

CAMTOS - A Software Suite Combining Direct and Indirect Trajectory Optimization Methods

Von der Fakultät Luft- und Raumfahrttechnik und Geodäsie der Universität Stuttgart
zur Erlangung der Würde eines Doktor-Ingenieurs (Dr.-Ing.)
genehmigte Abhandlung

Vorgelegt von

Peter Friedrich Gath

aus Waiblingen

Hauptberichter:	Prof. Klaus H. Well, Ph.D.
Mitberichter:	Prof. Anthony J. Calise, Ph.D.
	Prof. Dr.-Ing. Arnold Kistner
	Honorarprof. Dr.-Ing. Eveline Gottzein

Tag der mündlichen Prüfung:	29. November 2002
-----------------------------	-------------------

Institut für Flugmechanik und Flugregelung
Universität Stuttgart

2002

Abstract

Many engineering applications involve the solution of optimal control problems. Such problems cover a wide variety of different disciplines, starting from typical aerospace applications, e.g. a fuel minimal launcher ascent trajectory, up to more exotic applications such as the optimal motion of the human body in different sport disciplines.

This thesis presents a new method for solving a general class of multi-phase trajectory optimization problems. This new method uses a combination of direct multiple shooting and direct collocation. Depending on the specific demands of the problem to solve, a different kind of transcription method can be used for each phase. In addition, the method can be combined with the indirect method. Once the adjoint differential equations are available in addition to the dynamic system, one or more phases can be modeled as indirect phases. Since the transversality conditions that are usually required for solving the multi-point boundary value problem are contained in the Karush-Kuhn-Tucker conditions of the NLP-solver, they need not be formulated explicitly.

The benefits of combining direct and indirect methods is demonstrated on an Ariane 5 dual payload mission. While the atmospheric parts of this mission are modelled with the direct multiple-shooting technique, the upper stage is using the indirect method. An analysis of the switching function, which is generated as a by-product of the indirect method, yields an improved mission profile. This profile involves an additional coast-arc and improves the payload performance by 66%.

An additional example demonstrates the advantages of using direct collocation and direct multiple shooting to solve a complex branched trajectory optimization problem. The ascent and return of the suborbital Hopper vehicle is optimized such that the payload delivered to the geostationary transfer orbit by an upper stage is maximized. The robustness and large convergence radius of the direct collocation method significantly simplifies the generation of an approximate solution of the problem which is then refined by changing the transcription method of the atmospheric parts of the trajectory to the direct multiple shooting technique.

Zusammenfassung

Viele technische Anwendungen beinhalten die Lösung von Optimalsteuerungsproblemen. Derartige Probleme decken einen weiten Bereich verschiedener Fachrichtungen ab, angefangen bei typischen Raumfahrtanwendungen, wie z.B. der treibstoffoptimale Aufstieg einer Rakete, bis hin zu exotischeren Gebieten wie z.B. der optimalen Bewegung des menschlichen Körpers in verschiedenen Sportdisziplinen.

Diese Arbeit stellt eine neue Methode zur Lösung einer allgemeinen Klasse von mehrphasigen Bahnoptimierungsproblemen vor. Dieses Verfahren verwendet eine Kombination aus direktem Mehrzielverfahren und direkter Kollokation. In Abhängigkeit von den spezifischen Anforderungen des zu lösenden Problems kann für jede Phase die geeignete Transkriptionsmethode ausgewählt werden. Außerdem kann das Verfahren mit der indirekten Methode kombiniert werden. Sobald neben der Systemdynamik auch die Adjungierten-Differentialgleichungen verfügbar sind, können eine oder mehrere Phasen auch mit Hilfe der indirekten Methode modelliert werden. Da die Transversalitätsbedingungen bereits in den Karush-Kuhn-Tucker Bedingungen des NLP-Verfahrens enthalten sind, müssen sie nicht explizit formuliert werden.

Die Vorteile einer Kombination aus direktem und indirektem Verfahren wird am Beispiel einer Ariane 5 Mission mit zwei Nutzlasten demonstriert. Dabei werden die Missionsabschnitte innerhalb der Atmosphäre mit Hilfe des direkten Mehrzielverfahrens modelliert, während das indirekte Verfahren in der Oberstufe zum Einsatz kommt. Eine Analyse der Schaltfunktion, die sich als Nebenprodukt des indirekten Verfahrens ergibt, führt dabei zu einer Verbesserung des Missionsablaufes. Durch Hinzufügen einer zusätzlichen Freiflugphase kann die Nutzlast um 66% verbessert werden.

Ein weiteres Beispiel zeigt die Vorteile einer Kombination aus direkter Kollokation und direktem Mehrzielverfahren für die Lösung eines verzweigten Bahnoptimierungsproblems. Die Aufstiegs- und Rückkehrbahn des suborbitalen Hopper-Fahrzeuges wird derart optimiert, dass die durch eine Oberstufe in den geostationären Transferorbit beförderte Nutzlast maximiert wird. Dabei vereinfacht die Robustheit und der große Konvergenzradius des direkten Kollokationsverfahrens die Berechnung einer Näherungslösung erheblich. Durch eine Umschaltung der Transkription auf die direkte Mehrzielmethode in den atmosphärischen Abschnitten der Trajektorie ermöglicht schließlich die Berechnung einer genauen Lösung.

Acknowledgements

*The dream of yesterday
is the hope of today
and the reality of tomorrow.*

Robert H. Goddard, 1904

This work was carried out during my time as a research scientist at the Institute of Flight Mechanics and Control (IFR) at the University of Stuttgart.

I greatly appreciate and value the guidance and support of my advisor and director of IFR, Professor Klaus H. Well. His experience and passion for trajectory optimization always encouraged and motivated me during the past few years. In addition, he always gave me a lot of freedom to carry out my ideas.

The software that was developed in the framework of this thesis is part of the HISTOS project that was financed by the European Space Research and Technology Center ESTEC. I would like to acknowledge this support and thank our project monitor, Dr. Klaus H. Mehlem from ESTEC, for many useful discussions and giving me the required freedom within this project to write this thesis. I was always looking forward to our interesting progress meetings during the course of the HISTOS project.

My sincerest thanks also go to Professor Anthony J. Calise who has introduced me to the methods of indirect trajectory optimization during my stay at the Georgia Institute of Technology in 1998 and who was member of the reading committee. Ever since, I always look forward to meeting him again and I deeply admire his tremendous engineering expertise and knowledge.

I would also like to express my gratitude to Professor Arnold Kistner and Professor Eveline Gottzein who were also members of the reading committee.

Dr. Werner Grimm did an excellent job in proof-reading one of the first versions of this thesis report, and I thank him very much for many discussions about mathematical problems and details.

I would also like to thank Prof. Philip E. Gill for kindly providing his SQP-solver SNOPT which serves as a fundamental basis of the new trajectory optimization method.

Moreover, I would like to express my special thanks to three individuals from The Boeing Company: Dr. John T. Betts and Steve Paris for valuable discussions about direct collocation methods, optimal control and some numerical “tricks”, and Rocky Nelson for many discussions about launcher trajectory optimization and the application of SNOPT.

Andreas Wiegand did an outstanding job in enhancing GESOP’s graphical user interface, implementing branching into ASTOS, updating PROMIS and TROPIC to the latest version of SNOPT, and assisting me during the implementation of the indirect method into the ASTOS library. I always appreciated our discussions about several software engineering problems and the enhancement of the ASTOS model library, as well as the software training sessions and presentations we performed together for our GESOP and ASTOS customers.

Working within the GESOP/ASTOS team, together with Denis Fischer, Jörg Kindler, Dr. Albert Markl, and Andreas Wiegand, was a great pleasure and experience for me. The many discussions we had with numerous customers have proved to be very beneficial also for the development and implementation of the methods presented in this thesis.

