for the disturbed problem $NLP_{(5.4)}(p)$. Furthermore $\overset{\star}{z}(p)$ is a strong local minimum for $NLP_{(5.4)}(p)$ with Lagrange multiplier $\overset{\star}{\mu}(p)$.*

Proof: The proof can be found in [Fia83]. □

Based on Theorem (5.6) an explicit formulation of the sensitivity differentials of the primal- and dual-variables can be given. The KKT conditions (4.63) for $\text{NLP}_{(5.4)}(p)$ are

$$K(\overset{\star}{z},\overset{\star}{\mu}_a,p) = \begin{pmatrix} \nabla_z L(\overset{\star}{z},\overset{\star}{\mu}_a,p) \\ g_a(\overset{\star}{z},p) \end{pmatrix} = \begin{pmatrix} \nabla_z f(\overset{\star}{z},p) + (\overset{\star}{\mu}_a)^\intercal \nabla_z g_a(\overset{\star}{z},p) \\ g_a(\overset{\star}{z},p) \end{pmatrix} = 0. \tag{5.7}$$

Under the conditions of Theorem (5.6) the Jacobian matrix of $K(\overset{\star}{z},\overset{\star}{\mu}_a,p)$ w.r.t. (z,μ_a) can be obtained.

$$\nabla_{z,\mu_a} K(\overset{\star}{z},\overset{\star}{\mu}_a,p) = \begin{pmatrix} \nabla_z^2 L(\overset{\star}{z},\overset{\star}{\mu}_a,p) & (\nabla_z g_a(\overset{\star}{z},p))^\intercal \\ \nabla_z g_a(\overset{\star}{z},p) & 0 \end{pmatrix} \tag{5.8}$$

The differentiation of $K(\overset{\star}{z}(p),\overset{\star}{\mu}_a(p),p) \equiv 0$ with respect to p at the nominal parameter p_0 leads to a linear system for the PS of the entire optimal solution and the associated Lagrangian multipliers.

$$\begin{pmatrix} \nabla_z^2 L(\overset{\star}{z}_0,\overset{\star}{\mu}_{a,0},p_0) & (\nabla_z g_a(\overset{\star}{z}_0,p_0))^\intercal \\ \nabla_z g_a(\overset{\star}{z}_0,p_0) & 0 \end{pmatrix} \begin{pmatrix} \frac{\mathrm{d}z}{\mathrm{d}p}(p_0) \\ \frac{\mathrm{d}\mu_a}{\mathrm{d}p}(p_0) \end{pmatrix} + \begin{pmatrix} \nabla_{zp}^2 L(\overset{\star}{z}_0,\overset{\star}{\mu}_{a,0},p_0) \\ \nabla_p g_a(\overset{\star}{z}_0,p_0) \end{pmatrix} = 0 \tag{5.9}$$

Under the conditions of Theorem (5.6) the matrix (5.8) is invertible, which leads to the following explicit expression for the PS.

$$\begin{pmatrix} \frac{\mathrm{d}z}{\mathrm{d}p}(p_0) \\ \frac{\mathrm{d}\mu_a}{\mathrm{d}p}(p_0) \end{pmatrix} = -\begin{pmatrix} \nabla_z^2 L(\overset{\star}{z}_0,\overset{\star}{\mu}_{a,0},p_0) & (\nabla_z g_a(\overset{\star}{z}_0,p_0))^\intercal \\ \nabla_z g_a(\overset{\star}{z}_0,p_0) & 0 \end{pmatrix}^{-1} \begin{pmatrix} \nabla_{zp}^2 L(\overset{\star}{z}_0,\overset{\star}{\mu}_{a,0},p_0) \\ \nabla_p g_a(\overset{\star}{z}_0,p_0) \end{pmatrix} \tag{5.10}$$

The Hessian $\nabla_z^2 L(\overset{\star}{z}_0,\overset{\star}{\mu}_{a,0},p_0)$ evaluated at the nominal solution is guaranteed to be positive definite under the assumptions and the right hand side is computable.

With the multiple shooting transcription of Chapter 4.4 the matrix $\frac{\mathrm{d}z}{\mathrm{d}p}$ is of the form (5.11).

$$\left(\frac{\mathrm{d}z}{\mathrm{d}p}(p_0)\right)_{n_z \times n_p} = \begin{array}{c} \begin{array}{ccc} \overbrace{\hphantom{\frac{\mathrm{d}u^{(:,1)}}{\mathrm{d}p^{(1)}}(p_0)}}^{1} & \hphantom{\dots} & \overbrace{\hphantom{\frac{\mathrm{d}u^{(:,1)}}{\mathrm{d}p^{(n_p)}}(p_0)}}^{1} \end{array} \\ \left(\begin{array}{ccc} \frac{\mathrm{d}u^{(:,1)}}{\mathrm{d}p^{(1)}}(p_0) & \dots & \frac{\mathrm{d}u^{(:,1)}}{\mathrm{d}p^{(n_p)}}(p_0) \\ \vdots & \ddots & \vdots \\ \frac{\mathrm{d}u^{(:,n_u)}}{\mathrm{d}p^{(1)}}(p_0) & \dots & \frac{\mathrm{d}u^{(:,n_u)}}{\mathrm{d}p^{(n_p)}}(p_0) \\ \frac{\mathrm{d}x^{(:,1)}}{\mathrm{d}p^{(1)}}(p_0) & \dots & \frac{\mathrm{d}x^{(:,1)}}{\mathrm{d}p^{(n_p)}}(p_0) \\ \vdots & \ddots & \vdots \\ \frac{\mathrm{d}x^{(:,n_x)}}{\mathrm{d}p^{(1)}}(p_0) & \dots & \frac{\mathrm{d}x^{(:,n_x)}}{\mathrm{d}p^{(n_p)}}(p_0) \\ \frac{\partial t_l}{\partial p^{(1)}}(p_0) & \dots & \frac{\partial t_l}{\partial p^{(n_p)}}(p_0) \end{array}\right) \begin{array}{l} \}\, l_u \\ \\ \}\, l_u \\ \}\, l_x \\ \\ \}\, l_x \\ \}\, 1 \end{array} \end{array} \tag{5.11}$$

Block $\frac{\mathrm{d}u^{(:,j)}}{\mathrm{d}p^{(k)}} \in \mathbb{R}^{l_u} \times \mathbb{R}$, $1 \le j \le n_u$, $1 \le k \le n_p$ contains the parametric sensitivities of the nominal discretized control function $u^{(:,j)}$ on the grid T_u (4.76) against a perturbation in dimension k of the parameter vector p. The layout of the blocks for the discretized state trajectories follows analogously on the grid T_x (4.81) respectively.

Before the wide popularity of direct optimization methods PSA was conducted based on indirect methods, using a linearization of the PMP conditions. Pesch [Pes89a][Pes89b] transforms an OCP with mixed state and control constraints into a multi-point boundary value problem. The multi-point boundary value problem is solved numerically using a modified multiple shooting method which allows to obtain the PS as a byproduct. For problems with mixed state and control constraints an expanded set of necessary conditions is required to characterize the entry, exit and contact points of the active constraint arcs and the induced conditions on the adjoint variables. If the control function is only piecewise continuously differentiable, additionally the points of discontinuity must be determined. The linearization of the expanded PMP conditions then leads to sensitivities including the shifting behavior of the switching points.

The main advantage of the direct approach is that the KKT conditions are more compact and easier to handle than the PMP conditions. Direct methods often omit the explicit determination of points of discontinuity of the control function through the immediate discretization of the OCP. If the points of discontinuity are not explicilty considered in the transcription, the related jump condititions, as used in PMP, are not reflected in the KKT conditions, which simplifies the analysis, but it can also result into a lower solution accuracy, because the discontinuities and corner points are only determined up to the accuracy of the discretization grid.

5.3 Real-Time Approximation of NLP Solutions

In control engineering an important application of PSA is the computation of fast near optimal feedback controls. Real-time capable optimal control techniques are highly demanded technologies that have vast potential in many application fields. For systems with fast nonlinear dynamics the online solution of an optimal control problem is sometimes not feasible on computationally weak embedded systems. An alternative to online optimization is the use of parametric sensitivity analysis to compute neighboring optimal solutions using a Taylor expansion of the nominal solution.

5.3.1 Taylor Expansion of the Nominal NLP Solution

If the nominal solution $\overset{\star}{z}(p_0), \overset{\star}{\mu}_a(p_0)$ of $\text{NLP}_{(5.4)}(p_0)$ and the parametric sensitivities of the nominal solution are known, a linear approximation of the optimal solution $\overset{\star}{z}(p), \overset{\star}{\mu}_a(p)$ of $\text{NLP}_{(5.4)}(p)$ for disturbed parameters p can be obtained using a Taylor expansion of first order.

$$\begin{pmatrix} \overset{\star}{z}(p) \\ \overset{\star}{\mu}_a(p) \end{pmatrix} \approx \begin{pmatrix} \overset{\star}{z}_0 \\ \overset{\star}{\mu}_{a,0} \end{pmatrix} + \begin{pmatrix} \frac{\mathrm{d}z}{\mathrm{d}p}(p_0) \\ \frac{\mathrm{d}\mu_a}{\mathrm{d}p}(p_0) \end{pmatrix} (p - p_0) \tag{5.12}$$

The perturbed parameter p is assumed to be known from measurement or estimation at update time. All other quantities in (5.12) can be computed offline and are known at process runtime. For control applications usually only an update of the control function and/or the state trajectories are of interest, while the Lagrange multipliers are not required.

Equation (5.12) can then be reduced to

$$\overset{\star}{z}(p) \approx \tilde{z}^{[1]}(p) := z_0 + \frac{\mathrm{d}z}{\mathrm{d}p}(p_0)\,(p - p_0) \tag{5.13a}$$

$$= \underbrace{z_0 - \frac{\mathrm{d}z}{\mathrm{d}p}(p_0)\,p_0}_{\text{prepared offline}} + \underbrace{\frac{\mathrm{d}z}{\mathrm{d}p}(p_0)\,p}_{\text{online adaption}}\,. \tag{5.13b}$$

The evaluation of (5.13b) only requires one matrix vector multiplication and one vector addition, such that it can be computed almost instantly even on computationally weak processing hardware. The Taylor expansion of the optimal solution under KKT-conditions takes account feasibility and optimality. The approximated solution contains an adapted near optimal control sequence for the entire process and, depending on the state discretization, the adapted state trajectory.

5.3.2 Iterative Feasibility and Optimality Restoration

For optimization problems with constraints that are nonlinear in the variables or the parameters, the approximation $\tilde{z}^{[1]}(p)$ does not fulfill the constraints exactly, due to the truncation of the Taylor expansion (5.13b). The resulting constraint error can be computed exactly as

$$g_a(\tilde{z}^{[1]},p) = \varepsilon^{[1]} \neq 0. \tag{5.14}$$

The approximate solution $\tilde{z}^{[1]}(p)$ can be significantly improved [Büs02] by taking the error $\varepsilon^{[1]}$ into account. Consider linear perturbations $q \in \mathbb{R}^{n_g}$ in the constraint function of $\mathrm{NLP}_{(5.4)}(p)$.

$$g(z,p) - q \leq 0. \tag{5.15}$$

The linear constraint perturbations q are a specialization of the general perturbations p. Obviously for a nominal value of $q_0 = 0$ the original problem is retained. The parameter q is also of the same type as the constraint error $\varepsilon^{[1]}$. In the sensitivity analysis the perturbations q can be considered analogously to the parameters p. Let $\frac{\mathrm{d}z}{\mathrm{d}q_a}$ be the sensitivity of the active constraints against linear perturbations. With a transcription as in Chapter 4.4 this matrix is of the form (5.16).

$$\left(\frac{\mathrm{d}z}{\mathrm{d}q_a}\right)_{n_z \times n_a} = \begin{array}{cccl} \overbrace{}^{|C_a|} & \overbrace{}^{|D|} & \overbrace{}^{|\Psi_a|} & \\ \frac{\mathrm{d}u}{\mathrm{d}q_C} & \frac{\mathrm{d}u}{\mathrm{d}q_D} & \frac{\mathrm{d}u}{\mathrm{d}q_\Psi} & \}\, l_u n_u \\ \frac{\mathrm{d}x}{\mathrm{d}q_C} & \frac{\mathrm{d}x}{\mathrm{d}q_D} & \frac{\mathrm{d}x}{\mathrm{d}q_\Psi} & \}\, l_x n_x \\ \frac{\mathrm{d}t_d}{\mathrm{d}q_C} & \frac{\mathrm{d}t_d}{\mathrm{d}q_D} & \frac{\mathrm{d}t_d}{\mathrm{d}q_\Psi} & \}\, 1 \end{array} \tag{5.16}$$

Using (5.16) a better approximation than (5.13b) can be obtained by

$$\tilde{z}^{[k+1]}(p) = \tilde{z}^{[k]}(p) + \frac{\mathrm{d}z}{\mathrm{d}q_a}(q_0)\, g_a(\tilde{z}^{[k]},p), \tag{5.17}$$

starting with $k = 1$. Equation (5.17) defines an iterative procedure for $k = 2, 3, \ldots$. Büskens [Büs02] has shown, that under additional assumptions on the functions f and g iteration

(5.17) converges linearly against a fixpoint $\tilde{z}^{[\infty]}(p)$, if the perturbation Δp is sufficiently small. At the fixpoint the active constraints are fulfilled exactly.

Theorem 5.18 (Convergence on Invariant Subspaces)
Let the conditions on Theorem (5.6) be satisfied and let the functions f and g be three times continuously differentiable with respect to z and p. Then there exists a neighborhood $\mathcal{U}(p_0)$ of p_0, such that for all $p = p_0 + \Delta p \in \mathcal{U}(p_0)$ the following statements are true. Consider the orthogonal decomposition

$$\frac{1}{2}(\Delta p)^{\intercal}\frac{\mathrm{d}^2 z}{\mathrm{d}p^2}(p_0,0)\Delta p = v + w, \tag{5.18a}$$

with $v \in Kern(\nabla_z g_a(p_0)) \subset \mathbb{R}^{n_z}$ and $w \in (Kern(\nabla_z g_a(p_0)))^{\perp} \subset \mathbb{R}^{n_z}$, then the following error bounds hold

$$\|v\| = \mathcal{O}(\|\Delta p\|^2) \tag{5.18b}$$
$$\left\|z(p) - \tilde{z}^{[\infty]}(p)\right\| = \mathcal{O}(\|\Delta p\|^2) \tag{5.18c}$$
$$\left\|f(z(p),p) - f(\tilde{z}^{[\infty]}(p),p)\right\| = \mathcal{O}(\|\Delta p\|^3) \tag{5.18d}$$
$$\left\|g_a(\tilde{z}^{[\infty]}(p),p)\right\| = 0. \tag{5.18e}$$

If additionally the Lagrange multipliers are iterated with

$$\tilde{\mu}_a^{[k+1]}(p) := \tilde{\mu}_a^{[k]}(p) - \frac{\mathrm{d}\mu_a}{\mathrm{d}q_a}(p_0,0)g_a(\tilde{z}^{[k]}(p),p), \tag{5.18f}$$

then (5.18f) also converges against a fixpoint $\tilde{\mu}^{[\infty]}(p)$ with the following error bounds:

$$\left\|\mu_a(p) - \tilde{\mu}_a^{[\infty]}(p)\right\| = \mathcal{O}(\|\Delta p\|^2) \tag{5.18g}$$
$$\left\|\nabla_z L(\tilde{z}^{[\infty]}(p),\tilde{\mu}_a^{[\infty]}(p),p)\right\| = \mathcal{O}(\|\Delta p\|^2) \tag{5.18h}$$

The iterations (5.17) and (5.18f) converge linearly and the fixpoints $\tilde{z}^{[\infty]}(p)$ and $\mu_a^{[\infty]}(p)$ depend on the nominal value p_0 and the perturbation p: $\tilde{z}^{[\infty]}(p; p_0,p)$, $\mu_a^{[\infty]}(p; p_0,p)$.

Proof: The proof can be found in [Büs02]. □

Theorem (5.18) offers a significant improvement over the Taylor expansion (5.13b). In particular the corrector iteration (5.17) allows to obtain a trajectory which fulfills the discretized dynamics exactly: The discrete dynamics (represented through the shooting equality constraints) are always part of the active set, thus the differential defects are eliminated through the corrector iteration. At the fixpoint the integration of $\tilde{u}^{[\infty]}(p) \subset \tilde{z}^{[\infty]}(p)$ yields a state trajectory which is an exact solution to the discrete dynamics.

The real-time solution approximation is summarized in Algorithm (1).

Algorithm 1 Real-time solution approximation [Büs02]

```
1: procedure RTS($p$,$\varepsilon^{[\infty]}$)
2:     $\tilde{z}^{[k]} = z_0 - \frac{\mathrm{d}z}{\mathrm{d}p}\, p_0 + \frac{\mathrm{d}z}{\mathrm{d}p}\, p$                ▷ (5.13b)
3:     while $\left\| g_a(z^{[k]},p)\right\| > \varepsilon^{[\infty]}$ do
4:         $\tilde{z}^{[k+1]} = \tilde{z}^{[k]} + \frac{\mathrm{d}z}{\mathrm{d}q_a}\, g_a(z^{[k]},p)$                ▷ (5.17)
5:     end while
6:     return $\tilde{z}^{[\infty]}$
7: end procedure
```

5.4 Instructive Example: Control of a Rocket Car

In this section the real-time approximation scheme is applied to an academic optimal control problem that we use as basis for discussion.

We consider the thrust optimal control of a *rocket car* on a rail (see Figure 5.2). A straight forward modelling of this motion corresponds to a double integrator. In addition we consider an efficiency loss on the control force and a quadratic drag law. The resulting dynamics are easy to grasp, yet nonlinear in the control and the state, which makes them an interesting discussion example. The control problem is to transfer the car from the initial state to the final state in a fixed amount of time, using minimum fuel.

Problem 5.19 (Rocket Car)

$$\min_u J(x(\tau),u(\tau),p) = t_d \int_0^1 u(\tau)^2 \,\mathrm{d}\tau \tag{5.19a}$$

$$\text{s.t.} \quad \dot{r} = t_d v \tag{5.19b}$$

$$\dot{v} = t_d \left(u - e u|u| \; - \; c_D v|v|\right), \quad u \in \mathbb{R} \tag{5.19c}$$

$$\psi_s(x(0),p) = \begin{pmatrix} r_s \\ v_s \end{pmatrix} \tag{5.19d}$$

$$\psi_f(x(1),p) = \begin{pmatrix} r_f \\ v_f \end{pmatrix} \tag{5.19e}$$

where $t_d \in \mathbb{R}^+$ is the fix process duration, $e \in [0;\, 1[$ is the engine efficiency loss and $c_D \in \mathbb{R}^+$ is the drag coefficient. The parameter vector p and its nominal value p_0 are

$$p := \begin{pmatrix} e \\ c_D \\ r_s \\ v_s \\ r_f \\ v_f \\ t_d \end{pmatrix}, \quad p_0 := \begin{pmatrix} e_0 \\ c_{D,0} \\ r_{s,0} \\ v_{s,0} \\ r_{f,0} \\ v_{f,0} \\ t_{d,0} \end{pmatrix} = \begin{pmatrix} 0.3 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 3 \end{pmatrix}. \tag{5.19f}$$

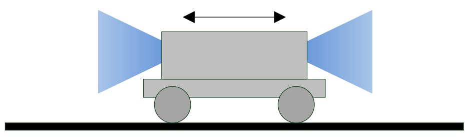


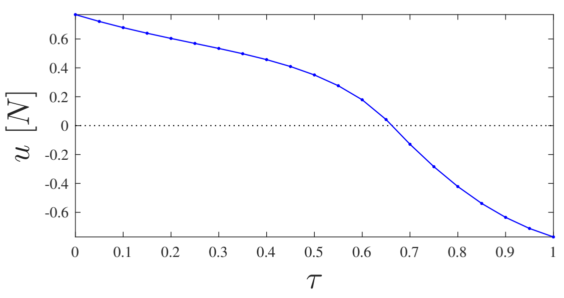
Figure 5.2: Rocket car

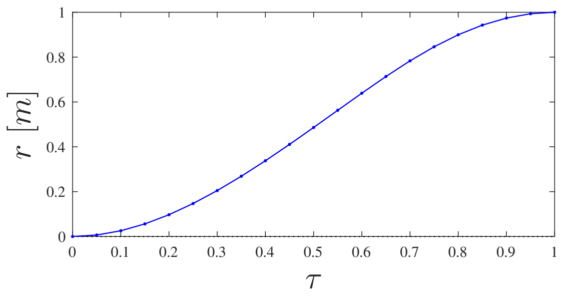
5.4.1 Nominal Solution and Parametric Sensitivities

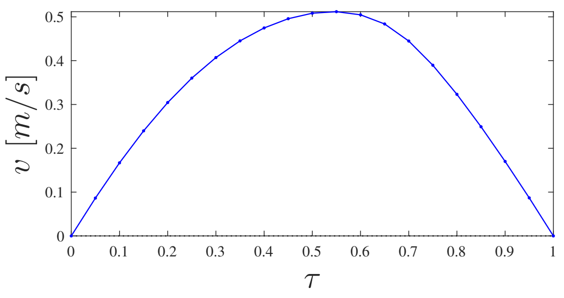
OCP$_{(5.19)}$ is solved for the nominal case $p := p_0$. The problem is discretized using single shooting with a Runge-Kutta scheme (RK4) on an equidistant control grid. The control function is approximated using piecewise linear intervals.

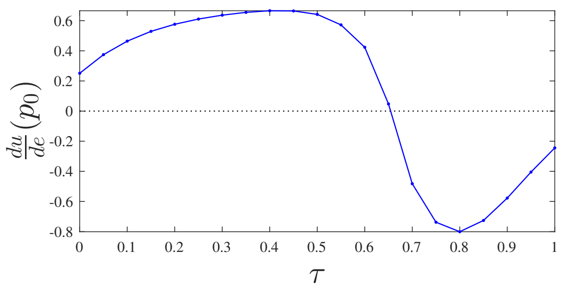
The nominal optimal control function and the integrated state trajectories are shown in Figures 5.3a to 5.3c. Due the drag, more fuel is required for accelerating the car than for stopping it, such that the switching of the control occurs at $\tau \approx 0.65$.

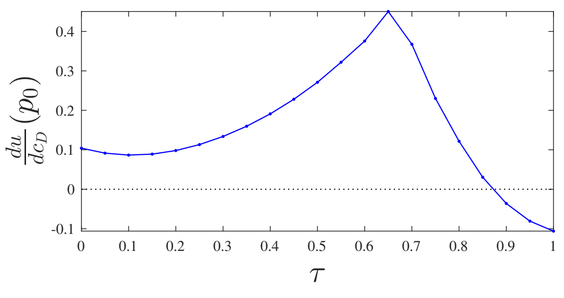
The sensitivity differentials of the optimal control function (Figures 5.3d to 5.3j) predict the effects of perturbations on the optimal solution. The simple system nature allows us to compare the sensitivities against common sense expectations: If the engine inefficiency increases, the required fuel increases as well, before and after the switching point, but it does not impact the switching time itself (Figure 5.3d). An increase in drag however causes a delay of the switching point (strong peak in Figure 5.3e). The sensitivities with respect to the initial position (Figure 5.3f) and the final position (Figure 5.3h) are mirrored. Obviously both impact the distance that the car must move equally, except that the sign is reversed. The sensitivities with respect to initial and final velocity (Figures 5.3g and 5.3i) however are not mirrored, because of the drag influence. An increased process duration obviously reduces the amount of required fuel (Figure 5.3j).

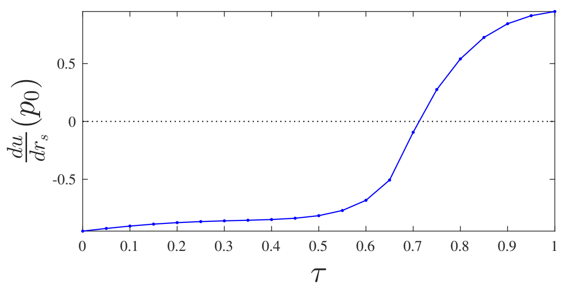

(a) Nominal control function

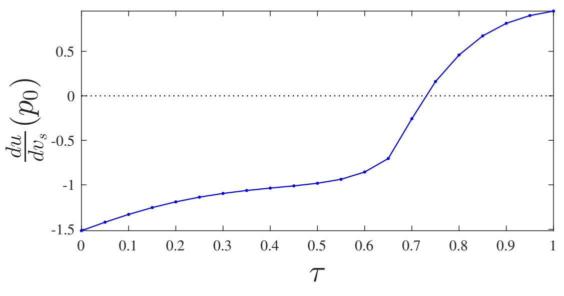

(b) Nominal position

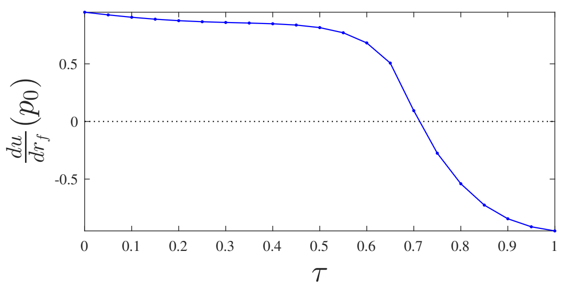

(c) Nominal velocity

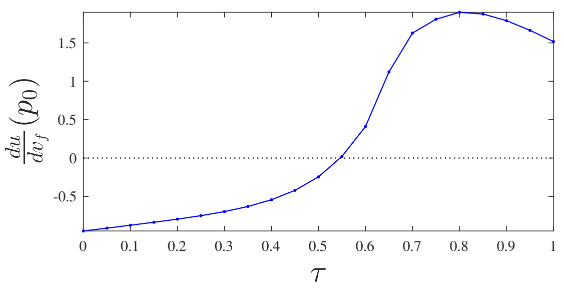

(d) Control sen. w.r.t. efficiency


(e) Control sen. w.r.t. drag coef.


(f) Control sen. w.r.t. initial position


(g) Control sen. w.r.t. initial velocity


(h) Control sen. w.r.t. final position


(i) Control sen. w.r.t. final velocity

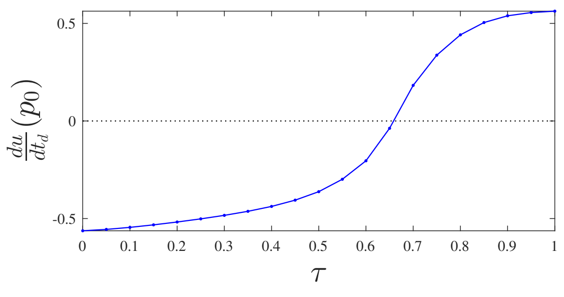

(j) Control sen. w.r.t. process duration

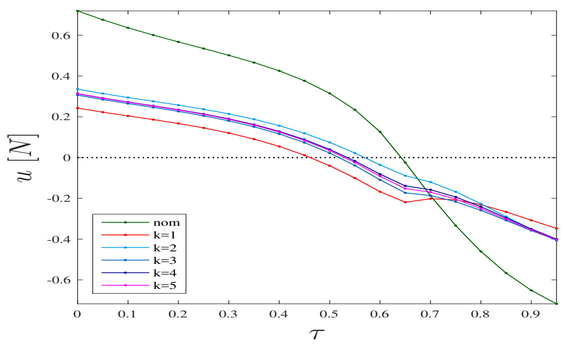
Figure 5.3: The images show the nominal control function (a), the integrated nominal state trajectories (b, c) and the parametric sensitivity of the nominal control function w.r.t. the perturbation parameters (d-j).

5.4.2 Real-Time Solution Approximation

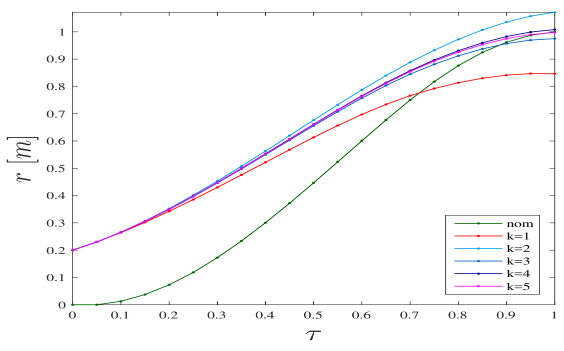
In the following we apply Algorithm 1 to counteract a perturbation in initial state, let

$$p := \begin{pmatrix} e \\ c_D \\ r_s \\ v_s \\ r_f \\ v_f \\ t_d \end{pmatrix} = \underbrace{\begin{pmatrix} 0.3 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 3 \end{pmatrix}}_{p_0} + \underbrace{\begin{pmatrix} 0 \\ 0 \\ 0.2 \\ 0.2 \\ 0 \\ 0 \\ 0 \end{pmatrix}}_{\Delta p}. \tag{5.20}$$

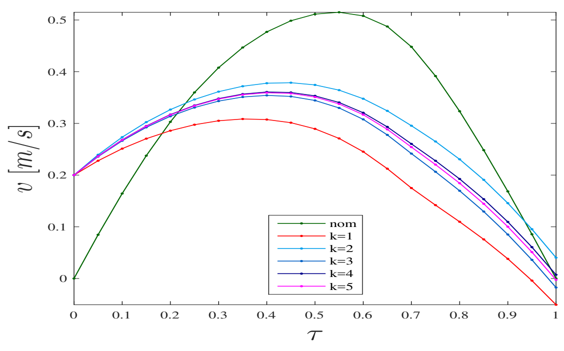
The application of the nominal control function to the perturbed system results into an error at the final state. Using Algorithm 1 the control function can be adapted to correct the error. Figure 5.4 shows the resulting iterates. The initial Taylor expansion yields a linear solution approximation ($k = 1$, red). The remaining constraint error is successively reduced through the feasibility restoration iteration. The iteration sequence is converging towards a fix-point at which the error in the active constraints is eliminated (cf. Theorem 5.18). After four corrector iterations ($k = 5$) the constraint error has been reduced to about $2 \cdot 10^{-3}$. Additional iterations continue to reduce the error, but the changes are visually indiscernible.



(a) Real-time approximation of the optimal control function



(b) Position



(c) Velocity

Figure 5.4: Real-time solution approximation for $\Delta r_s = 0.2$ and $\Delta v_s = 0.2$ based on single shooting. The state trajectories are obtained from integration of the dynamics using the adapted control sequence.

5.5 Exploring Parametric-Sensitivity-Based Control

To derive a closed loop control law on basis of the real-time solution approximation algorithm, the solution obtained in the open loop NLP frame has to be led back to the closed loop optimal control problem. In this chapter we prepare this step by exploring sensitivity-based open loop controls to improve our understanding of the effects that different problem formulations and discretizations have on Algorithm 1. The main points of investigation are:

1. The use of state discretization in conjunction with Algorithm 1.
2. The convergence region of iteration (5.17).
3. Monitoring of the convergence progress at runtime.
4. The limitations of the approach.

5.5.1 Discretization of the State Functions

According to Betts [Bet10] multiple shooting offers decisive advantages over single shooting: Multiple shooting is less dependent on an accurate optimization start value and it does not suffer convergence problems in case of highly sensitive initial states or controls. In addition multiple shooting often has a better convergence rate. The question arises if multiple shooting possibly has similar advantages for the real-time solution approximation scheme.

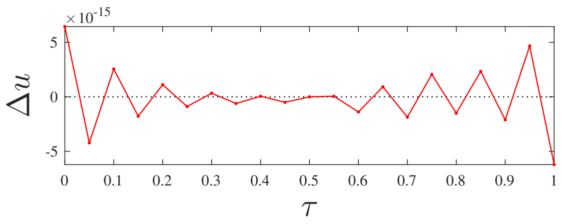
We investigate this question numerically at the example introduced in Section 5.4. We compare two different NLP formulations, $\mathrm{RC}_{\mathrm{SS}}(p)$ and $\mathrm{RC}_{\mathrm{FD}}(p)$ of $\mathrm{OCP}_{(5.19)}(p)$. Both formulations are based on an identical control grid, but the discretization of the state functions differs. $\mathrm{RC}_{\mathrm{SS}}(p)$ is based on single shooting, while $\mathrm{RC}_{\mathrm{FD}}(p)$ is based on full discretization. Both NLPs are solved for an identical nominal parameter p_0.

We investigate the influence of the state discretization on the parametric sensitivities in a highly precise numerical framework. The optimal solutions that are the basis for this analysis fulfill the KKT conditions to 10^{-15}, the required derivatives are quasi-analytic, determined through algorithmic differentiation. The achieved numerical accuracy is thus close to machine precision within the 64-bit IEEE-754 standard, which is $\varepsilon \approx 1.11 \cdot 10^{-16}$.

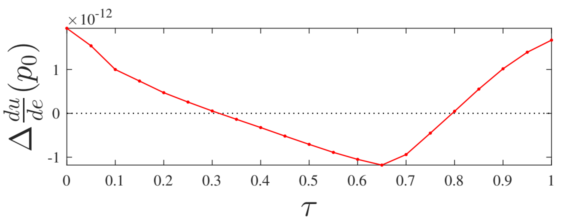
The parametric sensitivities are computed at a KKT point; the numerical error in the KKT conditions (compared to analytical analysis) propagates into the numeric sensitivity differentials (5.10). This effect can be seen in Figure 5.5, which compares the control function values and the sensitivity differentials of both formulations. The differences in the controls are only slightly above the enforced precision[1] (Figure 5.5a). However the differences between the parametric sensitivities (Figure 5.5b to 5.5h) are two to three orders of magnitude larger[2]. This is the consequence from approximating the analytical PMP optimality conditions through structurally differing KKT conditions. As the analytic sensitivity differentials are unknown it is impossible to quantize the absolute error, or to determine which NLP formulation yields more accurate sensitivity differentials. But a noteworthy observation is that the sensitivities obtained via (5.10) depend on the specific NLP formulation and they should not be mixed with sensitivities of other NLP formulations of the same OCP.

1 The difference can exceed the enforced precision, because the optimal objective values of $\mathrm{RC}_{\mathrm{SS}}(p_0)$ and $\mathrm{RC}_{\mathrm{FD}}(p_0)$ differ by approximately 4ε.

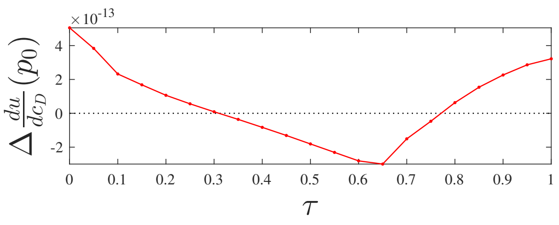
2 The switching point of the control function can be clearly identified as numerically critical point.



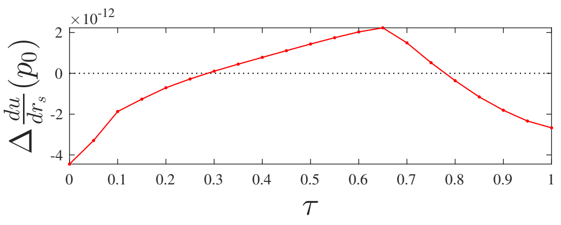
(a) Difference of the control functions



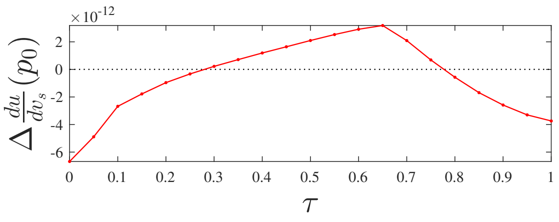
(b) Diff. control sen. w.r.t. efficiency



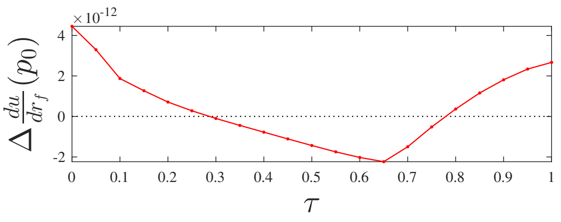
(c) Diff. control sen. w.r.t. drag



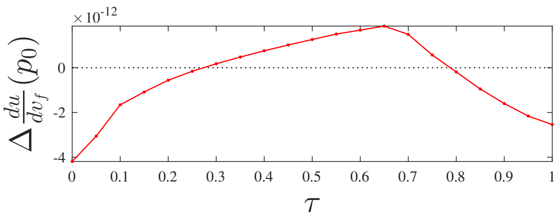
(d) Diff. control sen. w.r.t. initial position



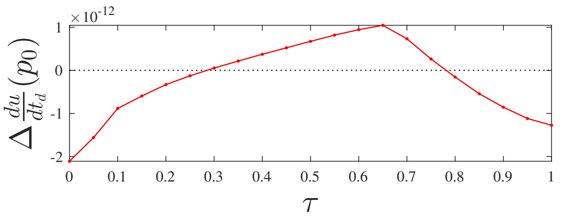
(e) Diff. control sen. w.r.t. initial velocity



(f) Diff. control sen. w.r.t. final position



(g) Diff. control sen. w.r.t. final velocity



(h) Diff. control sen. w.r.t. duration

Figure 5.5: Numeric differences between the control function and the parametric sensitivities of the control function for single shooting and full discretization of the rocket car problem $\mathrm{OCP}_{(5.19)}(p_0)$.

We now take a close look at the iteration sequences obtained by Algorithm 1 for both discretizations. Therefore we exemplary perturb the initial state by $\Delta x_s = (0.2, 0.2)^\intercal$. The optimal control function iterate is obtained directly as output of Algorithm 1; the state trajectory for single shooting is obtained by integration of the control function, while the state trajectory for full discretization is part of the optimal solution and therefore directly obtained from the iteration scheme.

Figure 5.6a and 5.6b show the first two iterates of the optimal control function. While the Taylor expansion is identical up to numerical accuracy, the structural difference between $\mathrm{RC}_{\mathrm{SS}}(p)$ and $\mathrm{RC}_{\mathrm{FD}}(p)$ becomes apparent in the first feasibility correction step. The approximation error of the Taylor expansion of the control variables

$$\left\|\overset{\star}{u}(p) - \tilde{u}^{[1]}(p)\right\| = \mathcal{O}(\|\Delta p\|^2) \tag{5.21}$$

is distributed in differently for SS and FD:

In *Single Shooting* the approximation error (5.21) for each control variable accumulates, because $\tilde{u}^{[1]}(p)$ is integrated nonstop from the initial state to the final state to obtain the terminal error, i.e. a single initial value problem is solved.

For *full discretization* the state trajectories are part of the optimal solution and the Taylor expansion contains $\tilde{x}^{[1]}_{\mathrm{FD}}(p)$ directly. The approximation error of the state variables is thus also bounded by

$$\left\|\overset{\star}{x}(p) - \tilde{x}^{[1]}_{\mathrm{FD}}(p)\right\| = \mathcal{O}(\|\Delta p\|^2). \tag{5.22}$$

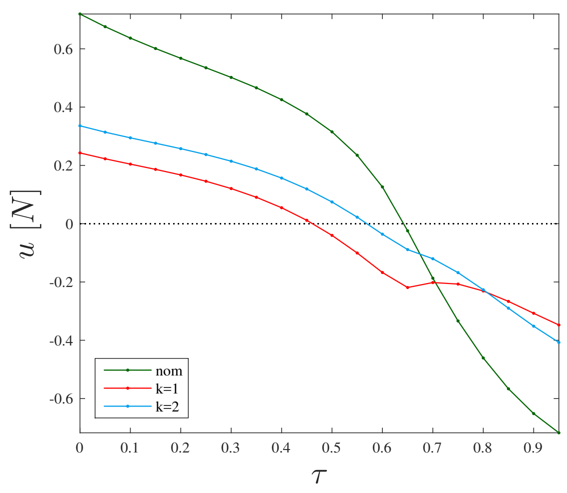
FD solves one initial value problem per control interval. Each integration step is performed based on a new initial state given by $\tilde{x}^{[1]}_{\mathrm{FD}}(p)$. FD prevents the cumulative error of the integration chain by *resetting* the error for each integration interval based on the state approximation. As a result the state trajectory for full discretization (Figures 5.6d and 5.6f) deviates muss less than the state trajectory of single shooting (Figures 5.6c and 5.6e).

Using the direct state approximation as integration start value causes discontinuities of the state trajectory, which can be clearly seen in Figure 5.6f. Note that in this example there is no continuity error in the position trajectory $\tilde{r}^{[1]}_{\mathrm{FD}}$ (Figure 5.6d) because the position (5.19b) does not depend on p directly, but only indirectly via v. The position is updated in each iteration to match the velocity trajectory without error.

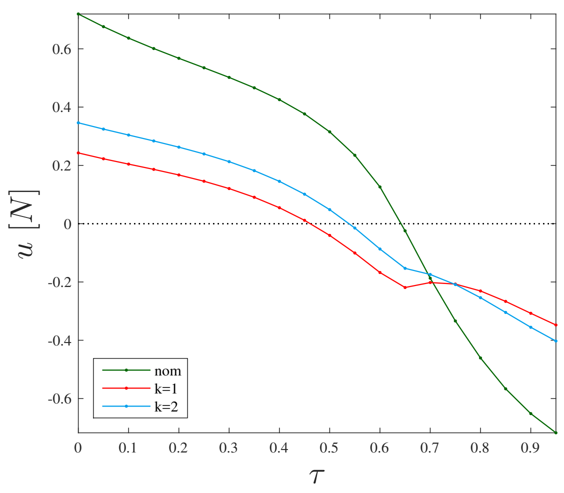
From iteration $k = 2$ onwards the control function iterates differ significantly and the iteration sequences of SS and FD converge to different fixpoints. Figure 5.7 shows the iterates for $k = 1, ..., 5$ and Figure 5.8 compares the final iterates at $k = 5$ for SS and FD with the true optimal solution.

Remark 5.23 (State Sensitivities for Single Shooting)
The parametric sensitivities of the state functions can be obtained using the chain rule, from the relations (4.77b) and (4.79) [Büs00].

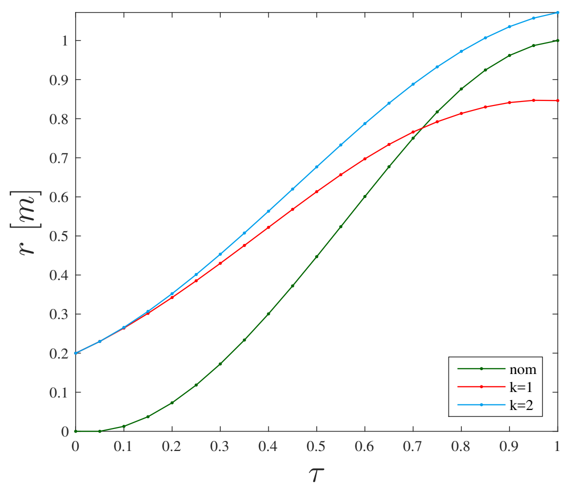
$$\frac{\mathrm{d}x(\tau^{(i)})}{\mathrm{d}p}(p_0) = \frac{\partial x(\tau^{(i)})}{\partial z}(\overset{\star}{z}_0,p_0)\left(\frac{\mathrm{d}z}{\mathrm{d}p}\right)_{\mathrm{SS}}(p_0) + \frac{\partial x(\tau^{(i)})}{\partial p}(\overset{\star}{z}_0,p_0), \quad i = 1,...,n_u.$$



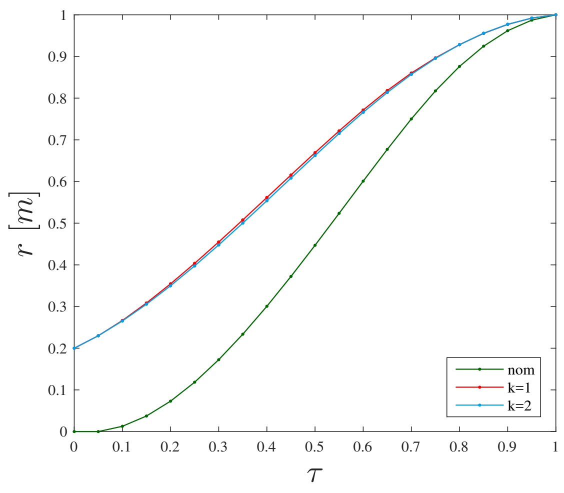
(a) u single shooting



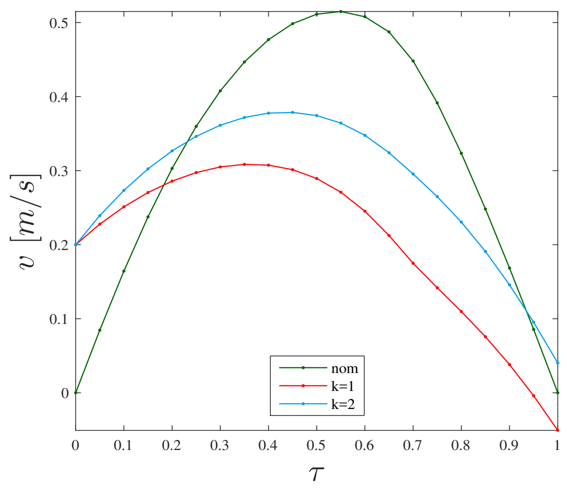
(b) u full discretization



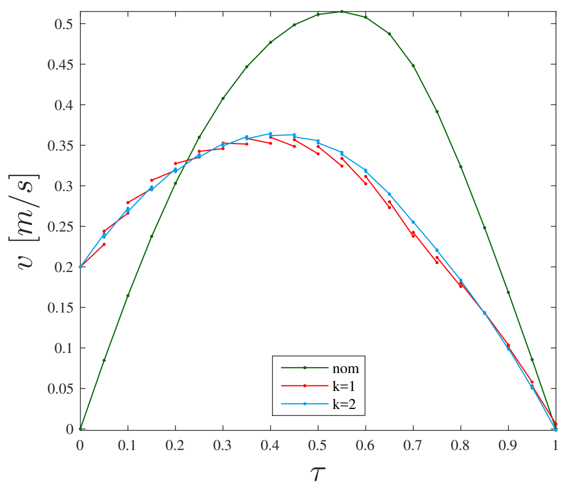
(c) r single shooting



(d) r full discretization

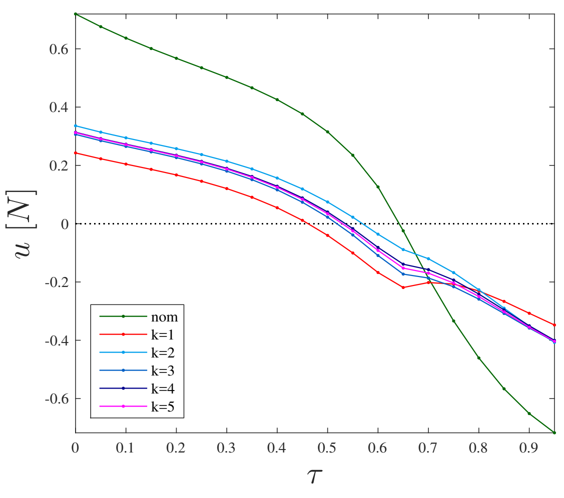


(e) v single shooting

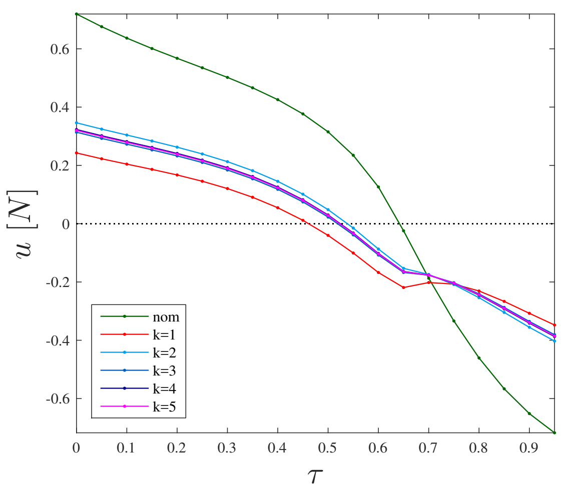


(f) v full discretization

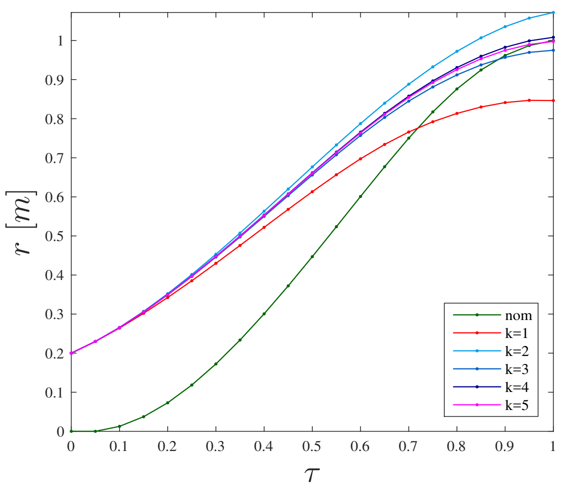
Figure 5.6: Solution approximation using algorithm 1 for problem $\text{OCP}_{(5.19)}(p)$ for a perturbed initial state $x = x_s + (0.2, 0.2)^\mathsf{T}$. Iterate $k = 1$ is the approximation after Taylor expansion, iterate $k = 2$ is the approximation after the first feasibility restoration step.



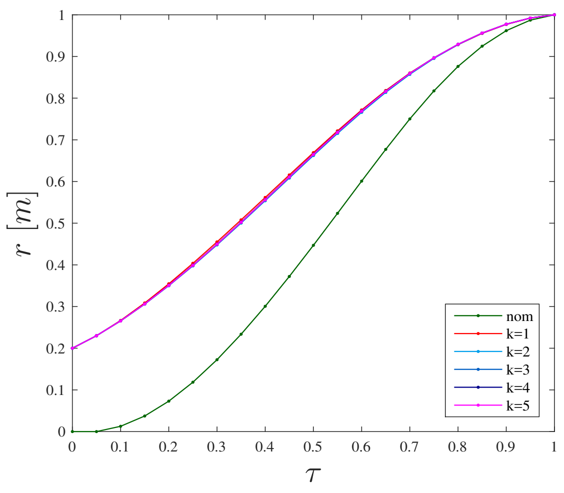
(a) u single shooting



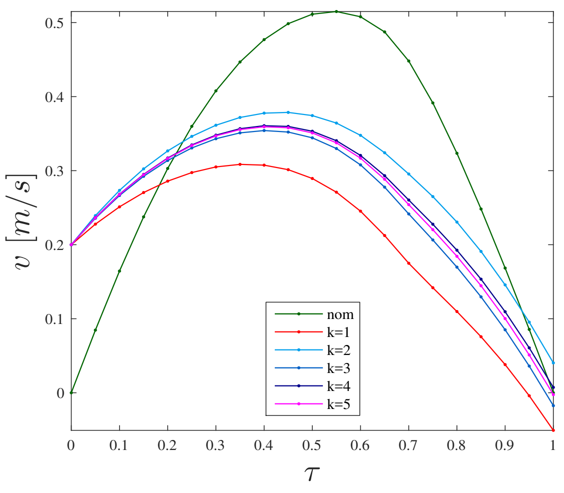
(b) u full discretization



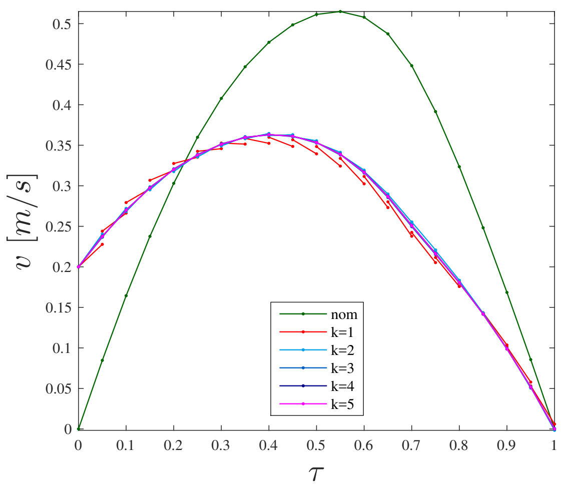
(c) r single shooting



(d) r full discretization

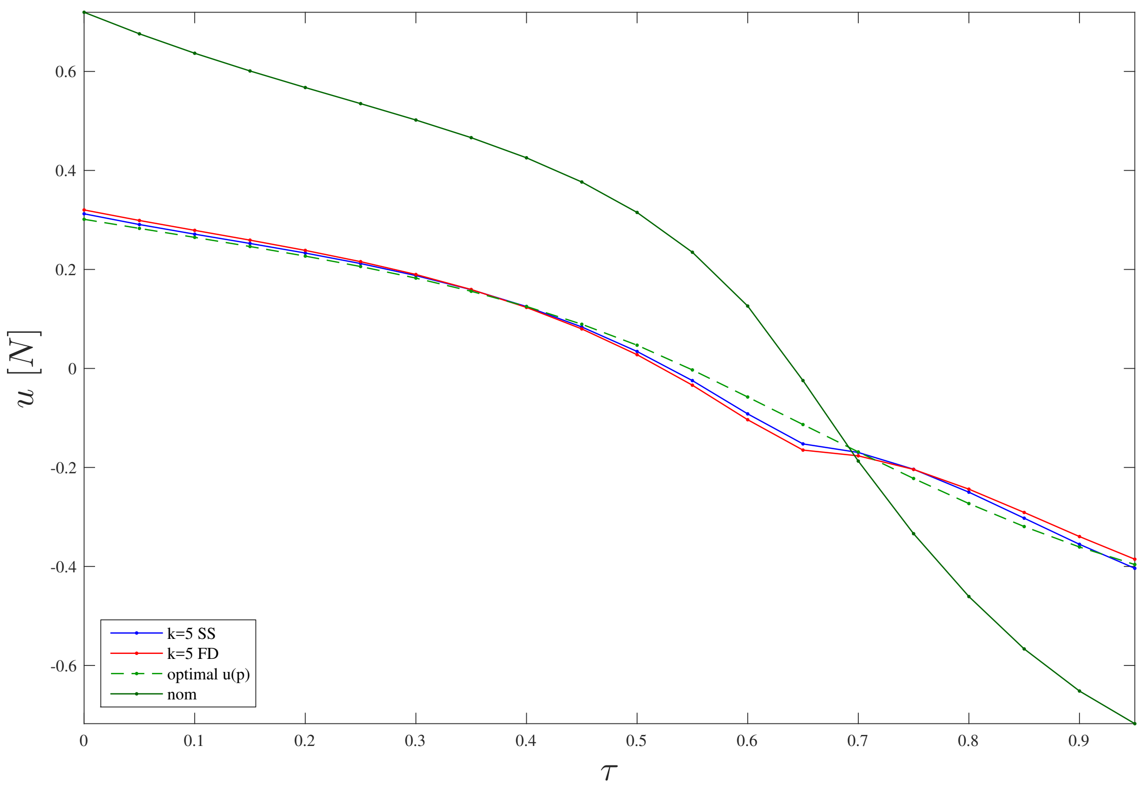


(e) v single shooting

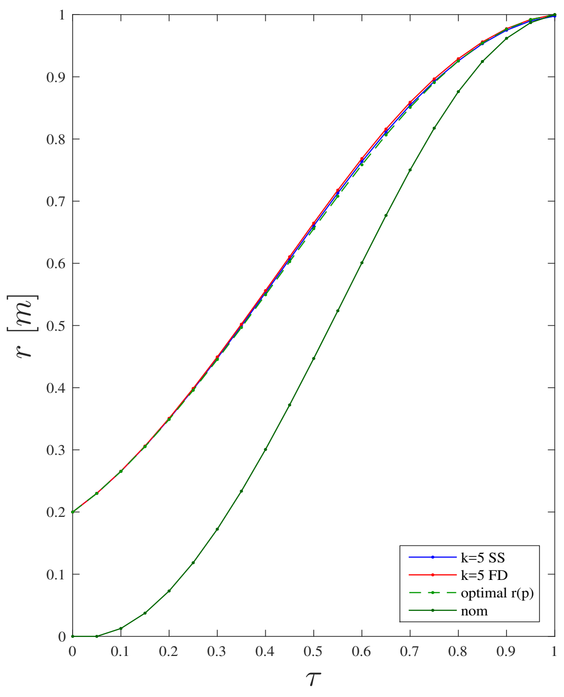


(f) v full discretization

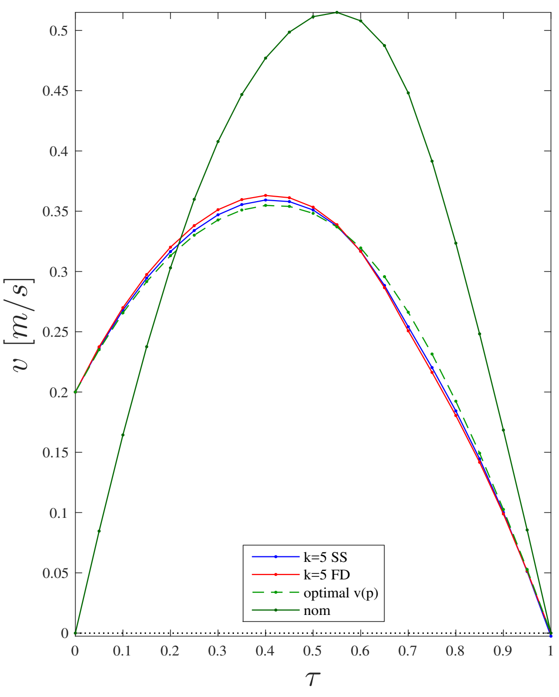
Figure 5.7: Iterations 1 to 5 of Algorithm 1 for problem $\mathrm{OCP}_{(5.19)}(p)$ for a perturbed initial state $x = x_s + (0.2, 0.2)^\intercal$.



(a) control function u



(b) position r



(c) velocity v

Figure 5.8: Comparison of the solution approximation obtained by Algorithm 1 for single shooting (blue) and full discretization (red) after 5 iterations. The true optimal solution for the disturbed parameter is shown in dashed green.

Table 5.1 shows the maximum absolute constraint error for each iteration step. The maximum absolute error is naturally smaller for FD, but the convergence rate is roughly equal. By only considering the maximal NLP constraint error, the cumulative effect of the discontinuities in the state trajectory, in case of FD, is neglected. Therefore we compare the state error resulting from the open loop application of the FD control function, as shown in column *FD (open loop)*. The comparison of the final state errors show that FD still converges faster.

Table 5.1: Iteration comparison for algorithm 1 applied to $\mathrm{OCP}_{(5.19)}(p)$

Iter. k	**SS**		**FD**		**FD (open loop)**	
	$\lvert\Delta r_f\rvert$	$\lvert\Delta v_f\rvert$	$\lVert\Delta r\rVert_2$	$\lVert\Delta v\rVert_2$	$\lvert\Delta r_f\rvert$	$\lvert\Delta v_f\rvert$
1	0.153601	0.050754	0	0.036548	0.153601	0.050754
2	0.071722	0.040331	0	0.011021	0.042097	0.008368
3	0.024853	0.017433	0	0.003086	0.011835	0.002185
4	0.008285	0.007266	0	0.000916	0.003456	0.000565
5	0.002362	0.002555	0	0.000272	0.001016	0.000153
6	0.000556	0.000809	0	0.000081	0.000301	0.000043
7	0.000080	0.000221	0	0.000009	0.000024	0.000012
8	0.000014	0.000048	0	0.000007	0.000027	0.000004

Table 5.2 shows the memory consumption of the sensitivity matrices for SS and FD, depending on the number of discretization points l_u, controls n_u, states n_x and parameters n_p. For single shooting the memory consumption grows linearly in all factors. For FD however the grows is quadratic in l_u and in l_x, because the amount of defect constraints increases if discretization points are added. Though for up to several hundred discretization points this is usually of no concern, even on embedded systems.

The largest contribution to the computational cost is to be expected from evaluating $g_a(\tilde{z}^{[k]}, p)$. The required number of operations are roughly equal for SS and FD as the same total number of integration steps has to be performed. The best way to reduce memory consumption and the required operations is obviously to reduce the amount of discretization points.

Table 5.2: Sensitivity matrix sizes

Item	**SS**	**FD**
NLP variables, n_z	$n_u l_u$	$(n_x + n_u)l_u$
Defect Constraints, n_d	n_x	$n_x(l_u - 1)$
Size $\frac{\mathrm{d}z}{\mathrm{d}p}$, $n_z \times n_p$	$n_u n_p l_u$	$(n_x + n_u)n_p l_u$
Size $\frac{\mathrm{d}z}{\mathrm{d}q_D}$, $n_z \times n_d$	$n_u n_x l_u$	$(n_x^2 + n_x + n_u)(l_u^2 - l_u)$

The conclusion we draw from this investigation is that a middle ground between the two extremes SS and FD is the most advantageous transcription choice for the RTS algorithm. The advantages of FD are the faster convergence, a potentially increased convergence region and the stationary behavior of the state trajectory during the iteration sequence. The disadvantage lies in the increased memory consumption.

Starting from single shooting, one additional state discretization point in the middle of the trajectory roughly cuts the error accumulation through the forward integration in half. But the ratio between error reduction and additional memory consumption gets worse with every additional state discretization point. The obvious strategy is hence to start with a SS transcription and to check if the required performance envelop can be covered. Otherwise the number of state discretization points can be increased, to achieve a more stationary behavior of the state trajectory, and possibly enlarge the region of convergence. Concerning the convergence speed of Algorithm 1, FD also shows a slight advantage (using the final state error obtained by the open loop application of the control function as a comparison basis).

5.5.2 Special Case: Linearly Perturbed and Linearly Constrained Quadratic Problem

We briefly consider the special case of a quadratic problem with linear constraints.

$$\min z^\intercal A z + b^\intercal z + c^\intercal p + d \tag{5.24a}$$

$$\text{s.t.} \qquad Gz + Hp + k = 0 \tag{5.24b}$$

where $A \in \mathbb{R}^{n_z} \times \mathbb{R}^{n_z}$, $G \in \mathbb{R}^{n_g} \times \mathbb{R}^{n_z}$, $H \in \mathbb{R}^{n_g} \times \mathbb{R}^{n_p}$ and b, c, d, k are vectors of appropriate size. The objective function is at most quadratic in z and linear in p, and the constraints are linear in z and p, i.e. we have a linearly perturbed, linearly constrained quadratic problem, LPLCQP(p).

We can adjust the rocket car example to match this problem type by removing the nonlinear elements by using the coefficients $e := 0$ and $c_D := 0$. The nominal solution of the now linear system is shown in Figure 5.9 in dark green.

For LPLCQP(p) the first order Taylor expansion of the nominal solution is identical to the *exact* optimal solution for a perturbed parameter. Figure 5.9 also shows a comparison between the Taylor expansion and the true optimal solution for the initial state perturbation (5.20). The difference between the optimal solution and the approximations based on single shooting and full discretization is smaller than 10^{-15}.

Corollary 5.25 (Optimal Solutions of LPLCQP(p))
Let $\overset{\star}{z}(p_0)$ be the nominal solution of LPLCQP(p_0) for the nominal parameter $p_0 \in \mathbb{R}^{n_p}$. The optimal solution $\overset{\star}{z}(p)$ for a perturbed parameter $p = p_0 + \Delta p \in \mathbb{R}^{n_p}$ is

$$\overset{\star}{z}(p) = \tilde{z}^{[1]}(p) = \overset{\star}{z}(p_0) + \frac{\mathrm{d}z}{\mathrm{d}p}(p_0)\ \Delta p.$$

Proof: For LPLCQP(p) the necessary optimality conditions $K(\overset{\star}{z}(p), \overset{\star}{\mu}_a(p), p) \equiv 0$ depend linearly on z and p (cf. (5.7)). The differentiation (5.9) thus leads to a constant expression for the sensitivity differentials $\frac{\mathrm{d}z}{\mathrm{d}p}$, and all higher order derivatives of the optimality conditions w.r.t. p are zero. The linear dependence of (5.7) on the parameter p can be described exactly by a support point, i.e. $\overset{\star}{z}(p_0)$, and a first order derivative w.r.t. p, i.e. the parametric sensitivity $\frac{\mathrm{d}z}{\mathrm{d}p}(p_0)$. Because there are no higher order terms in p and there is no dependence on z, the first order Taylor expansion (5.26) does not have a truncation error and is thus identical to the true optimal solution.

□

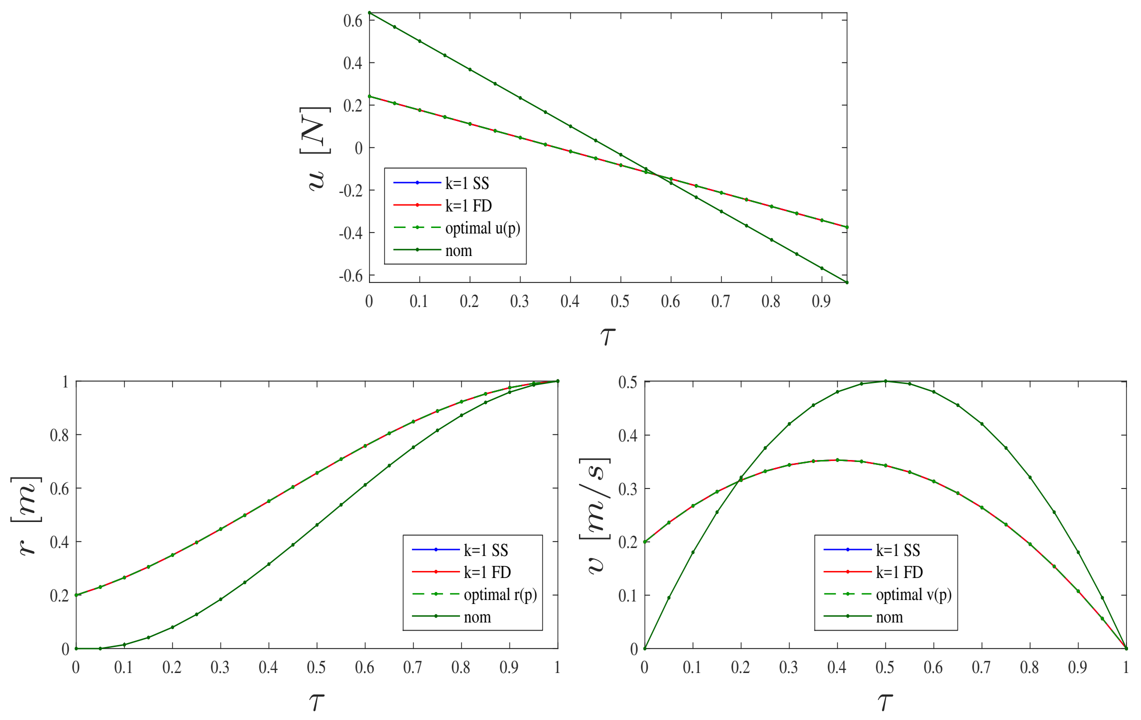


Figure 5.9: The optimal solution for the perturbed parameter (dashed light green) can be found using the nominal solution (dark green) and the first order Taylor expansion for single shooting (blue) or full discretization (red). The perturbed optimal solution and the Taylor expansions are superimposed.

5.5.3 Convergence Verification at Runtime

In case that the feasibility restoration (5.17) is divergent, Algorithm 1 must be prevented from iterating endlessly; an abort criterion is required, thus we need to determine if the algorithm is convergent for a perturbed parameter $p = p_0 + \Delta p$ at runtime.

The convergence progress at stage k can be measured by a norm on the remaining constraint error $g_a(\tilde{z}^{[k]}(p),p)$. At the fixpoint $\tilde{z}^{[\infty]}(p;p_0,p)$ it holds that

$$\left\| g_a(\tilde{z}^{[\infty]}(p;p_0,p),p) \right\| = 0. \tag{5.26}$$

The iteration sequence defined by the remaining constraint error is a contraction if and only if

$$\left\| g_a(\tilde{z}^{[k+1]}(p;p_0,p),p) \right\| < \left\| g_a(\tilde{z}^{[k]}(p;p_0,p),p) \right\|, \quad k > 0. \tag{5.27}$$

Condition (5.27) can be used as an abort criterion, however a violation of this criterion does not imply that iteration (5.17) is divergent. There are cases, in which Condition (5.27) is violated for a small number of steps, but subsequently iteration (5.17) converges. In this case the optimality error bound of Theorem (5.18) is violated, but at the fixpoint the active constraints are still satisfied and the solution is feasible.

If feasibility is more important than optimality, such a non-optimal but feasible solution might still be valuable. Under this premise Condition (5.27) can be heuristically relaxed, using a factor $r \in]0;1[$.

$$r \left\| g_a(\tilde{z}^{[k+1]}(p;p_0,p),p) \right\| < \left\| g_a(\tilde{z}^{[k]}(p;p_0,p),p) \right\| \tag{5.28}$$

If the contraction criterion is relaxed, it can be beneficial to additionally consider heuristics to identify divergent behavior. It has proven effective to monitor the defect constraints for the state equation independently of each other. Referring to (E.4), the defects $d^{(i,j)} \subset g_a$, $1 \le i \le l_x$, $1 \le j \le n_x$ are monitored. A norm of the defect error is computed separately for each state equation:

$$D^{[k]} = \begin{pmatrix} d^{(1)} \\ \vdots \\ d^{(n_x)} \end{pmatrix} = \begin{pmatrix} \|(d^{(1,1)}, \ldots, d^{(l_x,1)})^\intercal\| \\ \vdots \\ \|(d^{(1,n_x)}, \ldots, d^{(l_x,n_x)})^\intercal\| \end{pmatrix} \tag{5.29}$$

In an optimistic manner iteration (5.17) is continued if at least one component of $D^{[k+1]}$ is smaller than in $D^{[k]}$. This heuristic is justified because the relative importance between different constraint errors, i.e. the *skew* of the optimization problem, is unknown. The proposed heuristic assumes that errors in the same state equation can be compared amongst each other, but the comparison between different states is eliminated. If the norm of the error increases simultaneously for all state equations, the divergence of (5.17) is very likely, such that the iteration can be stopped.

In conclusion the following abort criteria are checked in each iteration:

1. The maximal number of iterations is exceeded.
2. The active set changed.
3. The relaxed contraction criterion is violated.
4. The norm of the defect errors increased in all state equations.

5.5.4 Limitations

We want to illustrate the dependence of parametric sensitivity differentials on the active set and the consequences for the use of the real-time solution approximation scheme as a control strategy. A closely related topic is the estimation of the region of convergence of Algorithm (1).

Let $\mathcal{U}(p_0)$ be the neighborhood in which Theorem 5.18 holds. In the context of stability, we would like to ascertain a priori that a perturbed parameter $p = p_0 + \Delta p$ is in $\mathcal{U}(p_0)$. In other words we would like to prove that a ball around p_0 with radius $r > 0$ is contained within $\mathcal{U}(p_0)$. Such a proof remains an open problem. In the following we illustrate the related difficulties by considering the conditions that limit $\mathcal{U}(p_0)$:

1. The active set $a(\overset{\star}{z}_0,p_0)$ must remain constant.
2. The iteration sequence (5.17) must converge.

Constancy of the Active Set

The linearization of the KKT-conditions implicitly assumes that the active set of the nominal and the perturbed optimal solutions are identical. The active set can be imagined to limit the directions in which the optimal solution can shift as a result of perturbations. If the active set changes, the parametric sensitivities would change too, to adopt to the gained or lost degrees of freedom.

Obviously only inequality constraints can cause a change of the active set. We differentiate the following cases:

1. An inactive constraint $g^{(m)}(\overset{\star}{z}_0,p_0) < 0$, $m \notin a(\overset{\star}{z}_0,p_0)$ becomes active.
2. An active constraint $g^{(m)}(\overset{\star}{z}_0,p_0) = 0$, $m \in a(\overset{\star}{z}_0,p_0)$ becomes inactive.

Note that *both* cases compromise the update strategy:

The first case occurs when the solution iterate $\tilde{z}^{[k]}$ is moved towards and across the boundary of an inactive inequality constraint, i.e. the iterate $\tilde{z}^{[k+1]}$ becomes infeasible.

The second case can only occur, when an inequality constraint is active at the nominal solution. An active inequality constraint becomes inactive, when the solution iterate moves away from the boundary. The additional degrees of freedom gained from a constraint becoming inactive are not taken into account in the next iteration, which results into a non optimal update.

Remark 5.30 (Activation of Pure Control Constraints)
For the special case that a pure control constraint becomes active, a heuristic can be used to possibly retain the feasibility of the approximation: The newly violating control variable is forced to the boundary, and the sensitivity of this control variable with respect to all perturbations is set to zero. In subsequent iterations the affected control variable remains on the boundary. The error introduced by the fixation can possibly be mitigated through the feasibility correction using other variables, which may still be adjusted. This method can prevent the solution approximation from becoming infeasible, but it compromises its optimality.

Remark 5.31 (Deactivation of Inequality Constraints)
If a constraint becomes inactive, the parametric sensitivities, i.e. the correction direction, needs to be updated to take into account the new degree of freedom and retain optimality of the correction direction. If however only a feasible solution is desired, the feasibility restoration can be continued using the old, no-longer optimal correction direction.

The neighborhood in which the active set remains constant can be approximated as follows: Let $p_0 = (p_0^{(1)}, \dots p_0^{(n_p)})^\intercal$ and consider a perturbation in a single parameter

$$\Delta p^{(j)} = p^{(j)} - p_0^{(j)}, \quad j \in \{1,\dots,n_p\}. \tag{5.32}$$

The point at which $\Delta p^{(j)}$ causes the constraint $g^{(m)}(z(p),p)$, $m \notin a(\overset{\star}{z}_0,p_0)$ to become active can be approximated as

$$g^{(m)}(z(p),p) = 0 \approx g^{(m)}(\overset{\star}{z}_0,p_0) + \frac{\mathrm{d}g^{(m)}}{\mathrm{d}p^{(j)}}(\overset{\star}{z}_0,p_0)(p^{(j)} - p_0^{(j)}). \tag{5.33a}$$

Let $\tilde{p}^{(m,j)}$ denote the approximative value of $p^{(j)}$ which causes the constraint $g^{(m)}$ to become active. $\tilde{p}^{(m,j)}$ can be obtained by solving (5.33a) for $p^{(j)}$.

$$\tilde{p}^{(m,j)} \approx p_0^{(j)} - \frac{g^{(m)}(\overset{\star}{z}_0,p_0)}{\frac{\mathrm{d}g^{(m)}}{\mathrm{d}p^{(j)}}(\overset{\star}{z}_0,p_0)}, \quad m \notin a(\overset{\star}{z}_0,p_0),\ j \in \{1,\dots,n_p\}. \tag{5.33b}$$

If $p^{(j)}$ does affect $g^{(m)}$, that implies that $\frac{\mathrm{d}g^{(m)}}{\mathrm{d}p^{(j)}}(\overset{\star}{z}_0,p_0) \neq 0$.

The interval in which constraint $g^{(m)}$ remains inactive for perturbation $\Delta p^{(j)}$ is

$$I^{(m,j)} \approx \begin{cases} [-\infty;\ +\infty] & \text{if } \frac{\mathrm{d}g^{(m)}}{\mathrm{d}p^{(j)}}(\overset{\star}{z}_0,p_0) = 0, \\ [-\infty;\ \tilde{p}^{(m,j)}] & \text{if } \Delta p^{(j)} > 0, \\ [\tilde{p}^{(m,j)}; +\infty] & \text{if } \Delta p^{(j)} < 0, \end{cases} \qquad m \notin a(\overset{\star}{z}_0,p_0),\ j \in \{1,...,n_p\}. \tag{5.34}$$

Similarly a perturbation that causes an active constraint $g^{(m)}(\overset{\star}{z}_0,p_0) = 0$, $i \in a(\overset{\star}{z}_0,p_0)$ to become inactive can be approximated using the parametric sensitivity of the associated Lagrange multiplier $\mu_a^{(m)}(p_0) > 0$. If $g^{(m)}(\overset{\star}{z}_0,p_0)$ is about to move away from the boundary, the Lagrange multiplier $\mu_a^{(m)}$ tends to zero.

$$0 \approx \mu_a^{(m)}(\overset{\star}{z}_0,p_0) + \frac{\mathrm{d}\mu_a^{(m)}}{\mathrm{d}p^{(j)}}(\overset{\star}{z}_0,p_0)(p^{(j)} - p_0^{(j)}), \quad m \in a(\overset{\star}{z}_0,p_0),\ j \in \{1,...,n_p\} \tag{5.35a}$$

$$\tilde{p}^{(m,j)} \approx p_0^{(j)} - \frac{\mu_a^{(m)}(\overset{\star}{z}_0,p_0)}{\frac{\mathrm{d}\mu_a^{(m)}}{\mathrm{d}p^{(j)}}(\overset{\star}{z}_0,p_0)}. \tag{5.35b}$$

If $p^{(j)}$ does affect $g^{(m)}$ and $\mu_a^{(m)}$, that implies that $\frac{\mathrm{d}\mu_a^{(m)}}{\mathrm{d}p^{(j)}}(\overset{\star}{z}_0,p_0) \neq 0$.

The interval in which constraint $g^{(m)}$ remains active for perturbation $\Delta p^{(j)}$ is

$$I^{(m,j)} \approx \begin{cases} [-\infty;\ +\infty] & \text{if } \frac{\mathrm{d}\mu^{(m)}}{\mathrm{d}p^{(j)}}(\overset{\star}{z}_0,p_0) = 0, \\ [-\infty;\ \tilde{p}^{(m,j)}] & \text{if } \Delta p^{(j)} > 0, \\ [\tilde{p}^{(m,j)}; +\infty] & \text{if } \Delta p^{(j)} < 0, \end{cases} \qquad m \in a(\overset{\star}{z}_0,p_0), \quad j \in \{1,...,n_p\} \tag{5.36}$$

Let $I^{(m,j)} = \left[\tilde{p}_-^{(m,j)};\ \tilde{p}_+^{(m,j)}\right]$. The interval around $p_0^{(j)}$ in which the active set remains unchanged is

$$A^{(j)}(p_0) = \left[\max_m \tilde{p}_-^{(m,j)};\ \min_m \tilde{p}_+^{(m,j)}\right], \quad m \in \{1,...,n_g\},\ j \in \{1,...,n_p\} \tag{5.37}$$

A linear approximation of the neighborhood $\mathcal{A}(p_0)$ in which the active set remains unchanged for a perturbation $p \in \mathbb{R}^{n_p}$ is

$$\mathcal{A}(p_0) \approx A^{(1)}(p_0) \times ... \times A^{(n_p)}(p_0). \tag{5.38}$$

Remark 5.39 (Maximal Parameter Deviation With Constant Active Set)
For later purposes we note, that the maximal absolute deviation that $p^{(j)}$ may have from $p_0^{(j)}$ before set active set changes is

$$\underset{\max}{\Delta} p^{(j)} \approx \min\ (\min_m |\tilde{p}_-^{(m,j)} - p_0^{(j)}|,\ \min_m |\tilde{p}_+^{(m,j)} - p_0^{(j)}|), \quad m \in \{1,...,n_g\},\ j \in \{1,...,n_p\}.$$

Active (or almost active) inequality constraints in the nominal solution should be avoided to achieve a large correction space. Consider the active boundary arc $[b;\ c;\ d]$ in Figure 5.10. If a perturbation *pushes* knot f in direction of the boundary, a change of the active set occurs after the margin Δf. A denser discretization (i.e. adding knot e) reduces the margin and aggravates the problem.

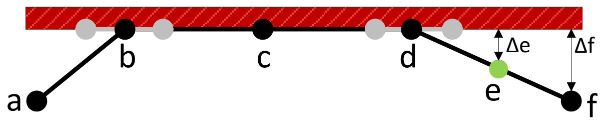


Figure 5.10: Schematic of a constrained arc (b to d) and the neighboring discretization points.

A problem transformation can be used to mitigate this problem: The problem is split into three phases: $[a, b]$, $[b, c, d]$ and $[d, f]$. Each phase has its own duration, such that the knots b and d can shift to the optimal entry and exit points of the boundary arc. Linkage constraints are used to enforce continuity of the trajectory at the phase transitions, similar to multiple shooting continuity constraints. In the transformed formulation perturbations effect the entry and exit points, i.e. there are additional degrees of freedom, which can lead to an increased margin before the active set change occurs. However there are also drawbacks to this approach: The afloat sequence of multiple phases can compromise the regularity of the problem formulation, and require a very good optimization start value to converge. In addition the implementation effort to (automatically) support this transformation for different types of constraints is considerable. Problems with multiple constrained arcs or a ragged control switching structure are thus not handled well by this method.

Another difficult example are min-max or max-min problems, which are solved through the optimization of a slack variable. Consider the following example, with $x_1, x_2 \in \mathbb{R}$.

$$\begin{aligned} &\min \max \{x_1, x_2\} \\ &\text{s.t.} \qquad x_1 + x_2 - 2 = 0 \end{aligned}$$

By introducing the slack variable $z \in \mathbb{R}$ the problem can be reformulated as

$$\begin{aligned} &\min z \\ &\text{s.t. } x_1 + x_2 - 2 = 0 \\ &\qquad x_1 - z \qquad \leq 0 \\ &\qquad x_2 - z \qquad \leq 0. \end{aligned}$$

At the optimal solution $z = x_1 = x_2 = 1$, all slack constraints are active. The optimization of slack generally leads to at least one, and often multiple, active inequality constraint. Hence slack variables should be avoided when approximating optimal solutions using parametric sensitivities.

Convergence of the Feasibility Restoration Iteration

Büskens [Büs02] shows that Iteration (5.17) is not a contraction on the space of optimization variables $\mathbb{R}^{n_z}$. But the optimization variables of each iterate $\tilde{z}^{[k]} \in \mathbb{R}^{n_z}$ can be rearranged, such that there exists $v^{[k]}, w^{[k]}$ with

$$\tilde{z}^{[k]}(p) = \begin{pmatrix} v^{[k]} \\ w^{[k]} \end{pmatrix} = \begin{pmatrix} v^{[k]} \\ w^{[k]}(v^{[k]}) \end{pmatrix}, \tag{5.40}$$

where $w^{[k]}$ is an affine linear transformation. This allows a partition of the iteration and Büskens shows that the partitioned iteration w.r.t. $v^{[k]}$ is a contraction on the subspace of $v^{[k]}$ if Δq is sufficiently small. It is however not possible to determine the partition (5.40) or the affine transformation $w^{[k]}$ explicitly. The neighborhood around q_0 in which the RTS algorithm converges can only be determined numerically. In consequence the stability of a system that is controlled by a control law based on the RTS algorithm cannot be proven analytically. The controllable space can be estimated using numerical, stochastic methods, e.g. Monte Carlo simulation.

5.6 Closed loop Near-Optimal Feedback in the Neighborhood of a Nominal Trajectory

In closed loop the control is determined as a function of the current state. The time measured from the start of the closed loop process is called closed loop time $\boldsymbol{t}$. The state trajectory of the closed loop system as seen from the outside, i.e. including perturbations, is denoted $\boldsymbol{x}(\boldsymbol{t})$. The goal of this section is to synthesize a local feedback law Θ, that computes a near-optimal control $\tilde{u}(\boldsymbol{t})$ depending on the current state $\boldsymbol{x}(\boldsymbol{t})$ and the current parameters $\bar{p}(\boldsymbol{t})$.

$$\overset{\star}{u}(\boldsymbol{t}) \approx \tilde{u}(\boldsymbol{t}) = \Theta(\boldsymbol{x}(\boldsymbol{t}), \bar{p}(\boldsymbol{t})), \quad \boldsymbol{t} \in [0;\, \boldsymbol{t}_d] \tag{5.41}$$

The loop is closed, if the feedback control affects the state. This is in contrast to the open loop problem $\mathrm{OCP}_{(5.1)}(\bar{p})$ in which the control function for the entire process is determined ahead of time.

A closed loop is said to be *ideal*, if the feedback control affects the state without delay. We consider the ideal closed loop optimal feedback problem at discrete times $\boldsymbol{t}^{(i)}$ with

$$T_{cl} := \{\boldsymbol{t}^{(1)}, ..., \boldsymbol{t}^{(i)}, ..., \boldsymbol{t}^{(l_T)}\}, \quad \boldsymbol{t}^{(1)} = 0, \quad \boldsymbol{t}^{(i)} < \boldsymbol{t}^{(i+1)}, \quad \boldsymbol{t}^{(l_T)} < \boldsymbol{t}_d. \tag{5.42}$$

The determination of the instantaneous closed loop optimal control $\overset{\star}{u}(\boldsymbol{t}^{(i)})$ requires to solve an open loop OCP at each $\boldsymbol{t}^{(i)} \in T_{cl}$. Therefore we distinguish between the closed loop time $\boldsymbol{t} \in [0;\, \boldsymbol{t}_d]$ and the problem specific open loop time $t_i \in [0;\, t_{i,go}]$ for the subproblem on the interval $[\boldsymbol{t}^{(i)};\, \boldsymbol{t}_d]$. The following relations hold between closed loop time and problem specific open loop time:

$$t_i(\boldsymbol{t}^{(i)}) := 0 \tag{5.43a}$$

$$t_{i,go}(\boldsymbol{t}^{(i)}) := \boldsymbol{t}_d - \boldsymbol{t}^{(i)} \tag{5.43b}$$

The open loop time $t_i \in [0;\, t_{i,go}]$ is normalized to $\tau_i \in [0;\, 1]$, because the remaining time-to-go $t_{i,go}$ is *free*.

$$\tau_i := \frac{t_i}{t_{i,go}}, \quad \tau_i \in [0;\, 1] \tag{5.44}$$

The OCP to be solved at $\boldsymbol{t}^{(i)}$ is

$$\begin{aligned}
\min_{u,t_{go}} J(x(\tau_i),u(\tau_i),t_{i,go},\bar{p}(\tau_i)) &= m(x(1),t_{i,go},\bar{p}(\tau_i)) + t_{i,go}\int_0^1 l(x(\tau_i),u(\tau_i),\bar{p}(\tau_i))\,\mathrm{d}\tau_i && (5.45a)\\
\text{s.t.} \qquad \dot{x} &= t_{i,go} f(x(\tau_i),u(\tau_i),\bar{p}(\tau_i)) && (5.45b)\\
x(0) &= \boldsymbol{x}(\boldsymbol{t}^{(i)}) && (5.45c)\\
\bar{p}(0) &= \bar{p}(\boldsymbol{t}^{(i)}) && (5.45d)\\
\psi_f(x(1),\bar{p}(\tau_i)) &= 0 && (5.45e)\\
c(x(\tau_i),u(\tau_i),\bar{p}(\tau_i)) &\leq 0. && (5.45f)
\end{aligned}$$

In contrast to the open loop view we now consider that the parameter may change during the process, i.e. $\bar{p}(\boldsymbol{t})$ depends on time. We assume that $\boldsymbol{x}(\boldsymbol{t}^{(i)})$ and $\bar{p}(\boldsymbol{t}^{(i)})$ are known at $\boldsymbol{t}^{(i)}$, but not in advance.

Note that the initial condition of $\text{OCP}_{(5.45)}$ is different from the initial condition of $\text{OCP}_{(5.1)}$. In the open loop problem the initial state may be free, whereas in closed loop the initial state is determined by the closed loop state. Analogously the initial condition of $\bar{p}$ is determined by $\bar{p}(\boldsymbol{t}^{(i)})$.