I always enjoyed the excellent work atmosphere at IFR and the very good team spirit, not only within the GESOP/ASTOS team, but also between the other teams at IFR. I would like to thank all of my colleagues and our students for many technical and not-so-technical discussions we had in the past few years. My special thanks go to our system administrator Dirk Wimmer, who took over the burden of taking care of our Windows NT environment with great passion in addition to his own research project.

Last but not least, I would like to thank my parents, my friends, and especially Ms. Barbara Nucera for their ever-lasting support and encouragement during this endeavour.

Markdorf, December 2002

Peter F. Gath

Table of Contents

Abstract	3
Zusammenfassung	4
Acknowledgements	5
Table of Contents	7
Table of Figures	13
List of Tables	19
Nomenclature	21
1 Introduction	25
1.1 Motivation	25
1.2 General Problem Formulation	27
1.3 Indirect Trajectory Optimization Methods	28
1.4 Direct Trajectory Optimization Methods	30
1.4.1 Direct Shooting Methods	31
1.4.2 Direct Collocation Methods	32

1.5	Problems with the Current Methods	33
1.6	Design of a New Trajectory Optimization Method	34
2	The Direct Trajectory Optimization Method	37
2.1	The Direct Multiple Shooting Method	37
2.1.1	Discretization of a Multiple Shooting Phase	38
2.1.2	Integration Methods	40
2.1.3	Comparison of the Integration Methods	42
2.2	The Direct Collocation Method	43
2.2.1	Discretization of a Collocation Phase	43
2.2.2	State Approximation Using Hermite-Simpson Polynomials	44
2.2.3	Other Collocation Schemes	46
2.2.3.1	Trapezoidal Scheme	47
2.2.3.2	Runge-Kutta 4 Scheme	47
2.2.4	Handling Path Constraints	47
2.3	Software Interfaces	48
2.3.1	General User Model Interface	49
2.3.2	General Interface to NLP Solvers	49
2.3.3	General Interface to Integration Methods	52
2.3.4	General Interface to Collocation Schemes	54
2.4	Implementation Considerations	57
2.4.1	Roundoff and Truncation Errors	57
2.4.2	Parameter and Function Scaling	59
2.4.2.1	Parameter Scaling	60
2.4.2.2	Scaling of the Performance Index	61
2.4.2.3	Default Scaling of Constraints	61
2.4.2.4	Scaling of Continuity Conditions in Multiple Shooting Phases	61
2.4.2.5	Scaling of Defects in Collocation Phases	61
2.4.2.6	Scaling of Phase Connect Conditions	61
2.4.3	Selecting the Parameter Perturbation Size for Gradient Calculation	62
2.5	Setup of Multi-Phase Problems	62

2.6	NLP Solvers	63
2.6.1	SNOPT 6.1	63
2.6.2	Other Methods	64
2.7	Performance of the Direct Optimization Method	64
2.7.1	Performance of the Direct Multiple Shooting Method	64
2.7.2	Performance of the Direct Collocation Method and its Combination with Direct Multiple Shooting	66
3	Indirect Method for Optimal Thrust Arcs in Vacuum	69
3.1	Formulation of Dynamics	69
3.1.1	Gravity Field with J_2 terms	70
3.1.2	Altitude over an Oblate Planet	71
3.1.3	Including Backpressure in the Thrust Calculation	72
3.2	Adjoint Differential Equations	73
3.2.1	Adjoint Differential Equations including J_2 Terms	73
3.2.2	Correction for Backpressure Term	74
3.3	Optimal Control	74
3.4	Boundary Conditions	75
3.4.1	Boundary Conditions for Circular Orbits with a Fixed Inclination .	76
3.4.2	Boundary Conditions for Equatorial Orbits	78
3.4.3	Boundary Conditions for Elliptical Orbits with a Fixed Argument of Peri-gee	79
3.4.4	Boundary Conditions for Elliptical Orbits with a Fixed Inclination	79
3.5	Optimal Coast Arcs and Switching Function	81
3.6	Generating an Initial Guess for the Initial Costates	82
3.6.1	Ad-Hoc Initial Guess for P and Q	82
3.6.2	Estimate based on Thrust-Acceleration and Target Radius	82
3.7	Solution of the Two-Point Boundary Value Problem	83
3.8	Solution in the Framework of an SQP Method	84
3.9	Verification of the Indirect Method	86

4	Example 1: Ariane 5 Dual Payload Mission Analysis	91
4.1	Reference Mission Scenario	92
4.1.1	Generating the Initial Guess	93
4.1.2	Optimization using CAMTOS	94
4.2	Improved Mission Profile with Two Coast Arcs	98
4.2.1	Generating the Initial Guess	99
4.2.2	Optimization using CAMTOS	99
4.3	Summary	103
5	Example 2: Branched Mission Scenario	109
5.1	Mission Scenario	110
5.2	Initial Guess Generation for the Hopper Mission	113
5.3	Optimization of the Branched Mission Scenario	115
5.4	Optimization Results	117
6	Conclusions and Future Research	123
6.1	Conclusions	123
6.2	Future Research	125
	References	127
A	Integration Methods for Direct Multiple Shooting	135
A.1	Runge-Kutta 2/3 Method	135
A.2	Runge-Kutta 4/5 Method	137
A.3	Integration Method of Paus	138
B	Derivation of Important Derivatives	143
B.1	Derivatives of Inclination w.r.t. Radius and Velocity Vectors	143
B.2	Derivatives of Orbital Momentum w.r.t. Radius and Velocity Vectors	144
B.3	Derivatives of Semimajor Axis w.r.t. Radius and Velocity Vectors	145
B.4	Derivatives of Orbital Energy w.r.t. Radius and Velocity Vectors	145

B.5	Derivatives of Eccentricity w.r.t. Radius and Velocity Vectors	146
B.6	Derivatives of Apogee and Perigee Radius w.r.t. Radius and Velocity Vectors 146	
B.7	Derivatives of the Argument of Perigee w.r.t. Radius and Velocity Vectors	147
C	Derivation of Transversality Conditions	151
C.1	Preliminaries	151
C.2	Transversality Conditions for Circular Orbit with Free Argument of Perigee	152
C.3	Transversality Conditions for Circular Orbit with Zero Argument of Perigee	152
C.4	Transversality Conditions for Circular Orbit with Fixed Argument of Perigee	154
C.5	Transversality Conditions for Elliptical Orbits	155
D	Initial Guesses for the Initial Costates	159
D.1	Preliminaries	159
D.2	Derivation of Initial Costate Approximation	160
E	Environment Models	163
E.1	Earth Model	163
E.2	Atmosphere Models	163
	Deutsche Kurzfassung	165

Table of Figures

Fig. 1.1:	Integration of CAMTOS into the GESOP environment	27
Fig. 1.2:	Discretization of a shooting phase without multiple shooting points	32
Fig. 1.3:	Verification of an optimization result generated with a collocation method	33
Fig. 1.4:	Structure of GESOP including the new CAMTOS optimizer	34
Fig. 2.1:	Application of multiple shooting to a phase with one state	38
Fig. 2.2:	Relation between discretization grids and integration intervals	40
Fig. 2.3:	Sparsity pattern of the Jacobian matrix in a multiple shooting phase	40
Fig. 2.4:	Gradient computation with and without stepsize control	42
Fig. 2.5:	Discretization of a collocation phase	44
Fig. 2.6:	Sparsity pattern of Jacobian matrix in a collocation phase	44
Fig. 2.7:	State approximation using Hermite-Simpson polynomials	46
Fig. 2.8:	Interfaces of CAMTOS	48