We consider the closed loop state as a parameter and denote $\text{OCP}_{(5.45)}$ at $\boldsymbol{t}^{(i)}$ as $\text{OCP}_{(5.45)}(p^{(i)})$ with the joined parameter vector

$$p^{(i)} = p(\boldsymbol{t}^{(i)}) := \begin{pmatrix} \bar{p}(\boldsymbol{t}^{(i)}) \\ \boldsymbol{x}(\boldsymbol{t}^{(i)}) \end{pmatrix}, \quad \boldsymbol{t}^{(i)} \in T_{cl}. \tag{5.46}$$

If we consider any perturbations on the system, $p(\boldsymbol{t})$ is unknown for $\boldsymbol{t} > \boldsymbol{t}^{(i)}$ at time $\boldsymbol{t}^{(i)}$. But for the special case that there are no perturbations during closed loop execution, the closed loop trajectory matches the optimal open loop trajectory and the closed loop parameter matches the nominal parameter. To obtain the nominal trajectory we solve the open loop problem $\text{OCP}_{(5.1)}(\bar{p}_0)$ for the nominal parameter value $\bar{p}_0$. Let $\overset{\star}{x}(t;\bar{p}_0)$, $t \in [0;\overset{\star}{t}_d]$ be the optimal open loop trajectory. If there are no perturbations in the closed loop system, we can specify the closed loop trajectory $\boldsymbol{x}(\boldsymbol{t})$ ahead of time:

$$\boldsymbol{x}(\boldsymbol{t},\bar{p}(\boldsymbol{t})) = \overset{\star}{x}(t;\bar{p}_0), \quad \boldsymbol{t} \in [0;\boldsymbol{t}_d], \quad \boldsymbol{t}_d = \overset{\star}{t}_d. \tag{5.47}$$

Likewise we can specify the parameter ahead of time:

$$\bar{p}(\boldsymbol{t}) = \bar{p}_0, \quad \boldsymbol{t} \in [0;\boldsymbol{t}_d]. \tag{5.48}$$

The control problems that arise in unperturbed closed loop are thus

$$\begin{aligned}
\min_{u,t_{i,go}} J(x(\tau_i),u(\tau_i),t_{i,go},\bar{p}_0) &= m(x(1),t_{i,go},\bar{p}_0) + t_{i,go}\int_0^1 l(x(\tau_i),u(\tau_i),\bar{p}_0)\,\mathrm{d}\tau_i && (5.49a)\\
\text{s.t.} \qquad \dot{x} &= t_{i,go} f(x(\tau_i),u(\tau_i),\bar{p}_0) && (5.49b)\\
x(0) &= \overset{\star}{x}(\boldsymbol{t}^{(i)};\bar{p}_0) && (5.49c)\\
\bar{p}(0) &= \bar{p}_0 && (5.49d)\\
\psi_f(x(1),\bar{p}_0) &= 0 && (5.49e)\\
c(x(\tau_i),u(\tau_i),\bar{p}_0) &\leq 0. && (5.49f)
\end{aligned}$$

The nominal parameters for these control problems are

$$p_0^{(i)} = p_0(\boldsymbol{t}^{(i)}) := \begin{pmatrix} \bar{p}_0 \\ \overset{\star}{x}(\boldsymbol{t}^{(i)}; \bar{p}_0) \end{pmatrix}, \quad \boldsymbol{t}^{(i)} \in T_{cl}. \tag{5.50}$$

Note that in the nominal problems $\text{OCP}_{(5.49)}(p_0^{(i)})$ the parameter $\bar{p}_0$ is constant, but the initial state still depends on time.

The proposed feedback law is based on the parametric sensitivity analysis of the problems $\text{OCP}_{(5.49)}(p_0(\boldsymbol{t}^{(i)}))$. In an offline phase, before closed loop, we obtain the parametric sensitivities of $\text{OCP}_{(5.49)}(p_0(\boldsymbol{t}^{(i)}))$, for all $\boldsymbol{t}^{(i)} \in T_{cl}$. In the online phase we use the precomputed sensitivities to execute the RTS algorithm to approximate a solution of the control problem for the disturbed closed loop state $\boldsymbol{x}$ and the disturbed parameter $\bar{p}$. In the following the strategy is outlined, details are discussed in the following sections.

Offline Preparation Phase

1. Solve $\text{OCP}_{(5.1)}(\bar{p}_0)$ open loop via direct optimization. Let $\overset{\star}{t}_d$ be the optimal open loop process duration. Denote the normalized time with respect to the full length process as

$$\boldsymbol{\tau} := \frac{t}{\overset{\star}{t}_d}, \quad t \in [0;\, \overset{\star}{t}_d]. \tag{5.51}$$

   Furthermore let $\overset{\star}{x}(\boldsymbol{\tau}; \bar{p}_0)$, $\boldsymbol{\tau} \in [0;\, 1]$ be the nominal open loop trajectory of $\text{OCP}_{(5.1)}(\bar{p}_0)$.
2. Define the nominal closed loop system $\text{OCP}_{(5.49)}(p_0(\boldsymbol{\tau}))$ having the nominal parameter

$$p(\boldsymbol{\tau})_0 := \begin{pmatrix} \bar{p}_0 \\ \overset{\star}{x}(\boldsymbol{\tau}; \bar{p}_0) \end{pmatrix}, \quad \boldsymbol{\tau} \in [0;\, 1]. \tag{5.52}$$

3. Choose a grid

$$T_{nom} := \{\boldsymbol{\tau}^{(1)}, ..., \boldsymbol{\tau}^{(i)}, ..., \boldsymbol{\tau}^{(l_p)}\}, \quad \boldsymbol{\tau}^{(1)} = 0, \quad \boldsymbol{\tau}^{(i)} < \boldsymbol{\tau}^{(i+1)}, \quad \boldsymbol{\tau}^{(l_p)} < 1, \tag{5.52a}$$

   and obtain the set

$$P_{nom} = \{p_0(\boldsymbol{\tau}^{(1)}), ..., p_0(\boldsymbol{\tau}^{(l_p)})\} = \{p_0^{(1)}, ..., p_0^{(l_p)}\}. \tag{5.52b}$$

4. For each $p_0^{(i)} \in P_{nom}$
   a) $\text{OCP}_{(5.49)}(p_0^{(i)})$ has the initial condition

$$x(0) = \overset{\star}{x}(\boldsymbol{\tau}^{(i)}, \bar{p}_0). \tag{5.52c}$$

$$\bar{p}(0) = \bar{p}_0 \tag{5.53}$$

      Note that the optimal solution of $\text{OCP}_{(5.49)}(p_0^{(1)})$ is identical to the optimal solution of $\text{OCP}_{(5.1)}(\bar{p}_0)$.
   b) Transcribe $\text{OCP}_{(5.49)}(p_0^{(i)})$ into a nonlinear program $\text{NLP}(p_0^{(i)})$.
   c) Consider parameters $q \in \mathbb{R}^{n_g}$, with $q_0 = 0$, that enter the constraint function $g(z,p)$ linearly. Let the augmented problem be $\text{NLP}(p_0^{(i)}, q_0)$.

d) Solve NLP($p_0^{(i)}$,q_0) to obtain the nominal solution

$$\overset{\star}{z}(p_0^{(i)},q_0). \tag{5.53a}$$

e) Obtain the parametric sensitivity of $\overset{\star}{z}(p_0^{(i)},q_0)$ with respect to $p_0^{(i)}$ and q_0.

$$\frac{\mathrm{d}z}{\mathrm{d}p}(p_0^{(i)},q_0), \quad \frac{\mathrm{d}z}{\mathrm{d}q}(p_0^{(i)},q_0). \tag{5.53b}$$

Online Feedback Phase

1. At closed loop time $\boldsymbol{t}$ receive the input: $p = \begin{pmatrix} \bar{p} \\ \boldsymbol{x} \end{pmatrix}$.
2. Set

$$p_0^{(r)} = \arg\min_{p_0^{(i)}} \left\| p - p_0^{(i)} \right\|, \quad p_0^{(i)} \in P_{nom}, \tag{5.53c}$$

   where $\|\cdot\|$ is a norm on $\mathbb{R}^{n_p}$.
3. Set

$$\Delta p = p - p^{(r)} \tag{5.53d}$$

4. Execute the RTS algorithm with the arguments

$$\Delta p, \quad \overset{\star}{z}(p^{(r)},q_0), \quad \frac{\mathrm{d}z}{\mathrm{d}p}(p^{(r)},q_0), \quad \frac{\mathrm{d}z}{\mathrm{d}q}(p^{(r)},q_0). \tag{5.53e}$$

   and obtain the adapted solution $\tilde{z}(p)$.
5. Extract the control sequence $\tilde{u}(p)$ from $\tilde{z}(p)$ and use $\tilde{u}$ as control input. Goto 1.

5.6.1 An Advantageous Norm

In the online phase the control law receives the perturbed parameter $p \in \mathbb{R}^{n_p}$ as input. To apply the RTS algorithm we must determine a reference point $p_0^{(r)} \in P_{nom}$ that defines the perturbation direction $\Delta p = p - p_0^{(r)}$.

Let the RTS algorithm converge on the neighborhoods $N = \{\mathcal{N}^{(1)}, ..., \mathcal{N}^{(l_p)}\}$. Then $p_0^{(r)}$ must be chosen such that $p \in \mathcal{N}^{(r)} \in N$. As a secondary goal $p_0^{(r)}$ should be as close to p as possible, because the solution approximation error increases with $\|\Delta p\|$. This leads to the constrained optimization problem

$$p_0^{(r)} = \arg\min_{p_0^{(i)} \in P_{nom}} \left\| p - p_0^{(i)} \right\| \tag{5.54a}$$

$$\text{s.t. } p \in \mathcal{N}^{(r)},\ \mathcal{N}^{(r)} \in N. \tag{5.54b}$$

Problem (5.54) only has a solution for

$$p \in \mathfrak{N} = \bigcup_{i}^{l_p} \mathcal{N}^{(i)} \subset \mathbb{R}^{n_p}. \tag{5.55}$$

and is hard to solve efficiently, because the only known method to evaluate the constraint (5.54b) is to apply the RTS algorithm at $p_0^{(r)}$.

Problem (5.54) can be solved by applying the RTS algorithm for all $p_0^{(i)} \in P_{nom}$ and selecting the convergent solution with minimal $\|\Delta p^{(i)}\|$. The space that is controllable by $\Theta(p)$ is thus

$$\mathcal{C}(\Theta(p)) = \mathfrak{N}. \tag{5.56}$$

This brute force approach however is computationally infeasible for real-time feedback.

To achieve a sufficient reduction of the required computations the reference point must be selected without evaluating (5.54b). An unconstrained selection of $p_0^{(r)}$ however results in a reduction of the controllable space, because it is possible that $p_0^{(r)}$, $p \notin \mathcal{N}^{(r)}$, although $p \in \mathcal{N}^{(j)}$, $j \neq r$.

One approach to select $p_0^{(r)}$ is to use an independent variable, e.g. the closed loop time $\boldsymbol{t}$. Under the assumptions that the temporal displacement between p and p_0 is low, and that the closed loop duration is similar to $\overset{\star}{t}_d$, the reference index can be defined as

$$r := \arg\min_i \left| \boldsymbol{\tau}^{(i)} - \frac{\boldsymbol{t}}{\overset{\star}{t}_d} \right|, \quad \boldsymbol{\tau}^{(i)} \in T_{nom}. \tag{5.57}$$

The feedback law then also depends on time, i.e. $\Theta(p,\boldsymbol{t})$ and the controllable space is a subset of $\mathfrak{N}$, determined by the accuracy of the time collocation between the closed loop and the open loop solution.

We suggest the following alternative approach: If hypothetically we could find a norm such that

$$p_0^{(r)} = \arg \min_{p_0^{(i)} \in P_{nom}} \left\| p - p_0^{(i)} \right\| \quad \Longrightarrow \quad p \in \mathcal{N}^{(r)}, \quad p \in \mathfrak{N} \tag{5.58}$$

the constraint (5.54b) would automatically be satisfied. Unfortunately for non convex $\mathcal{N}^{(i)}$ such a norm does not exist, but it is possible to find an approximation.

If the parameters enter the nominal optimal solution nonlinearly, or if the parameters are of different scale, $\mathcal{N}^{(1)}, ..., \mathcal{N}^{(l_p)}$ are skewed in standard Euclidean space. This is illustrated in Figure 5.11 in a two dimensional example. Consequentially a ball resulting from the standard Euclidean norm (blue) is not a good convex approximation of the skewed convergence neighborhoods; the purple striped area is erroneously mapped to $p_0^{(1)}$ and the area striped in brown is erroneously mapped to $p_0^{(2)}$. In these areas a feasible solution approximation is attainable, but it is not found by $\Theta(p)$ because the wrong reference point is chosen. The reference determination can be improved by using a weighted Euclidean norm

$$\|p\|_W = \sqrt{p^\intercal V^\intercal V p} = \sqrt{p^\intercal W p}, \tag{5.59}$$

that takes into account the extend of the convergence neighborhoods in different directions. For the weighted norm (red) the areas striped in green are mapped correctly, but the orange area is now erroneously mapped to $p_0^{(1)}$. Obviously without knowledge of the intersection of $\mathcal{N}^{(1)}, \mathcal{N}^{(2)}$ an error free separation is not possible.

The positive definite $n_p \times n_p$ transformation matrix W defines the scaling and rotation

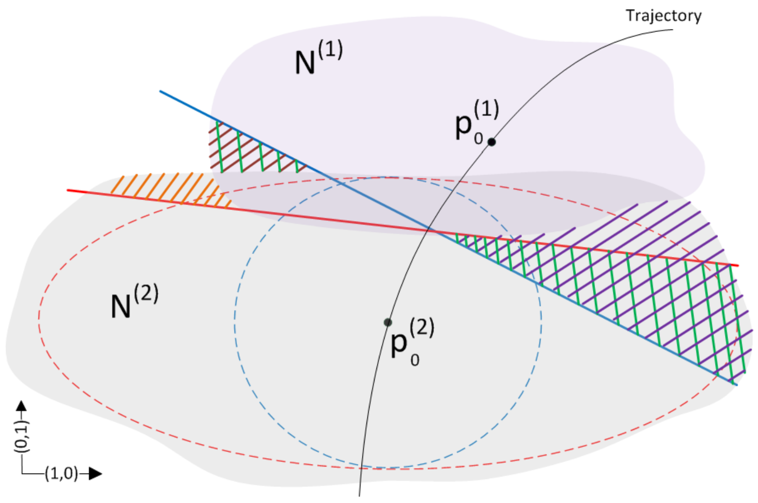


Figure 5.11: Nearest neighbor functions based on differently weighted Euclidean distances.

operations which map a hyper-sphere in unweighted euclidean space into an hyper-ellipsoid. In machine learning and statistics the matrix W is often chosen as the covariance of the data set, centered at its mean (Mahalanobis distance). Unfortunately in our case the means and covariances of $\mathcal{N}^{(1)}, ..., \mathcal{N}^{(l_p)}$ are unknown, such that we cannot use this approach directly, but we can apply a similar idea.

We approximate $\mathcal{N}^{(i)} \in N$ with a ball $\mathcal{B}^{(i)}_{\|\cdot\|_{W^{(i)}}}$ in the norm $\|\cdot\|_{W^{(i)}}$. $W^{(i)}$ is a diagonal matrix with the diagonal elements

$$w^{(j,j)} := \begin{cases} \frac{1}{\underset{\max}{\Delta} p^{(j)}} & \text{if } \underset{\max}{\Delta} p^{(j)} \neq \infty \\ c & \text{if } \underset{\max}{\Delta} p^{(j)} = \infty \end{cases}, \quad 0 < c < \min_j \frac{1}{\underset{\max}{\Delta} p^{(j)}}, \quad j \in \{1, ..., n_p\}. \tag{5.60}$$

$\underset{\max}{\Delta} p^{(j)}$ is the maximal distance that the j-th component of $p_0^{(i)}$ may change in $\text{OCP}_{(5.49)}(p_0^{(i)})$ without causing a change of the active set, as determined by (5.39). The diagonal elements define the scaling of the coordinate axes. For an accurate approximation of $\mathcal{N}^{(i)}$ we also require the off diagonal elements (or covariances) that specify the rotation of the (hyper-) ellipse we use to approximate $\mathcal{N}^{(i)}$. The covariances must be determined empirically or through numerical tests.

We can measure the distance of a fixed parameter p to a nominal parameter $p_0^{(i)} \in P_{nom}$ using any of the norms $W^{(i)}$. To find the closest nominal parameter we must compare these distances, which requires to measure in a common norm, i.e. the comparison must take place in a common normed vector space $(\mathbb{R}^{n_p}, \|\cdot\|_W)$.

The only feasible option is to assume that $W^{(i+1)} \sim W^{(i)}$, for all $i \in \{1,...,l_p - 1\}$, meaning we assume that the dominate skew of all $\mathcal{N}^{(i)} \in N$ is similar. If this assumption is not justified, choosing p_0^r based on an independent variable maybe the better option. If the assumption holds we can define W as mean of the normalized $W^{(i)}$.

$$W := \frac{1}{l_p} \left(\frac{W^{(1)}}{\|\text{trace}(W^{(1)})\|_2} + \ldots + \frac{W^{(l_p)}}{\|\text{trace}(W^{(l_p)})\|_2} \right). \tag{5.61}$$

Choosing $p_0^{(r)}$ based on the norm $\|\cdot\|_W$ is independent of time and adaptive to perturbations, because the choice of the reference point takes into account the entire perturbation vector. This can lead to an increase of the controllable space compared to time tracking under certain conditions:

Consider the convex approximations $\mathcal{B}^{(1)}, ..., \mathcal{B}^{(l_p)}$ of $\mathcal{N}^{(1)}, ..., \mathcal{N}^{(l_p)}$ in weighted Euclidean space $(\mathbb{R}^{n_p}, \|\cdot\|_W)$. Figure 5.12a illustrates that even with an adequate norm the convergence of the RTS algorithm on the balls $\mathcal{B}^{(1)}, \mathcal{B}^{(2)}$ is not sufficient for $\Theta(p)$ to be convergent on $\mathcal{B}^{(1)} \cup \mathcal{B}^{(2)}$, as indicated by the black striped area.

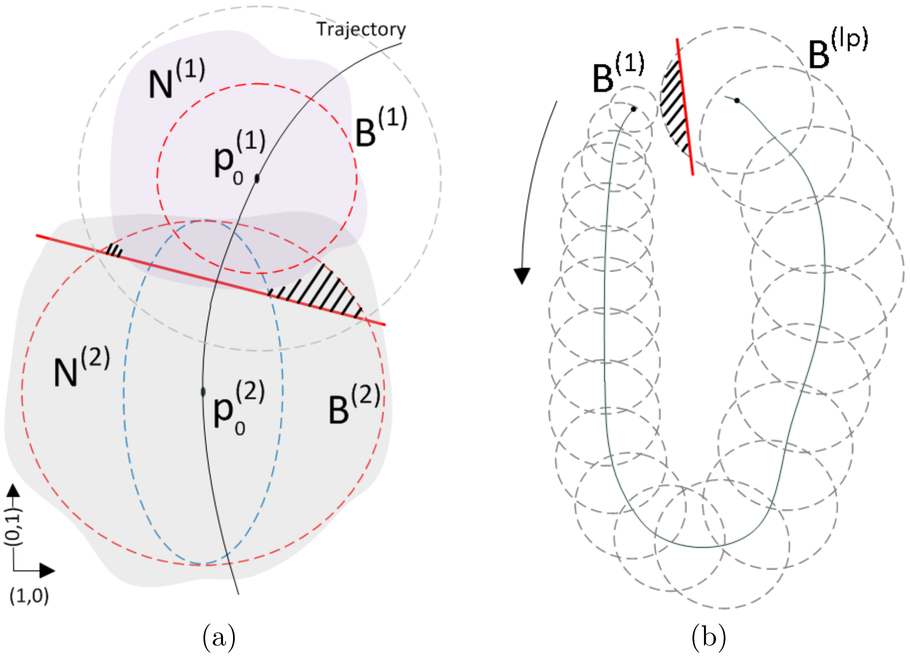


Figure 5.12: Nearest neighbor pitfalls: (a) Insufficient discretization; (b) Trajectory loop.

But it can be easily verified geometrically, that the nearest neighbor is chosen correctly if $p_0^{(i-1)} \in \mathcal{B}^{(i)}$, for all $i \in \{2, ..., l_p\}$ and $p_0^{(i+1)} \in \mathcal{B}^{(i)}$, $i \in \{1, ..., l_p - 1\}$, i.e. the balls must not only overlap, but each ball must contain the center of it's neighbors. Such a dense discretization is indicated in Figure 5.12b, however the nearest neighbor search can still fail if the trajectory contains a loop. For such cases the control space does obviously not include the miss matched subset.

The considerations of this section are summarized in the following corollary.

Corollary 5.62 (Near Optimal Feedback in the Neighborhood of a Nominal Trajectory) *Let* $\overset{\star}{x}(\boldsymbol{\tau}; \bar{p}_0)$, $\boldsymbol{\tau} \in [0; 1]$ *be the nominal trajectory of the open loop problem* $OCP_{(5.1)}(\bar{p}_0)$. *Let*

$$T_{nom} := \{\boldsymbol{\tau}^{(1)}, ..., \boldsymbol{\tau}^{(i)}, ..., \boldsymbol{\tau}^{(l_p)}\}, \quad \boldsymbol{\tau}^{(1)} = 0, \quad \boldsymbol{\tau}^{(i)} < \boldsymbol{\tau}^{(i+1)}, \quad \boldsymbol{\tau}^{(l_p)} < 1, \qquad (5.63)$$

$$p_0^{(i)} := \begin{pmatrix} \bar{p}_0 \\ \overset{\star}{x}(\boldsymbol{\tau}^{(i)}; \bar{p}_0) \end{pmatrix}, \quad \boldsymbol{\tau}^{(i)} \in T_{nom}. \qquad (5.64)$$

Let $OCP_{(5.49)}(p_0^{(i)})$ *be the nominal closed loop subproblem at* $\boldsymbol{\tau}^{(i)} \in T_{nom}$. *Let* $OCP_{(5.49)}(p_0^{(i)})$ *have the discretized form* $NLP(p_0^{(i)}, q_0)$ *with the optimal solution* $\overset{\star}{z}(p_0^{(i)}, q_0)$ *and the parametric* *sensitivities* $\frac{\mathrm{d}z}{\mathrm{d}p}(p_0^{(i)},q_0)$ *and* $\frac{\mathrm{d}z}{\mathrm{d}q}(p_0^{(i)},q_0)$. *Let the following assumptions hold:*

A1: (Convergence of the RTS algorithm)

Given $\overset{\star}{z}(p_0^{(i)},q_0)$, $\frac{\mathrm{d}z}{\mathrm{d}p}(p_0^{(i)},q_0)$, $\frac{\mathrm{d}z}{\mathrm{d}q}(p_0^{(i)},q_0)$ *let the RTS algorithm converge to a near optimal solution of NLP*(p,q_0) *on the ball* $\mathcal{B}^{(i)}_{\|\cdot\|}$.

A2: (Norm)

Let $\|\cdot\|$ *have the property*

$$p_0^{(r)} = \arg \min_{p_0^{(i)} \in P_{nom}} \left\|p - p_0^{(i)}\right\| \quad \Longrightarrow \quad p \in \mathcal{B}^{(r)}, \quad p \in \bigcup_i^{l_p} \mathcal{B}^{(i)}. \tag{5.65}$$

$\Theta(p)$ *is a near optimal feedback law for OCP*$_{(5.1)}(\bar{p})$ *on the controllability space*

$$\mathcal{C} = \bigcup_i^{l_p} \mathcal{B}^{(i)}. \tag{5.66}$$

Proof: If $p \in \mathcal{C}$, p is also in one or more balls $\mathcal{B}^{(1)}, ..., \mathcal{B}^{(l_p)}$. Given a parameter $p = (\bar{p},x)$ (5.53c) determines the nearest nominal parameter $p_0^{(r)}$ with convergence neighborhood $\mathcal{B}^{(r)}$, such that $p \in \mathcal{B}^{(r)}$ (A2). Equation (5.53e) executes the RTS algorithm with the arguments $\Delta p = p - p_0^{(r)}$, $\overset{\star}{z}(p_0^{(r)},q_0)$, $\frac{\mathrm{d}z}{\mathrm{d}p}(p_0^{(r)},q_0)$, $\frac{\mathrm{d}z}{\mathrm{d}q}(p_0^{(r)},q_0)$ for which the algorithm approximates a near optimal solution of NLP(p,q_0) (A1). The NLP solution contains a discrete near optimal control function for OCP$_{(5.49)}(p)$. □

5.6.2 Data Reduction Through Dynamic Programming

It is obvious that the problems OCP$_{(5.49)}(p_0^{(i)})$, $p_0^{(i)} \in P_{nom}$ are subproblems of OCP$_{(5.49)}(p_0^{(1)})$. In addition the objective function of OCP$_{(5.49)}(p_0^{(1)})$ is separable, thus the Principle of Optimality 4.28 can be applied to reduce the amount of data required to facilitate $\Theta(p)$:

The optimal solution of OCP$_{(5.49)}(p_0^{(1)})$ on the interval $[\boldsymbol{\tau}^{(i)};\, 1]$ is identical to the optimal solution of OCP$_{(5.49)}(p_0^{(i)})$. The transformation from normalized nominal open loop time $\boldsymbol{\tau} \in [\boldsymbol{\tau}^{(i)};\, 1]$ to normalized problem specific time $\tau_i \in [0;\, 1]$ is

$$\tau_i(\boldsymbol{\tau}) = \frac{\boldsymbol{\tau} - \boldsymbol{\tau}^{(i)}}{1 - \boldsymbol{\tau}^{(i)}}, \quad \tau_i \in [0;\, 1], \quad \boldsymbol{\tau} \in [\boldsymbol{\tau}^{(i)};\, 1] \tag{5.67}$$

The reverse transformation is

$$\boldsymbol{\tau}(\tau_i) = \boldsymbol{\tau}^{(i)} + (1 - \boldsymbol{\tau}^{(i)})\tau_i, \quad \tau_i \in [0;\, 1], \quad \boldsymbol{\tau} \in [\boldsymbol{\tau}^{(i)};\, 1]. \tag{5.68}$$

The optimal solution of subproblem OCP$_{(5.49)}(p_0^{(i)})$ can be obtained from the optimal solution of OCP$_{(5.49)}(p_0^{(1)})$ as

$$\overset{\star}{u}_i(\tau_i,p_0^{(i)}) = \overset{\star}{u}_1(\boldsymbol{\tau}(\tau_i),p_0^{(1)}), \quad \tau_i \in [0;\, 1], \quad 1 \le i \le l_p, \tag{5.69a}$$

$$\overset{\star}{x}_i(\tau_i,p_0^{(i)}) = \overset{\star}{x}_1(\boldsymbol{\tau}(\tau_i),p_0^{(1)}), \quad \tau_i \in [0;\, 1], \quad 1 \le i \le l_p. \tag{5.69b}$$

The relationship between the problem specific nominal local time-to-go and the nominal time of the full process is

$$\overset{\star}{t}_{i,go} = \overset{\star}{t}_{1,go} - t^{(i)}. \tag{5.70}$$

Hence only the optimal solution of $\mathrm{OCP}_{(5.49)}(p_0^{(1)})$ is required for $\Theta(p)$. The optimal solution of $\mathrm{OCP}_{(5.49)}(p_0^{(1)})$ is in turn identical to the solution of the nominal open loop problem $\mathrm{OCP}_{(5.1)}(\bar{p}_0)$.

It is important to note, that the Principle of Optimality does not apply to the parametric sensitivities, i.e. the optimal solution derivative. This becomes clear by considering that the relationship between an infinitesimally small change in the initial state and the attainable optimal value is described by the Hamilton-Jacobi-Bellman partial differential equation (4.36). The parametric sensitivity of a single open loop solution does not contain the information to approximate the entire extremal field described by a solution of the HJB equation, but only the part in the neighborhood of the open loop initial condition. The parametric sensitivities of all subproblems $\mathrm{OCP}_{(5.49)}(p_0^{(i)})$ are required to approximate a solution to the HJB equation locally around the nominal trajectory.

Figure 5.13 shows the parametric sensitivity w.r.t. perturbations in the initial state and the mass computed for the different subproblems of the rocket car problem, $\mathrm{OCP}_{(5.19)}(p_0)$. For this comparison the parametric sensitivities of all subproblems are plotted in the normalized open loop time $\boldsymbol{\tau} \in [0;\, 1]$.

The optimal value as function of the initial state (as defined in dynamic programming) can be approximated around the initial state of $\mathrm{OCP}_{(5.49)}(p_0^{(i)})$ by using the parametric sensitivity of the objective function:

$$\frac{\mathrm{d}f}{\mathrm{d}p}(\overset{\star}{z}(p_0^{(i)}),p_0^{(i)}) = \frac{\partial f}{\partial z}(\overset{\star}{z}(p_0^{(i)}),p_0^{(i)})\frac{\mathrm{d}z}{\mathrm{d}p}(p_0^{(i)}) + \frac{\partial f}{\partial p}(\overset{\star}{z}(p_0^{(i)}),p_0^{(i)}). \tag{5.71}$$

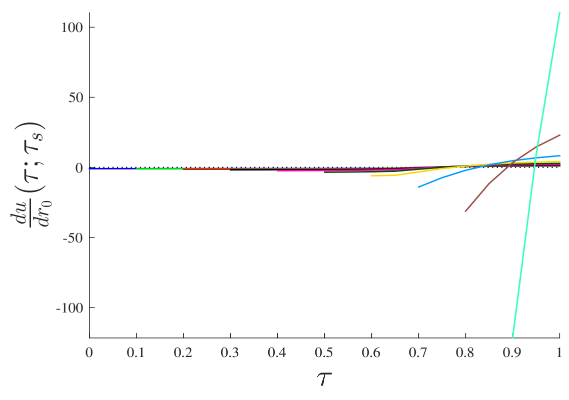
Equation (5.71) contains the sensitivity of the objective function w.r.t. to the initial state, which is a first order approximation of the optimal value gradient:

$$\nabla_{x_s} V(\overset{\star}{x}(\boldsymbol{\tau}^{(i)},\bar{p}_0)) \approx \frac{\mathrm{d}f}{\mathrm{d}x_s}(p_0^{(i)}), \quad \boldsymbol{\tau}^{(i)} \in T_{nom}, \quad p_0^{(i)} \in P_{nom}. \tag{5.72}$$

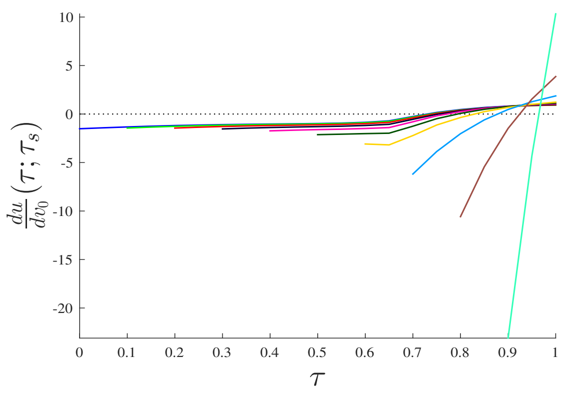
The use of the parametric sensitivities of all $\mathrm{OCP}_{(5.49)}(p_0^{(i)})$ allows an approximation of the optimal value in the neighborhood of the nominal open loop trajectory $\overset{\star}{x}(\boldsymbol{\tau};\bar{p}_0)$, $\boldsymbol{\tau} \in [0;1]$ as

$$V(p) \approx f(\overset{\star}{z}(p^{(r)}),p^{(r)}) + \frac{\mathrm{d}f}{\mathrm{d}p}(\overset{\star}{z}(p^{(r)}),p^{(r)}) \left[\underbrace{\begin{pmatrix} \bar{p} \\ x \end{pmatrix}}_{p} - \underbrace{\begin{pmatrix} \bar{p}_0 \\ \overset{\star}{x}(\boldsymbol{\tau}^{(r)},\bar{p}_0) \end{pmatrix}}_{p^{(r)}} \right], \tag{5.73}$$

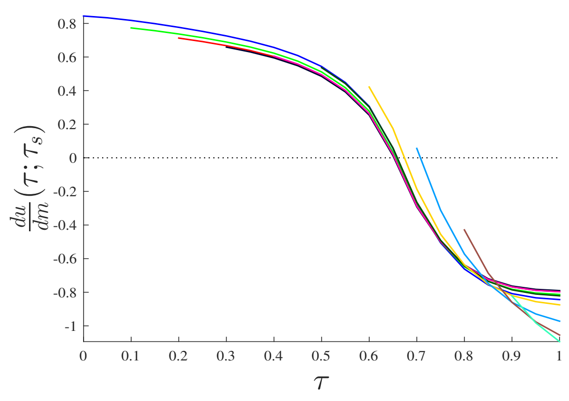
if $p^{(r)}$ is obtained according to (5.53c).



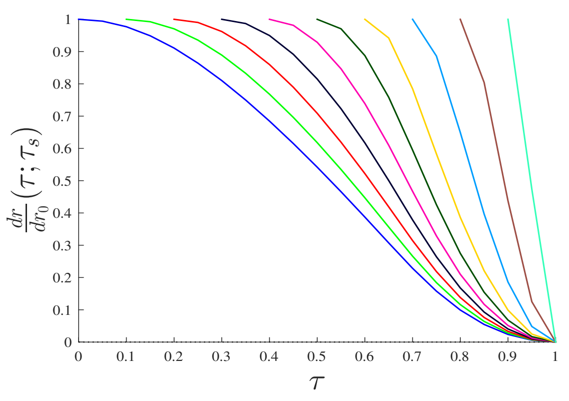
(a) Control sen. w.r.t. initial position



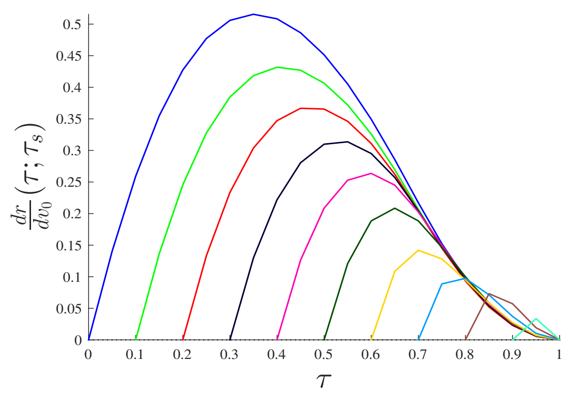
(b) Control sen. w.r.t. initial velocity



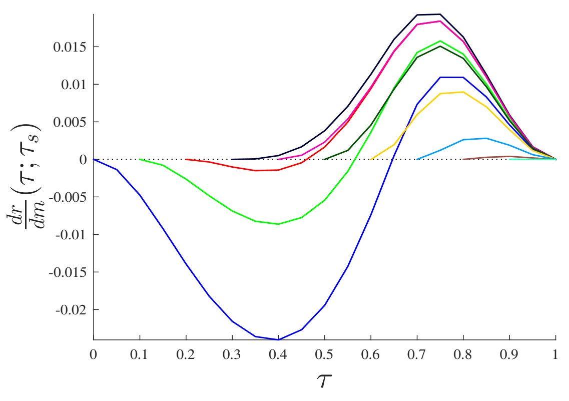
(c) Control sen. w.r.t. mass



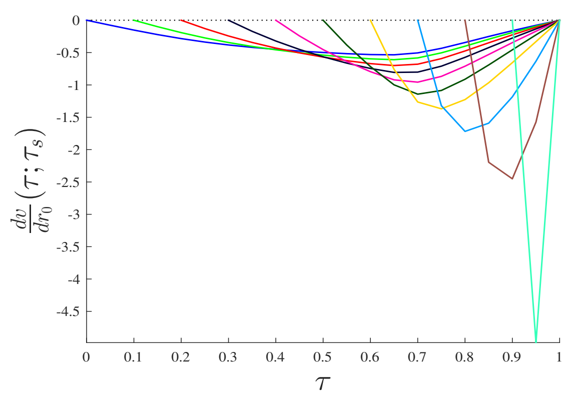
(d) Position sen. w.r.t. initial position



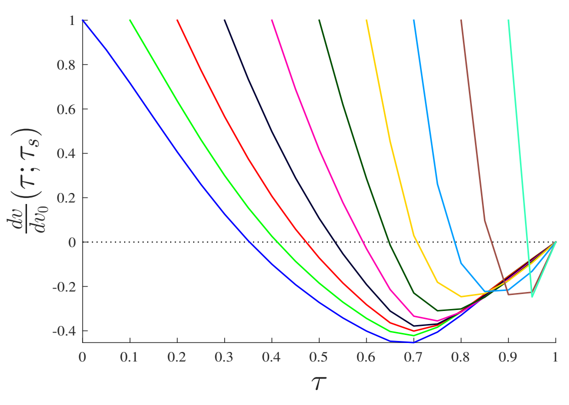
(e) Position sen. w.r.t. initial velocity



(f) Position sen. w.r.t. mass



(g) Velocity sen. w.r.t. initial position



(h) Velocity sen. w.r.t. initial velocity

Figure 5.13: Parametric sensitivity of the control, position and velocity against perturbations in the mass and the initial position and velocity at different points of the optimal trajectory of $\mathrm{OCP}_{(5.19)}(p_0)$.

5.6.3 Interpolation of Parametric Sensitivities

The number of discrete initial conditions that need to be analyzed to facilitate $\Theta(p)$ can be reduced by interpolating the sensitivity differentials between neighboring initial conditions. Depending on the curvature of the sensitivity differentials, i.e. the required interpolation order, this achieves a reduction of the memory consumption in trade-off for additional CPU load caused by the interpolation.

Interpolation with Respect to Time

Let $\frac{\mathrm{d}z}{\mathrm{d}p}(p_0^{(i)})$ be the parametric sensitivity of NLP$(p_0^{(i)})$, cf. (5.53b). $\frac{\mathrm{d}z}{\mathrm{d}p}(p_0^{(i)})$ contains the sensitivity of the control variables on the grid T_u, and the sensitivity of the state variables on the grid T_x:

$$\frac{\mathrm{d}u^{(j)}}{\mathrm{d}p^{(r)}}(p_0^{(i)}) = \begin{pmatrix} \frac{\mathrm{d}u^{(j)}[\boldsymbol{\tau}^{(1)}]}{\mathrm{d}p^{(r)}}(p_0^{(i)}) \\ \vdots \\ \frac{\mathrm{d}u^{(j)}[\boldsymbol{\tau}^{(l_u)}]}{\mathrm{d}p^{(r)}}(p_0^{(i)}) \end{pmatrix} \quad 1 \le j \le n_u, \quad 1 \le r \le n_p, \quad \boldsymbol{\tau}^{(i)} \in T_u \tag{5.74a}$$

$$\frac{\mathrm{d}x^{(j)}}{\mathrm{d}p^{(r)}}(p_0^{(i)}) = \begin{pmatrix} \frac{\mathrm{d}x^{(j)}[\boldsymbol{\tau}^{(1)}]}{\mathrm{d}p^{(r)}}(p_0^{(i)}) \\ \vdots \\ \frac{\mathrm{d}x^{(j)}[\boldsymbol{\tau}^{(l_x)}]}{\mathrm{d}p^{(r)}}(p_0^{(i)}) \end{pmatrix} \quad 1 \le j \le n_x, \quad 1 \le r \le n_p, \quad \boldsymbol{\tau}^{(i)} \in T_x \tag{5.74b}$$

$$\frac{\mathrm{d}t_d}{\mathrm{d}p^{(r)}}(p_0^{(i)}) \qquad 1 \le j \le n_p \tag{5.74c}$$

The convergence of the direct solution to the PMP solution, for an infinitesimally small discretization step size, implies the convergence of the direct solution derivative to a piecewise continuous function that is an approximation of the sensitivities obtained with respect to the PMP conditions. This justifies the interpolation of the sensitivity differentials between the discretization points of the control and state functions. For each initial condition $p_0^{(i)} \in P_{nom}$ we approximate the PMP sensitivities as

$$\left.\frac{\mathrm{d}u^{(j)}}{\mathrm{d}p^{(r)}}(\boldsymbol{\tau},p_0^{(i)})\right|_{\boldsymbol{\tau}\in[0;\,1]} \approx \underset{\boldsymbol{\tau}^{(i)}}{\text{interp}}\left(\frac{\mathrm{d}u^{(j)}}{\mathrm{d}p^{(r)}}(\boldsymbol{\tau}^{(i)},p_0^{(i)})\right), \qquad \boldsymbol{\tau}^{(i)} \in T_u,\ 1 \le j \le n_u,\ 1 \le r \le n_p \tag{5.75a}$$

$$\left.\frac{\mathrm{d}x^{(j)}}{\mathrm{d}p^{(r)}}(\boldsymbol{\tau},p_0^{(i)})\right|_{\boldsymbol{\tau}\in[0;\,1]} \approx \underset{\boldsymbol{\tau}^{(i)}}{\text{interp}}\left(\frac{\mathrm{d}x^{(j)}}{\mathrm{d}p^{(r)}}(\boldsymbol{\tau}^{(i)},p_0^{(i)})\right), \qquad \boldsymbol{\tau}^{(i)} \in T_x,\ 1 \le j \le n_x,\ 1 \le r \le n_p \tag{5.75b}$$

If the solution contains corners or points of discontinuity, the interpolation is admissible on each continuously differentiable arc.

Interpolation with Respect to the Initial Condition

The dependency of $\frac{\mathrm{d}z}{\mathrm{d}p}(p_0^{(1)}), \ldots, \frac{\mathrm{d}z}{\mathrm{d}p}(p_0^{l_p})$ on the initial condition is expressed as dependency on the normalized nominal open loop time $\boldsymbol{\tau} \in [0;\,1]$ via (5.52). We can thus interpolate between sensitivities of neighboring initial conditions, $p_0(\boldsymbol{\tau}^{(i)})$ and $p_0(\boldsymbol{\tau}^{(i+1)})$. The interpolation variable for the initial condition is denoted $\boldsymbol{\tau}_s \in [0;\,1]$. For all discretization points $\boldsymbol{\tau}^{(i)} \in T_u$ of the control function, and for all discretization points of the state functions

$\boldsymbol{\tau}^{(i)} \in T_x$, and for the time-to-go, we can approximate the change of the sensitivities between initial conditions $p_0^{(i)} \in P_{nom}$ as

$$\frac{\mathrm{d}u^{(j)}}{\mathrm{d}p^{(r)}}(\boldsymbol{\tau}^{(i)}, p_0(\boldsymbol{\tau}_s))\bigg|_{\boldsymbol{\tau}_s \in [0;\,1]} \approx \underset{p_0(\boldsymbol{\tau}^{(i)})}{\mathrm{interp}} \left(\frac{\mathrm{d}u^{(j)}}{\mathrm{d}p^{(r)}}(\boldsymbol{\tau}^{(i)}, p_0(\boldsymbol{\tau}^{(i)})) \right), \quad \boldsymbol{\tau}^{(i)} \in T_u,\ 1 \le j \le n_u, \tag{5.76a}$$

$$\frac{\mathrm{d}x^{(j)}}{\mathrm{d}p^{(r)}}(\boldsymbol{\tau}^{(i)}, p_0(\boldsymbol{\tau}_s))\bigg|_{\boldsymbol{\tau}_s \in [0;\,1]} \approx \underset{p_0(\boldsymbol{\tau}^{(i)})}{\mathrm{interp}} \left(\frac{\mathrm{d}x^{(j)}}{\mathrm{d}p^{(r)}}(\boldsymbol{\tau}^{(i)}, p_0(\boldsymbol{\tau}^{(i)})) \right), \quad \boldsymbol{\tau}^{(i)} \in T_x,\ 1 \le j \le n_x \tag{5.76b}$$

$$\frac{\mathrm{d}t_{go}}{\mathrm{d}p^{(r)}}(p_0(\boldsymbol{\tau}_s))\bigg|_{\boldsymbol{\tau}_s \in [0;\,1]} \approx \underset{p_0(\boldsymbol{\tau}^{(i)})}{\mathrm{interp}} \left(\frac{\mathrm{d}t_{go}}{\mathrm{d}p^{(r)}}(p_0(\boldsymbol{\tau}^{(i)})) \right). \tag{5.76c}$$

Sensitivity Surface Interpolation

The interpolation with respect to time and the initial condition is combined into a 2D surface interpolation, by transforming the sensitivities of all problems $\mathrm{OCP}_{(5.49)}(p_0^{(i)})$, $p_0^{(i)} \in P_{nom}$ to the normalized nominal open loop time (via (5.68)). Then the sensitivities can be arranged on a common 2D grid. This grid can be used as input for a surface interpolation algorithm, denoted as $\Xi_{\boldsymbol{\tau},\boldsymbol{\tau}_s}(\cdot)$, $\boldsymbol{\tau} \in [\boldsymbol{\tau}_s; 1]$, $\boldsymbol{\tau}_s \in [0;\,1]$. The surface fitting is performed offline, as last step of the preparation phase. Depending on the choice of interpolation algorithm the data basis may change in size and structure. We assume that the interpolation is exact at the data points.