Fig. 2.9:	User model interface and internal data flow	49
Fig. 2.10:	Structure of the NLP solver interface	50
Fig. 2.11:	Iteration log output	51
Fig. 2.12:	Graphical Iteration Sequence Monitor (GISMO)	51
Fig. 2.13:	Structure of the general integrator interface	52
Fig. 2.14:	Structure of the general collocation method interface	55
Fig. 2.15:	Accordion-like grid-point movement when final time is changed	56
Fig. 2.16:	Power series of Golden Mean value evaluated with different algorithms .	59
Fig. 2.17:	Sparsity pattern of Jacobian matrix in a multi-phase problem	63
Fig. 2.18:	Altitude profile for Ariane 5 dual payload mission	65
Fig. 2.19:	Altitude profile of Ariane 5 mission to a Molnija orbit	67
Fig. 2.20:	Thrust profile of during the first phase of the rocket ascent	67
Fig. 3.1:	Definition of the inertial coordinate frame	70
Fig. 3.2:	Altitude over an oblate planet	72
Fig. 3.3:	Definition of RAAN and inclination	77
Fig. 3.4:	Definition of important vectors	77
Fig. 3.5:	Altitude profile of indirect rocket ascent example	87
Fig. 3.6:	Hamiltonian for rocket ascent example	87

Fig. 3.7:	Convergence behavior of CAMTOS for solving a two-point boundary value problem that involves dissipative terms	88
Fig. 3.8:	Thrust profile of example rocket ascent including backpressure	88
Fig. 3.9:	Convergence behavior of CAMTOS for a rocket ascent without backpressure 89	
Fig. 4.1:	Altitude profile for the dual payload reference mission	95
Fig. 4.2:	3D trajectory representation of the dual payload reference mission	96
Fig. 4.3:	Switching function during the upper stage burns of the dual payload reference mission	96
Fig. 4.4:	Control time histories for the dual payload reference mission	97
Fig. 4.5:	Hamiltonian during upper stage burns	97
Fig. 4.6:	Altitude profile for the modified dual payload mission	100
Fig. 4.7:	3D trajectory representation of the modified dual payload mission	101
Fig. 4.8:	Control time histories for the modified dual payload mission	101
Fig. 4.9:	Switching function for the modified dual payload mission	102
Fig. 4.10:	Switching function during burn arcs for the modified dual payload mission 102	
Fig. 4.11:	Performance index and merit function value during optimization of the dual payload reference mission	104
Fig. 4.12:	Primal and dual infeasibility and optimization step size during the optimization of the dual payload reference mission.	105
Fig. 4.13:	Performance index and merit function value during optimization of the modified dual payload mission	105

Fig. 4.14:	Primal and dual infeasibility and optimization step size during the optimization of the modified dual payload mission	106
Fig. 4.15:	Sparsity structure of the Jacobian matrix for the dual payload reference mission	106
Fig. 4.16:	Sparsity structure of the Jacobian matrix for the modified dual payload mission	107
Fig. 5.1:	The Hopper concept [89]	109
Fig. 5.2:	3D trajectory representation of the Hopper mission scenario	111
Fig. 5.3:	Drag coefficient for Hopper vehicle	111
Fig. 5.4:	Lift coefficient for Hopper vehicle	112
Fig. 5.5:	Initial guess trajectory for branched Hopper mission	114
Fig. 5.6:	Altitude drop right after launch	115
Fig. 5.7:	Sparsity patten of the Jacobian matrix for the branched Hopper mission ..	116
Fig. 5.8:	Ground track of the Hopper vehicle and the upper stage	117
Fig. 5.9:	Altitude profile of the Hopper mission	118
Fig. 5.10:	Control time histories for the Hopper vehicle	119
Fig. 5.11:	Path constraints during the Hopper re-entry	119
Fig. 5.12:	Drag velocity diagram for Hopper re-entry	120
Fig. 5.13:	Control time histories of upper stage	120
Fig. 5.14:	Thrust direction during upper stage ascent	121

Fig. 5.15:	Orbital elements during the upper stage ascent	121
Fig. D.1:	Definition of orbit coordinate frame	159

List of Tables

Table 2.1:	Computational performance of CAMTOS versus PROMIS	65
Table 2.2:	Computational performance of CAMTOS versus TROPIC	66
Table 4.1:	Phase structure of the dual payload reference mission scenario	92
Table 4.2:	Phase structure for the initial guess of the dual payload reference mission scenario	93
Table 4.3:	Phase structure of the modified dual payload mission scenario	98
Table 4.4:	Phase structure for the initial guess of the modified dual payload reference mission scenario	99
Table 5.1:	Vehicle components of the Hopper mission	110
Table 5.2:	Propulsion system properties	110
Table 5.3:	Phase structure of the branched Hopper mission scenario	112
Table 5.5:	Initial guess for the ascent branch of the Hopper mission	113
Table 5.4:	Initial conditions of the branched Hopper mission scenario	113

Table E.1:	Properties of the Earth model	163
Table E.2:	Coefficients for the interpolated US-Standard atmosphere	164

Nomenclature

Scalars

a	semimajor axis	[m]
A_e	nozzle exit area	[m ²]
e	eccentricity	
f	dynamics or flattening factor	
g	path constraint	
g_0	gravity acceleration	[m/s ²]
h	step size or altitude	[s], [m]
H	Hamiltonian	
i	orbit inclination or index	[rad], -
I_{sp}	specific impulse	[s]
J	performance index	
J_2	J_2 harmonic of gravity field	
L	integrand of Lagrange cost	
m	mass	[kg]
M	Mach number	
n	load factor	
o_c	scaling offset	
p	parameter or norm of primer vector	

p_a	ambient pressure	$[\text{N}/\text{m}^2]$
q	massflow or dynamic pressure	$[\text{kg}/\text{s}], [\text{N}/\text{m}^2]$
\dot{Q}	heatflux	$[\text{W}/\text{m}^2]$
r	radius or norm of radius vector	$[\text{m}]$
s_c	scaling factor	
S	switching function	
t	time	$[\text{s}]$
T	Thrust	$[\text{N}]$
u	control	
v	velocity or norm of velocity vector	$[\text{m}/\text{s}]$
x	state value	
α	angle of attack	$[\text{rad}]$
γ	flight-path angle	$[\text{rad}]$
δ	declination	$[\text{rad}]$
Δ	defect or delta	
ε	specific orbital energy	$[\text{m}^2/\text{s}^2]$
ε_m	machine accuracy	
η	throttle factor	
λ	longitude	
λ_m	mass adjoint variable	
μ	gravity constant of central body or bank angle	$[\text{m}^3/\text{s}^2], [\text{rad}]$
τ	normalized time	
ϕ	Mayer cost term	
ω	argument of perigee	$[\text{rad}]$

Matrices and Vectors

\mathbf{g}	path constraint vector	
\mathbf{G}	gravity acceleration vector	$[\text{m}/\text{s}^2]$
\mathbf{i}	unity vector	
\mathbf{i}_f	unity vector perpendicular to orbit plane	

I_3	3×3 identity matrix	
P	primer vector	
p	vector of optimizable parameters	
Q	Q-vector	
R	radius vector	[m]
u	control vector	
V	velocity vector	[m/s]
x	state vector	
v	vector of constant Lagrange multipliers	
λ	vector of adjoint variables	
ψ	boundary constraint vector	

Subscripts and Superscripts

b	body axis
E	Earth
f	final value
i	initial value
l	lower bound
N	North
p	perigee
r	radius
rel	relative
u	upper bound
v	velocity
vac	vacuum
0	reference value
T	Transpose of a matrix or a vector

Abbreviations

ASTOS	Aerospace Trajectory Optimization Software
-------	--

CAMTOS	Collocation and Multiple Shooting Trajectory Optimization Software
DIRCOL	Direct Collocation
ESTEC	European Space Research and Technology Center
GESOP	Graphical Environment for Simulation and Optimization
GISMO	Graphical Iteration Sequence Monitor
GTO	Geostationary Transfer Orbit
HISTOS	High Speed Trajectory Optimization Software
IFR	Institut für Flugmechanik und Flugregelung (Institute of Flight Mechanics and Control)
LEO	Low Earth Orbit
NLP	Nonlinear Program
ODE	Ordinary Differential Equation
OTIS	Optimal Trajectories by Implicit Simulations
PROMIS	Parameterized Trajectory Optimization by Direct Multiple Shooting
RAAN	Right Ascension of Ascending Node
RAM	Random Access Memory
SNOPT	Sparse Nonlinear Optimizer
SOCS	Sparse Optimal Control Software
SRB	Solid Rocket Booster
SQP	Sequential Quadratic Program
TAEM	Target Area Energy Management
TOPS	Trajectory Optimization Specification
TROPIC	Trajectory Optimization by Direct Collocation

1 Introduction

Solving an optimal control problem is not easy. Pieces of the puzzle are found scattered throughout many different disciplines.