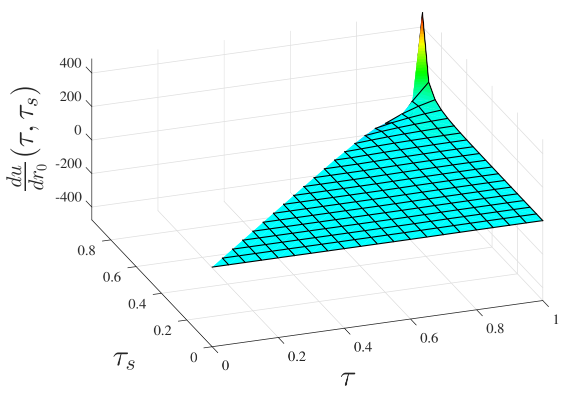
$$\frac{\mathrm{d}u^{(j)}}{\mathrm{d}p^{(r)}}(\boldsymbol{\tau},\boldsymbol{\tau}_s) \approx \Xi_{\boldsymbol{\tau},\boldsymbol{\tau}_s}\left[\begin{pmatrix} \frac{\mathrm{d}u^{(j)}[\boldsymbol{\tau}^{(1)},\boldsymbol{\tau}_s^{(1)}]}{\mathrm{d}p^{(r)}} & \cdots & \frac{\mathrm{d}u^{(j)}[\boldsymbol{\tau}^{(1)},\boldsymbol{\tau}_s^{(l_p)}]}{\mathrm{d}p^{(r)}} \\ \vdots & \ddots & \vdots \\ \frac{\mathrm{d}u^{(j)}[\boldsymbol{\tau}^{(l_u)},\boldsymbol{\tau}_s^{(1)}]}{\mathrm{d}p^{(r)}} & \cdots & \frac{\mathrm{d}u^{(j)}[\boldsymbol{\tau}^{(l_u)},\boldsymbol{\tau}_s^{(l_p)}]}{\mathrm{d}p^{(r)}} \end{pmatrix}\right] \tag{5.77a}$$

$$\frac{\mathrm{d}x^{(j)}}{\mathrm{d}p^{(r)}}(\boldsymbol{\tau},\boldsymbol{\tau}_s) \approx \Xi_{\boldsymbol{\tau},\boldsymbol{\tau}_s}\left[\begin{pmatrix} \frac{\mathrm{d}x^{(j)}[\boldsymbol{\tau}^{(1)},\boldsymbol{\tau}_s^{(1)}]}{\mathrm{d}p^{(r)}} & \cdots & \frac{\mathrm{d}x^{(j)}[\boldsymbol{\tau}^{(1)},\boldsymbol{\tau}_s^{(l_p)}]}{\mathrm{d}p^{(r)}} \\ \vdots & \ddots & \vdots \\ \frac{\mathrm{d}x^{(j)}[\boldsymbol{\tau}^{(l_x)},\boldsymbol{\tau}_s^{(1)}]}{\mathrm{d}p^{(r)}} & \cdots & \frac{\mathrm{d}x^{(j)}[\boldsymbol{\tau}^{(l_x)},\boldsymbol{\tau}_s^{(l_p)}]}{\mathrm{d}p^{(r)}} \end{pmatrix}\right] \tag{5.77b}$$

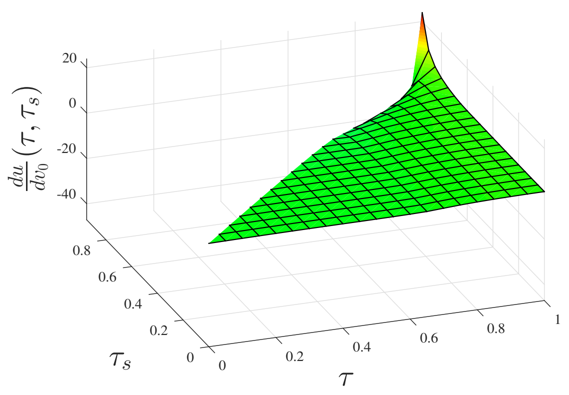
$$\frac{\mathrm{d}t_{go}}{\mathrm{d}p^{(r)}}(\boldsymbol{\tau}_s) \approx \Xi_{\boldsymbol{\tau}_s}\left[\begin{pmatrix} \frac{\mathrm{d}t_{go}[\boldsymbol{\tau}_s^{(1)}]}{\mathrm{d}p^{(r)}} & \cdots & \frac{\mathrm{d}t_{go}[\boldsymbol{\tau}_s^{(l_p)}]}{\mathrm{d}p^{(r)}} \end{pmatrix}\right] \tag{5.77c}$$

During the online phase an approximation of the parametric sensitivity differentials at all nominal initial conditions of the closed loop system can be obtained by the surface evaluation (5.77). Figure 5.14 shows the sensitivity surfaces of the rocket car example using bilinear interpolation (compare Figure 5.13). The sensitivities for the linear constraint perturbations q can be handled analogously.

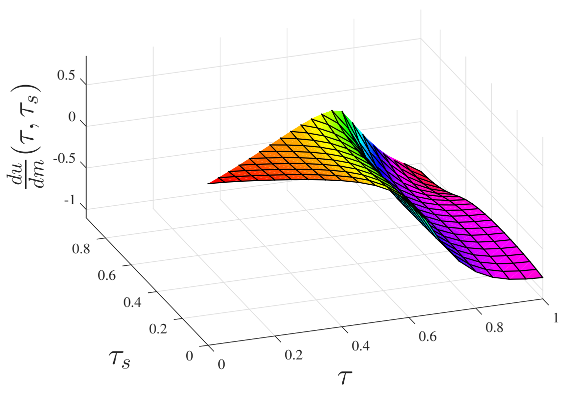
Remark 5.78 (Sensitivity Sampling)
Using the same grid T_u for $OCP_{(5.49)}(p_0^{(i)})$, $p_0^{(i)} \in P_{nom}$ results into an increasing solution accuracy as the same number of discretization points is used for the remaining nominal trajectory $\overset{\star}{x}(\boldsymbol{\tau}, p_0)$ on the shrinking interval $[\boldsymbol{\tau}; 1]$, $\boldsymbol{\tau} \to 1$. In addition the uniform grid resulting from the equal amount of discretization points can be advantageous for the surface interpolation. An alternative is to reduce the number of discretization points step wise, while keeping the accuracy of the solution constant. This results into less sensitivity samples and therefore a reduced memory consumption, but requires the surface interpolation to operate on a nonuniform grid.



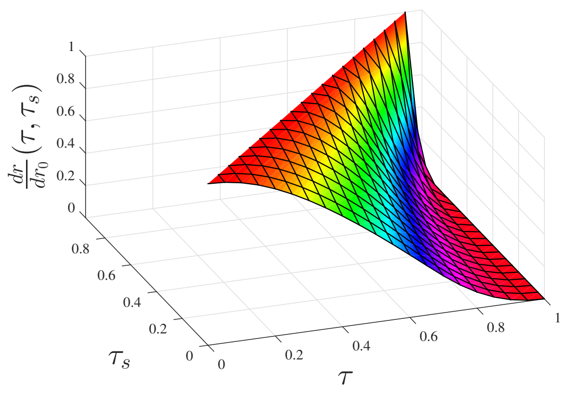
(a) Control sen. w.r.t. initial position



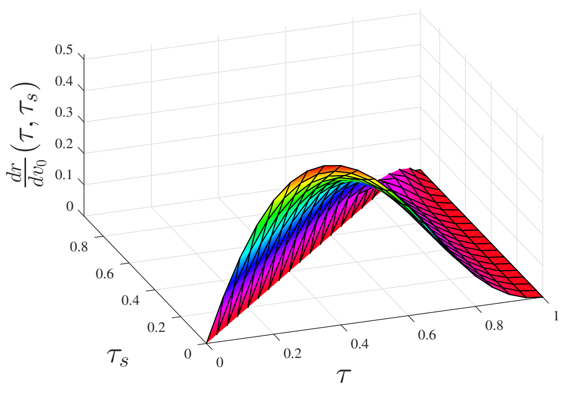
(b) Control sen. w.r.t. initial velocity



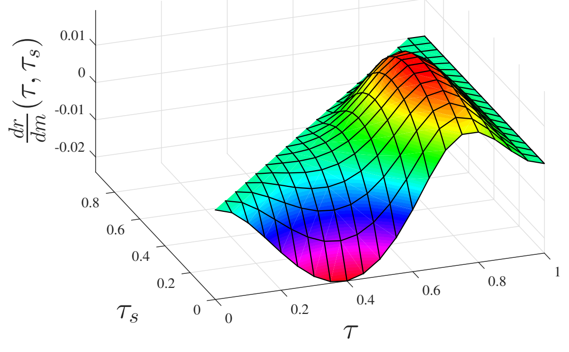
(c) Control sen. w.r.t. mass



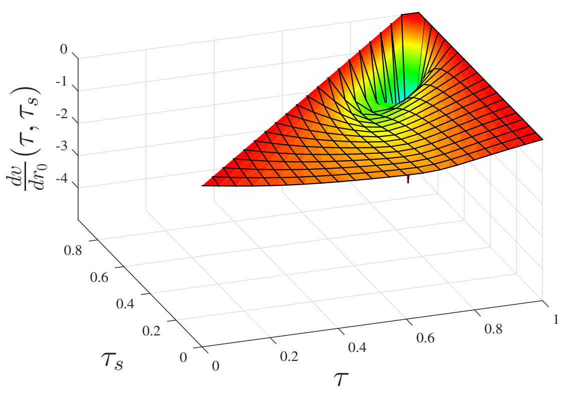
(d) Position sen. w.r.t. initial position



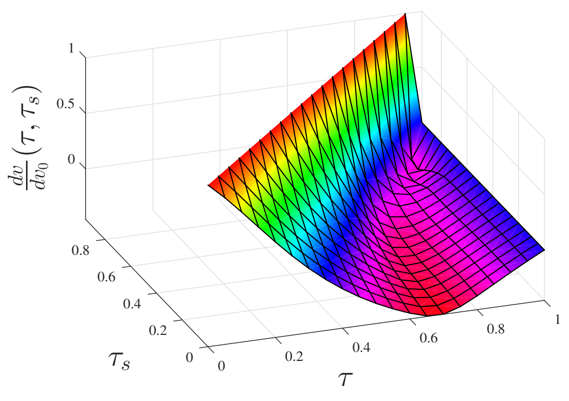
(e) Position sen. w.r.t. initial velocity



(f) Position sen. w.r.t. mass



(g) Velocity sen. w.r.t. initial position



(h) Velocity sen. w.r.t. initial velocity

Figure 5.14: Interpolated parametric sensitivity of the control, position and velocity against perturbations in the mass and the position and velocity, occurring at different points of the optimal trajectory of OCP$_{(5.19)}(p_0)$ (cf. Figure 5.13).

Sensitivity Surface Evaluation in $\Theta(p)$

In the online phase the sensitivity surfaces are evaluated at a reference initial condition $p_0(\boldsymbol{\tau}^{(r)})$, $\boldsymbol{\tau}^{(r)} \in [0;\,1[$ and on a reference grid $\widetilde{T}_u$ or $\widetilde{T}_x$ for control and state functions.

To determine $\boldsymbol{\tau}^{(r)}$ in between the grid points T_{nom} we refine the earlier proposed search strategy. Let $T_{dense} \subset [0;1[$ be a grid that fulfills the approximation accuracy requirement. Let $T_{nom} \subset T_{dense}$. $\boldsymbol{\tau}^{(r)}$ is determined by first finding the closest point

$$\bar{\boldsymbol{\tau}}^{(r)} = \arg\min_{\boldsymbol{\tau}^{(i)}} \left\| p - p_0(\boldsymbol{\tau}^{(i)}) \right\|, \quad \boldsymbol{\tau}^{(i)} \in T_{nom}. \tag{5.79}$$

The search is then refined on an interval around $\bar{\boldsymbol{\tau}}^{(r)}$ to determine $\boldsymbol{\tau}^{(r)} \in T_{dense}$.

Remark 5.80 (Alternative Approach: Distance to a Piecewise Polynomial)
A more accurate, but also more costly approach is to interpolate $p_0(\boldsymbol{\tau})$, $\boldsymbol{\tau} \in [0;\,1]$ using piecewise polynomials, e.g. B-splines. The determination of $\boldsymbol{\tau}^{(r)}$ then requires a root finding problem for each spline interval.

It remains to choose the evaluation points for the control and state functions. To minimize the distance to the original discretization points it is natural to choose

$$\widetilde{T}_u := \boldsymbol{\tau}^{(r)} \cup \{\boldsymbol{\tau} \mid \boldsymbol{\tau} \in T_u,\, \boldsymbol{\tau} > \boldsymbol{\tau}^{(r)}\}, \tag{5.81a}$$
$$\widetilde{T}_x := \boldsymbol{\tau}^{(r)} \cup \{\boldsymbol{\tau} \mid \boldsymbol{\tau} \in T_x,\, \boldsymbol{\tau} > \boldsymbol{\tau}^{(r)}\}. \tag{5.81b}$$

The surfaces can now be evaluated at $(\boldsymbol{\tau}^{(r)}, \widetilde{T}_u)$, $(\boldsymbol{\tau}^{(r)}, \widetilde{T}_x)$ and $(\boldsymbol{\tau}^{(r)})$ to obtain an approximation of the sensitivities for the initial condition at $p_0(\boldsymbol{\tau}^{(r)})$.

$$\widetilde{\frac{\mathrm{d}u^{(j)}}{\mathrm{d}p^{(r)}}}(\boldsymbol{\tau},\boldsymbol{\tau}^{(r)}) \quad \boldsymbol{\tau} \in \widetilde{T}_u, \quad 1 \le j \le n_u, \quad 1 \le r \le n_p, \tag{5.82a}$$
$$\widetilde{\frac{\mathrm{d}x^{(j)}}{\mathrm{d}p^{(r)}}}(\boldsymbol{\tau},\boldsymbol{\tau}^{(r)}) \quad \boldsymbol{\tau} \in \widetilde{T}_x, \quad 1 \le j \le n_x, \quad 1 \le r \le n_p, \tag{5.82b}$$
$$\widetilde{\frac{\mathrm{d}t_d}{\mathrm{d}p^{(r)}}}(\boldsymbol{\tau}^{(r)}) \qquad 1 \le r \le n_p, \tag{5.82c}$$
$$\widetilde{\frac{\mathrm{d}u^{(j)}}{\mathrm{d}q_a^r}}(\boldsymbol{\tau},\boldsymbol{\tau}^{(r)}) \quad \boldsymbol{\tau} \in \widetilde{T}_u, \quad 1 \le j \le n_u, \quad 1 \le r \le n_{q_a}, \tag{5.82d}$$
$$\widetilde{\frac{\mathrm{d}x^{(j)}}{\mathrm{d}q_a^r}}(\boldsymbol{\tau},\boldsymbol{\tau}^{(r)}) \quad \boldsymbol{\tau} \in \widetilde{T}_x, \quad 1 \le j \le n_x, \quad 1 \le r \le n_{q_a}, \tag{5.82e}$$
$$\widetilde{\frac{\mathrm{d}t_d}{\mathrm{d}q_a^r}}(\boldsymbol{\tau}^{(r)}) \qquad 1 \le r \le n_{q_a}. \tag{5.82f}$$

The nominal open loop trajectory $\overset{\star}{x}(p_0)$ and control function $\overset{\star}{u}(p_0)$ must be interpolated on $\widetilde{T}_u$, $\widetilde{T}_x$ to obtain the optimal solution at the chosen discretization points. With these inputs the RTS algorithm can be executed at $p_0(\boldsymbol{\tau}^{(r)})$.

The sensitivity differentials are the main memory consumer for the feedback law. The interpolation is a trade-of between lower memory consumption and increased computational cost. The number of required surface evaluations is equal to the combinations of parameters with control or state equations and the time:

$$n_{surf} = (n_u + n_x + 1)(n_p + n_a). \tag{5.83}$$

In addition to memory reduction, the interpolation reduces $\|\Delta p\| = \|p - p_0(\boldsymbol{\tau}^{(r)})\|$, by allowing to start the interpolation from a closer initial condition. The optimality error of the solution approximation is quadratic in $\|\Delta p\|$, a reduction thus results into a quadratic optimality increase.

At the same time note that Theorem (5.18) only holds at T_{nom}, i.e. the interpolated sensitivities do not fulfill the assumptions. The error incurred from the interpolation of the sensitivity differentials depends on the interpolation order. We will however not pursue a theoretical investigation of this matter, but rather move on to a numerical test and an application oriented validation.

5.6.4 Synthesis Conclusion and Pseudo Code Statement

We conclude the derivation by incorporating the refinements discussed in the previous sections into the feedback law. As main result of this chapter the pseudo code of $\Theta(p)$ is stated, which is the basis for the implementation used during the numerical evaluations in Chapter 6. The offline preparation phase is stated in Algorithm (2), the online feedback phase is stated in Algorithm (3).

Remark 5.84 (Determination of Discretization Points)
The local discretization error of the grid T_u can be approximated using a Runge-Kutta-Fehlberg method. An iterative grid refinement, as suggested e.g. in [Büs98], is used to distribute the local discretization error equally and thereby minimize the global discretization for the given number of discretization points.

1. *$T_u \subset [0;1]$ is determined such that the global discretization error is below the desired threshold.*
2. *$T_x \subset T_u$ is as coarse as possible to minimize memory and CPU requirements, while sufficiently dissecting the trajectory, as discussed in Section 5.5.1.*
3. *$T_{nom} \subset T_u$ is as coarse as possible, such that the convergence neighborhoods of the RTS algorithm are a cover of the nominal trajectory (determined through iterative numeric tests).*
4. *$T_{dense} \subset [0;1]$ is a refinement of T_{nom} that is used to determine the closest initial condition.*

This concludes the chapter. In the next chapter we focus on evaluating the feasibility and performance of $\Theta(p)$ for guiding the hypersonic entry of a small capsule into the Martian atmosphere.

Algorithm 2 Θ offline phase

```
 1: procedure PREPARATION($T_{dense}$, $T_{nom}$, $T_u$, $T_x$)
 2:     $\mathrm{NLP}^{(0)}(\bar{p}_0) \leftarrow$ Full discretization of $\mathrm{OCP}_{(5.1)}(\bar{p}_0)$ on $T_{dense}$
        $\overset{\star}{u}_{(5.1)}(\bar{p}_0)$
 3:     $\overset{\star}{x}_{(5.1)}(\bar{p}_0) \leftarrow$ Solve $\mathrm{NLP}^{(0)}(\bar{p}_0)$
        $\overset{\star}{t}_{(5.1)}(\bar{p}_0)$
 4:     $p_0(\boldsymbol{\tau}) = \begin{pmatrix} \bar{p}_0 \\ \overset{\star}{x}_{(5.1)}(\boldsymbol{\tau},\bar{p}_0) \end{pmatrix}$                ▷ (5.52)
 5:     $P_{nom} \leftarrow$ Evaluate $p_0(\boldsymbol{\tau})$ on $T_{nom}$                ▷ (5.52b)
 6:     for $p_0^{(i)} \in P_{nom}$ do
 7:         $\mathrm{NLP}^{(i)}(p_0^{(i)},q_0) \leftarrow$ Multiple shooting transcription of $\mathrm{OCP}_{(5.49)}(p_0^{(i)})$ on $T_u$,$T_x$
            $\overset{\star}{u}_{(5.49)}(p_0^{(i)})$
 8:         $\overset{\star}{x}_{(5.49)}(p_0^{(i)}) \leftarrow$ Solve $\mathrm{NLP}^{(i)}(p_0^{(i)},q_0)$
            $\overset{\star}{t}_{(5.49)}(p_0^{(i)})$
            $\frac{\mathrm{d}u}{\mathrm{d}p}(p_0^{(i)})$
            $\frac{\mathrm{d}x}{\mathrm{d}p}(p_0^{(i)})$
            $\frac{\mathrm{d}t}{\mathrm{d}p}(p_0^{(i)})$
 9:         $\frac{\mathrm{d}u}{\mathrm{d}q_a}(p_0^{(i)})$ $\leftarrow$ Param. sensitivity analysis of $\overset{\star}{u}_{(5.49)}(p_0^{(i)}), \overset{\star}{x}_{(5.49)}(p_0^{(i)}), \overset{\star}{t}_{(5.49)}(p_0^{(i)})$
            $\frac{\mathrm{d}x}{\mathrm{d}q_a}(p_0^{(i)})$
            $\frac{\mathrm{d}t}{\mathrm{d}q_a}(p_0^{(i)})$
10:     end for
11:     for each parameter $p^{(k)}$, $k = 1,...,n_p$ do
12:         for each control equation $u^{(j)}$, $j = 1,...,n_u$ do
13:             $\widetilde{\frac{\mathrm{d}u^{(j)}}{\mathrm{d}p^{(k)}}}(\boldsymbol{\tau},\boldsymbol{\tau}_s) = \Xi_{\boldsymbol{\tau},\boldsymbol{\tau}_s}\left[\frac{\mathrm{d}u^{(j)}}{\mathrm{d}p^{(k)}}(p_0^{(1)}),...,\frac{\mathrm{d}u^{(j)}}{\mathrm{d}p^{(k)}}(p_0^{(l_p)})\right]$
14:         end for
15:         for each state equation $x^{(j)}$, $j = 1,...,n_x$ do
16:             $\widetilde{\frac{\mathrm{d}x^{(j)}}{\mathrm{d}p^{(k)}}}(\boldsymbol{\tau},\boldsymbol{\tau}_s) = \Xi_{\boldsymbol{\tau},\boldsymbol{\tau}_s}\left[\frac{\mathrm{d}x^{(j)}}{\mathrm{d}p^{(k)}}(p_0^{(1)}),...,\frac{\mathrm{d}x^{(j)}}{\mathrm{d}p^{(k)}}(p_0^{(l_p)})\right]$
17:         end for
18:         $\widetilde{\frac{\mathrm{d}t}{\mathrm{d}p^{(k)}}}(\boldsymbol{\tau}_s) = \Xi_{\boldsymbol{\tau}_s}\left[\frac{\mathrm{d}t}{\mathrm{d}p^{(k)}}(p_0^{(1)}),...,\frac{\mathrm{d}t}{\mathrm{d}p^{(k)}}(p_0^{(l_p)})\right]$
19:     end for
20:     for each parameter $q_a^{(k)}$, $k = 1,...,n_a$ do
21:         for each control equation $u^{(j)}$, $j = 1,...,n_u$ do
22:             $\widetilde{\frac{\mathrm{d}u^{(j)}}{\mathrm{d}q_a^{(k)}}}(\boldsymbol{\tau},\boldsymbol{\tau}_s) = \Xi_{\boldsymbol{\tau},\boldsymbol{\tau}_s}\left[\frac{\mathrm{d}u^{(j)}}{\mathrm{d}q_a^{(k)}}(p_0^{(1)}),...,\frac{\mathrm{d}u^{(j)}}{\mathrm{d}q_a^{(k)}}(p_0^{(l_p)})\right]$
23:         end for
24:         for each state equation $x^{(j)}$, $j = 1,...,n_x$ do
25:             $\widetilde{\frac{\mathrm{d}x^{(j)}}{\mathrm{d}q_a^{(k)}}}(\boldsymbol{\tau},\boldsymbol{\tau}_s) = \Xi_{\boldsymbol{\tau},\boldsymbol{\tau}_s}\left[\frac{\mathrm{d}x^{(j)}}{\mathrm{d}q_a^{(k)}}(p_0^{(1)}),...,\frac{\mathrm{d}x^{(j)}}{\mathrm{d}q_a^{(k)}}(p_0^{(l_p)})\right]$
26:         end for
27:         $\widetilde{\frac{\mathrm{d}t}{\mathrm{d}q_a^{(k)}}}(\boldsymbol{\tau}_s) = \Xi_{\boldsymbol{\tau}_s}\left[\frac{\mathrm{d}t}{\mathrm{d}q_a^{(k)}}(p_0^{(1)}),...,\frac{\mathrm{d}t}{\mathrm{d}q_a^{(k)}}(p_0^{(l_p)})\right]$
28:     end for
29:     return $\overset{\star}{u}_{(5.49)}(p_0^{(1)})$, $\overset{\star}{x}_{(5.49)}(p_0^{(1)})$, $\overset{\star}{t}_{(5.49)}(p_0^{(1)})$, $\widetilde{\frac{\mathrm{d}u}{\mathrm{d}p}}$, $\widetilde{\frac{\mathrm{d}x}{\mathrm{d}p}}$, $\widetilde{\frac{\mathrm{d}t}{\mathrm{d}p}}$, $\widetilde{\frac{\mathrm{d}u}{\mathrm{d}q_a}}$, $\widetilde{\frac{\mathrm{d}x}{\mathrm{d}q_a}}$, $\widetilde{\frac{\mathrm{d}t}{\mathrm{d}q_a}}$
30: end procedure
```

Algorithm 3 Θ online phase

```
 1: procedure FEEDBACK($p$)
 2:     data $\left\langle \overset{\star}{u}_{(5.49)}(p_0^{(1)}),\ \overset{\star}{x}_{(5.49)}(p_0^{(1)}),\ \overset{\star}{t}_{(5.49)}(p_0^{(1)}),\ \widetilde{\frac{\mathrm{d}u}{\mathrm{d}p}},\ \widetilde{\frac{\mathrm{d}x}{\mathrm{d}p}},\ \widetilde{\frac{\mathrm{d}t}{\mathrm{d}p}},\ \widetilde{\frac{\mathrm{d}u}{\mathrm{d}q_a}},\ \widetilde{\frac{\mathrm{d}x}{\mathrm{d}q_a}},\ \widetilde{\frac{\mathrm{d}t}{\mathrm{d}q_a}} \right\rangle$
 3:     $\boldsymbol{\tau}_{\min} = \arg\min_{\boldsymbol{\tau}^{(i)}} \left\| p - p_0(\boldsymbol{\tau}^{(i)}) \right\|_{2W}, \quad \boldsymbol{\tau}^{(i)} \in T_{nom}$
 4:     $\boldsymbol{\tau}^{(r)} = \arg\min_{\boldsymbol{\tau}^{(ii)}} \left\| p - p_0(\boldsymbol{\tau}^{(ii)}) \right\|_{2W}, \quad \boldsymbol{\tau}^{(ii)} \in [\boldsymbol{\tau}^{(i)}; \boldsymbol{\tau}^{(i+1)}] \ \subset T_{dense}$
 5:     $p^{(r)} = p_0(\boldsymbol{\tau}^{(r)})$
 6:     $\Delta p = p - p^{(r)}$
 7:     $\widetilde{T}_u = \boldsymbol{\tau}^{(r)} \cup \{\tau^{(i)} \in T_u : \tau^{(i)} > \boldsymbol{\tau}^{(r)}\}$
 8:     $\widetilde{T}_x = \boldsymbol{\tau}^{(r)} \cup \{\tau^{(k)} \in T_x : \tau^{(k)} > \boldsymbol{\tau}^{(r)}\}$
 9:     $z(p^{(r)}) = \begin{pmatrix} \overset{\star}{u}_{(5.49)}(\tau^{(i)}, p_0), & \tau^{(i)} \in \widetilde{T}_u \\ \overset{\star}{x}_{(5.49)}(\tau^{(k)}, p_0), & \tau^{(k)} \in \widetilde{T}_x \\ \overset{\star}{t}_{(5.49)}(p_0) - \overset{\star}{t}_{(5.49)}(p_0)\boldsymbol{\tau}^{(r)} & \end{pmatrix}$     ▷ eval. sol. at $\widetilde{T}_u$,$\widetilde{T}_x$
10:     for each parameter $p^{(k)}$, $k = 1, \dots, n_p$ do
11:         for each control equation $u^{(j)}$, $j = 1, \dots, n_u$ do
12:             $\widetilde{\frac{\mathrm{d}z}{\mathrm{d}p}}(p^{(r)}) \leftarrow \widetilde{\frac{\mathrm{d}u^{(j)}}{\mathrm{d}p^{(k)}}}(\widetilde{T}_u, \boldsymbol{\tau}^{(r)})$     ▷ eval. sen. surf. at $(\widetilde{T}_u, p^{(r)})$
13:         end for
14:         for each state equation $x^{(j)}$, $j = 1, \dots, n_x$ do
15:             $\widetilde{\frac{\mathrm{d}z}{\mathrm{d}p}}(p^{(r)}) \leftarrow \widetilde{\frac{\mathrm{d}x^{(j)}}{\mathrm{d}p^{(k)}}}(\widetilde{T}_x, \boldsymbol{\tau}^{(r)})$     ▷ eval. sen. surf. at $(\widetilde{T}_x, p^{(r)})$
16:         end for
17:         $\frac{\mathrm{d}z}{\mathrm{d}p}(p^{(r)}) \leftarrow \widetilde{\frac{\mathrm{d}t}{\mathrm{d}p^{(k)}}}(\boldsymbol{\tau}^{(r)})$
18:     end for
19:     for each parameter $q_a^{(k)}$, $k = 1, \dots, n_a$ do
20:         for each control equation $u^{(j)}$, $j = 1, \dots, n_u$ do
21:             $\widetilde{\frac{\mathrm{d}z}{\mathrm{d}q_a}}(p^{(r)}) \leftarrow \widetilde{\frac{\mathrm{d}u^{(j)}}{\mathrm{d}q_a^{(k)}}}(\widetilde{T}_u, \boldsymbol{\tau}^{(r)})$
22:         end for
23:         for each state equation $x^{(j)}$, $j = 1, \dots, n_x$ do
24:             $\widetilde{\frac{\mathrm{d}z}{\mathrm{d}q_a}}(p^{(r)}) \leftarrow \widetilde{\frac{\mathrm{d}x^{(j)}}{\mathrm{d}q_a^{(k)}}}(\widetilde{T}_x, \boldsymbol{\tau}^{(r)})$
25:         end for
26:         $\frac{\mathrm{d}z}{\mathrm{d}q_a}(p^{(r)}) \leftarrow \widetilde{\frac{\mathrm{d}t}{\mathrm{d}q_a^{(k)}}}(\boldsymbol{\tau}^{(r)})$
27:     end for
28:     $\tilde{z}_1(p) = z(p^{(r)}) + \widetilde{\frac{\mathrm{d}z}{\mathrm{d}p}}(p^{(r)})\ \Delta p$     ▷ (5.13b)
29:     while $\left\| g_a(z^{[k]}, p)] \right\| > \varepsilon$ do
30:         $\tilde{z}^{[k+1]}(p) = \tilde{z}^{[k]}(p) + \widetilde{\frac{\mathrm{d}z}{\mathrm{d}q_a}}(p^{(r)})\ g_a(z^{[k]}, p)$     ▷ (5.17)
31:         if not convergence heuristic or $k > k_{\max}$ then     ▷ see Section 5.5.3
32:             error!
33:         end if
34:     end while
35:     $\tilde{u}(p) \leftarrow \tilde{z}_\varepsilon(p)$
36:     $\tilde{x}(p) \leftarrow$ integrate$[f(\tilde{u}(p); p, x)]$
37:     return $\tilde{u}(p), \tilde{x}(p)$
38: end procedure
```

CHAPTER 6

On-Board Trajectory Computation for Mars Atmospheric Entry

6.1 Mars Entry Reference Scenario

For the future exploration of Mars the European Space Agency (ESA) investigates Mars precision landing. Oriented at the ESA Mars precision landing study we consider a scenario which has the goal to deliver a rover to the surface of Mars, in close vicinity to already present mission assets, as e.g. required for the return of a Mars soil sample. The preliminary GNC accuracy requirement for this scenario is to achieve a 10 km radial error at parachute opening. This requires a guided atmospheric entry and the use of atmospheric maneuvering to precisely steer the vehicle from the entry interface to the parachute opening point.

The assumed kinematic conditions at the entry interface point correspond to a direct hyperbolic entry as described in MPL [Wol07]. The entry is characterized by a steep flight path angle of $-14.5°$. The touchdown landing site is assumed to be at sea level. The opening of the parachute is scheduled at an altitude of 10 000 m, in close vicinity to the landing site. The parachute opening may not occur at a higher dynamic pressure than 0.55 kPa, which corresponds to a Mach number of 2.2. An opening at lower dynamic pressure/Mach is desirable to reduce the strain on the parachute system. The entry interface and the parachute opening conditions are summarized in Table 6.1.

The cone shaped entry capsule has an average lift over drag ratio at hyper- and supersonic speed of approximately 0.2 at a trim angle of attack of $-12.5°$. The aerodynamic coefficients are obtained from the Mars Precision Lander aerodynamic database. Additional details of the vehicle model have been given in Section 2.3.

The maximal values for the path constraints on the heat flux $\dot{Q}$, the dynamic pressure d and the load factor n, as considered in the MSR mission [Wol07], are shown in Table 6.2.

The atmosphere model is obtained from the European Mars Climate Database 5.0. The nominal scenario is based on average yearly UV solar flux and average dust concentration.

Table 6.1: Entry interface point (EIP) and parachute opening conditions (POC)

EIP		Value	POC		Value
h_0	$=$	$120\,000\,\mathrm{m}$	h_f	$\geq$	$10\,000\,\mathrm{m}$
λ_0	$=$	$0°$	λ_f	$=$	$11.3°$
φ_0	$=$	$25°$	φ_f	$=$	$23.3°$
v_0	$=$	$5440.8\,\frac{\mathrm{m}}{\mathrm{s}}$	$d_{\max_f}$	$\leq$	$0.55\,\mathrm{kPa}$
γ_0	$=$	$-14.5°$			
χ_0	$=$	$97.4°$			

Table 6.2: Path constraints

Property		Limit
$\dot{Q}_{\max}$	$=$	$1600\,\frac{\mathrm{kW}}{\mathrm{m}^2}$
$d_{\max}$	$=$	$17\,\mathrm{kPa}$
$n_{\max}$	$=$	15

6.2 Formulation of the Optimal Control Problem

There are multiple ways to formulate the Mars entry scenario as optimal control problem. Our overall goal is to use the available degrees of freedom to reduce the sensitivity of the control function as much as possible, because this can translate into an increased convergence region of the RTS algorithm and thus increase the control envelop of the proposed guidance law. The available degrees of freedom are

1. the formulation of the boundary conditions,
2. the formulation of the objective function,
3. the choice of the perturbation parameters,
4. the choice of the independent variable.

6.2.1 Final Boundary Constraints

The final boundary constraints can either be enforced directly, or they can be considered as an objective, by minimizing the (weighted) quadratic distance to the target value in the objective function. If a terminal state constraint is treated as objective, a potentially active constraint is removed from the formulation. As a consequence the feasibility corrector iteration (5.17) cannot be used to achieve the satisfaction of this constraint. The transformation of a terminally constrained state into an objective is a trade-off between relaxing the optimality conditions (potentially resulting in lower sensitivities) and the ability to achieve the desired terminal state exactly.

Numerical analysis shows, that active terminal constraints on the altitude, longitude, latitude and velocity result into a *rigid* and *sensitive* problem, which is undesirable because it diminishes the correction space. Achieving a low velocity and a high altitude at the same time are competing objectives, because only a certain amount of the vehicle's mechanical energy can be dissipated. To relax the problem it can thus be advantageous to fix only either the terminal altitude or velocity and consider the other one as objective.

The relationship between the terminal altitude and velocity is constrained by the dynamic pressure limit of the parachute system. This can be expressed with the nonlinear boundary constraint

$$\frac{1}{2}\rho(h_f)v_f^2 - d_{\max_f} = 0 \tag{6.1}$$

We choose to simplify this constraint and approximate it using box constraints on the state[1]. Because of the competitive relationship between altitude and velocity, a constraint on the terminal altitude is active at the optimal solution, whenever the objective includes a minimization of the terminal velocity and vice versa. If an equality constraint is enforced on the terminal altitude, (6.1) can be solved for the terminal velocity, using the reference atmosphere $\rho_{\text{ref}}(h_f) > 0$.

$$v_{\max_f} = \sqrt{2\frac{d_{\max_f}}{\rho_{\text{ref}}(h_f)}} \tag{6.2}$$

At an altitude of 10 000 m, the critical velocity $v_{\max_f}$ evaluates to approximately 500 $\frac{\text{m}}{\text{s}}$; the nominal dynamic pressure would be achieved at 440 $\frac{\text{m}}{\text{s}}$. The linear constraints

$$h_f - 10000 = 0 \tag{6.2a}$$
$$v_f - 500 \quad \leq 0 \tag{6.2b}$$

ensure that the nonlinear dynamic pressure constraint is satisfied under nominal conditions. But because we do not capture the nonlinear relationship with the atmospheric density, the limit can be exceeded in a thicker atmosphere. This is prevented in the following by penalizing a high terminal velocity in the objective function.

6.2.2 Objective Function

The performance index is a linear combination of multiple objective terms:

In addition to constraining the terminal velocity (6.2b) it is desirable to incentivise $v_f \approx 440$ to achieve the nominal dynamic pressure for the parachute opening. The objective thus includes the term

$$J_V = (v_f - 440)^2. \tag{6.3}$$

The weight of this term is chosen such that we achieve $v_f \approx 440$ in the nominal solution. This strategy has proven to be sufficient to satisfy the terminal pressure constraint even for a strongly perturbed atmosphere.

During the high velocity phase of the entry, the feasible space is limited by the constraint on the maximal heat flux (2.26). It is desirable to minimize the maximal heat flux during this phase, to stay as far away from the constraint boundary as possible. This leads to a min-max problem, which is problematic to use with the real-time update scheme (cf. Section 5.5.4). An alternative that has a similar effect is to minimize the total square of the heat flux.

1 While theoretically (6.1) is only one nonlinear constraint among many, the numerical implementation of nonlinear boundary constraints increases the implementation effort.

$$J_H = \int_{t_s}^{t_f} \left(k_p \sqrt{\frac{\rho(h)}{r_n}} v^3 \right)^2 \mathrm{d}t \tag{6.4}$$

Minimizing the total heat flux itself (without taking the square) is often not beneficial towards a minimization of the peak heat flux.

The control authority of the entry capsule is determined by the generated lift, more specifically by the ability to alter the vertical lift to drag ratio. The control reserves would be maximized by planning the nominal trajectory for a flight at zero vertical lift, over the entire time. Then all lifting capability could be used to react to perturbations. Some lifting is required to regulate the sink rate to acceptable levels, but it is desirable to use a minimum of vertical lift.

$$J_L = \int_{t_s}^{t_f} \cos^2 \sigma \, \mathrm{d}t \tag{6.5}$$

This penalizes flight at full lift-up or lift-down and thus helps to prevent encountering the bank angle control singularities.

Lastly the objective can be used to shape the control function. A smooth control profile is desired, to enable an accurate and immediate realization of the commanded values by the attitude control system. Smoothing of the control function can be achieved by minimizing the integral over the squared control input, the so called control energy. The control input is chosen as the bank angle rate $\dot{\sigma}$, the bank angle σ is treated as an auxiliary state.

$$J_R = \int_{t_s}^{t_f} \dot{\sigma}^2 \, \mathrm{d}t \tag{6.6}$$

The objective function is a linear combination of the terms J_V, J_H, J_L, J_R. Each term is weighted with a factor $w_V, w_H, w_L, w_R \in \mathbb{R}_0^+$.

$$\min J = w_V J_V + w_H J_H + w_L J_L + w_R J_R \tag{6.7}$$

The weights are chosen such that all objective terms contribute roughly equally, with a slight priority on the heat-flux and vertical lift[1].

6.2.3 Perturbation Parametrization

We differentiate between two kinds of perturbations:

1. *Known perturbations* can be represented in the OCP by a mathematical description of their effect: the perturbation model.
2. *Unknown perturbations* are not represented in the OCP, because either their existence is not known a priori, or because their effect cannot be described mathematically.

Perturbations can either be detected by *direct measurement* using a sensor, or they can be

1 The objective function weights are available in Appendix G

estimated by a filter algorithm, on the basis of indirectly related sensor measurements. Note that if a perturbation cannot be measured or estimated at control time, it is effectively an unknown perturbation, even though a perturbation model might be available.

Known perturbations that can be measured or estimated are added to the parameter vector p and in the parametric sensitivity analysis we obtain the related sensitivity $\frac{\mathrm{d}z}{\mathrm{d}p}$. In the online phase the Taylor expansion (5.13b) directly adapts the solution to the known perturbations. The effects of unknown perturbations (and the remaining error of the Taylor expansion) is determined indirectly as the violation of the NLP constraint function. The solution is adapted to these unknown perturbations in the feasibility correction (5.17), by modeling the unknown perturbations as linear perturbation of the constraint function. For known perturbations the approximation error $\|\tilde{z}^{[\infty]}(p) - \overset{\star}{z}(p)\|$ is of the order $\|\Delta q\|^3$, while for unknown perturbations the approximation error increases to $\|\Delta q\|^2$ [Büs02].

In principle we are thus interested in modeling as many perturbations as possible. To be able to use $\Theta(p)$ it is clear that we parametrize the initial state:

$$x(0) = x_{s,0} + \Delta x_s = \begin{pmatrix} h_{s,0} \\ \lambda_{s,0} \\ \varphi_{s,0} \\ v_{s,0} \\ \gamma_{s,0} \\ \chi_{s,0} \end{pmatrix} + \begin{pmatrix} \Delta h_s \\ \Delta \lambda_s \\ \Delta \varphi_s \\ \Delta v_s \\ \Delta \gamma_s \\ \Delta \chi_s \end{pmatrix} . \tag{6.8}$$

In addition perturbations in the aerodynamic forces are most important, this includes perturbations of the temperature, the pressure, the atmospheric density, the aerodynamic coefficients, the trim angle of attack and the vehicle mass. All these perturbations affect the lift and drag accelerations:

$$L = \frac{1}{2}\rho(h)v^2 C_L(M(v,h))\frac{S}{m} \tag{6.9a}$$

$$D = \frac{1}{2}\rho(h)v^2 C_D(M(v,h))\frac{S}{m} \tag{6.9b}$$

One challenge of Mars atmospheric entry is the lack of direct measurements: In current entry systems only IMU data, i.e. the translational accelerations and rotational rates are available. The state is integrated using the measurements in an inertial strap-down integration, starting from the last known absolute position and orientation, which is typically briefly before reaching the entry interface point. The estimation error grows unboundedly with time. The dynamic system is thus not observable and most importantly the estimation error cannot be compensated by the guidance system.

The estimation of the perturbation parameters must also be based on inertial navigation. This is problematic, because there is no way to measure the contribution of any of the previously mentioned parameters to the sensed acceleration data, i.e. the error in each individual parameter is unknown. This problem can be elevated by a well tuned estimation algorithm, provided that accurate models and accurate knowledge of the parameter covariances are available. The development and tuning of a filter algorithm and the determination of the covariances is however not the focus of this work. We follow a different approach and parametrize the lift and drag acceleration directly:

$$L = \frac{1}{2}\rho(h)v^2 C_L(M)\frac{S}{m}\,(1+\mathit{\Delta}L), \tag{6.10a}$$

$$D = \frac{1}{2}\rho(h)v^2 C_D(M)\frac{S}{m}\,(1+\mathit{\Delta}D). \tag{6.10b}$$

The aerodynamics are treated as an unpredictable system, thus the perturbation model is chosen as a time invariant linear scaling. The accelerations L and D are obtained as measurements from the IMU. $\mathit{\Delta}L$ and $\mathit{\Delta}D$ are thus directly depending on the measurement:

$$\mathit{\Delta}L = L - L_r, \tag{6.11a}$$

$$\mathit{\Delta}D = D - D_r, \tag{6.11b}$$

where L_r and D_r are the acceleration profiles of the nominal open loop trajectory. For academic purposes we choose to also parametrize the mass m and the terminal position $h_f, \lambda_f, \varphi_f$. The parametrization of the terminal position opens the possibility to change the parachute opening point, by artificially introducing a perturbation. Although this is not demanded in the scenario, it has potential uses and does not require additional effort from a mathematical point of view.