John T. Betts in “Practical Methods for Optimal Control Using Nonlinear Programming”

Many engineering applications involve the calculation of “optimal trajectories”. Typical examples in the aerospace sector are a rocket ascent trajectory to maximize the payload delivered into orbit (e.g. Refs. [3], [37], [39], [84] and many others), re-entry missions with several different performance criteria (e.g. Refs. [2], [52], [58], [68], [87]), or the calculation of a minimum time, minimum fuel, or “minimum direct operating cost” trajectory for a supersonic transport aircraft (Refs. [4] and [88]). Other space applications can be found in the framework of orbital transfer maneuvers, including aeroassisted transfer vehicles (e.g. Ref. [66]) and low-thrust engines (e.g. Ref. [14]).

Besides the aerospace sector, many other engineering disciplines involve problems which can be formulated as optimal control problems. Examples are an energy-optimal robot motion (Ref. [77]) or the optimization of chemical processes (Ref. [69]), human body motion (Ref. [73]) or economic models (Refs. [15] and [53]).

The methodology developed in this thesis is a very general approach that is applicable to all such problems. However, most examples presented are using the **Aerospace Trajectory Optimization Software ASTOS**, which is a special, data-driven model for launch, re-entry, and orbital transfer problems (Ref. [5]).

1.1 Motivation

Basically, two different methods are available for solving trajectory optimization problems: The calculus of variation, which is also referred to as “indirect method” or the “Maximum Principle of Pontryagin”, and the “direct methods”, which transform the original optimal control problem into a nonlinear parameter optimization problem. It turns out that existing soft-

ware for trajectory optimization problems is often highly specialized on a rather small class of problems. This especially applies for any software that is using the indirect methods (see Section 1.3). Due to this high level of specialization, these methods are often hard to use for a novice user and require a detailed knowledge on optimal control theory, calculus of variation, and sometimes even about implementation specific behaviors. On the other hand, these methods often give additional information about the optimality of the problem structure itself. Such information is usually not directly available when using alternative methods.

Nevertheless, several general purpose methods have been developed in recent years that are able to handle a large variety of optimal control problems. By using the direct methods (see Section 1.4), the user need not be very familiar with the basics of optimal control theory. However, such methods are often slow compared to indirect methods, and a specific problem discretization may not be suitable during certain phases of a trajectory.

Finally, most available methods are coded in a way that makes them hard to read and understand by anyone who is not familiar with the code itself, thus making them hard to maintain or even change in order to incorporate improvements. In addition, in cases where FORTRAN 77 is used, the resulting codes always require an a-priori knowledge about the required workspace sizes since FORTRAN 77 does not allow for a dynamic memory allocation during run-time.

The major goal of this thesis is to eliminate the major drawbacks of direct and indirect methods discussed further below in detail, while their advantages are preserved. The approach is a hybrid method that allows the user to select the appropriate optimization method for each phase of a multi-phase trajectory optimization problem. This method combines direct multiple shooting with direct collocation and is called CAMTOS (**C**ollocation **A**nd **M**ultiple **S**hooting **T**rajectory **O**ptimization **S**oftware). In addition, an indirect method is developed, which is applicable for fuel-optimal rocket thrust arcs occurring in vacuum (Section 3). This method can be used in conjunction with CAMTOS in order to obtain the switching function for further optimality analysis. The resulting software involves a comprehensive user interface which matches already existing standards in the GESOP/ASTOS environment, thus making it easy to use for the end user by simply selecting the appropriate method with the click of a mouse button (see Fig. 1.1).

On the software engineering side, the major attention is drawn on applying a modular software design and defining simple interfaces to all major elements of a trajectory optimization method. It is not intended to incorporate the most elegant ways of object oriented design, involving inheritance or generic classes and packages. However, a clear and straight forward implementation is preferred wherever it seems to be reasonable in order to simplify future maintenance and enhancement efforts. By using a modern computer language such as Ada-95, features like dynamic memory allocation are readily available. Another important issue is the source code compatibility over a large range of operating systems, from Windows NT and 2000 over Linux up to several different flavors of Unix. This compatibility is mainly achieved by using the GNAT and GCC compiler suite which is available for almost every target platform.

Since the software is integrated into the GESOP environment, its design must allow for the solution of a very general class of multi-phase trajectory optimization problems. The following section gives details on the specific, mathematical formulation of this problem class.

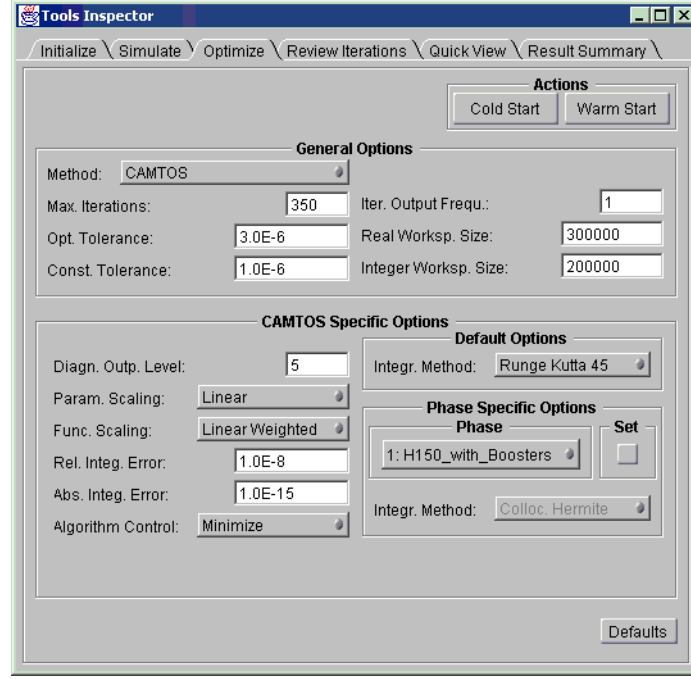


Fig. 1.1: Integration of CAMTOS into the GESOP environment

1.2 General Problem Formulation

In the framework of trajectory optimization, the time history of a control vector $\mathbf{u}(t)$ is calculated such that it minimizes a scalar performance index J . The performance index can be formulated in Bolza-form (Eq. (1.1)) and consists of a Mayer term ϕ and a Lagrange term with the integrand L . A set of path constraints \mathbf{g} and boundary constraints Ψ_0 and Ψ_f must be satisfied at the solution. Both, path and boundary constraints, may be equality or inequality constraints. A general mathematical formulation for a single-phase problem is presented in Eqs. (1.1)-(1.5).

$$\text{Minimize } J = \phi(\mathbf{x}_f, t_f) + \int_{t_0}^{t_f} L(\mathbf{x}, \mathbf{u}, t) dt \quad (1.1)$$

$$\text{subject to } \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t) \quad (1.2)$$

$$\text{and } \mathbf{g}(\mathbf{x}, \mathbf{u}, t) \geq \mathbf{0} \text{ for } t_0 \leq t \leq t_f, \quad (1.3)$$

$$\Psi_0(\mathbf{x}_0, t_0) \geq \mathbf{0}, \quad (1.4)$$

$$\Psi_f(\mathbf{x}_f, t_f) \geq \mathbf{0}. \quad (1.5)$$

This approach can easily be extended to multi-phase problems consisting of k phases by introducing a phase index i and phase connect conditions as shown in Eqs (1.11) and (1.12). In addition, an initial cost term ϕ^0 is introduced and each phase may have a final cost and a Lagrange cost. Besides control time histories, technical optimization problems often involve the optimization of design parameters \mathbf{p} . For example, the mass of a payload is to be maximized while certain sizing factors for a rocket engine are optimally chosen as well (design optimization, see Ref. [86]). In order to support the solution of such problems, design or real parameters are included in the problem formulation. These real parameters can be optimizable within user specified bounds and are constant within a phase ($\dot{\mathbf{p}} = \mathbf{0}$). Each phase may have its own set of real parameters.

$$\text{Minimize } J = \phi^0(\mathbf{x}_0^1, \mathbf{p}^1, t_0^1) + \sum_{i=1}^k \phi^i(\mathbf{x}_f^i, \mathbf{p}^i, t_f^i) + \sum_{i=1}^k \int_{t_0^i}^{t_f^i} L^i(\mathbf{x}^i, \mathbf{u}^i, \mathbf{p}^i, t) dt \quad (1.6)$$

$$\text{subject to } \dot{\mathbf{x}}^i = \mathbf{f}(\mathbf{x}^i, \mathbf{u}^i, \mathbf{p}^i, t) \text{ for } t_0^i \leq t \leq t_f^i, i = 1, \dots, k, \quad (1.7)$$

$$\mathbf{g}^i(\mathbf{x}^i, \mathbf{u}^i, \mathbf{p}^i, t) \geq \mathbf{0} \text{ for } t_0^i \leq t \leq t_f^i, i = 1, \dots, k, \quad (1.8)$$

$$\Psi_0^1(\mathbf{x}_0^1, \mathbf{p}^1, t_0^1) \geq \mathbf{0}, \quad (1.9)$$

$$\Psi_f^i(\mathbf{x}_f^i, \mathbf{p}^i, t_f^i) \geq \mathbf{0}, i = 1, \dots, k, \quad (1.10)$$

$$\begin{bmatrix} \mathbf{x}_0^{i+1} & \mathbf{u}_0^{i+1} & \mathbf{p}^{i+1} \end{bmatrix}^T = \mathbf{h}(\mathbf{x}_f^i, \mathbf{u}_f^i, \mathbf{p}^i, t_f^i) \text{ for } i < k, \quad (1.11)$$

$$t_o^{i+1} = t_f^i \text{ for } i < k. \quad (1.12)$$

In order to simplify the notation, the phase index i is omitted in most parts of the following discussion. The following sections give a brief overview of direct and indirect optimization methods, formulated for a single phase problem as shown in Eqs. (1.1)-(1.5). For an extensive discussion of these methods applied to multi-phase problems, the reader is referred to other references.

1.3 Indirect Trajectory Optimization Methods

Calculus of variation techniques were successfully applied for the first time by Sir Isaac Newton in 1686 for calculating the minimum-drag nose shape in hypersonic flow (see Ref. [23],

p. 52ff). Since then, the theory was continuously developed. In 1744, the first order necessary conditions were formulated by Euler and Lagrange (the so-called Euler-Lagrange equations). In 1834/35, these conditions were posed in a more clear form by the introduction of the Hamilton function. For a problem without path constraints, with terminal equality constraints, and with an initial state vector whose components are either fixed or optimizable, the first order necessary conditions can be formulated as shown in Eqs. (1.13)-(1.20):

$$\text{Hamiltonian: } H = L + \boldsymbol{\lambda}^T \mathbf{f} \quad (1.13)$$

$$\text{Dynamics: } \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t) = \left(\frac{\partial H}{\partial \boldsymbol{\lambda}} \right)^T \quad (1.14)$$

$$\text{Adjoint differential equations: } \dot{\boldsymbol{\lambda}} = - \left(\frac{\partial L}{\partial \mathbf{x}} \right)^T - \left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right)^T \boldsymbol{\lambda} = - \left(\frac{\partial H}{\partial \mathbf{x}} \right)^T \quad (1.15)$$

$$\text{Optimality condition: } \mathbf{0} = \left(\frac{\partial H}{\partial \mathbf{u}} \right)^T = \left(\frac{\partial \mathbf{f}}{\partial \mathbf{u}} \right)^T \boldsymbol{\lambda} + \left(\frac{\partial L}{\partial \mathbf{u}} \right)^T \quad (1.16)$$

$$\text{Initial conditions: } x_k(t_0) \text{ given or } \lambda_k(t_0) = 0 \quad (1.17)$$

$$\text{Terminal constraints: } \boldsymbol{\Psi}_f(\mathbf{x}_f, t_f) = \mathbf{0} \quad (1.18)$$

$$\text{Transversality conditions: } \boldsymbol{\lambda}_f = \left(\frac{\partial \phi}{\partial \mathbf{x}} + \mathbf{v}^T \frac{\partial \boldsymbol{\Psi}_f}{\partial \mathbf{x}} \right)_{t=t_f} \quad (1.19)$$

$$\text{Transversality condition for optimizable } t_f: \boldsymbol{\Omega} = \left[\frac{\partial \phi}{\partial t} + \mathbf{v}^T \frac{\partial \boldsymbol{\Psi}_f}{\partial t} + H \right]_{t=t_f} = 0 \quad (1.20)$$

During the course of the 18th and 19th century, the Euler-Lagrange equations were extended by the introduction of further necessary conditions, such as the Legendre-Clebsch condition (1786)

$$\frac{\partial^2 H}{\partial \mathbf{u}^2} \geq \mathbf{0}, \quad (1.21)$$

the Jacobi condition (1837), and the Weierstrass condition (1879). A detailed discussion on these conditions can be found in Ref. [23]. Except for the Legendre-Clebsch condition and the first order necessary conditions, none of the other necessary conditions are used in the framework of this thesis.

An important extension to the indirect methods as far as technical problems are concerned is the Maximum Principle introduced in 1954 by the Russian mathematician Pontryagin. His method allowed the introduction of constrained variables (path constraints), which are inevitable in the formulation of technical problems. This achievement is also often attributed to Hestenes (1950) and, together with Pontryagin, to Boltyanskii and Gamkrelidze. An excellent summary of all necessary conditions for a very general class of trajectory optimization problems can be found in Ref. [23]. According to Ref. [63], precursors of the Maximum Principle can already be found in a book from Carathéodory, published in 1935. Carathéodory's approach was once called the "Royal Road in the Calculus of Variations" (Königsweg in der Variationsrechnung). Ref. [63] gives an excellent summary of the historical background on the development and relations between different approaches for solving optimal control problems.

The major drawback of indirect optimization methods is the requirement for a detailed mathematical analysis of each single problem. Even slight changes in the dynamics or in the boundary constraints can lead to a completely different solution structure, and often requires a complete revision of any previous derivation of the necessary equations. On the other hand, such an analysis yields an in-depth insight into the problem. The availability of additional indicators for the optimality of the calculated solution often leads to even better solutions by improving an a-priori estimate for the solution structure (e.g. a different burn/coast sequence for rocket trajectories, see Ref. [39]).

The solution of single-phase trajectory optimization problems with indirect methods requires the solution of a two-point boundary value problem using an appropriate, multi-dimensional zero finding algorithm. In a more general setup, a multi-point boundary value problem has to be solved. Several state-of-the-art algorithms such as BNDSCO (Ref. [59]) are available for an efficient solution of these problems. When applying these methods, another major difficulty arises in the requirement of supplying very accurate initial guesses for the adjoint, or costate, variables λ . Usually, these variables do not have any physical meaning, but the trajectory is highly sensitive to even small changes in the costates. This inevitably leads to solution difficulties when new problems have to be solved and little knowledge is available on the structure of the optimal solution. A good approach to enlarge the convergence radius is to use a collocation approach like in Ref. [32] instead of using a multiple shooting approach as it is used in BNDSCO. But still, such boundary value problems remain difficult to solve as soon as the dynamic equations involve dissipative terms (see Ref. [23], Chapter 7).