The combined perturbation parameters for the sensitivity analysis of the nominal closed loop are thus

$$p(t) = (\bar{p},\, x_s) = (L,D,m,h_s,\lambda_s,\varphi_s,v_s,\gamma_s,\chi_s,h_f,\lambda_f,\varphi_f). \tag{6.12a}$$

6.2.4 Time Domain Problem Statement

The entry problem can now be formulated as parametric optimal control problem. The formulation uses the extended state vector $x \in \mathbb{R}^7$,

$$x = (h, \lambda, \varphi, v, \gamma, \chi, \sigma), \tag{6.13}$$

and the control vector $u \in \mathbb{R}$,

$$u = \dot{\sigma}. \tag{6.14}$$

Using the models of Section 2, the problem can be written in compact, normalized form:

Problem 6.15 (Mars Entry (time dynamics, non-rotating planet))

$$\begin{aligned}\min_{u,t_d}\ J(x(\tau),u(\tau),t_d,p) &= w_V(v_f-430)^2 \\ &\quad + w_H t_d \int_0^1 \left(k_p\sqrt{\frac{\rho(h)}{r_n}}v^3\right)^2 + w_L\left(\cos\sigma\right)^2 + w_R\left(\dot{\sigma}\right)^2 \,\mathrm{d}\tau\end{aligned} \tag{6.16a}$$

$$\text{s.t.} \qquad \dot{x} = t_d \begin{pmatrix} v\sin\gamma \\ v\frac{\cos\gamma\sin\chi}{r\cos\varphi} \\ v\frac{\cos\gamma\cos\chi}{r} \\ -D-g\sin\gamma \\ \frac{L}{v}\cos\sigma + \left(\frac{v}{r}-\frac{g}{v}\right)\cos\gamma \\ \frac{L\sin\sigma}{v\cos\gamma} + \frac{v}{r}\cos\gamma\sin\chi\tan\varphi \\ \dot{\sigma} \end{pmatrix} \tag{6.16b}$$

$$\psi_s(x(0),p) = \begin{pmatrix} 120000 \\ 0 \\ 25 \\ 5440.8 \\ -14.5 \\ 97.4 \end{pmatrix} + \begin{pmatrix} \Delta h_0 \\ \Delta\lambda_0 \\ \Delta\varphi_0 \\ \Delta v_0 \\ \Delta\gamma_0 \\ \Delta\chi_0 \end{pmatrix} - \begin{pmatrix} x^{(1)}(0) \\ x^{(2)}(0) \\ x^{(3)}(0) \\ x^{(4)}(0) \\ x^{(5)}(0) \\ x^{(6)}(0) \end{pmatrix} = 0 \tag{6.16c}$$

$$\psi_f(x(1),p) = \begin{pmatrix} 10000 \\ 11.3 \\ 23.3 \\ 450 \end{pmatrix} + \begin{pmatrix} \Delta h_f \\ \Delta\lambda_f \\ \Delta\varphi_f \\ 0 \end{pmatrix} - \begin{pmatrix} x^{(1)}(1) \\ x^{(2)}(1) \\ x^{(3)}(1) \\ x^{(4)}(1) \end{pmatrix} \begin{matrix} = \\ = \\ = \\ \leq \end{matrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{6.16d}$$

$$c(x(\tau),u(\tau),p) = \begin{pmatrix} k_p\frac{\sqrt{\rho(h)}}{1000r_n}v^3 \\ \frac{1}{2000}\rho(h)v^2 \\ \frac{\sqrt{D^2+L^2}}{g_E} \end{pmatrix} \leq \begin{pmatrix} 1600 \\ 17 \\ 15 \end{pmatrix} \tag{6.16e}$$

$$\text{with} \qquad L = L_r(1+\Delta L)\frac{1050}{1050+\Delta m} \tag{6.16f}$$

$$D = D_r(1+\Delta D)\frac{1050}{1050+\Delta m}. \tag{6.16g}$$

6.2.5 Energy Domain Problem Statement

Time can be replaced as independent variable by energy as illustrated in Section 2.5.2. Energy is a function of altitude and velocity, i.e. time is substituted for a relationship between two states. This opens a new degree of freedom and potentially reduces the sensitivity of the system against perturbations in these states.

The transformed optimal control process is discretized with respect to energy; discretization points correspond to energy levels and the state trajectory and the control are functions of energy. Let $\tau_e \in [0;\, 1]$ be the normalized energy over the domain $[E_s;\, E_f]$, and let $E_d = E_f - E_s$. The normalized energy dynamics are

$$x'(\tau_e) = E_d \begin{pmatrix} -\frac{1}{D}\sin\gamma \\ -\frac{1}{D}\frac{\cos\gamma\sin\chi}{r\cos\varphi} \\ -\frac{1}{D}\frac{\cos\gamma\cos\chi}{r} \\ \frac{D+g\sin\gamma}{Dv} \\ -\frac{L}{D}\frac{\cos\sigma}{v^2} - \left(\frac{v}{r} - \frac{g}{v}\right)\frac{\cos\gamma}{Dv} \\ -\frac{L}{D}\frac{\sin\sigma}{v^2\cos\gamma} - \frac{1}{Dr}\cos\gamma\sin\chi\tan\varphi \\ -\frac{\dot{\sigma}}{vD}. \end{pmatrix} \tag{6.17}$$

Because the problem is discretized with respect to energy, the objective function and in particular the Lagrange term are defined with respect to energy. While mathematically this is of no concern, it is a change of the optimization goal, which is not desired. Moreover the physical interpretation of the Lagrange terms with respect to energy is unclear (e.g. the energy integral over the heat-flux).

Our goal is a comparison of the guidance and control performance based on the time and the energy domain formulation. Thus we want to keep the optimization goal identical in both formulations. This requires to integrate the Lagrange term with respect to time, while the problem is discretized with respect to energy. The transformation from time to energy is explicitly given by (2.20), but the inverse function cannot be obtained explicitly. We can however use the differential relationship $\dot{E} = -vD$ (cf. (2.21)) to write the process duration as

$$\begin{aligned} t_d = t_f - t_s &= -\int_{E_s}^{E_f} \frac{1}{v(E)D(E)}\,\mathrm{d}E = \int_{E_f}^{E_s} \frac{1}{v(E)D(E)}\,\mathrm{d}E \\ &\approx 0.5\sum_{i=1}^{l_u-1} \frac{1}{v(E_{i+1})D(E_{i+1})} + \frac{1}{v(E_i)D(E_i)} \end{aligned} \tag{6.18}$$

with $E_s = E(t_s)$ and $E_f = E(t_f)$. The time that passes in the interval $\Delta E_i = E_{i+1} - E_i$ is

$$\begin{aligned} \Delta t_i &\approx 0.5\left(\frac{1}{v(E_{i+1})D(E_{i+1})} + \frac{1}{v(E_i)D(E_i)}\right) \\ &= 0.5\left(\frac{1}{v(\tau_e^{(i+1)})D(\tau_e^{(i+1)})} + \frac{1}{v(\tau_e^{(i)})D(\tau_e^{(i)})}\right)(E_f - E_s)(\tau_e^{(i+1)} - \tau_e^{(i)}) \end{aligned} \tag{6.19}$$

where $\tau_e^{(i)} \in [0;\, 1]$, $1 \le i \le l_u - 1$ are the discretization points of the normalized energy.

Relation (6.19) allows to approximate the time integral over a function of normalized energy $f(\tau_e) : [0;\, 1] \to \mathbb{R}$ as

$$\int_{t_s}^{t_f} f(\tau_e)\, dt \approx 0.5 \sum_{i=1}^{l_u - 1} \Delta t_i \left(f(\tau_e^{(i+1)}) + f(\tau_e^{(i)}) \right). \tag{6.20}$$

This allows to solve the following optimal control process:

Problem 6.21 (Mars Entry (energy dynamics, non-rotating planet))

$$\min_{u,t_d} J(x(\tau_e),u(\tau_e),E_l,p) = w_V (v_f - 430)^2 \tag{6.22a}$$

$$+ \int_0^{t_f} w_H \left(k_p \sqrt{\frac{\rho(h)}{r_n}} v^3 \right)^2 + w_L \left(\cos\sigma\right)^2 + w_R \left(\dot\sigma\right)^2 \,\mathrm{d}t$$

$$\text{s.t.} \qquad x'(\tau_e) = E_l \begin{pmatrix} -\frac{1}{D}\sin\gamma \\ -\frac{1}{D}\frac{\cos\gamma\sin\chi}{r\cos\varphi} \\ -\frac{1}{D}\frac{\cos\gamma\cos\chi}{r} \\ \frac{D+g\sin\gamma}{Dv} \\ -\frac{L}{D}\frac{\cos\sigma}{v^2} - \left(\frac{v}{r} - \frac{g}{v}\right)\frac{\cos\gamma}{Dv} \\ -\frac{L}{D}\frac{\sin\sigma}{v^2\cos\gamma} - \frac{1}{Dr}\cos\gamma\sin\chi\tan\varphi \\ -\frac{\dot\sigma}{vD} \end{pmatrix} \tag{6.22b}$$

$$\psi_s(x(0),p) = \begin{pmatrix} 120000 \\ 0 \\ 25 \\ 5440.8 \\ -14.5 \\ 97.4 \end{pmatrix} + \begin{pmatrix} \Delta h_0 \\ \Delta\lambda_0 \\ \Delta\varphi_0 \\ \Delta v_0 \\ \Delta\gamma_0 \\ \Delta\chi_0 \end{pmatrix} - \begin{pmatrix} x^{(1)}(0) \\ x^{(2)}(0) \\ x^{(3)}(0) \\ x^{(4)}(0) \\ x^{(5)}(0) \\ x^{(6)}(0) \end{pmatrix} = 0 \tag{6.22c}$$

$$\psi_f(x(1),p) = \begin{pmatrix} 10000 \\ 11.3 \\ 23.3 \\ 450 \end{pmatrix} + \begin{pmatrix} \Delta h_f \\ \Delta\lambda_f \\ \Delta\varphi_f \\ 0 \end{pmatrix} - \begin{pmatrix} x^{(1)}(1) \\ x^{(2)}(1) \\ x^{(3)}(1) \\ x^{(4)}(1) \end{pmatrix} \begin{matrix} = \\ = \\ = \\ \leq \end{matrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{6.22d}$$

$$c(x(\tau_e),u(\tau_e),p) = \begin{pmatrix} k_p \frac{\sqrt{\rho(h)}}{1000 r_n} v^3 \\ \frac{1}{2000}\rho(h) v^2 \\ \frac{\sqrt{D^2+L^2}}{g_E} \end{pmatrix} \leq \begin{pmatrix} 1600 \\ 17 \\ 15 \end{pmatrix} \tag{6.22e}$$

$\mathrm{OCP}_{(6.15)}(p)$ and $\mathrm{OCP}_{(6.21)}(p)$ have identical objective functions, boundary conditions and path constraints. Their dynamics are related through a strongly nonlinear state transformation, which changes the dependencies within the dynamics through substitution of time with a relationship between altitude and velocity. In the next sections we compare the solutions and the parametric sensitivities of both formulations to determine the effects of this transformation and to conclude if either offers an advantage over the other for the purpose of sensitivity based real-time solution approximation.

6.3 Problem Analysis

$\mathrm{OCP}_{(6.15)}(p)$ and $\mathrm{OCP}_{(6.21)}(p)$ are analyzed offline using Algorithm (2). Figure 6.1 recalls the offline preparation phase schematically.

The problems are discretized with the standard Runge-Kutta (RK4) method. The control function is linearly interpolated. Identical objective weights are used for both problems (see Appendix G).

All first- and second order derivatives are computed quasi-analytically using automatic differentiation, which allows to determine the true sparsity patterns and to compute the derivatives accurately up to machine precision. The numerical accuracy for constraint satisfaction and optimality is 10^{-12}.

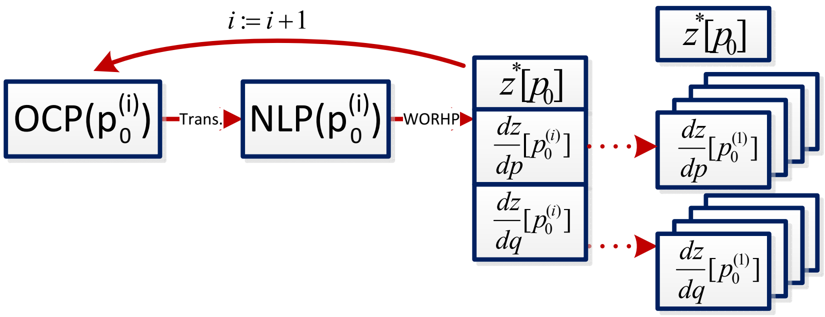

$i := i+1$
$\mathrm{OCP}(p_0^{(i)})$
Trans.
$\mathrm{NLP}(p_0^{(i)})$
WORHP
$\overset{*}{z}[p_0]$
$\frac{dz}{dp}[p_0^{(i)}]$
$\frac{dz}{dq}[p_0^{(i)}]$
$\overset{*}{z}[p_0]$
$\frac{dz}{dp}[p_0^{(1)}]$
$\frac{dz}{dq}[p_0^{(1)}]$

Figure 6.1: Preparation of the nominal solution and the sensitivity catalog.

6.3.1 Nominal Solution

To compare the nominal open loop solutions of $\mathrm{OCP}_{(6.15)}(p)$ and $\mathrm{OCP}_{(6.21)}(p)$, the time domain solution is transformed to the energy domain and vice versa. Consider Figure 6.2 which shows two pairs of graphs with the following coloring:

Dark blue: Nominal solution of (6.15) in normalized time.

Cyan: Nominal solution of (6.15) in normalized energy.

Red: Nominal solution of (6.21) in normalized energy.

Magenta: Nominal solution of (6.21) in normalized time.

Figure 6.2a shows the optimal control function $\dot{\sigma}$, Figure 6.2b shows the bank angle σ. The expectation that both problems have an identical optimal solution is fulfilled. The optimal controls of both problems are identical to a relative accuracy of 10^{-2} and the optimal objective function values are identical to a relative accuracy of 10^{-3}. This degree of equivalence can likewise be achieved with single shooting and full discretization.

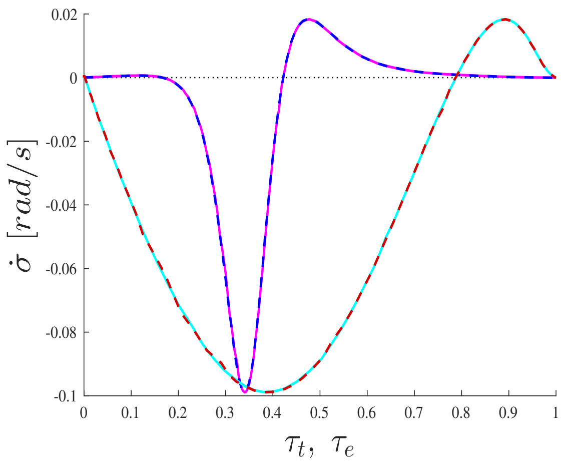
Image 6.2c and 6.2d illustrate the effect of the change of the independent variable. While the energy loss is rapid and strongly nonlinear in the time domain, this nonlinearity is removed by the transformation. In the energy domain the energy function E is the identity and $\dot{E}$ is roughly a quadratic function. Whereas in the time domain $\dot{E}$ is a higher order function, that is very shallow in the beginning and the end, and steep in between. The transformation to the energy domain is thus a strong compression at the beginning and the end, and a relaxation in between (cf. Figure 6.2e).

Using the transformed control grid of $\text{OCP}_{(6.15)}(p_0)$ for $\text{OCP}_{(6.21)}(p_0)$ leads to an increased discretization error and vice versa[1]. Thus $\text{OCP}_{(6.15)}(p_0)$ and $\text{OCP}_{(6.21)}(p_0)$ are discretized on different control grids[2] to achieve the same order of global discretization error, while minimizing the number of required discretization points by equally distributing the local discretization error as suggested by Büskens [Büs98].

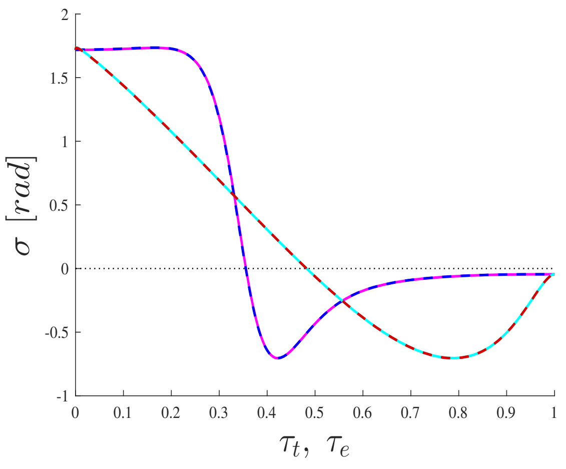
Figure 6.3 shows the optimal open loop state trajectories of both solutions. Note that in 6.3d the velocity increases until $\tau_t \approx 0.25$ (respectively $\tau_e \approx 0.015$), i.e. in this phase the kinetic energy is increasing. Figure 6.3a shows a rapid decrease of the altitude in the same interval, i.e. a reduction of the potential energy. The total amount of energy lost in this phase is very small, such that in the energy domain this entire dynamic is compressed into the interval $[0; 0.02]_e$. As a consequence the energy domain dynamics has stiff characteristics in this region, requiring a fine discretization. We will come back to this interval of rapid altitude loss when analyzing the sensitivities and the correction space. The phase of rapid deceleration $[0.25; 0.5]_t$ is stretched to $[0.02; 0.9]_e$.

Figure 6.4 shows the nominal aerodynamic acceleration profiles and the related path constraints. The path constraints do not become active, and there is ample margin between the maximal constraint values and the boundaries, which is beneficial towards the intended approach. Further tests have shown, that the path constraints will not become active, even under the worst expected conditions. Especially the margin on the heat flux is generous. If detailed heating models confirm the estimate of the Sutton-Graves method (2.26), it could be feasible to use a less resistant, but lighter heat shield, which could result in an increase of the payload mass.

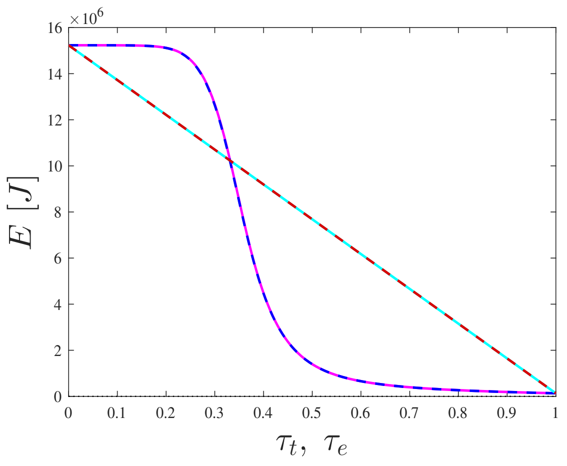
1 The global and local discretization errors are estimated using the Runge-Kutta-Fehlberg (RK45) scheme.
2 $\text{OCP}_{(6.15)}(p_0)$ uses 122 discretization points; $\text{OCP}_{(6.21)}(p_0)$ uses 127 discretization points.



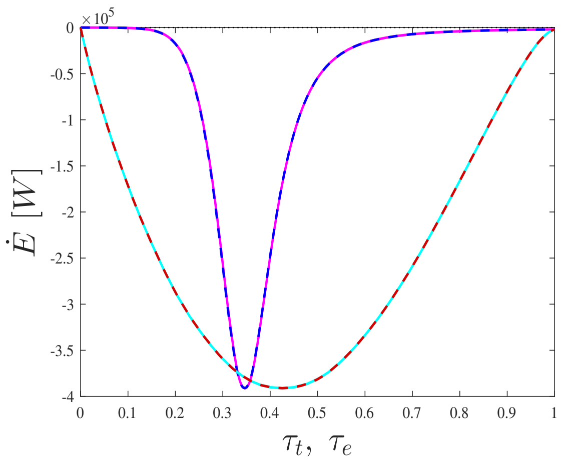
(a) bank angle time derivative



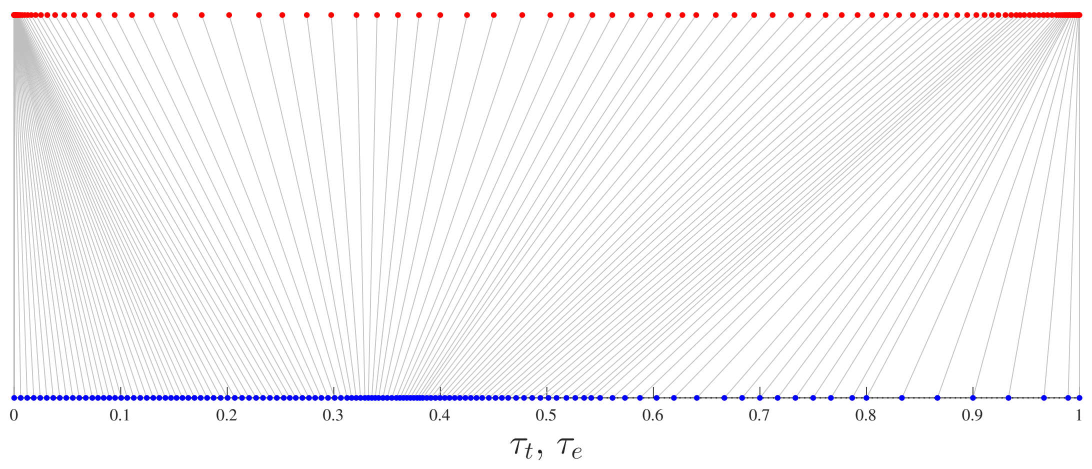
(b) bank angle



(c) energy

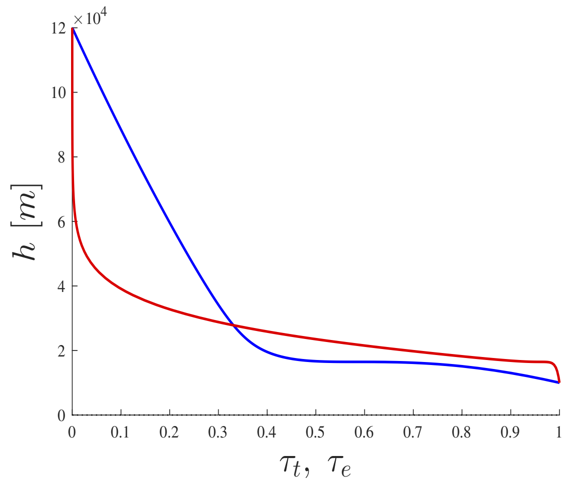


(d) energy time derivative

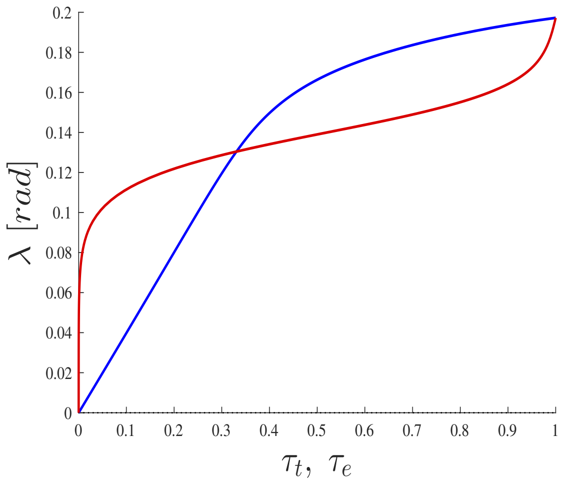


(e) distribution and mapping of discretization points

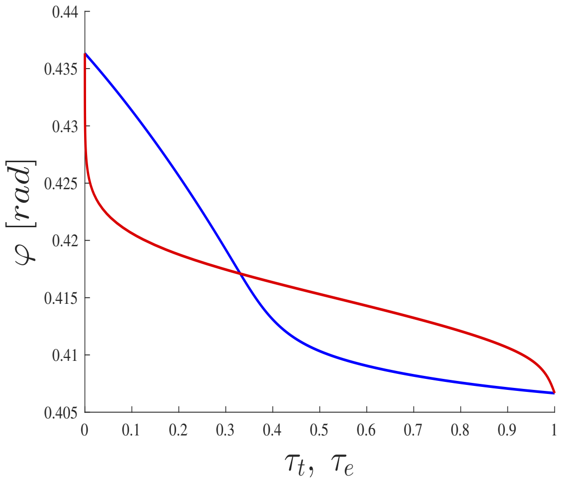
Figure 6.2: Subplots (a) and (b) show the nominal optimal control function for the problems (6.15) and (6.21) in the normalized time and energy domain. Subplots (c) and (d) show the energy and the energy derivative. Subplot (e) shows the transformation between the time based discretization (blue) to the energy based discretization (red). The grey lines indicate the corresponding transformed point.



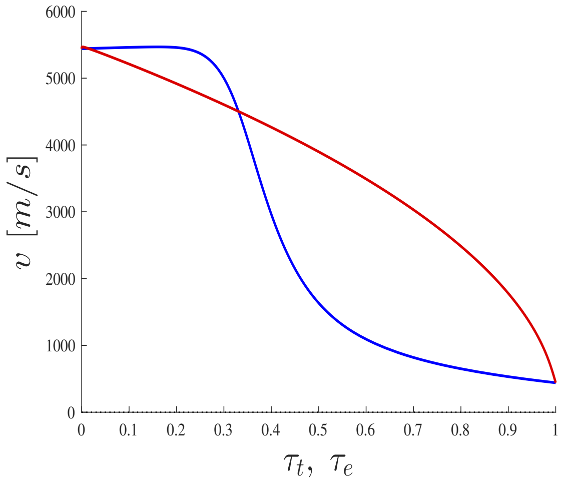
(a) altitude



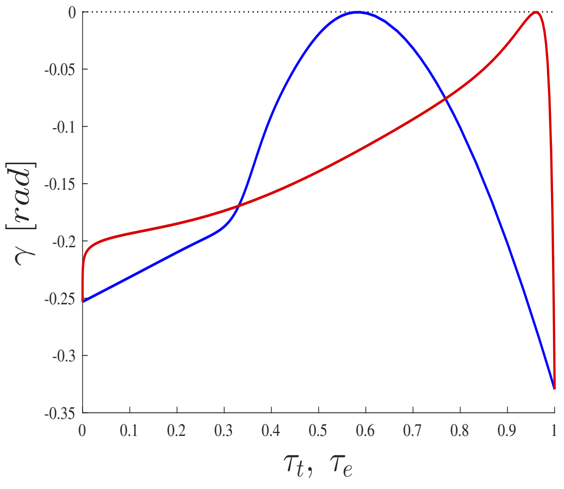
(b) longitude



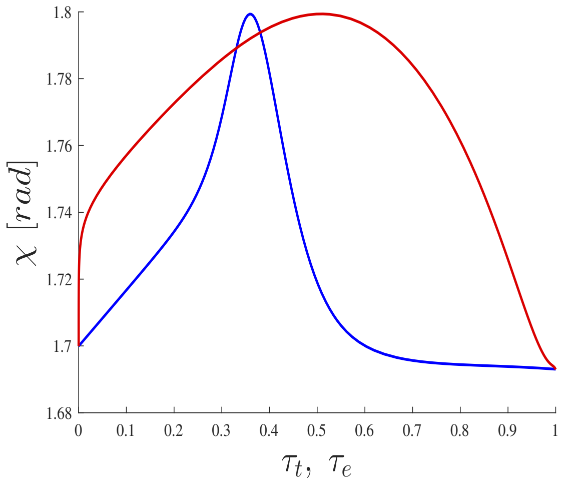
(c) latitude



(d) velocity

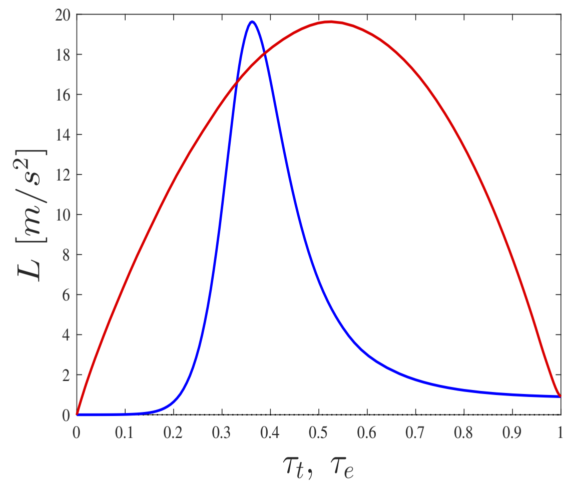


(e) flight path angle

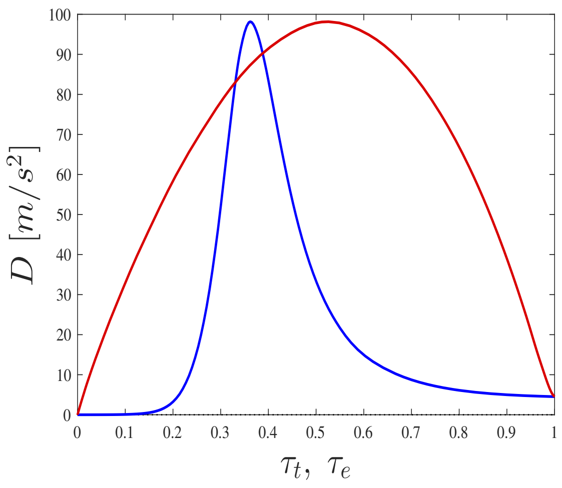


(f) heading angle

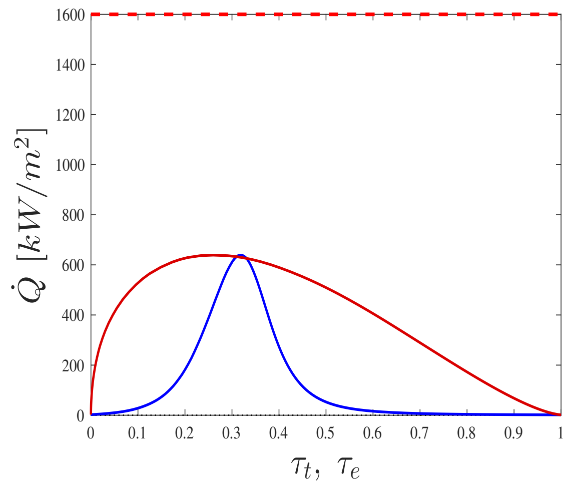
Figure 6.3: Nominal optimal state trajectories of the problems (6.15) and (6.21) in normalized time (blue) and energy (red).



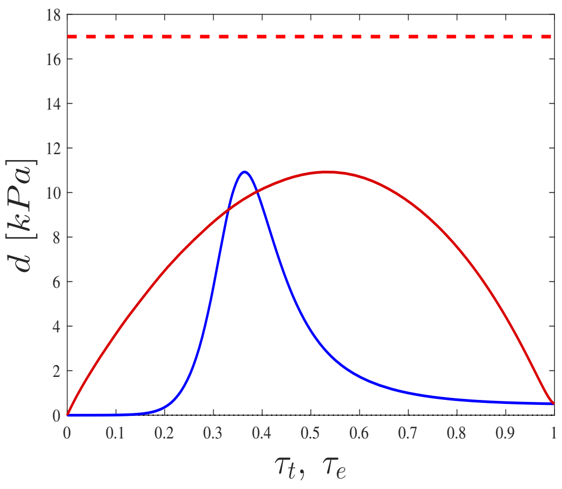
(a) lift acceleration



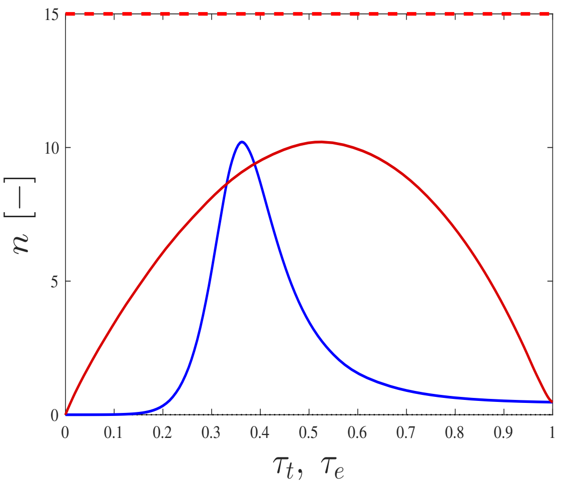
(b) drag acceleration



(c) heat flux



(d) dynamic pressure



(e) load factor

Figure 6.4: The aerodynamic accelerations and path constraints for the nominal solution of problems (6.15) and (6.21) plotted against normalized time (blue) and normalized energy (red). The boundary (dashed red) is identical for both problems.

6.3.2 Parametric Sensitivity Analysis

Parametric Sensitivity of the Open Loop Solution

In the previous section we verified that $\mathrm{OCP}_{(6.15)}(p_0)$ and $\mathrm{OCP}_{(6.21)}(p_0)$ have equivalent optimal solutions. Now we compare the parametric sensitivities of their optimal control functions. Therefore the sensitivities of $\mathrm{OCP}_{(6.21)}(p_0)$ are transformed into the time domain. In addition we compare the sensitivities of different state discretizations of $\mathrm{OCP}_{(6.15)}(p_0)$. All figures in this section contain six graphs:

Green:	$\mathrm{OCP}_{(6.15)}(p_0)$; single shooting.
Dark blue:	$\mathrm{OCP}_{(6.15)}(p_0)$; multiple shooting with 2 equally long shooting intervals.
Light blue:	$\mathrm{OCP}_{(6.15)}(p_0)$; multiple shooting with 3 equally long shooting intervals.
Cyan:	$\mathrm{OCP}_{(6.15)}(p_0)$; multiple shooting with 4 equally long shooting intervals.
Red:	$\mathrm{OCP}_{(6.15)}(p_0)$; full discretization.
Magenta:	$\mathrm{OCP}_{(6.21)}(p_0)$; full discretization; the sensitivities have been transformed to normalized time for these plots.

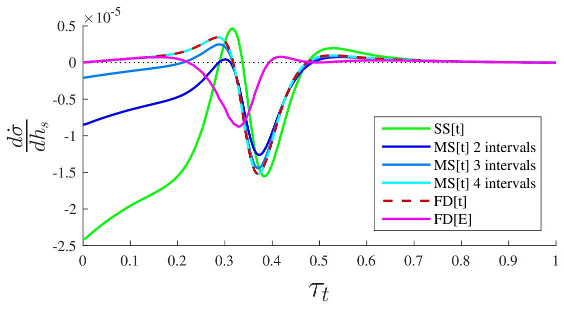
The entry problem shows the same relationships between the state discretization and the sensitivities as the analysis of the rocket car problem: Different state discretizations lead to different sensitivity differentials. As clearly shown in Figures 6.5a, d, e, f, increasing the number of shooting intervals, i.e. cutting off the direct dependency on the control variables, leads to a decrease of the sensitivity of the initial control variables. This confirms the empirical perception that multiple shooting decreases the sensitivity with respect to the initial condition. The reduced control sensitivity must however be put in context with the additionally gained sensitivities of the states. In particular the real-time approximation scheme only leads to correct results, if the complete set of sensitivities of the entire solution is used to approximate the perturbed optimum.

It is not surprising that also the sensitivities of the time domain formulation $\mathrm{OCP}_{(6.15)}(p_0)$ and the energy domain formulation $\mathrm{OCP}_{(6.21)}(p_0)$ differ. The differences include changes in magnitude, shifts and curvature structure (cf. Figure 6.5b and Figure 6.6b). It remains to be seen, if these differences amount to any significant advantages or disadvantages for the purpose of real-time solution approximation.

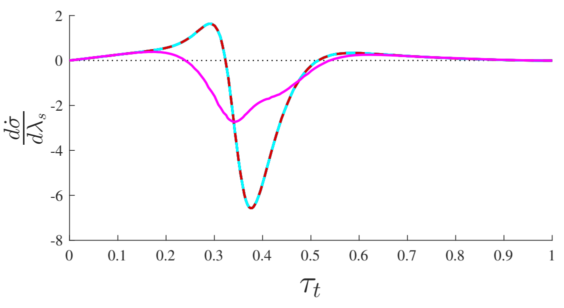
An NLP formulation that achieves an overall reduction of the sensitivity magnitude is likely to result into a larger correction space of the RTS algorithm. Obviously multiple shooting reduces the sensitivity of $\dot{\sigma}(\boldsymbol{\tau})$ against perturbations in h_s, v_s and γ_s compared to single shooting. Unfortunately the transition from the time domain to the energy domain does not achieve a further general reduction of the sensitivity. While the sensitivity against perturbations in h_s, λ_s, γ_s and λ_f is reduced, the sensitivity against perturbations in $\varphi_s, \chi_s, \varphi_f$ and L is increased. This indicates a shift of the sensitive relationships rather than a segmentation or redistribution.

The absolute value of the control sensitivity is highest for $\tau_t \in [0.2; 0.6]$. The absolute minima and maxima occur roughly for $\tau_t \in [0.3; 4.5]$. By comparison with Figure 6.4a and 6.4b it is found, that this exactly matches the interval, in which the aerodynamic forces are high. From the equations of motion we know that the lifting capability of the vehicle is directly related to the control authority. The majority of the optimal perturbation adaption thus happens, when the control authority is high.

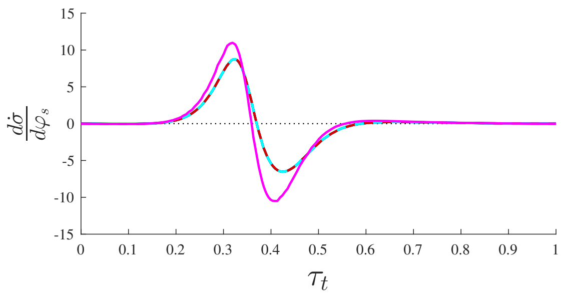
In the following we focus on the parametric sensitivities of the full discretization transcrip-



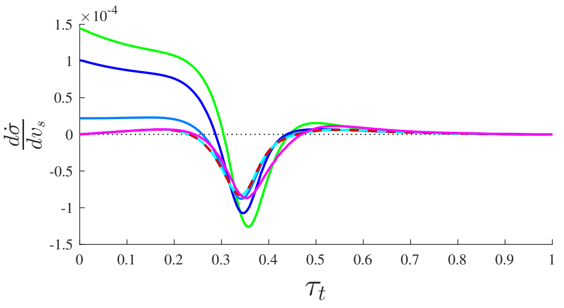
(a) Initial altitude



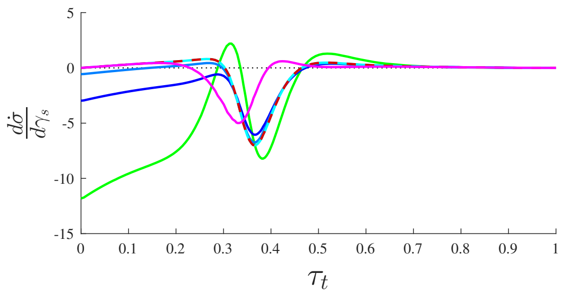
(b) Initial longitude



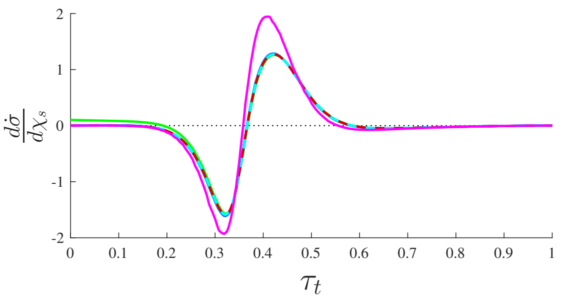
(c) Initial latitude



(d) Initial velocity



(e) Initial flight path angle



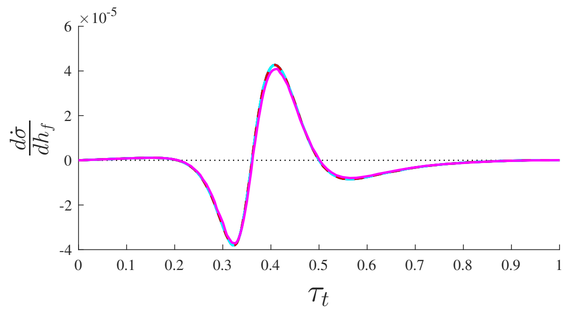
(f) Initial heading angle

Figure 6.5: Parametric sensitivity of the nominal control function $\dot{\sigma}$ against perturbations in the initial states.

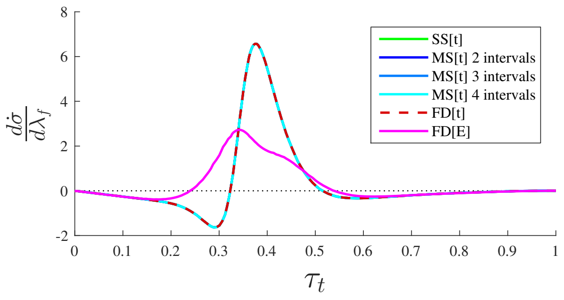
tion. We are interested in identifying the perturbations which are most problematic to compensate. Because the perturbations use different units and operate on different scales, the magnitude of the sensitivity alone is not of significance. Important is the magnitude of the sensitivity in relation with the expected perturbations. We define the impact $I(p^{(j)})$ of a scalar parameter $p^{(j)}$ as the maximal point-wise change of the optimal control function induced by the Taylor approximation of the optimal solution. The impact of parameter $p^{(j)}$ is thus

$$I(p^{(j)}) := |p^{(j)} - p_0^{(j)}| \max_{\tau} \left| \frac{\mathrm{d}\dot{\sigma}}{\mathrm{d}p^{(j)}}(\tau) \right|, \quad \tau \in T_u. \tag{6.23}$$

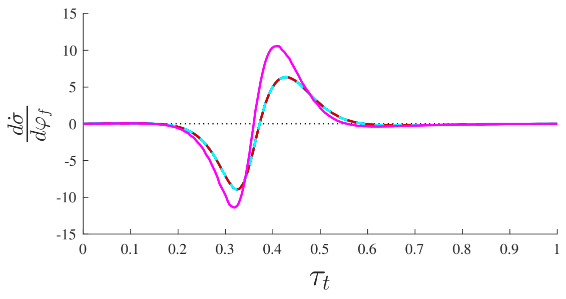
The expected perturbations and their impact are shown in Table 6.3. The state perturbation that we expect at the EIP is determined by the delivery accuracy of the previous mission leg [Wol07]. Concerning the lift and drag the strongest perturbations are expected from



(a) Final altitude

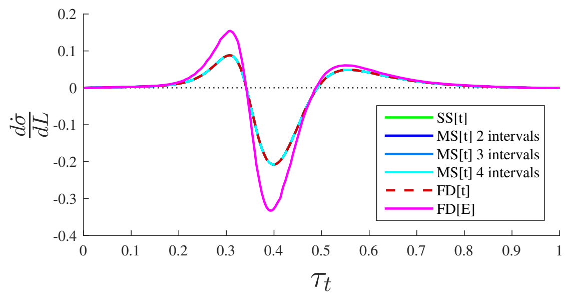


(b) Final longitude

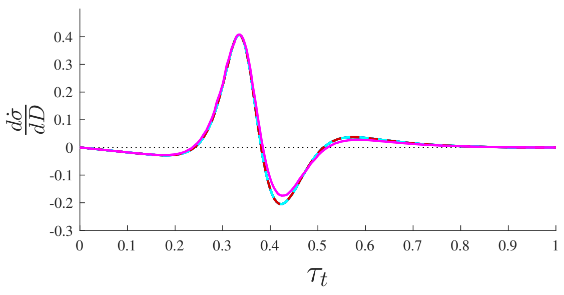


(c) Final latitude

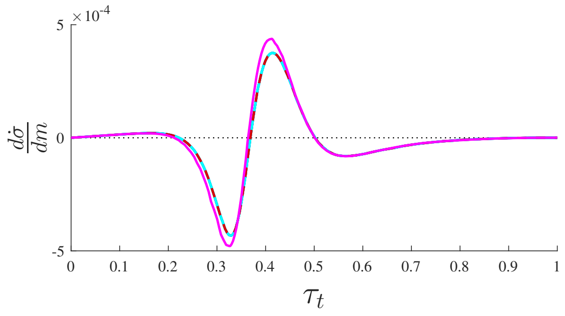
Figure 6.6: Parametric sensitivity of the nominal control function $\dot{\sigma}$ against perturbations in the final states.



(a) Lift



(b) Drag



(c) Mass

Figure 6.7: Parametric sensitivity of the nominal control function $\dot{\sigma}$ against perturbations in the lift (a), drag (b) and mass (c).

the atmospheric density and the aerodynamic coefficients. The aerodynamic coefficients are assumed to be accurate to 5 %. For the atmospheric density we distinguish between local and global perturbation. While local perturbations of up to 100 % are possible, the average perturbation is much smaller. The nominal atmosphere model already uses an atmospheric density profile, that is close to the lower limit. Reducing the density by more than ≈5 % results into an infeasible problem. For a warmer atmosphere we assume a worst case of 25 % average density increase. The perturbation of the aerodynamic coefficients and the density have an additive effect.

The expected perturbations of the aerodynamic accelerations in case of a denser atmosphere have the strongest impact on the control function. Among the initial states the perturbation of the flight path angle has the highest impact in both problem formulations, followed by altitude, longitude and latitude. These findings are in line with our expectations based on literature results. It remains to check, if the controllability space covers the expected perturbations.

Table 6.3: Perturbation impact estimate

Perturbation	**Expected at EIP**	**Time Domain**		**Energy Domain**	
		$\max \left\lvert \frac{\mathrm{d}\hat{\sigma}}{\mathrm{d}p^{(j)}} \right\rvert$	$I(p^{(j)})$	$\max \left\lvert \frac{\mathrm{d}\hat{\sigma}}{\mathrm{d}p^{(j)}} \right\rvert$	$I(p^{(j)})$
ΔL	−0.1 to 0.3	$2.07 \cdot 10^{-1}$	$6.24 \cdot 10^{-2}$	$3.32 \cdot 10^{-1}$	$9.98 \cdot 10^{-2}$
ΔD	−0.1 to 0.3	$4.07 \cdot 10^{-1}$	$1.22 \cdot 10^{-1}$	$4.07 \cdot 10^{-1}$	$1.22 \cdot 10^{-1}$
Δm	-	$4.32 \cdot 10^{-1}$	-	$4.78 \cdot 10^{-4}$	-
Δh_s	$1.2 \cdot 10^{3}\,\mathrm{m}$	$1.51 \cdot 10^{-5}$	$1.83 \cdot 10^{-2}$	$8.74 \cdot 10^{-6}$	$1.05 \cdot 10^{-2}$
$\Delta \lambda_s$	$2.87 \cdot 10^{-3}\,\mathrm{rad}$	6.57	$1.89 \cdot 10^{-2}$	2.72	$7.83 \cdot 10^{-3}$
$\Delta \varphi_s$	$8.63 \cdot 10^{-4}\,\mathrm{rad}$	8.74	$7.55 \cdot 10^{-3}$	10.98	$9.48 \cdot 10^{-3}$
Δv_s	$0.8\,\frac{\mathrm{m}}{\mathrm{s}}$	$8.27 \cdot 10^{-5}$	$6.64 \cdot 10^{-5}$	$8.67 \cdot 10^{-5}$	$6.95 \cdot 10^{-5}$
$\Delta \gamma_s$	$3.43 \cdot 10^{-3}\,\mathrm{rad}$	7.01	$2.41 \cdot 10^{-2}$	5.99	$1.71 \cdot 10^{-2}$
$\Delta \chi_s$	$4.32 \cdot 10^{-3}\,\mathrm{rad}$	1.58	$6.87 \cdot 10^{-3}$	1.94	$8.41 \cdot 10^{-3}$
Δh_f	-	$4.27 \cdot 10^{-5}$	-	$4.09 \cdot 10^{-5}$	-
$\Delta \lambda_f$	-	6.57	-	2.72	-
$\Delta \varphi_f$	-	8.93	-	11.40	-

Parametric Sensitivities For Closed Loop Control

For the application of $\Theta(p)$ (Algorithm 3) we compute the parametric sensitivity differentials for the control problems with the initial condition given by evaluating the nominal open loop trajectory $\overset{\star}{x}$ at the points $\boldsymbol{\tau}^{(i)} \in T_{nom}$. In the interest of keeping the plots readable, the images only show the sensitivity functions with respect to initial conditions of the first half of the trajectory, i.e. $\boldsymbol{\tau}^{(i)} \in T_{nom} \wedge \boldsymbol{\tau}^{(i)} \leq 0.5$.

$$\psi_s^{(i)}(x(0),p_0^{(i)}) := \overset{\star}{x}(\boldsymbol{\tau}^{(i)};\bar{p}_0) + \Delta x_s^{(i)}, \quad p_0^{(i)} := \begin{pmatrix} \bar{p}_0 \\ \overset{\star}{x}(\boldsymbol{\tau}^{(i)};\bar{p}_0) \end{pmatrix} \tag{6.24}$$

The sensitivity functions are represented on a grid of the normalized nominal open loop time. The sensitivities of the control problem with initial condition $\psi_s^{(i)}(x(0),p)$ are transformed from problem specific time to the normalized nominal open loop time $\boldsymbol{\tau}$ on the interval $[\boldsymbol{\tau}^{(i)}; 1]$.

Figure 6.8 shows the sensitivity interpolation based on 60 OCPs using a full discretization transcription. The left column shows the sensitivity functions obtained for the sub-problems. The right column shows the surface fits based on two dimensional cubic spline interpolation. The interpolation along the x-axis is with respect to the discretization of the control or state function. The interpolation along the y-axis is with respect to the initial condition. To distinguish between the interpolation directions the interpolation variable for the initial condition (y-axis) is denoted as $\boldsymbol{\tau}_s \in [0; 1]$. The color of the surface indicates the sensitivity magnitude to improve the plasticity of the image.

The initial and final conditions enforce some simple relations on the sensitivity surfaces, that can be used for plausibility checks:

1. The sensitivity of a state against a perturbation in itself at the current time is one. Example: Figure 6.8c and d show the sensitivity of the altitude against a perturbation in altitude. It holds that

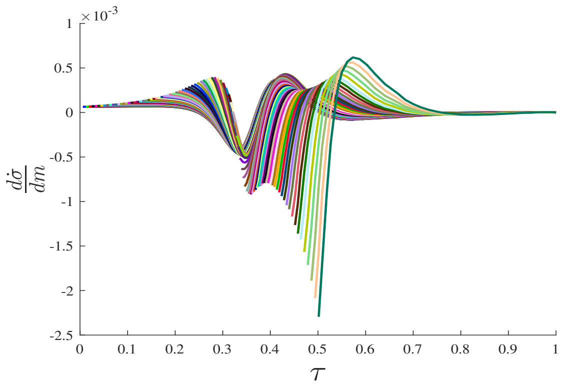
$$\frac{\mathrm{d}h}{\mathrm{d}h_s}(\boldsymbol{\tau},\boldsymbol{\tau}_s) = 1, \quad \text{at } \boldsymbol{\tau} = \boldsymbol{\tau}_s. \tag{6.25}$$

2. The sensitivity of a state against any other perturbation at the current time is zero. Example: Figure 6.8e and f show the sensitivity of the longitude against a perturbation in latitude. It holds that

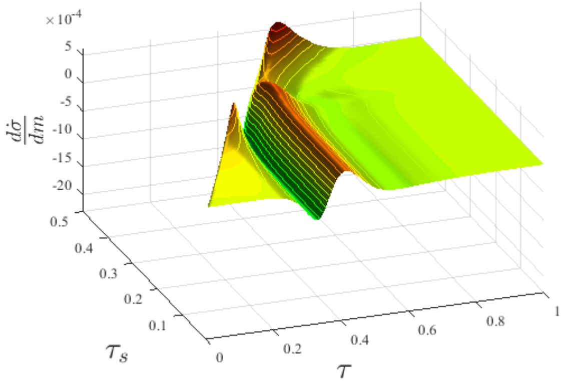
$$\frac{\mathrm{d}\lambda}{\mathrm{d}\varphi_s}(\boldsymbol{\tau},\boldsymbol{\tau}_s) = 0, \quad \text{at } \boldsymbol{\tau} = \boldsymbol{\tau}_s. \tag{6.26}$$

3. States that are fixed at the terminal time must have a terminal sensitivity of zero. Example: Figures 6.8c, d, e and f.

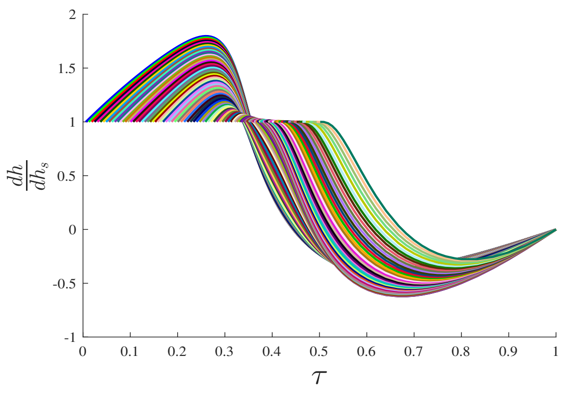
Due to the high number of states and parameters the surface fitting must be completely automated to allow for a sensible preparation time of the closed loop data set. The surface coefficients and the nominal open loop solution form the reference catalog that is required during the online phase. With this data we are ready to test $\Theta(p)$.



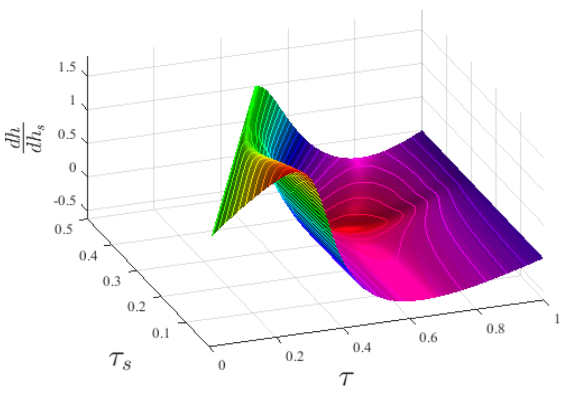
(a) Sen. bank angle rate w.r.t. mass



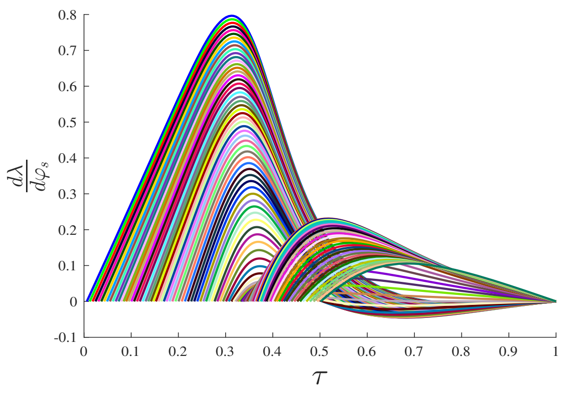
(b) Sen. bank angle rate w.r.t. mass



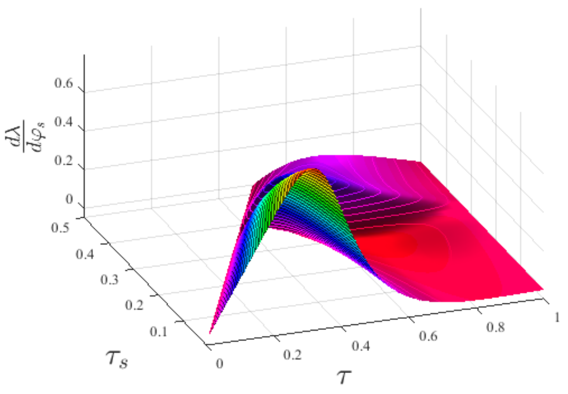
(c) Sen. altitude w.r.t. initial altitude



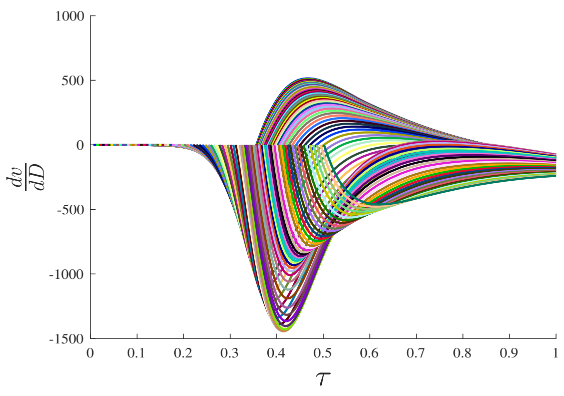
(d) Sen. altitude w.r.t. initial altitude



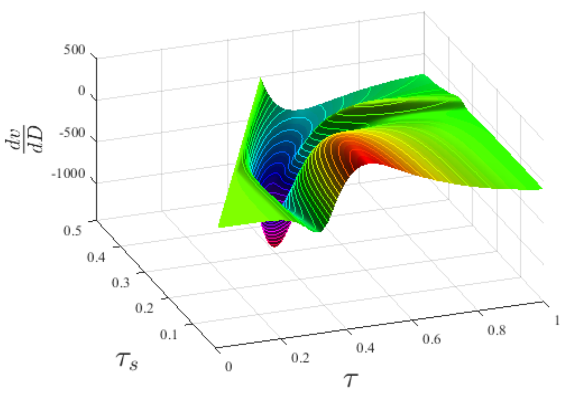
(e) Sen. longitude w.r.t. initial latitude



(f) Sen. longitude w.r.t. initial latitude



(g) Sen. velocity w.r.t. drag



(h) Sen. velocity w.r.t. drag

Figure 6.8: Interpolated parametric sensitivity of the control, position and velocity against perturbations in the mass and the position and velocity, occurring at different points of the optimal trajectory of $\text{OCP}_{(5.19)}(p_0)$ (cf. Figure 5.13).

6.3.3 Correction Space Analysis

Our goal is to show numerically that the correction space achieved by $\Theta(p)$ is large enough to cover the mission relevant parameter space, and thus $\Theta(p)$ is a feasible option for closed loop control. Furthermore we want to estimate the extent of the correction space and compare it to the at all feasible space.

Such numerical demonstrations are typically based on Monte Carlo analysis. A Monte Carlo method repeatedly chooses a random set of parameters and simulates the result. With a sufficient number of trials, the mean of the trials converges against the true mean of the investigated process. The main advantage of this method is that it captures conditional decisions and the interdependencies of parameters. But performing Monte Carlo analysis on a large parameter space requires a very high number of trials until this convergence can be observed with significant accuracy.

In preparation for a Monte Carlo analysis we roughly estimate the extent of the correction space. To this end we perform line searches in multiple search directions. We consider perturbed parameters

$$p = p_0 + \Delta p = p_0 + \alpha \frac{\Delta p}{\|\Delta p\|} = p_0 + \alpha\, d, \quad \alpha \in \mathbb{R}, \quad d \in \mathbb{R}^{n_p},\ \|d\| = 1. \tag{6.27}$$

The factor α is the perturbation strength and d is the perturbation direction. Our goal is to estimate the minimal and maximal values of α for which $\Theta(p_0 + \alpha d)$ computes a feasible solution.

$$\alpha_{\min}(p_0, d) := \arg\min_{\alpha}\ \Theta(p_0 + \alpha d), \quad \Theta(p_0 + \alpha d) \text{ feasible,} \tag{6.28a}$$
$$\alpha_{\max}(p_0, d) := \arg\max_{\alpha}\ \Theta(p_0 + \alpha d), \quad \Theta(p_0 + \alpha d) \text{ feasible.} \tag{6.28b}$$

To find $\alpha_{\min}(p_0, d)$ and $\alpha_{\max}(p_0, d)$ we start at a small value of α and perform line search until the solution becomes infeasible and then use binary search to improve the accuracy of the estimated boundary up to a certain threshold. A naive search algorithm that finds $\alpha_{\min}$, $\alpha_{\max}$ is stated in Appendix H. The boundary value is found easily, thus we omit the discussion of more efficient search variants.

To estimate the correction space boundary in perturbation direction d, the boundary search algorithm is called at all sensitivity sampling points

$$p_0^{(i)} := \begin{pmatrix} \bar{p}_0 \\ \overset{\star}{x}(\boldsymbol{\tau}_s^{(i)}; \bar{p}_0) \end{pmatrix}, \quad \boldsymbol{\tau}_s^{(i)} \in T_{nom}. \tag{6.29}$$

It remains to choose the search directions. Considering that the solution update for perturbations in multiple scalar parameters is the superposition of the update for each individual scalar, i.e. the Taylor expansion (5.13b) for the parameter vector

$$p = (L, D, m, h_f, \lambda_f, \varphi_f, h_s, \lambda_s, \varphi_s, v_s, \gamma_s, \chi_s) \tag{6.30}$$

can be written as

$$\begin{aligned}\tilde{z}_1(p) &= \overset{\star}{z}_0 + \frac{\mathrm{d}z}{\mathrm{d}p}(p_0)\begin{pmatrix}\Delta L\\ \vdots \\ \Delta\chi_s\end{pmatrix} \qquad (6.31)\\ &= \overset{\star}{z}_0 + \frac{\mathrm{d}z}{\mathrm{d}p}(p_0)\begin{pmatrix}1\\ \vdots \\ 0\end{pmatrix}\Delta L + \ldots + \frac{\mathrm{d}z}{\mathrm{d}p}(p_0)\begin{pmatrix}0\\ \vdots \\ 1\end{pmatrix}\Delta\chi_s,\end{aligned}$$

natural choices for d are perturbations in single scalar parameters.

$$d_1 := (1,0,0,0,0,0,0,0,0,0,0,0), \qquad (6.32a)$$
$$d_2 := (0,1,0,0,0,0,0,0,0,0,0,0), \qquad (6.32b)$$
$$\vdots \qquad (6.32c)$$
$$d_{12} := (0,0,0,0,0,0,0,0,0,0,0,1) \qquad (6.32d)$$

In the following we show approximations of the correction space in the directions $d_1, ..., d_{12}$.

In addition we estimate the boundary of the feasible space in direction d by using the same line search algorithm, but calling the NLP solver WORHP on the perturbed problem, instead of $\Theta(p)$. The feasible space boundary is traced backwards, starting at the second last discretization point. The solution is then used as start value at the third last discretization point and so on. Due to the large computational effort required to find the boundary at all discretization points, for multiple directions, the number of major SQP iterations is limited to 30. Thus the definition of *feasible* in this case is: The NLP solver WORHP could find a feasible solution in 30 or less iterations, using an optimization start value that was feasible for the neighboring discretization point. The relative accuracy limit for the refinement binary search is set to 5 %.

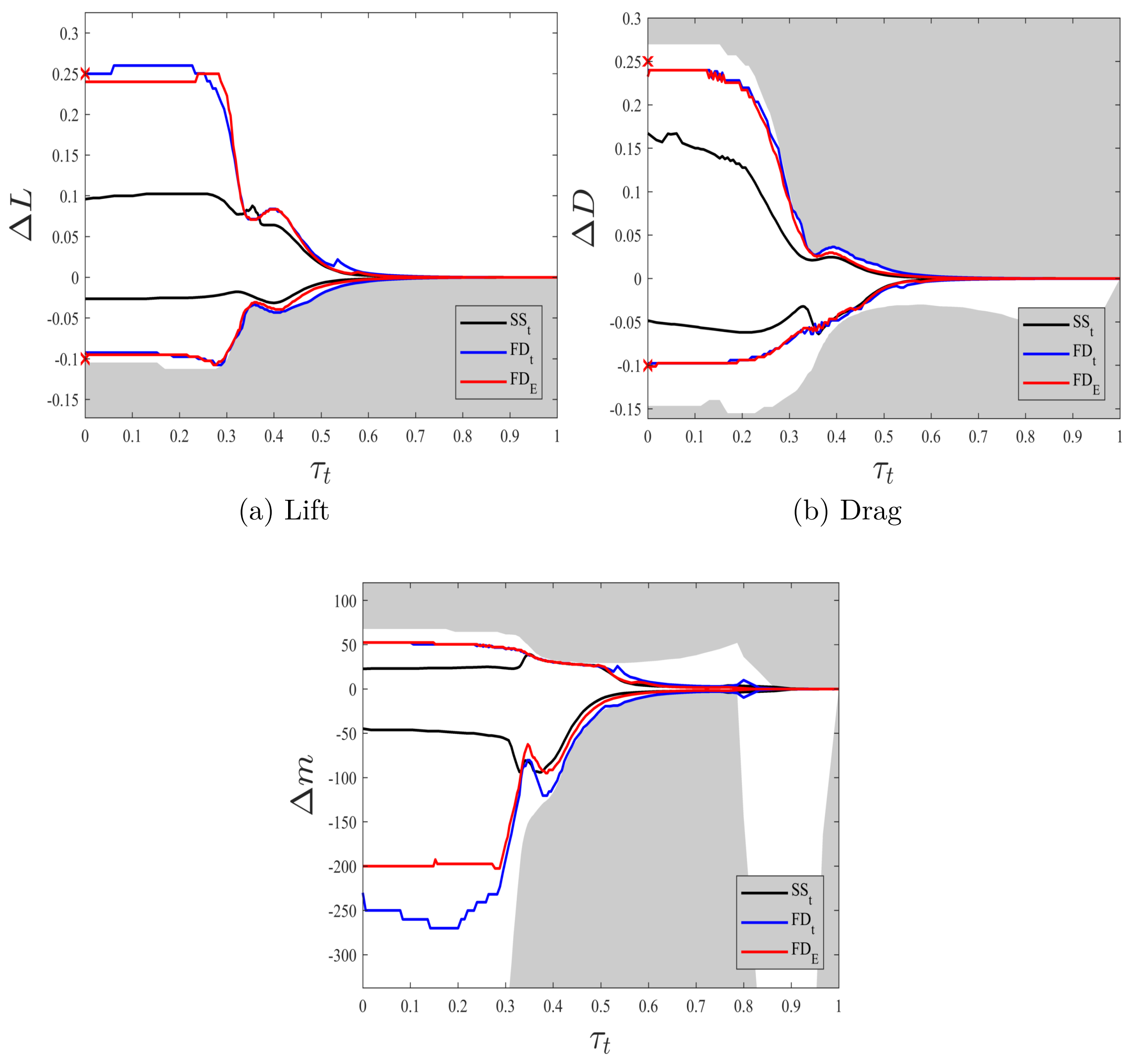
In the following figures *white background* shows the feasible space as determined by WORHP. The correction space boundaries are colored as:

Black: Correction space of $\Theta(p)$ for a single shooting transcription of the time domain formulation (6.15). The closest nominal parameter is obtained according to the independent variable time.

Blue: Correction space of $\Theta(p)$ for a full discretization transcription of the time domain formulation (6.15). The closest nominal parameter is obtained using a weighted euclidean distance function, as discussed in Section 5.6.1.

Red: Correction space of $\Theta(p)$ for a full discretization transcription of the energy domain problem formulation (6.21). The closest nominal parameter is obtained using the same weighted euclidean distance function as for the time domain problem.

The following images show the correction space w.r.t. normalized time; energy domain values have been transformed accordingly. Figure 6.9 shows the boundaries for perturbations in the aerodynamic accelerations and the mass. Figure 6.10 shows the boundaries for perturbations in the initial state (Figure 6.11 shows a zoomed view).

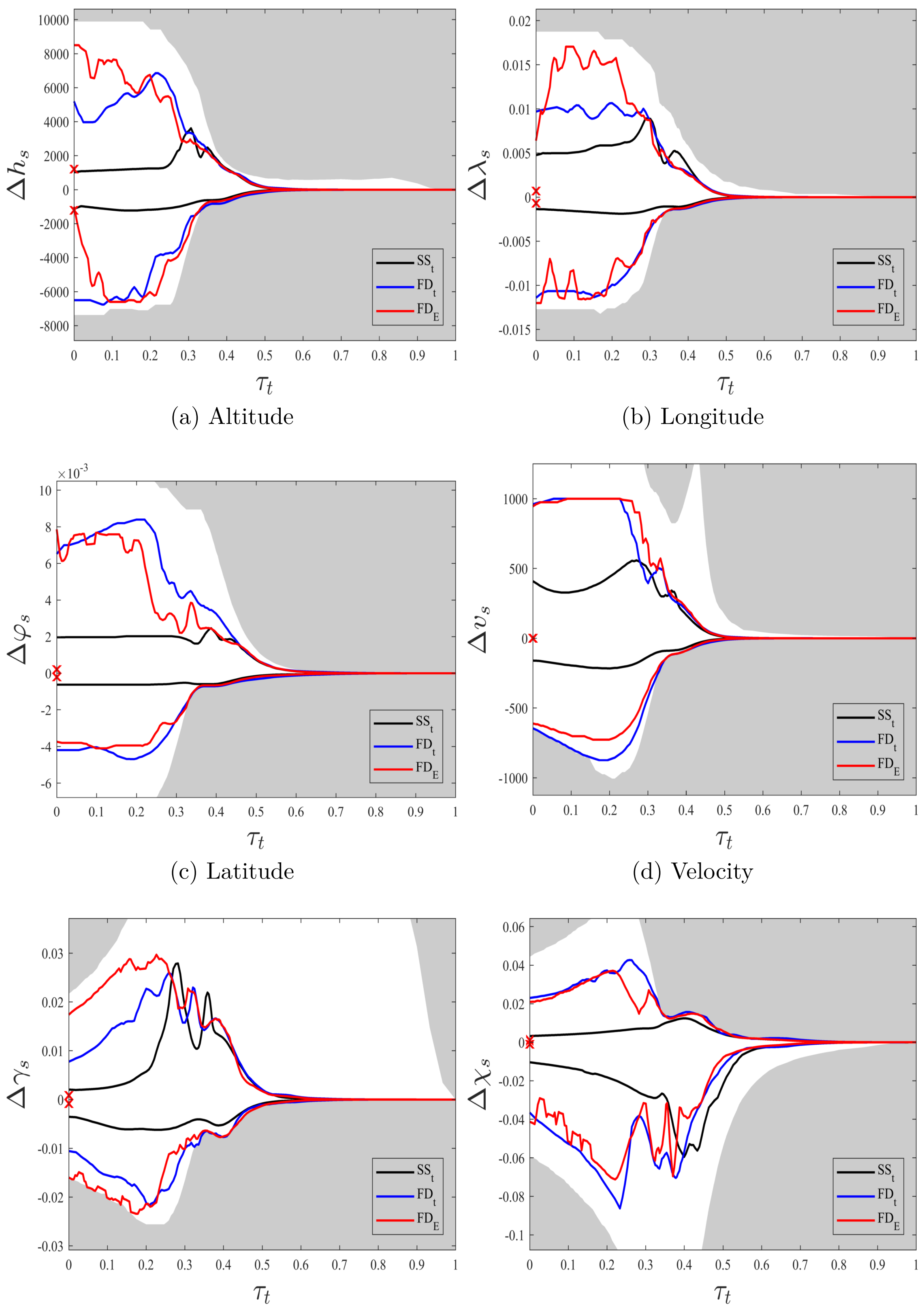


(a) Lift

(b) Drag

(c) Mass

Figure 6.9: Correction space obtained by $\Theta(p)$ for perturbations in different parameters based on time domain single shooting (black), time domain full discretization (blue) and energy domain full discretization (red). The feasible space as approximated by the NLP solver WORHP is shown in white, the grey area indicates the infeasible space. The red crosses mark the maximally expected perturbation at the EIP.

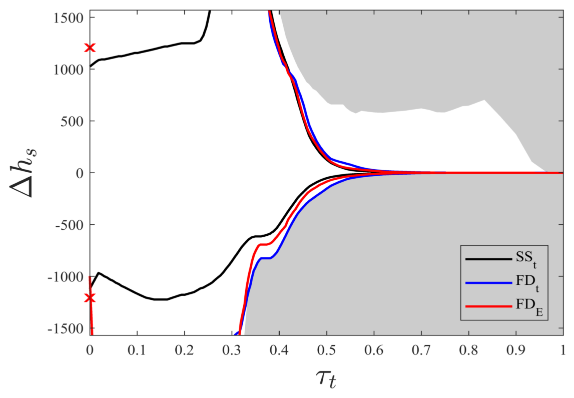


(a) Altitude (b) Longitude

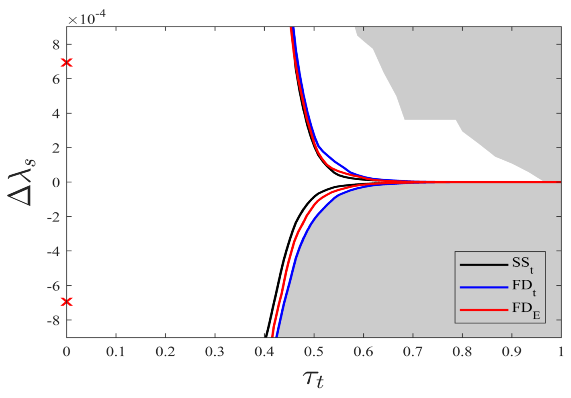
(c) Latitude (d) Velocity

(e) Flight path angle (f) Heading angle

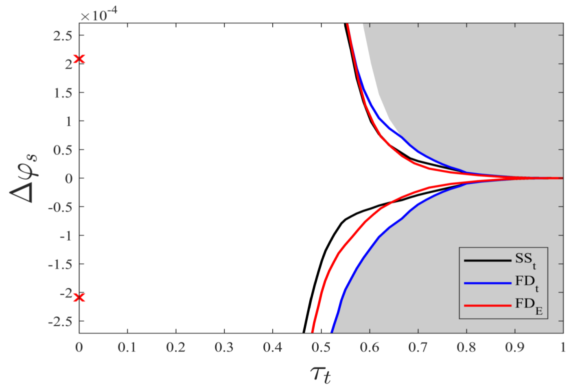
Figure 6.10: Correction space obtained by $\Theta(p)$ for perturbations in different parameters based on time domain single shooting (black), time domain full discretization (blue) and energy domain full discretization (red). The feasible space as approximated by the NLP solver WORHP is shown in white, the grey area indicates the infeasible space. The red crosses mark the maximally expected perturbation at the EIP.



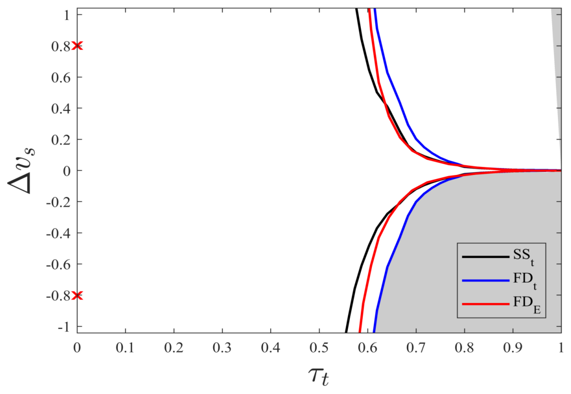
(a) Altitude



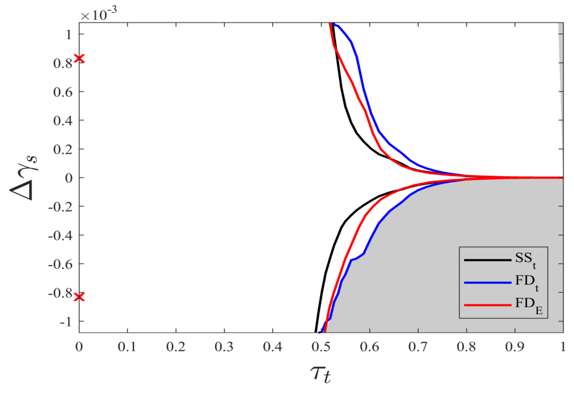
(b) Longitude



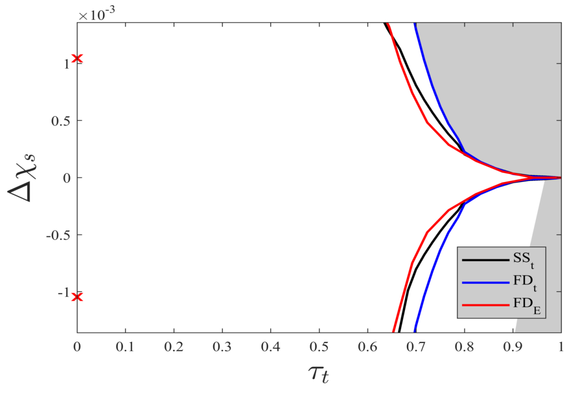
(c) Latitude



(d) Velocity



(e) Flight path angle



(f) Heading angle

Figure 6.11: Zoom on the perturbations expected at the entry interface point, and the enveloping correction space.

Hypersonic Re-Entry (Interval from 0 to 0.3)

This phase is mainly characterized by a rapid loss of altitude (cf. Figure 6.3a). The system can absorb much higher lift, but less lift or drag or too much drag can result into problem infeasibility. The expected perturbations $\Delta L, \Delta D \in [-0.10; 0.25]$ are close to the feasible limit. A thinner atmosphere is clearly shown to be a critical situation, leading to an infeasible problem[1]. The feasible space for perturbations of the state is large compared to the expected errors (cf. Table 6.3).

The full discretization correction space is large for all states in comparison to the expected EIP errors. Concerning the lift and drag acceleration, the correction space covers the expected errors at the EIP. There are feasible solutions for higher lift ($\Delta L > 0.25$), that are not covered by the correction space. For reduced lift the correction space roughly matches the feasibility boundary.

Full discretization shows to be superior to single shooting, achieving a universally larger correction space. This confirms the result that was observed in the analysis of the rocket car example.

Deceleration Phase (Interval from 0.3 to 0.65)

This phase is characterized through high aerodynamic accelerations (cf. Figure 6.4) and the resulting strong deceleration (cf. Figure 6.3d). We see a strong reduction of the feasible space for all parameters, except for positive lift and for a positive flight path angle.

If the correction space boundary lies in the interior of feasible space, the boundary is determined by the convergence failure of the corrector iteration (5.17). During the deceleration phase the feasibility space is strongly reduced, such that no longer the convergence of (5.17), but the feasibility boundary itself determines the extend of the correction space. In this case the corrector iteration converges, but a change of the active set occurs, i.e. the trajectory violates the path constraints or the terminal condition $v_f \leq 450\,\frac{\mathrm{m}}{\mathrm{s}}$.

Supersonic Phase (Interval from 0.65 to 1)

The feasible space concerning all perturbations, except a higher flight path angle, is small. The correction space is severely reduced. Figure (6.11) shows a zoom of Figure (6.10) to the level of the expected perturbations at the EIP. For all states, except the latitude, there exist feasible solutions for perturbations in either positive or negative direction.

Unfortunately the correction space achieved by $\Theta(p)$ is very small in the supersonic phase. Independently of the problem formulation or the chosen discretization, the corrector iteration does not converge to a feasible solution, resulting into a correction space that is too small to be practically applicable. Figure (6.11) shows that also the feasible space boundary for all states is very close to the solution on at least one side.

The strongly constrained problem has a very negative effect on the corrector scheme: The optimal solution sensitivities are based on the assumption, that all optimization variables that are not on their box boundary can be used to adapt to a perturbation. But sensitivity

1 A perturbation of the atmospheric density equally affects lift and drag, which is not represented in this test. Simulations indicate, that the simultaneous, equal perturbation of the aerodynamic forces is less problematic than an isolated perturbation of either one.

analysis does not include information on how large the tolerable perturbation is: Considering a scalar parameter $p^{(j)} = p_0^{(j)} + \Delta p^{(j)}$ the sensitivity $\frac{\mathrm{d}z}{\mathrm{d}p^{(j)}}$ can be thought of as optimal, feasible descent direction at the optimal solution for a parameter change $\Delta p^{(j)}$. It is however not taken into account, that the direction $\frac{\mathrm{d}z}{\mathrm{d}p^{(j)}}$ might only be feasible for an infinitesimally small perturbation $\Delta p^{(j)}$.

The corrector step (5.17) relies on the assumption, that the directions $\frac{\mathrm{d}z}{\mathrm{d}q^{(i)}}$, $1 \leq i \leq n_g$ are feasible with sufficient margin to adapt to the error remaining in solution after the forward correction (5.13b). If this assumption is broken, either because the margin is too small, or because there are multiple violated constraints with mutually adverse correction directions, it is not possible to reduce the constraint error to zero and the solution adaption fails. If on the other hand the NLP solver encounters the same situation, a new feasible descent direction is determined, that eventually may allow to find a new feasible solution.

In the next sections we update the guidance strategy to account for the insufficient correction space in the supersonic phase.

Comparison of the Correction Space for Time and Energy Discretization

Figure 6.9 and 6.10 do not show a clear advantage of either time or energy based formulation. Concerning longitude 6.10b and flight path angle 6.10e the energy formulation achieves a slightly better result, while for the rest of the states the performance is similar, except for the initial altitude, where the energy domain formulation is inferior. The similarity of the correction spaces suggest that, when taking into account the entire sensitivity set, the seemingly significant differences in single sensitivities are not meaningful. A hypothesis is that the changed dependencies in the dynamics are compensated by estimating the integral w.r.t. time in the objective function. In conclusion the more complicated energy domain analysis does not provide a significant correction space advantage and is thus not recommended for the purpose of sensitivity based real-time adaption.

Other Observations

The correction space boundary is more jagged when using the weighted euclidean distance function to determine the closest nominal parameter, compared to using the independent variable. This is caused by switching the closest nominal parameter: $\Theta(p)$ receives the perturbed parameter $p = p_0 + \alpha d$ as an argument, but it has no knowledge of p_0, α, d. Depending on the determination of the closest nominal parameter, cf. (5.53c), the correction starts from different $p_0^{(i)}$ and thus requires different correction directions and step sizes[1].

Concerning a possible increased payload mass Figure 6.9c shows that the vehicle mass cannot be increased by more than ≈60 kg. The upper mass boundary is determined by the ability to reduce the terminal velocity below the threshold. The lower mass boundary on the interval [0.4; 0.8] indicates that during this phase the mass, i.e. the gravitational acceleration, is the deciding factor that prevents the vehicle from skipping the atmosphere.

1 The weights of the distance function for were determined using to the strategy discussed in Section 5.6.1 as well manual tuning

Conclusions

The discretization of the state equations combined with the use of the state sensitivity in the correction scheme results into a larger correction space. Full discretization or multiple shooting with a high number of short shooting intervals are thus preferable to single shooting.

The advantage of an energy domain formulation is a higher tolerance to a perturbed flight path angle. Otherwise the time domain formulation achieves equal or better results. Both formulations are feasible approaches.

The correction space is sufficient at the EIP and there is a large margin. A violation of the EIP delivery accuracy up to a factor of three stays well within the convergence region, which is a very good result.

The ability to react to perturbed aerodynamic accelerations is sufficient. Even with the simple model of uniformly disturbed aerodynamic accelerations, the correction space covers a large part of the feasible space. It is noted that strong, local perturbations of the atmosphere are not described well with this perturbation model. A refined perturbation model, developed in collaboration with a robust parameter estimation technique could lead to further improvement.

Unfortunately the correction space of $\Theta(p)$ is not sufficient in the supersonic phase. The divergence of the corrector scheme does not allow to determine a feasible trajectory even for small perturbations. The hope that $\Theta(p)$ could be used directly for closed loop control command generation for the entire process is not fulfilled. At least in the supersonic phase a different guidance and control strategy is required.

6.4 Two Degree of Freedom Guidance System

To improve the insufficient control envelope in the supersonic phase we propose to combine $\Theta(p)$ with a drag controller (cf. Chapter 3.1) into a two-degree-of-freedom system. $\Theta(p)$ is used during the hypersonic phase and the deceleration phase to generate admissible, near optimal state trajectories and the corresponding control sequences. From the adapted state trajectory we obtain the drag acceleration profile, which is tracked by the drag controller. Optionally the vertical-lift-over-drag ratio of the adapted trajectory can be used as feed forward control. The drag controller generates the bank angle command which is the input for the attitude control system, cf. Figure 6.12.

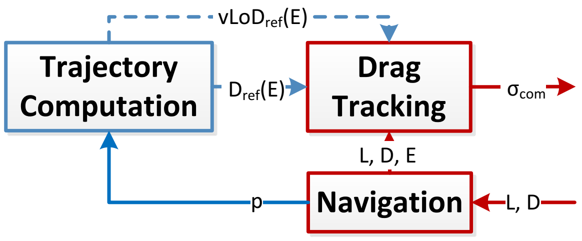


Figure 6.12: Two-degree-of-freedom guidance system.

The drag controller tracks the newest available, admissible trajectory. This capitalizes on the ability of $\Theta(p)$ to adapt to strong state errors during the hypersonic phase, while compensating the weak performance in the supersonic phase using drag control.

Because the system is reliant on inertial navigation the accuracy of the state estimate decreases. But the estimation accuracy of $\Delta L, \Delta D, \Delta m$ increases with flight time if the estimator converges. The parameters L, D, m are an adaptive model, which is used by both, $\Theta(p)$ and the drag controller. When the ability to adapt the trajectory is lost, the last parameter set, i.e. the newest model, is used for the remaining flight.

The guidance system can operate synchronously or asynchronously: In synchronous operation the trajectory generation and the tracking operate on fixed execution frequencies. The trajectory generation operates at a low rate, while the tracking operates at a high rate. Each time a new trajectory is accepted, the accumulated drag error in the integral term of the tracking controller is reset to zero.

In asynchronous mode, the tracking still operates at a fixed, high rate, but the trajectory generation is called only on demand. If the errors incurred from tracking the current trajectory are low compared to the expected accuracy, a recomputation of the trajectory is not required. A new trajectory is only requested when the state error or the total drag error exceed a certain threshold. The asynchronous operation mode makes better use of the integral action of the drag controller: In absence of significant errors, an unnecessary drag error reset is prevented, which can lead to improved tracking performance.

The drag control law (3.31) is sensitive during the low drag segment and tends to overreact. We compensate this behavior by using a one-sided gain scheduling, similar as suggested by Tu [Tu98]. During the low drag, hypersonic phase we rely on the near-optimal control determined by $\Theta(p)$. With increasing drag, we smoothly transition to the drag controller.

Recall Section 3.1.2: The drag controller commands the vertical lift over drag ratio

$$u_{\text{com}} = \frac{1}{b}\left(-a + D_r'' + \nu\right) \tag{6.33}$$

with

$$\nu = -K_P \Delta D - K_D \Delta D' - K_I \int \Delta D \, \mathrm{d}E. \tag{6.34}$$

The reference drag is given by the currently tracked trajectory: $D_r'' := D_\Theta''$. The commanded vertical lift over drag ratio is thus

$$u_{\text{com}} = \underbrace{\frac{-a + D_\Theta''}{b}}_{l} + \underbrace{\frac{\nu}{b}}_{o}. \tag{6.35}$$

In a low drag environment the outer, linear control o behaves as expected, but the synthetic linearization l is sensitive and tends to overreact. To prevent the overreaction we extract the reference lift-over-drag ratio u_Θ from the near optimal solution provided by $\Theta(p)$ and replace the linearizing control during the low drag phase:

$$u_{\text{com}} := u_\Theta + s\left(\frac{-a + D_\Theta''}{b} - u_\Theta\right) + \frac{\nu}{b}. \tag{6.36}$$

$s : \mathbb{R}^+ \to [0; 1]$ is a gain function, scheduled on sensed drag acceleration D. In the low drag, hypersonic phase s is close to zero, thus the command is determined by u_Θ. Note that the linear drag error feedback o is always active. This results into a damped, but as early as possible reaction to drag error. With increasing drag acceleration the gain increases to one and u_Θ fades out. For s we use a smooth sigmoid function, which is scaled such that the gain reaches one at 50 % of the expected maximal acceleration.

This strategy shows very good results. In most cases the commanded bank angle can be realized by the attitude control system with only a short delay, which is most important during the high drag segment of the flight.

The design is detailed in Figure 6.13, showing the data flow and the computation steps of the proposed system. The complete trajectory information consists of the state trajectories, the control sequence, and the dynamic model parameters. The acceleration profiles can be derived from this data as required.