1.4 Direct Trajectory Optimization Methods

With the development of digital computers in the middle of the 20th century, direct trajectory optimization methods were developed in the 1970s. The beauty of these methods is the possibility of solving very complex problems with a minimum effort of mathematical analysis. In fact, only the physical equations (1.6)-(1.12) need to be coded by the user of such methods. However, these methods require an efficient algorithm to solve constrained nonlinear programming problems with thousands of variables and (nonlinear) constraints (e.g. Ref. [42]).

Direct trajectory optimization methods are divided into two subclasses: Direct shooting methods (e.g. Ref. [13]) and direct collocation methods (e.g. Refs. [50], [51]). Both methods

are approximating the optimal control time history by using control approximations such as piecewise constant or piecewise linear functions, or even spline approximations. Besides the control discretization, path constraints are also discretized. The algorithms only satisfy path constraints at path constraint evaluation points, i.e. they are approximated with a set of interior point constraints.

Both, the control approximation and the discretization of path constraints, are usually not critical in the framework of engineering problems. In general, the error made by calculating a sub-optimal solution due to the discretization is well below the modelling error. The only exception are differential-algebraic equations (DAEs). When working with DAEs, an equality path constraint must be satisfied at each evaluation point of the dynamics.

It can be shown for the limiting case of the direct optimization methods that the generated solution is equivalent to the solution found by an indirect method (see Section 3.8 and Refs. [11] Chapter 4.2, [24], [25], and [71]). In addition, there are methods available which can be used to calculate the time histories of the adjoint variables based on the solution generated with a direct optimization method (Ref. [71]).

1.4.1 Direct Shooting Methods

Direct shooting methods are based on the integration of the trajectory during the optimization process. Such an integration is usually performed numerically with standard solvers for initial value problems (see Refs. [45], [72]).

Shooting methods are very popular for solving boundary value problems. An early application of such methods is to aim a cannon such that the cannonball hits its target, hence the name “shooting method”. The major problem of shooting methods is a great sensitivity of the terminal constraints with respect to changes in the initial conditions. A first approach to reduce this sensitivity is to “shoot” from both sides of the interval and try to match the two trajectories at an intermediate point. Such methods are sometimes referred to as “shooting to a fitting point” (Ref. [65]). In order to reduce the sensitivity of the terminal constraints with respect to changes in the initial conditions even further, multiple shooting points are inserted to restart the integration at intermediate points (see e.g. Ref. [74]). Such multiple-shooting points also improve the numerical stability by preventing the growth of the error introduced by poor initial guesses, discretization or round-off, and propagated by inherent instabilities in the dynamics (see Ref. [13], p. 242).

One of the first direct shooting methods is described by Brusch in Refs. [20], [21] and [22] and Hull in Ref. [46]. Direct multiple shooting methods have been proposed by several authors (see e.g. Refs. [13] (MUSCOD), [49] (STOMP), [50], [70] (PROMIS), [54], [55] (TOMP)). They are often based on the shooting methods originally developed to solve boundary value problems, but they can also be used in the framework of direct trajectory optimization. In the framework of direct shooting methods, the control time histories are discretized using a parameter dependent control approximation. In addition to the controls, path constraints are discretized as well, and they are only satisfied at path constraint evaluation points.

Fig. 1.2 shows a phase discretized for single shooting that includes one state, one control, and one path constraint. Note that the initial state remains a scalar optimization parameter

while the control becomes a vector of discrete optimization parameters. The path constraint is transformed into a vector of interior point constraints. A detailed description of the direct multiple shooting method implemented in the framework of this thesis is given in Section 2.1.

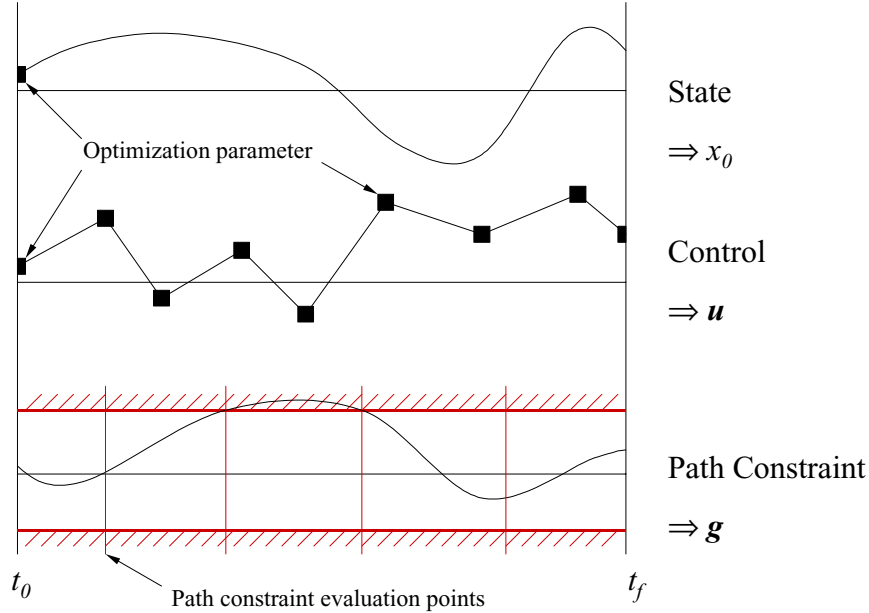


Fig. 1.2: Discretization of a shooting phase without multiple shooting points

1.4.2 Direct Collocation Methods

In contrast to the direct shooting methods, direct collocation methods are also approximating the state time histories by using e.g. Hermite-Simpson polynomials. The basics for collocation methods were first proposed by Dickmanns and Well in 1974 in the framework of indirect collocation (Ref. [32]). It was extended to a direct collocation method by Hargraves and Paris in 1987 (Ref. [48]) and implemented in their OTIS software (Ref. [60]). More recent implementations of similar algorithms can be found in Refs. [50], [75], [76] and others. A major improvement to collocation methods was made by Betts with the introduction of SOCS which includes an automatic mesh refinement algorithm and an SQP method that is specialized on solving very large, sparse problems including several ten-thousand parameters and constraints (see Ref. [11]). The special property of SOCS is that it is not only able to exploit a sparsity pattern in the gradients of the cost function and the constraint vector, but also in the Hessian matrix. This is achieved by directly calculating the Hessian matrix using sparse finite differencing. The code DIRCOL by von Stryk (Refs. [75] and [76]) also includes an algorithm for mesh refinement and the possibility of adjusting the position of collocation grid points during the optimization such that the discretization error is equidistributed.

The major advantages of using direct collocation methods in comparison to a direct multiple shooting method are a much better run-time performance as long as a relatively small number of collocation intervals is used and a larger convergence radius. The latter advantage can be attributed to the fact that a direct collocation method can be seen as an implicit integra-

tion of the dynamic system, while multiple shooting methods are using explicit integration formulas such as a Runge-Kutta method. In that sense, collocation methods are usually using many more “multiple shooting points” (in the sense of collocation intervals) than a multiple shooting method.

It is worth noting that direct collocation methods offer a convenient way to deal with optimization problems that involve differential-algebraic equations (see Ref. [11], p. 69ff) since it is possible to place a path constraint evaluation point at the exact same locations where the dynamics are evaluated. Such a consistent node placement is not possible when direct multiple shooting methods are used that involve an integration method with step-size control.

1.5 Problems with the Current Methods

One of the major advantages of using multiple-shooting methods is the possibility of running them with an ODE solver that includes a step-size control algorithm. Therefore, the algorithm is almost independent of the discretization grid and will, if it converges, at least deliver a sub-optimal solution. Using the computed control-time history of a converged solution in a simulation will always deliver a feasible trajectory which satisfies all boundary conditions and the path constraints at the path constraint evaluation nodes. However, the integration of the dynamic equations requires a lot of computational effort, especially due to the numerical calculation of the gradients which are required by the SQP methods for solving the NLP problem. The exact distribution of the run-time required for gradient calculation and other tasks in a trajectory optimization method cannot be generalized, but it is usual for direct multiple shooting methods that around 70% of the total execution time is spent for gradient calculations.