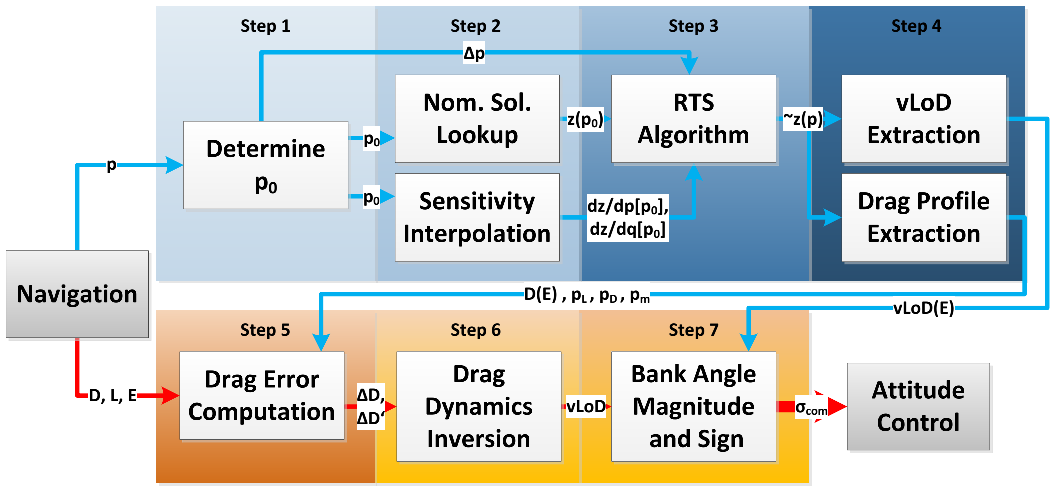


Figure 6.13: Data flow and computation steps of the two-degree-of-freedom guidance system.

6.5 Numerical Verification and Validation

The two-degree-of-freedom guidance is tested in a closed loop Monte Carlo campaign to assess the landing accuracy dispersion. In the following we detail the environment simulation, the considered perturbations and the guidance system configuration, before presenting the guidance performance results. Finally the guidance algorithm is tested on a flight representative processor.

The transcription and the analysis of the nominal solution and the parametric sensitivity differentials is performed by the code *Sensitivity Analysis Framework* (SAF) by D. Seelbinder. The solution of the parametric NLP and the computation of the sensitivity differentials is performed by the NLP solver WORHP [Was13].

The numeric quality of the sensitivity differentials strongly depends on the knowledge of the true sparsity structure of the Hessian matrices in (5.10). Standard finite difference routines do not provide sufficient accuracy. The required derivatives are computed using the automatic differentiation library ADOL-C [Wal12] which allows derivative computation close to machine precision.

The online part of the guidance algorithm is implemented in Embedded Matlab. Automatic code generation is used to obtain the guidance algorithm in C-code. For the processor in the loop test the tool chain TASTE [Per10] is used to specify the interfaces of the partition between the GNC algorithm and the simulation environment. The setup of the communication architecture and the final compilation for the LEON2 processor is also performed by TASTE, using the generated C-code. The major steps of the work-flow are summarized in Figure 6.14.

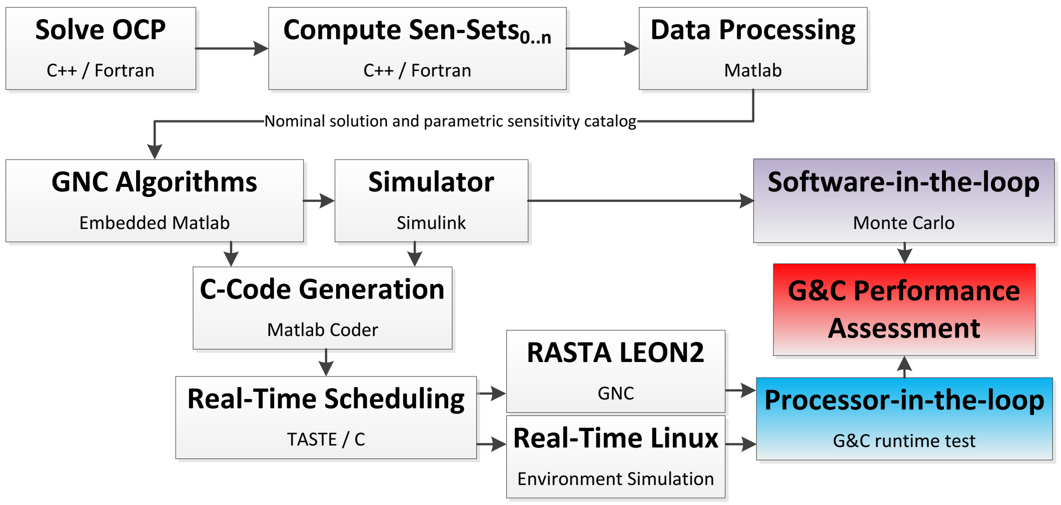


Figure 6.14: Verification and validation steps.

6.5.1 Monte Carlo Simulation

Environment and Perturbations

We integrate the translational equations of motion over a spherical, rotating planet, using (2.15) and (2.17). The simulation does not include rotational dynamics or an attitude control system, but we simulate the delay in realizing the commanded bank angle using a PID controller that tracks the guidance command. This setup is referred to as 3.5 degree of freedom simulation.

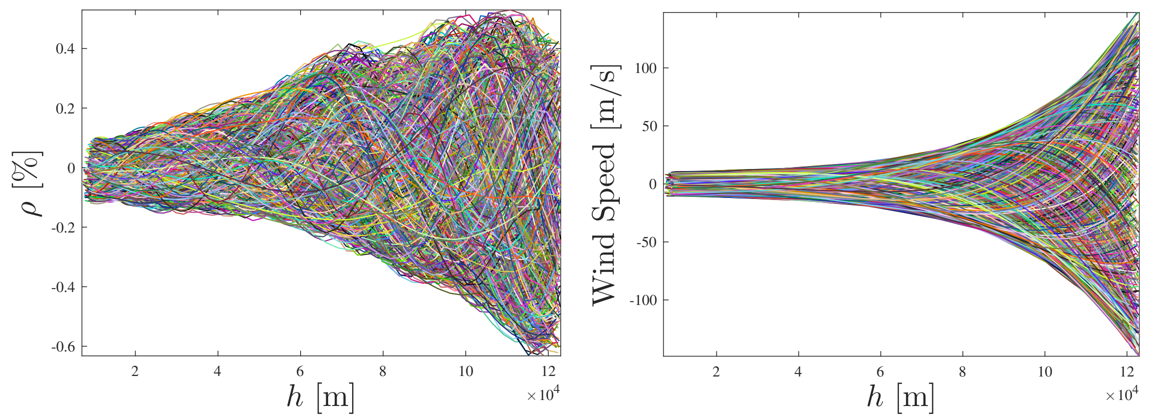
The main goal of this test campaign is to show, that the proposed guidance system is robust to perturbations. Hence we consider perturbations, that by far exceed the expected perturbation magnitude, cf. Table 6.3.

The atmosphere models are obtained from the European Mars Climate Database [Lew99]. We consider two extreme conditions:

1. The first one is a cold scenario, with an extremely clear atmosphere (low dust). The dust opacity is the minimum observed over 6 Mars years and further decreased by 30 %, combined with a solar minimum thermosphere.

2. The warm scenario corresponds to dusty atmosphere conditions. The dust opacity is set to the observed maximum and increased by an additional 30 %, topped with a solar maximum thermosphere.

As a first step, for each simulation an atmosphere is interpolated between those two extreme conditions. In a second step the interpolated atmosphere is perturbed with a sinusoidal. The amplitude and frequency are randomly chosen in specified intervals. The maximal amplitude is 50 % at maximum altitude, and reduces to 10 % at parachute opening. This resembles the expected strong, local perturbations of the upper atmosphere. Examples of the perturbation coverage are shown in Figure 6.15 for the atmospheric density and the wind speed.



(a) Atmospheric density (b) Wind speed

Figure 6.15: Random sinusoidal perturbation of atmosphere parameters during Monte Carlo simulation.

The aerodynamic coefficients are perturbed with a random sinusoidal of up to 10 % amplitude. We also consider a perturbation of the mass of up to 20 kg absolute value.

Table 6.4 shows the perturbation intervals of the initial state. The intervals around λ_0 and φ_0 roughly correspond to a maximum error of 20 km in downrange and 10 km in crossrange direction at the EIP. The errors are equally distributed, i.e. no error covariance has been assumed.

Table 6.4: Perturbed simulation parameters

State		Value	Parameter	max. Amp.
$h_{s,0}$	+/-	3000 m	ρ	temperature + 50%
$\lambda_{s,0}$	+/-	0.3265°	c_L	10%
$\varphi_{s,0}$	+/-	0.1632°	c_D	10%
$v_{s,0}$	+/-	200 m/s	wind speed	200 m/s
$\gamma_{s,0}$	+/-	1°	mass	up to 20 kg
$\chi_{s,0}$	+/-	1°		

GNC Configuration

The trajectory generation runs synchronously at 0.1 Hz. We use the time domain problem formulation (6.15) with a multiple shooting discretization. The problem is discretized using Runge-Kutta 4, on a nonuniform control grid with 81 points, which are determined by iterative grid refinement, such that the global discretization error is minimized. We use 20 shooting arcs with 4 control steps in each arc.

The trajectory tracking is performed at 10 Hz. The newest available, feasible trajectory is tracked, infeasible trajectories are discarded.

The guidance input is the vector

$$\tilde{p} = (\tilde{p}_L, \tilde{p}_D, \tilde{p}_m, \tilde{h}, \tilde{\lambda}, \tilde{\varphi}, \tilde{v}, \tilde{\gamma}, \tilde{\chi}). \tag{6.37}$$

The input is provided by a performance navigation function, which is simplified such that the *estimated* state is equal to the true state, falsified with white noise. The parameters $\tilde{p}_L, \tilde{p}_D, \tilde{p}_m$ are estimated from the true lift- and drag- acceleration using an extended Kalman filter to demonstrate that the estimation of the required model parameters is feasible based on the available sensor data, i.e. lift- and drag acceleration.

Simulation Results

In the following we discuss the dispersion around the nominal parachute opening point. The following results are based on 2000 Monte Carlo cases. Figure 6.16 shows the horizontal dispersion at parachute opening. The cross marks the position at which the simulation termination criterion is reached. The line connected to the termination position is the orientation of the terminal velocity vector.

Note that the desired terminal state is not a stable point of the system. Thus the stopping condition of the simulation, i.e. the point at which the error is measured, is of importance. We base the stopping criterion on the same variable, energy, as the trajectory tracking. The simulation is stopped, when the estimated energy is equal to the energy of the desired terminal state. In this way the state error is directly related to the tracking error. An alternative approach is to stop the simulation when either the horizontal euclidean target distance is no longer decreasing (point of closest approach), or when a minimum altitude value is reached. This would yield an equally valid, but different, error projection.

Table 6.5 quantifies the dispersion in terms of downrange, crossrange, altitude and velocity errors. The quantification assumes a normal distribution of the error. It is noted though, that this assumption is not accurate. The true covariance of the error is unknown.

With a 95 % confidence ($\mu + 2\sigma$) the downrange error is below 10.2 km, the crossrange error is below 2.7 km and the altitude error is below 1.7 km. Figures 6.17 and 6.18 show the σ-dispersion ellipses. The radial, horizontal miss distance is lower than 9.2 km with 95 % confidence. The Monte Carlo state trajectories are shown in Figure 6.19. Lastly the path constraints are shown in Figure 6.20. In none of the cases a path constraints becomes active.

To provide an assessment of the achieved results, we note that the threshold for Mars precision landing is attributed to roughly 10 km radial error, which has been undercut. A true qualitative comparison of the guidance law against other algorithms is very difficult, because of the high number of influential factors that make the scenario and the performance metric unique. A strict comparison against an error threshold, or an alternative guidance and control algorithm requires a standardization of the problem scenario, the perturbations, the simulation fidelity and the error quantification. Because of the complex nature of the problem the effect of changes in these criteria is not obvious, but could introduce a strong bias in the comparison. In terms of qualitative performance assessment we can state that the perturbations considered in this study are on par or exceed the perturbations considered in the references cited through the thesis, and we achieve a very satisfactory accuracy. In the same context we mention the major simplifications made in this test: a simplified attitude control, and a simplified state and parameter estimation.

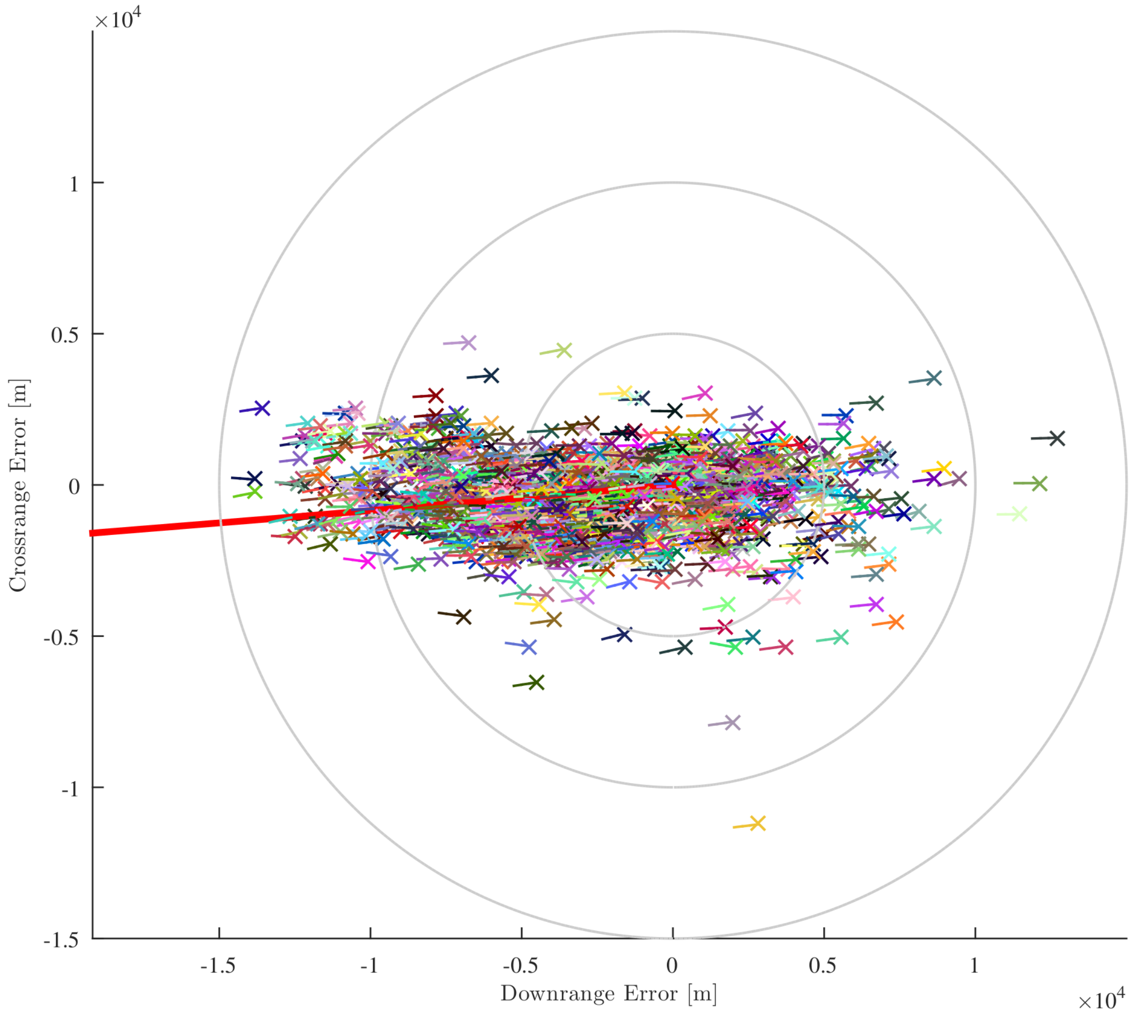


Figure 6.16: Dispersion at parachute opening plotted against 5 km, 10 km and 15 km radi. The incident lines indicate the terminal velocity direction.

Table 6.5: State error at parachute opening. The confidence values assume a normal distribution.

Type	Mean (μ)	Std. Deviation (σ)	68 % (1σ)	95 % (2σ)	99.7 % (3σ)
Horizontal Dist. [km]	3.9	2.6	<6.6	<9.3	<11.9
Downrange Error [km]	−2.4	3.8	<6.4	<10.3	<14.1
Crossrange Error [km]	−0.4	1.1	<1.6	<2.8	<3.8
Altitude Error [km]	−0.4	0.6	<1.1	<1.8	<2.5
Velocity Error $[\frac{m}{s}]$	4	6	<10	<16	<22

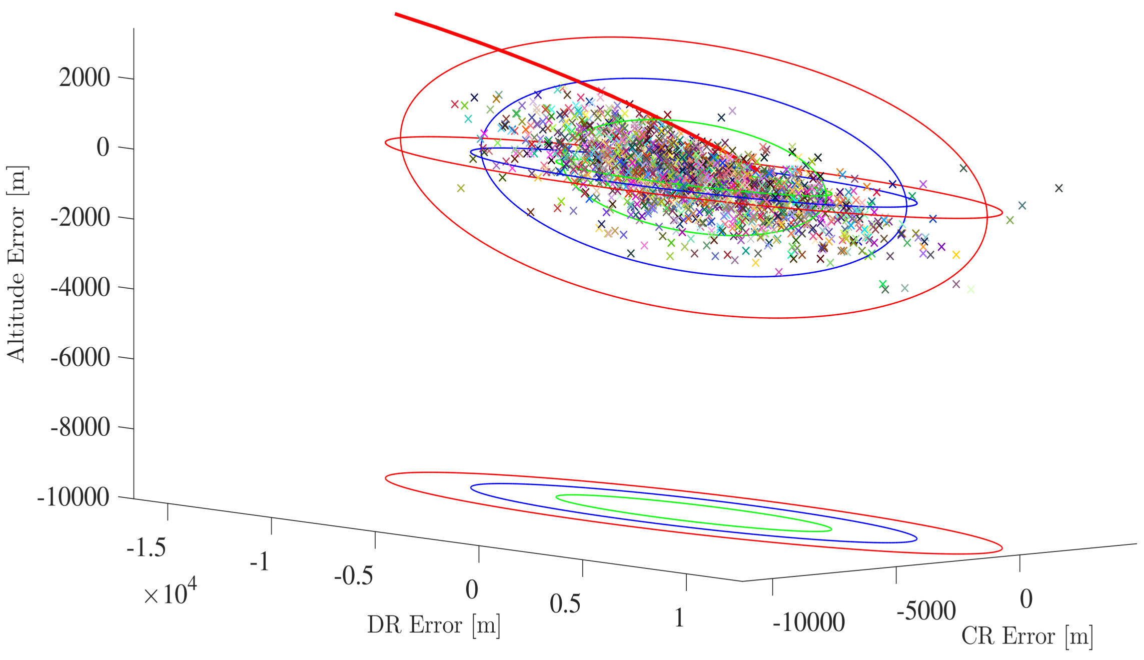


Figure 6.17: σ-confidence ellipses at parachute opening (3D).

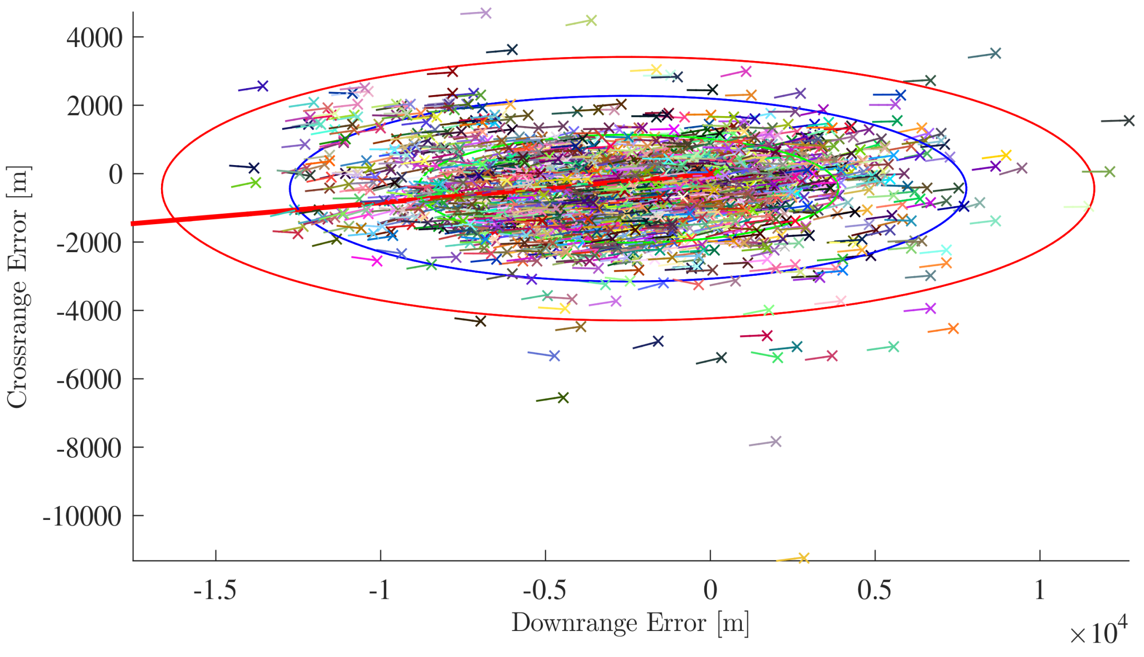
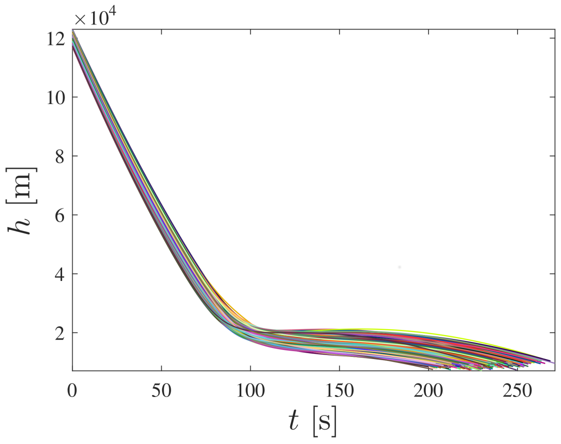
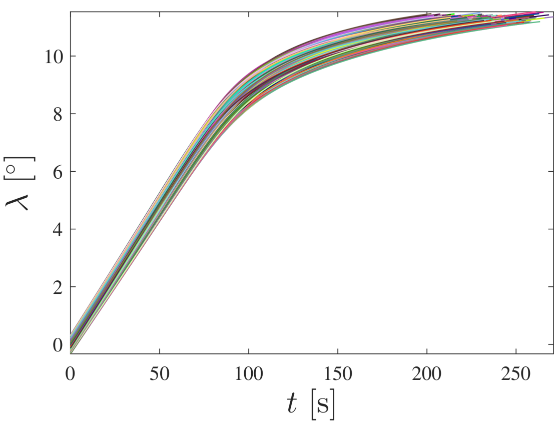


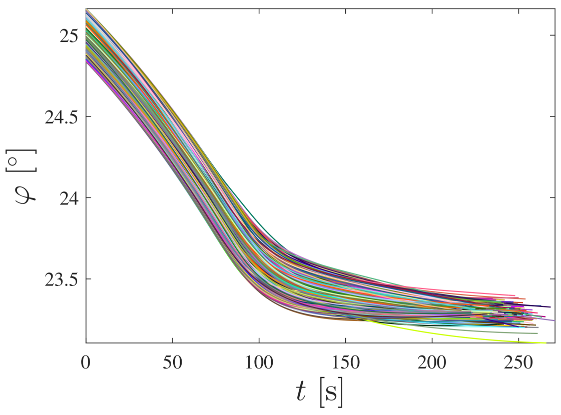
Figure 6.18: σ-confidence ellipses at parachute opening (top down). The incident lines indicate the terminal velocity direction.



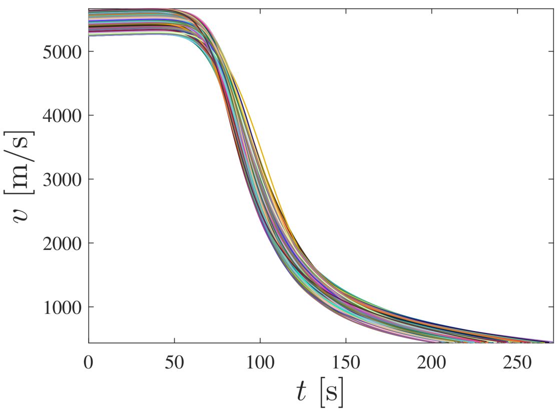
(a) Altitude



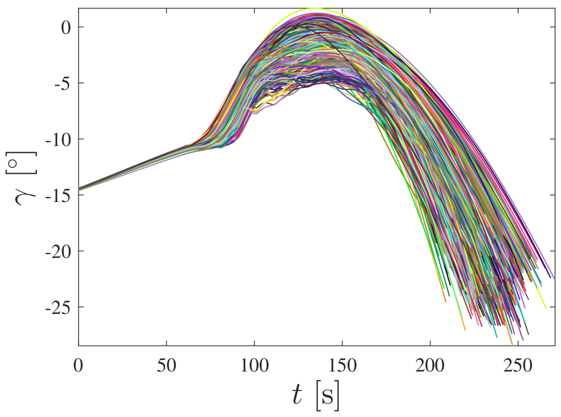
(b) Longitude



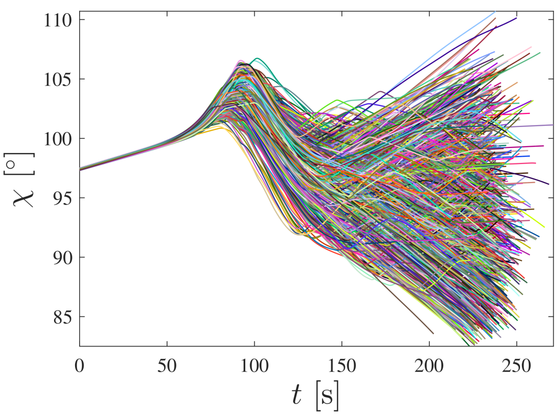
(c) Latitude



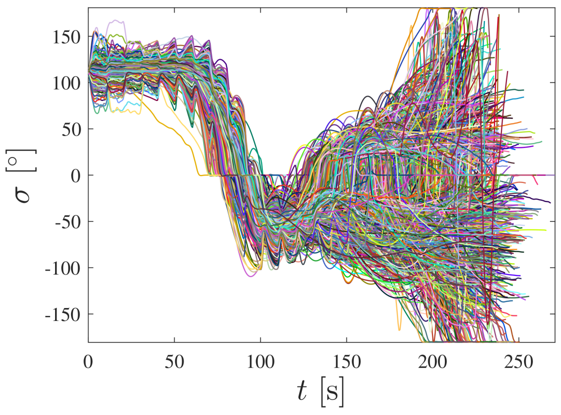
(d) Velocity



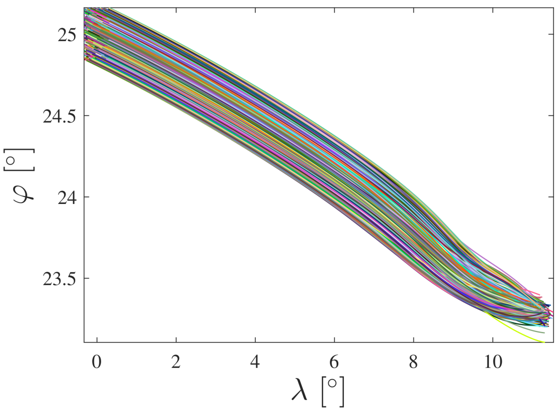
(e) Flight path angle



(f) Heading angle

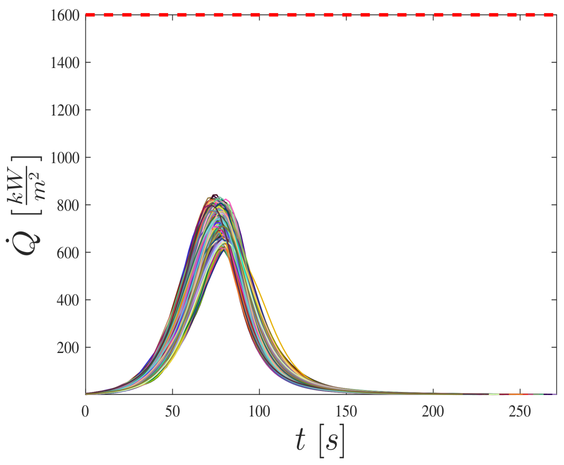


(g) Bank Angle

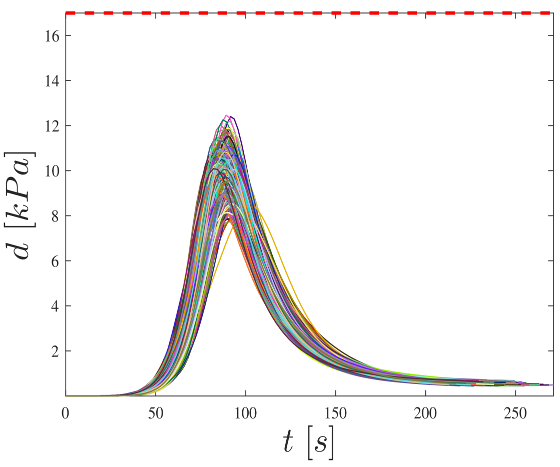


(h) Ground track

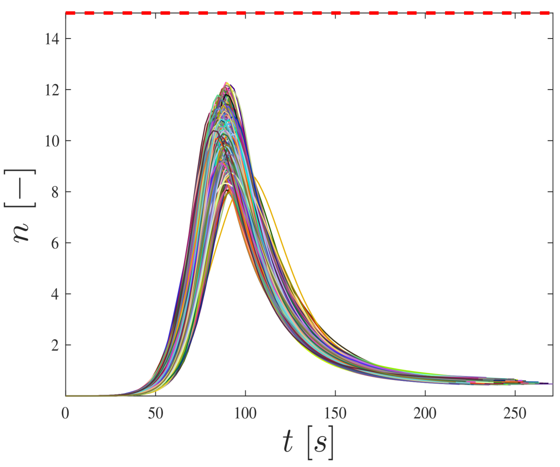
Figure 6.19: State trajectories of the Monte Carlo simulations.



(a) Heat flux



(b) Dynamic pressure



(c) Load factor

Figure 6.20: Path constraints during Monte Carlo simulations.

Lessons Learned

The negative downrange and altitude mean values (cf. Table 6.5) indicate a bias towards undershooting the target. The undershoot is a consequence from underestimating the atmospheric density: A local, negative density perturbation in the upper atmosphere can result into a temporary estimation of a thinner atmosphere. The adapted trajectory based on this estimation lets the vehicle dive deeper into the atmosphere to compensate the lack of drag. By the time the local density perturbation is moderated, or even swings into the opposite, the already applied corrective action was too strong. In the next trajectory iteration, when the density estimation has returned to a higher value, the vehicle cannot produce enough lift, to sufficiently moderate the earlier decision. Some of the bank angle profiles in Figure 6.19g show a bank angle of 0° (full lift-up) for an extend period of time. This leads to the divergence of drag tracking and the loss of crossrange control. This situation is very difficult to avoid for worst-case aerodynamic perturbations, due to the low $\frac{L}{D}$ ratio of the vehicle. Gain tuning and a delay of the model adaption could possibly improve the situation, but a true solution to this problem seems only possible through an increase of the $\frac{L}{D}$ ratio, i.e. a higher control authority margin.

In the hypersonic phase, the flown bank angle profiles 6.19g show a periodic pattern, roughly corresponding to the handover times of new trajectories. While the new trajectory is optimal with respect to the newest available information, the switch of the reference trajectory disrupts the tracking and requires a reset of the accumulated drag error (integral control). A future evolution could investigate asynchronous trajectory generation: A new trajectory is only requested, when the tracking error grows above a certain threshold.

In rare cases we see rapid changes of the realized bank angle in the supersonic phase. In these cases the pseudo bank angle tracking is not able to follow the commanded value and tends to overshoot. The rapid command changes can be caused by high frequency perturbations of the lift- or drag acceleration, or they indicate overtuning of the drag controller. In any case the gain scheduling of the drag controller and the interaction between the guidance command and the (pseudo) attitude control can be further improved.

6.5.2 LEON2 Processor in the Loop Test

Space mission on-board computer systems must fulfill special requirements to function reliably in space environment. CPUs and memory must be designed to withstand radiation and to operate under wide temperature conditions. The computing power of space qualified CPUs is at least one order of magnitude lower than that of current consumer CPUs. The restriction of computational power from space qualified hardware is an important factor for the design of GNC algorithms. The validation on space qualified, or equivalent, hardware is therefore a significant step in proving the feasibility of a GNC algorithm.

An example for a flight representative hardware environment is ESA's Reference Architecture System Test-Bed for Avionics (RASTA). RASTA uses LEON processors, which are available in fault tolerant, radiation hardened and space qualified versions. RASTA runs the real-time operating system RTEMS[1], which is also available in flight qualified versions.

In a real-time environment computational tasks are scheduled according to priority. A task

1 Real-Time Executive for Multiprocessor Systems

with higher priority preempts the CPU from a task with lower priority. If a task does not finish in the allotted time slot an overrun fault occurs, and the lower priority task is shut down. This can leave the computation result in an undefined state. This must be avoided at all costs, as it can lead to subsequent errors and ultimately to the loss of control and loss of the mission.

Commonly GNC algorithms are developed and tested in high level rapid prototyping environments, like e.g. Matlab/Simulink. But the execution in a real-time environment typically requires an implementation in a hardware-related programming language, like e.g. C or Ada.

In this project the guidance and control algorithm has been implemented in Embedded Matlab. Algorithms in Embedded Matlab can, under certain conditions, be automatically reimplemented in C code, using automatic code generation tools. The obtained C code is used as basis to compile executable binaries for the RTEMS real-time operating system. This approach has the advantage of enabling tests in a real-time environment without a manual reimplementation. The guidance and control algorithm is *cut* from the simulation loop (cf. Figure 1.1), and prepared to run on the LEON2 processor. The software framework TASTE [Per10] is used to support the compilation for the LEON2 target, the scheduling of the real-time tasks, and the setup of the communication pipelines between different GNC modules and the environment simulation.

Figure 6.21 shows a RASTA test bench in the ESA-ESTEC control hardware laboratory. A similar processor-in-the-loop setup is available at DLR Bremen, where the test was performed.

Table 6.6 shows the test results: the computation time of relevant algorithm steps on a standard Linux desktop PC, in comparison with the computation time on the RASTA. The desktop PC is between 27 and 60 times faster. While the nearest neighbor search and the p- and q-correction steps are computed in few milli seconds, the evaluation of the constraint function takes about 33 ms. With a runtime of 350 ms the evaluation of the sensitivity surfaces takes significantly longer. A maximum of 15 q-iterations was allowed in the Monte Carlo simulation. This results into a total execution time of 989 ms for one trajectory computation on the LEON2. Keep in mind that these result are obtained on

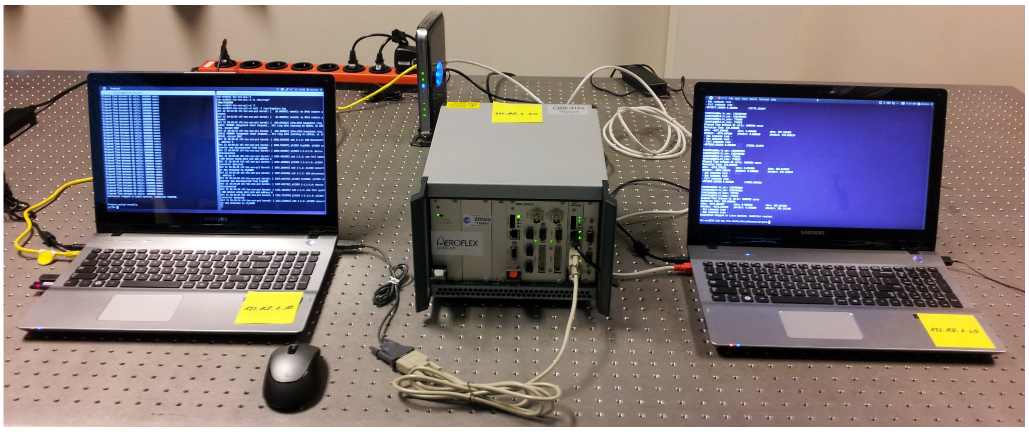


Figure 6.21: Processor-in-the-loop setup: simulation control (left), RASTA system with LEON2 processing board (middle), environment simulation (right).

the basis of a research and development implementation. Further speedup can be expected from industrial quality flight code.

Table 6.6: Runtime comparison

Computation	i7, 2.9 GHz	LEON2, 80 MHz	Factor
Nearest parameter search	0.17 ms	6 ms	≈ 35
Sensitivity interpolation	13 ms	350 ms	≈ 27
p-step (5.13b)	0.05 ms	3 ms	≈ 60
q-step (5.15)	0.21 ms	9 ms	≈ 43
$G(z,p)$	0.88 ms	33 ms	≈ 36

The processor-in-the-loop test demonstrates, that it is feasible to run the trajectory computation at a maximal rate of about 1 Hz. Considering safety margins, a realistic execution frequency is roughly one magnitude slower. Thus the trajectory generation could be scheduled at 0.1 Hz, with an expected mean delay of the computation result of about 1 s.

This demonstrates the real-time capability of the guidance approach in a flight equivalent environment. Using an industrial quality implementation and the next generation of processors it is realistic to achieve a sufficiently high execution frequency, such that the trajectory planning algorithm can be used directly for control command generation.

CHAPTER 7

Conclusion

This work connects control challenges of atmospheric entry guidance to mathematical methods from nonlinear optimization, parametric sensitivity analysis and dynamic programming. The major development steps and scientific contributions of this thesis are:

1. The synthesis of a local, closed loop, near-optimal feedback law for nonlinear dynamic systems. The approach extends preceding results in parametric sensitivity analysis of NLP problems to a closed loop control law. This includes the numerical analysis and characterization of effects of different transcription methods and problem formulations on parametric sensitivities of the related NLPs, in a high precision numerical environment.
2. The application of the proposed feedback law to the problem of controlling the atmospheric entry of a small capsule into the Martian atmosphere, in close resemblance of the ESA *Mars Precision Lander* reference scenario. The capabilities of the feedback law are analyzed and its limitations are carved out. The proposed control law is combined with the well-known drag tracking method, which results into a highly adaptable, two degree-of-freedom guidance system.
3. The numerical validation, verification and performance analysis of the combined, novel guidance system in a flight representative software and hardware environment. This includes the test of the guidance algorithm in a strict real-time environment on a flight representative LEON2 processor, as well as a software-in-the-loop Monte Carlo campaign.

The following conclusions are separated into three parts: In the first part we state our conclusion concerning the developed atmospheric entry guidance system. In the second part we sketch ideas for future research directions. The third part concludes the thesis.

7.1 Entry Guidance Performance

A major control challenge of atmospheric entry is the strongly nonlinear dynamic system, and the nonlinear state constraints. Our treatment of the entry problem as OCP, and the direct transcription into an NLP is a very good approach to obtaining a reference solution of the underlying ODE system while respecting the constraints. But for in-flight trajectory computation and control this approach is not computationally feasible on the targeted hardware.

We meet the need for minimal computational load by sacrificing on-board optimization. Instead we rely on a snapshot of the derivative of the optimality conditions, i.e. the parametric sensitivities. The approximation of the extremal field around the nominal solution via parametric sensitivities is computationally highly efficient, but at the same time has a strong drawback: The method depends on a constant active set. This leaves us unable to adjust to perturbations which cause a change of the active set, this includes activating bounds on the control variables and path constraints on the states. This disadvantage can be partially compensated by a congenial reformulation of the OCP. The predictive capability of the approach also allows early detection of critical trajectories, such that a combination with ulterior methods is possible. But ultimately the inability to change the active set remains a strong disadvantage, which in the end prevents the application of the sensitivity-based feedback past the deceleration phase. An approximation of the reachable space reveals that the ability to shape the trajectory is severely diminished after deceleration. But while a method based on online optimization would be able to capitalize on the remaining degrees of freedom, our approach fails in this flight regime.

The second major challenge are the uncertainties in the atmospheric model. The atmosphere is very close to a chaotic system, hence a prediction is very difficult. Our method profits from an accurate atmospheric model, and the performance decreases if the model is wrong. An advantage is that the parametric problem formulation easily allows for an adaption of the model, provided that the true parameters are known. This connects to the third challenge: The scarce availability of environmental information. Solely based on the IMU measurements it is very difficult to robustly adjust the model. We use a simple parametrization with few parameters, and limit the maximal adjustment to a predetermined confidence interval. The design of a filter to estimate parameters in the inner functions of the atmosphere model could be a subject of future work.

The illustrated weaknesses are for the most part overcome by combining the sensitivity based update with the drag tracking paradigm. Drag tracking is less dependent on an accurate atmosphere model and the drag control law is valid up until the parachute opening. A drag profile however relates to a predetermined downrange. Downrange errors, and other state errors at the EIP, are handled very well by the new feedback law. The combination of the sensitivity based trajectory update and drag tracking thus capitalizes on the strength of both methods, while compensating each others weaknesses.

The Monte Carlo results show that the combined guidance system is able to compensate perturbations that by far exceed the expected flight envelope. The processor-in-the-loop test demonstrates that the guidance system is feasible on currently available space qualified hardware.

The weak points that remain are the handling of path constraints and the lack of a formal prove, guaranteeing the convergence of the feedback law for perturbations up to a predetermined magnitude. It is however noted that these disadvantages are commonly shared amongst nonlinear control methods.

7.2 Way Forward

A driving factor for the research in this thesis was the specialization on a highly restricted computing environment with very limited computational resources. In the light of increasing computational power, this restriction will relax with future hardware generations, even for space applications. Without this restriction, solving optimization problems online in real-time becomes feasible. Online optimization based control methods, such as the receding horizon approach introduced in Section 3.2.3, have revolutionary potential.

Online optimization enables the consideration of constraints on the actuator and the state. Real-time optimal control has the potential to increases the region of controllability while at the same time achieving optimal performance. Control algorithms based on real-time convex optimization are already reality.

Sensitivity based control requires a thorough, case-by-case analysis for every application. The general principle is applicable to a wide range of problems, but it is not a *plug-and-play* solution. Parametric sensitivity analysis also requires exceptionally accurate numeric methods for derivative computation and OCP-NLP transcription, that at the time of writing are not commonly available. The preliminary work to achieve sensitivity based control is thus comparable to the effort to apply the receding horizon principle. If sufficient computational power is available, the receding horizon method is clearly superior.

Sensitivity based control is a precursor of online optimization based methods. A possible next step are hybrid systems, which perform online optimization at a low rate and use sensitivity based updates in between. Parametric sensitivities can be obtained after each optimization for a very small additional computational cost. This removes the need to precompute the sensitivity differentials at multiple points of the trajectory. Sensitivity based methods can be used to bridge the time between optimization runs.

Another option is to incorporate the Taylor expansion of the optimal solution into a receding horizon controller to obtain an improved initial guess for the next optimization run. The same principle can also be used to counteract the delay caused by the computation time: Once a receding horizon step finishes, a Taylor expansion of the optimal solution is used to adapt to the change of the state since the computation started.

7.3 Closing

The proposed guidance system and the lessons learned during its development will be used by the German Aerospace Center as input to Germany's reusable launch vehicle development program. The proposed approach demonstrates the power of optimal control and parametric sensitivity analysis. The author hopes that the discussion in this thesis contributes to stimulating further research in optimization based control methods to enable the development of flexible, robust, high performance guidance and control systems.

Bibliography

[Ath06] Athans, Michael and Peter L. Falb: *Optimal Control: An Introduction to the Theory and Its Applications*. Dover Books on Engineering. Dover Publications, 2006 (cit. on pp. 42, 43).

[Ber07] Bertsekas, Dimitri P.: *Dynamic Programming And Optimal Control*. Athena Scientific optimization and computation series. Athena Scientific, 2007 (cit. on pp. 40, 45).

[Ber99] Bertsekas, Dimitri P: *Nonlinear Programming*. 2nd. Athena Scientific, Sept. 1999 (cit. on p. 47).