When using a direct collocation method, the trajectory is not integrated with an ODE solver with step-size control. The direct collocation approach can be regarded as an implicit integration of the trajectory without step-size control. For most aerospace applications, the representation of state time histories using Hermite-Simpson polynomials is fairly accurate even when only a few collocation intervals are used. Therefore, fewer evaluations of the dynamics are required during the optimization process and the run-time critical part of collocation methods is shifted towards the routines dealing with vector and matrix algebra.

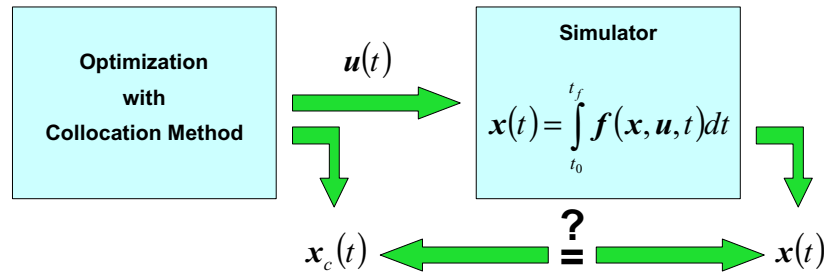


Fig. 1.3: Verification of an optimization result generated with a collocation method

The major drawback when using direct collocation methods is the possibility that the converged solution does not reflect the physical behavior of the dynamic system. However, the

solution can be verified by taking only the computed control time history for a simulation run which integrates the trajectory with an ODE solver that includes a step-size control (see Fig. 1.3). If the resulting simulation run does not satisfy the path and boundary constraints anymore, additional collocation intervals need to be inserted. However, this can be very tedious to do if certain parts of a trajectory involve large changes in the dynamics. Such regions require many collocation intervals, thus leading to many additional parameters and constraints (see e.g. Ref. [11]).

A typical example problem is a rocket ascent that involves solid rocket boosters (SRBs). Such SRBs usually have a predefined, time-dependent thrust and massflow profile. Both, thrust and massflow, may vary significantly during the ascent, thus leading to significant changes in the dynamics (see Section 2.7.2 for an example). If the collocation interval boundaries are not placed exactly on the data points of the SRB profiles, the Hermite-Simpson approximations of the state-time histories may be completely wrong when large collocation intervals are used.

While the problem arising with time-dependent SRB profiles can be reduced by selecting an appropriate collocation grid, similar problems will arise if the aerodynamic coefficients of a vehicle vary significantly with Mach number and/or angle of attack. Such dependencies cannot be reflected in a fixed collocation grid, since the time histories of Mach number and angle of attack may change significantly during the course of the optimization process.

1.6 Design of a New Trajectory Optimization Method

The main goal of this thesis is to eliminate the major drawbacks of all the trajectory optimization methods mentioned above by introducing a combination of direct multiple shooting and direct collocation in a single optimization algorithm called CAMTOS (**C**ollocation and **M**ultiple Shooting **T**rajectory **O**ptimization **S**oftware). Each phase of a multi-phase trajectory optimization problem can be discretized with a different transcription method. In addition, an indirect method specialized for rocket trajectories is introduced which can be used in combination with the direct optimization methods for exo-atmospheric, finite-thrust maneuvers.

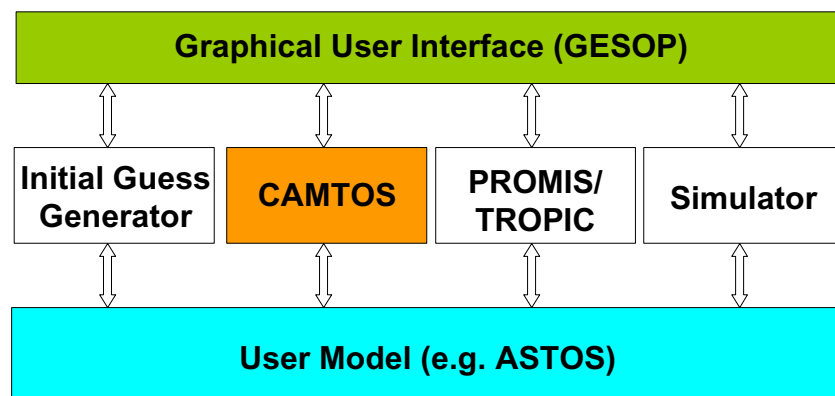


Fig. 1.4: Structure of GESOP including the new CAMTOS optimizer

As shown in Fig. 1.4, CAMTOS is completely integrated into the **Graphical Environment for Simulation and Optimization** (GESOP, see Ref. [40]). Since this environment also includes a direct multiple shooting algorithm (**Parameterized Optimal Control using Multiple Shooting**, PROMIS) and a direct collocation method (**Trajectory Optimization by Direct Collocation**, TROPIC), a direct comparison with these methods can be performed.

Since CAMTOS is completely embedded into the GESOP software, it also uses the GESOP model interface. Therefore, all models which are currently available for GESOP can also be used with CAMTOS. Most comparisons presented in the following chapters are performed by using the **Aerospace Trajectory Optimization Software** ASTOS, which represents a completely data-driven model for GESOP, specialized in modelling launcher, re-entry, and orbital transfer missions. In addition, the indirect method is implemented in the ASTOS software in order to use it in combination with the direct methods (see Section 4).

While CAMTOS currently relies on the SQP solver SNOPT (Ref. [42]) for solving the resulting NLP problem, the remainder of the software is coded in Ada95. This modern computing language allows for a structured, modular, and object-oriented design of the software. Thereby, readability, maintainability, and extensibility of the resulting algorithm is greatly improved. By the specification of very general and generic interfaces to ODE solvers, collocation schemes, and NLP solvers, it becomes an easy task to extend the software with new modules in future versions.

The hybrid approach of combining an indirect method with a more or less direct approach can also be found in papers from Goodson, et al. (Ref. [43]), and Hanson and Dukeman (Ref. [47]). These papers describe the optimization of orbit transfer missions involving a large number of burns. The finite thrust arcs are optimized one-by-one by using an indirect method and a two-point boundary value problem solver. In an outer loop, the orbital parameters of the coast arcs connecting the single burn arcs are optimized such that a performance index is minimized. The approach used in this thesis is a more integrated approach in which all parameters are optimized at the same time, avoiding the necessity of an outer loop.

The following chapters discuss the details of each of the elements of the new trajectory optimization software. Chapter 2 describes the design and implementation issues of the direct optimization method. Chapter 3 deals with the design and implementation of the indirect method for thrust arcs in vacuum. Chapters 4 and 5 are example applications and contain numerical results for the combination of the direct and indirect method as well as for the combination of direct collocation and direct multiple shooting. Conclusions are made in Chapter 6.

2 The Direct Trajectory Optimization Method

“The optimizer exploits the weakness of your model.”

Henry J. Kelley

This chapter gives a detailed description of the direct optimization method that is developed in the framework of this thesis. Due to the combination of a direct multiple shooting approach with a direct collocation approach, the resulting optimization program is called **Collocation and Multiple Shooting Trajectory Optimization Software (CAMTOS)**. In CAMTOS, a different discretization method can be selected for each single phase, i.e., some phases can be discretized using direct collocation while others can use a direct multiple shooting approach. If a closed form solution for the propagation of the state vector is available, it is also possible to avoid any discretization and insert an analytical phase which directly propagates the state time histories without any numerical integration.