[Bet10] Betts, John T.: 'Practical Methods for Optimal Control and Estimation Using Nonlinear Programming'. Cambridge University Press, 2010. Chap. 4. The Optimal Control Problem: pp. 123–217 (cit. on pp. 49, 52, 53, 67, 158).

[Bha98] Bharadwaj, Sanjay, Anil V. Rao, and Kenneth D. Mease: 'Entry Trajectory Tracking Law via Feedback Linearization'. *Journal of Guidance, Control, and Dynamics* (1998), vol. 21(5) (cit. on p. 22).

[Boc84] Bock, Hans Georg and Karl J. Plitt: 'A multiple shooting algorithm for direct solution of optimal control problems'. *Proceedings of the 9th IFAC World Congress*. 1984 (cit. on p. 49).

[Bol06] Bollino, K. P., I. M. Ross, and D. Doman: 'Optimal Nonlinear Feedback Guidance for Reentry Vehicles'. *AIAA Guidance Navigation, and Control Conference and Exhibit* (2006), vol. (cit. on p. 29).

[Bra06] Braun, Robert D. and Robert M. Manning: 'Mars Exploration Entry, Descent and Landing Challenges'. *Aerospace Conference, IEEE* (2006), vol. (cit. on p. 5).

[Bre07] Bressan, A. and B. Piccoli: *Introduction to the Mathematical Theory of Control*. AIMS on applied math. American Institute of Mathematical Sciences, 2007 (cit. on pp. 41, 43, 45).

[Bry69] Bryson, A.E. and Y.C. Ho: *Applied optimal control: optimization, estimation and control*. Massachusetts: Waltham, 1969.

[Büs02] Büskens, Christof: 'Echtzeitoptimierung und Echtzeitoptimalsteuerung parametergestörter Probleme'. ngerman. *Habil. Thesis* (2002), vol. (cit. on pp. 58, 61–63, 79, 101).

[Büs98] Büskens, Christof: 'Optimierungsmethoden und Sensitivitätsanalyse für optimale Steuerprozesse mit Steuer- und Zustands-Beschränkungen'. ngerman. PhD thesis. Westfälische Wilhelms-Universität Münster, 1998 (cit. on pp. 53, 57, 94, 107, 157).

[Büs00] Büskens, Christof and Helmut Maurer: 'SQP-methods for solving optimal control problems with control and state constraints: adjoint variables, sensitivity analysis and real-time control'. *Journal of Computational and Applied Mathematics* (2000), vol. 120: pp. 85–108 (cit. on pp. 57, 69).

[Cox10] Cox, Jake: 'MSL Code Analysis'. *NASA IV&V Annual Workshop.* 2010 (cit. on p. 3).

[Die02] Diehl, M., H. G. Bock, J. P. Schlöder, R. Findeisen, Z. Nagy, and F Allgöwer: 'Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations'. *Journal of Process Control* (2002), vol. (cit. on p. 29).

[Don01] Dontchev, A. L. and William W. Hager: 'The Euler Approximation in State Constrained Optimal Control'. *Math. Comput.* (2001), vol. 70(233): pp. 173–203 (cit. on p. 58).

[Eve80] Evers, A.H.: 'Sensitivity analysis in dynamic optimization'. *Journal of Optimization Theory and Applications* (1980), vol. 32(1): pp. 17–37.

[Fia83] Fiacco, Anthony V.: *Introduction to Sensitivity and Stability Analysis in Nonlinear Programming.* Vol. 165. New York: Academic Press, 1983 (cit. on pp. 57, 59).

[Gei02] Geiger, Carl and Christian Kanzow: *Theorie und Numerik restringierter Optimierungsaufgaben.* ngerman. Springer, 2002 (cit. on pp. 46, 47).

[Ger09] Gerdts, Matthias: *Lecture notes in Optimal Control.* University Würzburg. 2009 (cit. on p. 150).

[Hag00] Hager, William W: 'Runge-Kutta methods in optimal control and the transformed adjoint system'. *Numerische Mathematik* (2000), vol. 87(2): pp. 247–282 (cit. on p. 58).

[Har79] Harpold, J. D. and C. A. Graves: 'Shuttle Entry Guidance'. *Journal of Astronautical Sciences* (1979), vol. 27(3): pp. 239–268 (cit. on p. 31).

[Hir09] Hirschel, Ernst and Claus Weiland: *Selected Aerothermodynamic Design Problems of Hypersonic Flight Vehicles.* Springer, 2009 (cit. on p. 9).

[Isi95] Isidori, Alberto: *Nonlinear Control Systems.* Ed. by Thoma, M., E. D. Sontag, B. W. Dickinson, A. Fettweis, J. L. Massey, and J. W. Modestino. 3rd. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1995 (cit. on p. 19).

[Kar09] Karlgaard, Christopher D., Roger E. Back, Stephen A. Keefe, Paul M. Siemers, and Brady A. White: 'Mars Entry Atmospheric Data System Modeling and Algorithm Development'. *41st AIAA Thermophysics Conference.* 2009 (cit. on p. 30).

[Lew99] Lewis, S. R., M. Collins, P. L. Read, F. Forget, F. Hourdin, R. Fournier, C. Hourdin, O. Talagrand, and J.-P. Huot: 'A climate database for Mars'. *Journal of Geophysical Research* (1999), vol. (cit. on pp. 8, 127).

[Lu98] Lu, Ping and John M. Hanson: 'Entry Guidance for the X-33 Vehicle'. *Journal of Spacecraft and Rockets* (1998), vol. 35(3) (cit. on p. 31).

[Mal97] Malanowski, K., C. Büskens, and H. Maurer.: 'Mathematical Programming with Data Perturbations'. Vol. 195. Lecture Notes in Pure and Applied Mathematics. A. V. Fiacco, 1997. Chap. Convergence of approximations to nonlinear optimal control problems: pp. 253–284 (cit. on p. 58).

[Mau95] Maurer, H. and H.J. Pesch: 'Solution differentiability for parametric nonlinear control problems with control-state constraints'. *Journal of Optimization Theory and Applications* (1995), vol. 86(2): pp. 285–309 (cit. on p. 57).

[Mau94] Maurer, Helmut and Hans Josef Pesch: 'Solution Differentiability for Nonlinear Parametric Control Problems'. *SIAM Journal on Control and Optimization* (1994), vol. 32(6): pp. 1542–1554 (cit. on p. 57).

[Mau09] Maurer, Helmut and Georg Vossen: 'Sufficient Conditions and Sensitivity Analysis for Optimal Bang-Bang Control Problems with State Constraints'. *System Modeling and Optimization*. Ed. by Korytowski, Adam, Kazimierz Malanowski, Wojciech Mitkowski, and Maciej Szymkat. Vol. 312. Springer Berlin Heidelberg, 2009: pp. 82–99 (cit. on p. 57).

[Moo94] Mooij, Erwin: *The Motion of a Vehicle in a Planetary Atmosphere*. Tech. rep. TU Delft, 1994 (cit. on p. 16).

[Per10] Perrotin, Maxime, Eric Conquet, Pierre Dissaux, Thanassis Tsiodras, and Jerome Hugues: 'The TASTE Toolset: turning human designed heterogeneous systems into computer built homogeneous software'. *European Congress on Embedded Real-Time Software (ERTS 2010)* (2010), vol. (cit. on pp. 127, 136).

[Pes89a] Pesch, Hans Josef: 'Real-time computation of feedback controls for constrained optimal control problems. part 1: Neighbouring extremals'. *Optimal Control Applications and Methods* (1989), vol. 10(2): pp. 129–145 (cit. on p. 60).

[Pes89b] Pesch, Hans Josef: 'Real-time computation of feedback controls for constrained optimal control problems. part 2: A correction method based on multiple shooting'. *Optimal Control Applications and Methods* (1989), vol. 10(2): pp. 147–171 (cit. on p. 60).

[Pes12] Pesch, Hans Josef and Michael Plail: 'The Cold War and the Maximum Principle of Optimal Control'. *Documenta Mathematica* (2012), vol.: pp. 331–344 (cit. on p. 41).

[Pra08] Prakash, Ravi: 'Mars Science Laboratory Entry, Descent, and Landing System Overview'. *Aerospace Conference, IEEE* (2008), vol. (cit. on p. 5).

[Roe93] Roenneke, Axel J. and Philipp J. Cornwell: 'Trajectory Control for a Low-Lift Re-Entry Vehicle'. *Journal of Guidance, Control, and Dynamics* (Sept. 1993), vol. 16(5) (cit. on p. 26).

[Sch14] Schomakers, Carina: 'Nichtlineare modellprädiktive Regelung für die Bahnplanung eines Wiedereintrittsproblems'. MA thesis. Universität Bremen, 2014 (cit. on p. 30).

[Son98] Sontag, Eduardo D.: *Mathematical Control Theory*. 2nd ed. Vol. 6. Texts in Applied Mathematics. Springer-Verlag New York, 1998 (cit. on p. 40).

[Spr11] Spreng, F.: 'Robust Skip Entry Guidance for Accurate Low Earth Orbit or Lunar Returns'. *8th International ESA Conference on Guidance, Navigation & Control Systems* (2011), vol. (cit. on p. 27).

[Sus97] Sussmann, Hector J. and Jan C. Willems: *300 Years Of Optimal Control: From The Brachystochrone To The Maximum Principle*. 1997 (cit. on p. 41).

[Tu98] Tu, Kuang-Yang, Mohammed S. Munir, Kenneth D. Mease, and David S. Bayard: 'Drag-Based Predictive Tracking Guidance for Mars Precision Landing'. *American Institute of Aeronautics and Astronautics* (1998), vol. (cit. on p. 125).

[Tu00] Tu, Kuang-Yang, Mohammed S. Munir, Kenneth D. Mease, and David S. Bayard: 'Drag-Based Predictive Tracking Guidance for Mars Precision Landing'. *Journal of Guidance* (2000), vol. 23(4) (cit. on pp. 24, 31).

[Vin80] Vinh, Nguyen X., Adolf Busemann, and Robert D. Culp: *Hypersonic and Planetary Entry Flight Mechanics*. Ed. by Arbor, Ann. The University of Michigan Press, 1980 (cit. on p. 7).

[Vin10] Vinter, Richard: *Optimal Control*. Modern Birkhäuser Classics. Boston, MA: Birkhäuser Boston Inc., 2010 (cit. on p. 41).

[Wal12] Walther, A. and A. Griewank: 'Getting started with ADOL-C'. *Combinatorial Scientific Computing*. Ed. by Naumann, U. and O. Schenk. Chapman-Hall CRC Computational Science, 2012. Chap. 7: pp. 181–202 (cit. on p. 127).

[Was13] Wassel, D. and C. Büskens: 'Modeling and Optimization in Space Engineering'. Vol. 73. Springer Optimization and Its Applications. Springer Verlag, 2013. Chap. The ESA NLP Solver WORHP (cit. on p. 126).

[Wei10] Weiland, Claus: *Computational Space Flight Mechanics*. Springer, 2010 (cit. on pp. 12, 13).

[Wol07] Wolf, Marco and Tobias Lutz: *Mars Precision Lander Entry Descent and Landing Analysis*. Tech. rep. EADS Astrium, 2007 (cit. on pp. 97, 112).

APPENDIX A

Coordinate Reference Frames

Table A.1: Coordinate frames used for the definition of the state vector

Planet Centered Inertial Frame		
Origin	Axis	Definition
Planet center	X_{PCI}	in the equatorial plane, towards the vernal equinox (J2000).
	Y_{PCI}	completes the right hand system.
	Z_{PCI}	is parallel to the planet's angular momentum vector.
Planet Centered Planet Fixed Frame		
Origin	Axis	Definition
Planet center	X_{PCPF}	lies in the equatorial plane, towards the zero meridian.
	Y_{PCPF}	completes the right hand system.
	Z_{PCPF}	is parallel to the planet's angular momentum vector.
North-East-Down Frame		
Origin	Axis	Definition
Vehicle CoM	X_{NED}	lies inside the local horizontal plane, pointing north.
	Y_{NED}	lies inside the local horizontal plane, pointing east.
	Z_{NED}	is normal to the local horizontal plane, pointing down.
Trajectory Axis Frame		
Origin	Axis	Definition
Vehicle CoM	X_{TA}	lies in direction of the ground velocity vector.
	Y_{TA}	completes the right hand system, pointing starboard.
	Z_{TA}	inside the vertical plane, spanned by Z_{NED} and the ground velocity vector.

Table A.2: Coordinate frames used for the definition of the aerodynamic attitude

Body Reference Frame		
Origin	Axis	Definition
Vehicle CoM	X_b	lies on the longitudinal vehicle axis, pointing ahead.
	Y_b	completes the right hand system, pointing starboard.
	Z_b	lies inside the vertical plane of symmetry, pointing down.
Vertical Air-Path Axis Frame		
Origin	Axis	Definition
Vehicle CoM	X_{av}	lies in direction of the air relative velocity vector.
	Y_{av}	completes the right hand system, pointing starboard.
	Z_{av}	inside the vertical plane, spanned by Z_{ned} and the air relative velocity vector.
Note: When there is no wind, the ground velocity and the air relative velocity are identical, then the air-path axis frame coincides with the trajectory axis frame.		
Air-Path Axis Frame		
Origin	Axis	Definition
Vehicle CoM	X_a	lies in direction of the air relative velocity vector.
	Y_a	completes the right hand system, pointing starboard.
	Z_a	inside the vehicle's vertical plane of symmetry, pointing down.
Note: When the vehicle is not banking, the air-path axis frame coincides with the vertical air-path axis frame.		

APPENDIX B

Spaces and Norms

Definition B.1 (Vector Space)
A vector space consists of a set V who's elements are called vectors, a field F who's elements are called scalars, and two operations:

1. *Vector addition takes two vectors $u, v \in V$ and produces a third vector $w \in V$, written $u + v = w$.*
2. *Scalar multiplication takes a scalar $a \in F$ and a vector $v \in V$ and produces a new vector $w \in V$, written $av = w$.*

The two operations must satisfy the following axioms for all $u, v, w \in V$ and $a, b \in F$:

$$\begin{aligned}
u + v &= v + u && \textit{commutativity of vector addition}\\
(u + v) + w &= u + (v + w) && \textit{associativity of vector addition}\\
u + 0 &= u && \textit{existence of additive identity, } 0 \in V\\
u + (-u) &= 0 && \textit{existence of additive inverse, } -u \in V\\
(ab)u &= a(bu) && \textit{associativity of scalar multiplication}\\
(a + b)u &= au + bu && \textit{distributivity of scalar sums}\\
a(u + v) &= au + av && \textit{distributivity of vector sums}\\
1u &= u && \textit{existence of multiplicative identity, } 1 \in F
\end{aligned}$$

Definition B.2 (Norm)
Given a vector space V over a field F a norm is a mapping $\|\cdot\| : V \to \mathbb{R}_0^+$ with the following properties for all $u, v \in V$ and $a \in F$:

$$\begin{aligned}
\|u\| = 0 &\Rightarrow u = 0 && \textit{zero vector}\\
\|au\| &= |a| \, \|u\| && \textit{absolute homogeneity}\\
\|u + v\| &\leq \|u\| + \|v\| && \textit{triangle inequality}
\end{aligned}$$

where $|\cdot|$ is the absolute value.

Theorem B.3 (Equivalence of Norms on Finite Dimensional Vector Spaces)
On a finite dimensional vector space V all norms are equivalent: For two norms $\|\cdot\|_a$ and

$\|\cdot\|_b$ *there exist positive, bounded constants* c, C *such that*

$$c\,\|v\|_a \leq \|v\|_b \leq C\,\|v\|_a\,, \quad v \in V.$$

Equipping a vector space V with a norm $\|\cdot\|$ allows to measure the distance between two vectors $u, v \in V$ by using the distance metric $d(u,v) = \|u - v\|$ induced by the norm. More generally a metric is defined as:

Definition B.4 (Metric)
A metric d *over a set* X *is a mapping* $d : X \times X \to \mathbb{R}_0^+$ *such that for all* $x, y, z \in X$

$$\begin{array}{ll} d(x,y) \geq 0 & \textit{non-negativity} \\ d(x,y) = 0 \Leftrightarrow x = y & \textit{identity of indiscernibles} \\ d(x,y) = d(y,x) & \textit{symmetry} \\ d(x,z) \leq d(x,y) + d(y,z) & \textit{triangle inequality} \end{array}$$

The pair (X,d) *is called metric space.*

Definition B.5 (Cauchy Sequence)
A sequence x_n, $n \in \mathbb{N}$ *of elements of a metric space* (X,d) *is called Cauchy sequence, if*

$$\textit{For all } \varepsilon > 0,\ \exists n_0 \in \mathbb{N},\ \textit{for all } n, m \geq n_0 :\ d(x_n,x) < \varepsilon.$$

A metric space (X,d) is called complete, if every Cauchy sequence in (X,d) is convergent. A common strategy to show the convergence of a sequence if the limit is unknown, is showing that the sequence is a contraction and then applying the Fixpoint Theorem of Banach:

Definition B.6 (Contraction)
Let (X,d) *be a metric space and* $A \subseteq X$ *a complete subspace. A contraction on* A *is a mapping* $f : A \to A$*, with the property that* $\exists L$, $0 \leq L < 1$ *such that for all* $x, y \in A$

$$d(f(x),f(y)) \leq L d(x,y)$$

The smallest value L *is called the Lipschitz constant of* f*.*

Theorem B.7 (Fixpoint Theorem of Banach)
Let (X,d) *be a metric space and* $A \subseteq X$ *a complete subspace. Furthermore let* $f : A \to A$ *be a contraction on* (A,d)*. Then the sequence* $x_{k+1} = f(x_k)$ *converges to a uniquely determined fixpoint* $\overset{\star}{x}$ *for all start values* $x_s \in A$*.*

For later purposes in the following important spaces and norms are introduced, oriented at [Ger09].

Definition B.8 (Banach space)
A complete and normed vector space is called Banach space.

Definition B.9 (Space of Continuous Functions)
$C([a,b],\mathbb{R})$ *is the space of continuous functions* $x : [a,b] \to \mathbb{R}$*.*

Definition B.10 (Space of k-Times Continuously Differentiable Functions)
$C^k([a,b],\mathbb{R})$, $k > 0$ *is the space of* k*-times continuously differentiable functions* $x : [a,b] \to \mathbb{R}$*.*

Definition B.11 (L^p-Norm)
The L^p-norm is

$$\|x\|_p := \left(\int_a^b |x(t)|^p \, \mathrm{d}t \right)^{\frac{1}{p}} \qquad 1 \leq p \leq \infty \tag{B.11a}$$

$$\|x\|_\infty := \sup_{a \leq t \leq b} |x(t)| \tag{B.11b}$$

C^1 equipped with the norm $\| \cdot \|_\infty$ is a Banach space that is of importance in the calculus of variation. In Optimal Control the problems are usually defined on less restrictive function spaces than C^1. Commonly used are the Lebesque space L^∞ and the Sobolev space $W^{1,\infty}$.

Definition B.12 (Lebesque Space)
$L^\infty([a,b],\mathbb{R})$ *is the space of all measurable functions* $x : [a,b] \to \mathbb{R}$ *which are essentially bounded:*

$$\mathit{ess} \sup_{a \leq t \leq b} |x(t)| := \inf_{\substack{N \subset [a,b] \\ \mu(N)=0}} \sup_{t \in [a,b] \setminus N} |x(t)| < \infty \tag{B.12a}$$

$L^\infty([a,b],\mathbb{R})$ *is a Banach space with the norm*

$$\|x\|_\infty := \mathit{ess} \sup_{a \leq t \leq b} |x(t)| \tag{B.12b}$$

Definition B.13 (Sobolev Spaces)
$W^{q,p}([a,b],\mathbb{R})$ *is the space of all absolutely continuous functions* $x : [a,b] \to \mathbb{R}$ *that have absolutely continuous derivatives up and including to order* $q-1$ *and it holds that*

$$\|x\|_{q,p} < \infty. \tag{B.13a}$$

The norm $\| \cdot \|_{q,p}$ *is given by*

$$\|x\|_{q,p} := \left(\sum_{k=0}^{q} \|x^{(k)}\|_p^p \right)^{\frac{1}{p}} \tag{B.13b}$$

$$\|x\|_{q,\infty} := \max_{0 \leq k \leq q} \|x^{(k)}\|_\infty \tag{B.13c}$$

where $| \cdot |^{(k)}$ *denotes the derivative of order* k*.* $\| \cdot \|_p$ *denotes the* L^p*-norm.* $W^{q,p}([a,b],\mathbb{R})$*,* $1 \leq q,\ p \leq \infty$ *with the norm* $\| \cdot \|_{q,p}$ *are Banach spaces.*

APPENDIX C

Strong and Weak Minima

Admissible function pairs (x,u) for the solution of an OCP are compared based on the metric induced by the norm of their vector space. A minimum is therefore characterized by the underlying norm. In optimal control the control function is commonly assumed to be in the Lebesque space L^∞. A control function with points of discontinuity results into a state trajectory which is continuous, but not continuously differentiable everywhere. The trajectory is a piecewise continuously differentiable function belonging to the Sobolev space $W^{1,\infty}$. The minimum of an optimal control problem can be characterized based on the underlying norm:

Definition C.1 (Strong Minimum)
The pair $(\overset{\star}{x},\overset{\star}{u}) \in W^{1,\infty} \times L^\infty$ is called strong minimum of problem (4.7) if it exists $\varepsilon > 0$ such that

$$J(\overset{\star}{x},\overset{\star}{u}) \leq J(x,u)$$

for all admissible (x,u) with $\|x - \overset{\star}{x}\|_\infty < \varepsilon$.

Definition C.2 (Weak Minimum)
The pair $(\overset{\star}{x},\overset{\star}{u}) \in W^{1,\infty} \times L^\infty$ is called weak minimum of problem (4.7) if it exists $\varepsilon > 0$ such that

$$J(\overset{\star}{x},\overset{\star}{u}) \leq J(x,u)$$

for all admissible (x,u) with $\|x - \overset{\star}{x}\|_{1,\infty} < \varepsilon$ and $\|u - \overset{\star}{u}\|_\infty < \varepsilon$.

A strong minimum is also a weak minimum but not vice versa. A weak minimum is more restrictive concerning the admissible comparable control functions: For a weak minimum the ε-neighborhood around a control function with discontinuities only includes control functions with identical points of discontinuity. For a strong minimum the set of compared control functions includes those with variations in the points of discontinuity.

In optimal control the interest is on finding a strong minimum.

APPENDIX D

Factorized Form of the Drag Dynamics

In the derivation of the drag control law we use the fact that we can factor out u_{vLoD} from the second energy derivative of drag.

$$D'' = a(D,D',h,v,\gamma) + b(D,D',h,v,\gamma)u_{\mathrm{vLoD}}.$$

The linearizing terms a and b are:

$$\begin{aligned}
a = {} & \frac{1}{4D^2vc^4r^4}S\Bigg[D^2v^5\rho r^4\left(\frac{\mathrm{d}c}{\mathrm{d}h}\right)^2\left(\frac{\mathrm{d}h}{\mathrm{d}E}\right)^2\frac{\mathrm{d}^2C_D}{\mathrm{d}M^2}+4D^2v^4c\rho r^4\frac{\mathrm{d}c}{\mathrm{d}h}\frac{\mathrm{d}h}{\mathrm{d}E}\left(\frac{\mathrm{d}c}{\mathrm{d}h}\frac{\mathrm{d}h}{\mathrm{d}E}\frac{\mathrm{d}C_D}{\mathrm{d}M}-\frac{\mathrm{d}v}{\mathrm{d}E}\frac{\mathrm{d}^2C_D}{\mathrm{d}M^2}\right)- \\
& v^3c^2r^2\Bigg[4D^2vr^2\frac{\mathrm{d}c}{\mathrm{d}h}\left(\frac{\mathrm{d}h}{\mathrm{d}E}\right)^2\frac{\mathrm{d}\rho}{\mathrm{d}h}\frac{\mathrm{d}C_D}{\mathrm{d}M}+ \\
& 2\rho\Bigg[\frac{\mathrm{d}C_D}{\mathrm{d}M}\Bigg[D^2vr^2\frac{\mathrm{d}^2c}{\mathrm{d}h^2}\left(\frac{\mathrm{d}h}{\mathrm{d}E}\right)^2+\frac{\mathrm{d}c}{\mathrm{d}h}\left(r^2\left(6D^2\frac{\mathrm{d}h}{\mathrm{d}E}\frac{\mathrm{d}v}{\mathrm{d}E}+v\sin(\gamma)\frac{\mathrm{d}D}{\mathrm{d}E}\right)+D\cos^2(\gamma)\left(\mu_M-v^2r\right)\right)\Bigg]- \\
& D^2r^2\left(\frac{\mathrm{d}v}{\mathrm{d}E}\right)^2\frac{\mathrm{d}^2C_D}{\mathrm{d}M^2}\Bigg]\Bigg]+vc^3\frac{\mathrm{d}C_D}{\mathrm{d}M}\Bigg[\rho\Bigg[2D^2r^4\frac{\mathrm{d}v}{\mathrm{d}E}\left(4v\frac{\mathrm{d}v}{\mathrm{d}E}-1\right)- \\
& 2\mu_Mr\sin(\gamma)\left(vr\frac{\mathrm{d}D}{\mathrm{d}E}+Dr\frac{\mathrm{d}v}{\mathrm{d}E}+2Dv\frac{\mathrm{d}h}{\mathrm{d}E}\right)+2D\mu_M\cos^2(\gamma)\left(v^2r-\mu_M\right)\Bigg]+4D^2v^2r^4\frac{\mathrm{d}h}{\mathrm{d}E}\frac{\mathrm{d}v}{\mathrm{d}E}\frac{\mathrm{d}\rho}{\mathrm{d}h}\Bigg]+c^4C_D \\
& \Bigg[v^2r^2\left(\frac{\mathrm{d}\rho}{\mathrm{d}h}\left(2r^2\left(4D^2\frac{\mathrm{d}h}{\mathrm{d}E}\frac{\mathrm{d}v}{\mathrm{d}E}+v\sin(\gamma)\frac{\mathrm{d}D}{\mathrm{d}E}\right)+2D\cos^2(\gamma)\left(\mu_M-v^2r\right)\right)+2D^2vr^2\left(\frac{\mathrm{d}h}{\mathrm{d}E}\right)^2\frac{\mathrm{d}^2\rho}{\mathrm{d}h^2}\right)+ \\
& 4\rho\left(D^2r^4\frac{\mathrm{d}v}{\mathrm{d}E}\left(v\frac{\mathrm{d}v}{\mathrm{d}E}-1\right)-\mu_Mr\sin(\gamma)\left(vr\frac{\mathrm{d}D}{\mathrm{d}E}+Dr\frac{\mathrm{d}v}{\mathrm{d}E}+2Dv\frac{\mathrm{d}h}{\mathrm{d}E}\right)+D\mu_M\cos^2(\gamma)\left(v^2r-\mu_M\right)\right)\Bigg]\Bigg]
\end{aligned}$$

$$b = \frac{S\cos(\gamma)}{2vc^2r^2}\left[v^3\rho r^2\frac{\mathrm{d}c}{\mathrm{d}h}\frac{\mathrm{d}C_D}{\mathrm{d}M}+\mu_Mvc\rho\frac{\mathrm{d}C_D}{\mathrm{d}M}+c^2C_D\left(2\mu_M\rho-v^2r^2\frac{\mathrm{d}c}{\mathrm{d}h}\right)\right]$$

The drag tracking input is D,D',h,v,γ and the derivatives $\frac{\mathrm{d}c}{\mathrm{d}h}$, $\frac{\mathrm{d}^2c}{\mathrm{d}h^2}$, $\frac{\mathrm{d}\rho}{\mathrm{d}h}$, $\frac{\mathrm{d}^2\rho}{\mathrm{d}h^2}$, $\frac{\mathrm{d}C_D}{\mathrm{d}M}$, $\frac{\mathrm{d}^2C_D}{\mathrm{d}M^2}$ are available numerically as part of the model and must only be evaluated at the current state. The energy derivatives of the states, the drag and the Mach number have been derived in Chapter 3.1.2 and can also be evaluated at the current state. Thus all terms in a and b are known.

APPENDIX E

NLP Formulation Details

E.1 Variable Ordering

To reference the control and state variables on the grids T_u and T_x the matrices (E.1) are defined.

$$U \in \mathbb{R}^{l_u} \times \mathbb{R}^{n_u} \tag{E.1a}$$

$$X \in \mathbb{R}^{l_x} \times \mathbb{R}^{n_x} \tag{E.1b}$$

The rows of U are the control vectors on the grid $T_u = \{\tau^{(1)},...,\tau^{(l_u)}\}$. An element $u^{(i,j)} \subset U$ is the value of control channel j, $1 \leq j \leq n_u$ at $\tau^{(i)} \subset T_u$. A column $u^{(:,j)} \in \mathbb{R}^{l_u} \subset U$ is the discretized control function for control channel j. A column $u^{(i,:)} \in \mathbb{R}^{n_u} \subset U$ is the control vector at normalized time $\tau^{(i)} \in T_u$. Analogously the rows of matrix X are the state vectors at the shooting nodes T_x.

The logical ordering of the decision variables is chosen as

$$z := \begin{pmatrix} (u^{(:,1)}) \\ \vdots \\ (u^{(:,n_u)}) \\ (x^{(:,1)}) \\ \vdots \\ (x^{(:,n_x)}) \\ t_d \end{pmatrix}, \quad |z| = n_z = n_u l_u + n_x l_x + 1. \tag{E.2}$$

The ordering prioritizes the variable type over the dimension index, followed by the temporal index. This allows to easily extract state trajectories and control sequences as coherent blocks from the decision variables.

E.2 Constraint Ordering

The state at the shooting nodes is part of the decision variables, but the state at intermediate points must be obtained via the recursive evaluation of the discretized dynamics on each shooting interval. Let $l_{S_1},...,l_{S_{l_x-1}}$ be the number of control grid points in the shooting intervals $S_1,...,S_{l_x-1}$. Let

$$\tilde{X}_{S_k} \in \mathbb{R}^{l_{S^k}} \times \mathbb{R}^{n_x}, \quad 1 \leq k < l_x - 1, \tag{E.3}$$

be the state trajectory obtained from integration of the control function on interval S^k. An element $\tilde{x}_{S_k}^{(h,j)} \in \tilde{X}_{S_k}$ is the value of state function j, $1 \leq j \leq n_x$ after $h-1$, $1 \leq h \leq l_{S^k}$ integration steps in shooting interval k, $1 \leq k < l_x$. When the differential constraints are satisfied the last entry in each shooting interval matches the state at the next shooting node (or the terminal point). Thus the defect $d_{(k,j)}$ of state function j at $\tau^{(k)} \in T_x$ can be written as the defect vector

$$D = \begin{pmatrix} d^{(1,1)} \\ d^{(2,1)} \\ \vdots \\ d^{(l_x-1,1)} \\ d^{(1,2)} \\ d^{(2,2)} \\ \vdots \\ d^{(l_x-1,2)} \\ \vdots \\ d^{(1,n_x)} \\ d^{(2,n_x)} \\ \vdots \\ d^{(l_x,n_x)} \end{pmatrix} = \begin{pmatrix} \tilde{x}_{(S_1)}^{(l_{S_1},1)} - \tilde{x}_{(S_2)}^{(1,1)} \\ \tilde{x}_{(S_2)}^{(l_{S_2},1)} - \tilde{x}_{(S_3)}^{(1,1)} \\ \vdots \\ \tilde{x}_{(S_{l_x-1})}^{(l_{S_{l_x-1}},1)} - x^{(l_u,1)} \\ \tilde{x}_{(S_1)}^{(l_{S_1},2)} - \tilde{x}_{(S_2)}^{(1,2)} \\ \tilde{x}_{(S_2)}^{(l_{S_2},2)} - \tilde{x}_{(S_3)}^{(1,2)} \\ \vdots \\ \tilde{x}_{(S_{l_x-1})}^{(l_{S_{l_x-1}},2)} - x^{(l_u,2)} \\ \vdots \\ \tilde{x}_{(S_1)}^{(l_{S_1},n_x)} - \tilde{x}_{(S_2)}^{(1,n_x)} \\ \tilde{x}_{(S_2)}^{(l_{S_2},n_x)} - \tilde{x}_{(S_3)}^{(1,n_x)} \\ \vdots \\ \tilde{x}_{(S_{l_x-1})}^{(l_{S_{l_x-1}},n_x)} - x^{(l_u,n_x)} \end{pmatrix} = 0. \tag{E.4}$$

The path constraints $c(x,u)$ are evaluated at every point of T_u using the integrated state vectors $\tilde{x}_{S_k}^{(i,:)}$ given by the rows of the matrices $\tilde{X}_{S_k}$. The discretized path constraints can then be written as:

$$C = \begin{pmatrix} c^{(1,1)} \\ \vdots \\ c^{(l_u,1)} \\ c^{(1,2)} \\ \vdots \\ c^{(l_u,n_c)} \end{pmatrix} = \begin{pmatrix} c(\tilde{x}_{(S_1)}^{(1,:)}, u^{(1,:)}) \\ \vdots \\ c(\tilde{x}_{(S_{l_x-1})}^{(l_{S_{l_x-1}},:)}, u^{(l_u,:)}) \\ c(\tilde{x}_{(S_1)}^{(1,:)}, u^{(1,:)}) \\ \vdots \\ c(\tilde{x}_{(S_{l_x-1})}^{(l_{S_{l_x-1}},:)}, u^{(l_u,:)}) \end{pmatrix} \leq 0. \tag{E.5}$$

Element $c^{(i,j)}$ corresponds to path constraint j, $1 \leq j \leq n_c$ evaluated at $\tau^{(i)} \in T_u$.

Constraints on the admissible control set $u \in \mathbb{U} \subset \mathbb{R}^{n_u}$ can either be implemented as additional path constraints, or alternatively most NLP solvers support the use of simple lower and upper bounds on the decision variables. These so called box constraints must be constant values. Thus the corresponding entries in the Hessian matrix ∇L_z^2 are zero, which can be exploited by the NLP solver. If possible the use of box constraints is preferable from a numeric point of view compared to using additional path constraints.

The boundary constraints Ψ are

$$\Psi = \begin{pmatrix} \psi^{(1,1)}\left(x^{(1,1)}\right) \\ \vdots \\ \psi^{(1,n_{r_s})}\left(x^{(1,n_{r_s})}\right) \\ \psi^{(2,1)}\left(x^{(l_u,1)}\right) \\ \vdots \\ \psi^{(2,n_{r_f})}\left(x^{(l_u,n_{r_s})}\right) \end{pmatrix} = 0 \tag{E.6}$$

All constraints are joined in the function $g : \mathbb{R}^{n_z} \to \mathbb{R}^{n_g}$, with $n_g = n_x l_x + n_c l_u + n_{r_s} + n_{r_f}$.

$$g(z)^{(m)} := \begin{pmatrix} D \\ \Psi \end{pmatrix} = 0 \quad m = 1,...,n_{eq} \tag{E.7a}$$

$$g(z)^{(m)} := \quad C \quad \leq 0 \quad m = n_{eq} + 1,...,n_g \tag{E.7b}$$

This ordering results into a diagonally banded structure in the blocks C and D. If the temporal index would be prioritized higher than the component index the result would be a block diagonal structure.

APPENDIX F

NLP Scaling

NLPs are predominately solved using Newton or Quasi-Newton methods. The convergence rate of these algorithms depends on the conditioning of the KKT matrix (4.63). Scaling techniques aim at improving the condition of the KKT matrix. Even if theoretically an optimal solution is attainable, ill conditioned problems prevent the solution for numerical reasons. In practice problem scaling thus very important. In the following the quantities of a scaled NLP are stated following Büskens [Büs98].

Let $a^{(i)} \in \mathbb{R}^+, i = 1,\dots,n_z$ be the scale factors of the decision variables $z \in \mathbb{R}^{n_z}$ and let $b^{(i)} \in \mathbb{R}^+, i = 1,\dots,n_g$ be the scale factors of the constraint function $g \in \mathbb{R}^{n_g}$. Let $A := diag(a^{(1)},\dots,a^{(n_z)}) \in \mathbb{R}^{n_z} \times \mathbb{R}^{n_z}$, $B := diag(b^{(1)},\dots,b^{(n_g)}) \in \mathbb{R}^{n_g} \times \mathbb{R}^{n_g}$ and $\alpha \in \mathbb{R}^+$. Then the scaled system quantities, denoted by subscript s, can be written as:

$$z_s = Az, \tag{F.1a}$$
$$f_s(z) = \alpha f(z), \tag{F.1b}$$
$$g_s(z) = Bg(z), \tag{F.1c}$$
$$\mu_s = \alpha E^{-1}\mu, \tag{F.1d}$$
$$L_s(z_s,\mu_s) = f_s(z_s) + \mu_s^\intercal g_s(z_s) \tag{F.1e}$$

The scaled Jacobian with respect to the scaled variables can be written in terms of the unscaled quantities as

$$\nabla_{z_s} g_s(A^{-1}z_s) = \nabla_{z_s} Bg(A^{-1}z_s) = B\ \nabla_z g(z)\ A^{-1}. \tag{F.2}$$

The Hessian with respect to the scaled variables and function can be obtained as:

$$\begin{aligned}\nabla^2_{z_s} L_s(z_s,\mu_s) &= \nabla^2_{z_s} f_s(A^{-1}z_s) + \nabla^2_{z_s} \mu_s^\intercal g_s(A^{-1}z_s) \\ &= \alpha\nabla^2_{z_s} f(A^{-1}z_s) + \nabla^2_{z_s} \alpha(B^{-1}\mu)^\intercal Bg(A^{-1}z_s) \\ &= \alpha\nabla^2_{z_s} f(A^{-1}z_s) + \nabla^2_{z_s} \alpha\mu^\intercal g(A^{-1}z_s) \\ &= \alpha\nabla^2_{z_s} L(A^{-1}z_s,\mu) \\ &= \alpha A^{-1}\nabla^2_z L(z,\mu)A^{-1}.\end{aligned} \tag{F.3}$$

The remaining task is to chose the scale factors A, B and α. Finding optimal scale factors is not a trivial task that leads to another nonlinear optimization problem [Büs98].

A simpler, heuristic approach based on the optimization start value is suggested in [Bet10]: For an OCP the NLP scaling can be determined using the magnitude of the physical variables in the OCP regime. All decision variables and constraints are normalized to an interval $I = [-r,r]$, $r \in \mathbb{R}^+$. Only one scale factor is determined for each dimension of the state, the control and the path constraints.

It is heuristically assumed that the magnitude of the state/control/constraint dimensions will stay roughly the same during the control process, thus the same scale factor is used at all discretization points. In practice this has shown to result into an improved condition of the constraint Jacobian.

Determining the scale factors automatically requires a nontrivial optimization start value $\tilde{z} \neq 0$. Let $\tilde{x}^{(i,j)}$ be the start value for state function j, $1 \leq j \leq n_x$, at $\tau^{(i)} \in T_x$. The scale factors of the state functions are determined as

$$a_x^{(1)} := \min_i \frac{r}{|\tilde{x}^{(i,1)}|}, \quad 1 \leq i \leq l_x, \quad |\tilde{x}^{(i,1)}| \neq 0 \tag{F.4a}$$

$$\vdots$$

$$a_x^{(n_x)} := \min_i \frac{r}{|\tilde{x}^{(i,n_x)}|}, \quad 1 \leq i \leq l_x, \quad |\tilde{x}^{(i,n_x)}| \neq 0. \tag{F.5a}$$

Let $A_x = \operatorname{diag}(a_x^{(1)},\dots,a_x^{(n_x)}) \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_x}$. Then the scaled state variables are

$$x_s = A_x x. \tag{F.6}$$

The scale factors $a_u^{(1)},\dots,a_u^{(n_u)}$ of the control function are determined analogously.

The scale factors $b_c^{(1)},\dots,b_c^{(n_c)}$ of the path constraints are determined by evaluating the path constraints at the optimization start value and finding their maximal value:

$$b_c^{(1)} := \min_i \frac{r}{|c^{(i,1)}(\tilde{x}^{(i,:)}, \tilde{u}^{(i,:)})|}, \quad 1 \leq i \leq l_u, \quad c^{(i,1)} \neq 0 \tag{F.7a}$$

$$\vdots$$

$$b_c^{(n_c)} := \min_i \frac{r}{|c^{(i,n_c)}(\tilde{x}^{(i,:)}, \tilde{u}^{(i,:)})|}, \quad 1 \leq i \leq l_u, \quad c^{(i,n_c)} \neq 0. \tag{F.8a}$$

The defect constraints D (E.4) and boundary constraints Ψ (E.6) are scaled using the already known scale factors of the state $a_x^{(1)},\dots,a_x^{(n_x)}$. Thus B is given by using $a_x^{(1)},\dots,a_x^{(n_x)}$ and $b_c^{(1)},\dots,b_c^{(n_c)}$.

The scale factor α of the objective function is set as

$$\alpha := \frac{r}{|f(\tilde{z})|}. \tag{F.9}$$

This scales the objective and the constraints roughly to the same magnitude.

For ill conditioned problems with an inaccurate initial guess it has proven successful to recalculate the scale factors after a number of Newton steps.

APPENDIX G

Entry Optimal Control Problem: Objective Function Weights

The weighting coefficients used to obtain the nominal solution discussed in Section 6.3.1 are shown in the following table.

Table G.1: Objective function weights

Term	**Weight**	**Value**	**Time Domain**	**Energy Domain**	**Percent**
Terminal velocity	w_V	$1 \cdot 10^{-3}$	0.136	0.137	∼8 %
Heat load	w_H	$1 \cdot 10^{1}$	0.491	0.508	∼31 %
Vertical lift	w_L	$5 \cdot 10^{2}$	0.608	0.582	∼38 %
Bank angle rate	w_R	$1 \cdot 10^{-5}$	0.359	0.367	∼23 %
Optimal value			1.595	1.596	

APPENDIX H

Algorithm for the Approximation of the Correction Space Boundary

Considering a disturbed parameter vector

$$p = p_0 + \varDelta p = p_0 + \alpha \frac{\varDelta p}{\|\varDelta p\|} = p_0 + \alpha\, d, \quad \alpha \in \mathbb{R}, \quad d \in \mathbb{R}^{n_p},\ \|d\| = 1. \tag{H.1}$$

the goal is to find

$$\alpha_{\min}(p_0, d) := \arg\min_{\alpha}\ \varTheta(p_0 + \alpha d), \quad \varTheta(p_0 + \alpha d) \text{ feasible}, \tag{H.2a}$$

$$\alpha_{\max}(p_0, d) := \arg\max_{\alpha}\ \varTheta(p_0 + \alpha d), \quad \varTheta(p_0 + \alpha d) \text{ feasible}. \tag{H.2b}$$

One option to approximate the values $\alpha_{\min}(p_0\, d)$ and $\alpha_{\max}(p_0\, d)$ is a brute-force method, combining line search and binary search, as in Algorithm (4). The algorithm arguments are the nominal parameter value p_0, the perturbation direction d, the percentile accuracy $acc \in [0;\, 1]$, the line search step size $inc > 1$ and the step size sign $sgn \in \{-1, 1\}$.

The algorithm performs line search on in direction d until the feedback law fails. The boundary is then narrowed down using binary search in the interval between the last successfully handled perturbation and the first failure, until the desired accuracy is reached. For $sgn = 1$ the result is $\alpha_{\max}(p_0, d)$, for $sgn = -1$ the result is $\alpha_{\min}(p_0, d)$.

Algorithm 4 Find correction space boundary

```
procedure FindBoundary($p_0, d, acc, inc, sgn$)
    $\alpha := 1$
    $fail := 0$
    $success := 0$
    $diff := \infty$
    while $fail == 0$ or $diff > acc$ do
        $p = p_0 + sgn\,\alpha\,d$
        $\{\tilde{u}(p), \tilde{x}(p)\} \leftarrow$ Feedback($p$)
        if isAdmissible($\tilde{u}(p), \tilde{x}(p)$) then
            $success = \alpha$
        else
            $fail = \alpha$
        end if
        if $fail == 0$ then
            $\alpha = inc\,\alpha$
        else
            $\alpha = 0.5\,(fail + success)$
        end if
        if $success \neq 0$ then
            $diff = |(fail - success)/(success)|$
        end if
    end while
    return $success$
end procedure
```