For most technical problems it is essential that discontinuities in the state time histories can be handled. An example for such a discontinuity is a mass-drop during a launch vehicle ascent mission. Such problems are called multi-phase or multi-stage problems. For a detailed discussion on the formulation of such problems the reader is referred to Ref. [50]. Without any loss of generality, only single-phase problems will be discussed in the majority of the following Sections. However, this discussion is extended to multi-phase problems by introducing phase connect conditions between the different phases in Section 2.5.

After giving details on both, the direct multiple shooting and the direct collocation method, some overall implementation issues are discussed.

2.1 The Direct Multiple Shooting Method

As it is already mentioned in the introduction, one basic problem of shooting methods is their great sensitivity of the final states of a trajectory with respect to small changes in the initial states and controls. During the optimization process this may lead to numerical problems when

the initial guess is not accurate enough. The numerical integration fails or numerical overflows may occur due to the parameter changes performed by the optimization method.

In general, there are two possibilities to handle this difficulty. First, a better initial guess can be constructed. However, this is often very tedious or even impossible to do within a reasonable timeframe. A better approach is to improve the stability of the optimization algorithm by introducing intermediate guesses for the states and restart the integrator at these multiple-shooting points. Of course, such intermediate states must be optimizable parameters, and additional constraints must be introduced to the problem to ensure a continuous trajectory once the optimizer is converged. These constraints are called continuity conditions or defects (Ref. [11]).

2.1.1 Discretization of a Multiple Shooting Phase

The multiple shooting approach is shown in Fig. 2.1 for a phase with one state. Note that the scalar optimization parameter x_0 is replaced with a vector of optimization parameters \mathbf{x} that includes the guesses for the intermediate states. In addition, the defects or continuity conditions Δ are introduced.

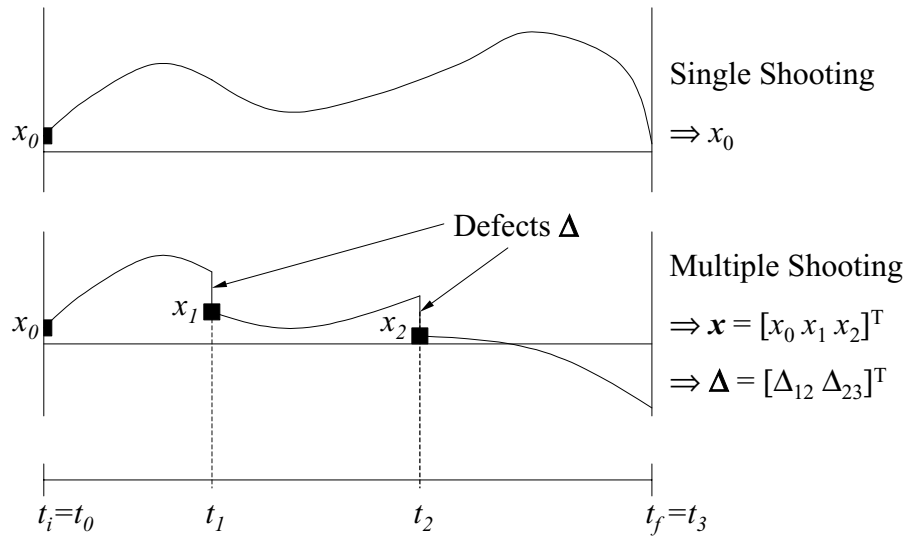


Fig. 2.1: Application of multiple shooting to a phase with one state

The continuity conditions are formulated as

$$\Delta_{ab} = \int_{t_a}^{t_b} f(x, u, t) dt + x_a - x_b, \quad (2.1)$$

where t_a is the initial time of the current shooting interval and t_b is the final time of the shooting interval.

When selecting an appropriate control approximation, the convergence behavior can be improved by selecting an approximation which has a local behavior. I.e., changing one param-

eter of the control only has a local effect on the trajectory. This can be achieved by using a piecewise constant or piecewise linear control approximation. Experiments using third order splines as a control approximation often lead to problems where the resulting splines tend to oscillate significantly, violating the user-specified bounds at the control nodes. Such a behavior is not acceptable for most technical applications where many control variables, e.g. an engine throttle setting, are specified with bounds that reflect physical boundaries. Since it is usually sufficient to use a piecewise linear control approximation in the framework of engineering problems, only a piecewise constant and a piecewise linear control approximation are implemented in this multiple shooting method. The local influence of the control approximation is also exploited during the numerical computation of the Jacobian matrix using finite differences and is required to preserve the sparsity pattern of the Jacobian matrix. Especially the preservation of a sparse Jacobian matrix significantly increases the run-time performance of the resulting algorithm. A very detailed analysis can be found in Ref. [11].

For the numerical integration of a multiple shooting phase, the whole phase is split into several separate integration intervals. These intervals are derived from the previously defined multiple shooting, control refinement, and path constraint evaluation grids. Overlaying all of these grids results in the structure of the integration intervals (see Fig. 2.2). In order to avoid a change in the initial and final times of each integration interval, the whole phase is normalized in time such that the initial normalized time is zero and the final normalized time is one. The required transformations are

$$\tau = \frac{t - t_0}{t_f - t_0} \text{ and } t = (t_f - t_0)\tau + t_0. \quad (2.2)$$

Note that this transformation also requires a change in the definition of the dynamics and the Lagrange cost:

$$\frac{d\mathbf{x}}{d\tau} = \frac{d\mathbf{x}}{dt} \cdot \frac{dt}{d\tau} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \tau, t_0, t_f) \cdot (t_f - t_0) \quad (2.3)$$

$$\int L dt = \int L \cdot \frac{dt}{d\tau} d\tau = \int L(\mathbf{x}, \mathbf{u}, \tau, t_0, t_f) \cdot (t_f - t_0) d\tau \quad (2.4)$$

Initial and final phase times need to be regarded as additional optimization parameters, just like the design parameters. The advantage of introducing a normalized time τ is that the integration grid does not change on this normalized time scale when the phase times t_0 and t_f are changed. This becomes important when calculating numerical gradients with slightly perturbed trajectories. The integration methods need not deal with changes in the integration interval length and can use the same step-size sequence (in normalized time) as for the reference trajectory.

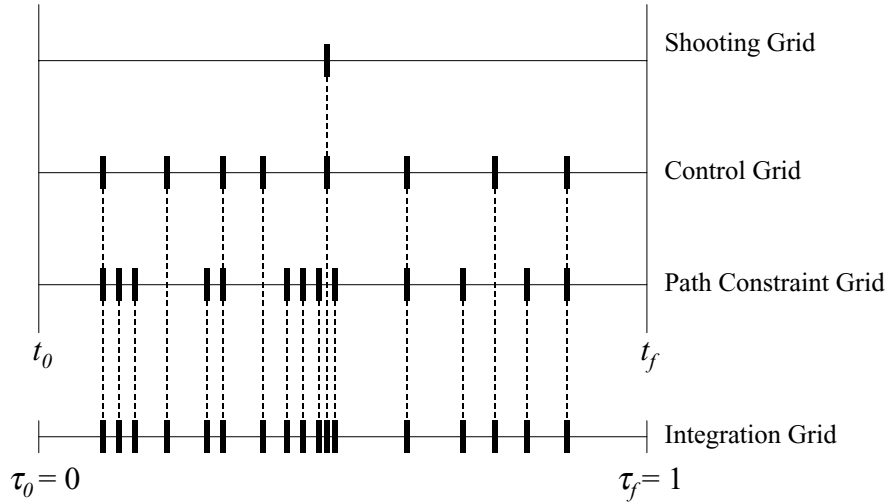


Fig. 2.2: Relation between discretization grids and integration intervals

By arranging constraints and parameters in a convenient order, the corresponding Jacobian matrix has a block diagonal structure as shown in Fig. 2.3 for a phase with one multiple shooting point. Note that the optimizable real parameters and phase times are duplicated at each multiple shooting point and connected to the next shooting interval with the corresponding continuity conditions. This method will produce a block diagonal structure for the Jacobian matrix. Such a structure greatly simplifies the construction of the associated sparsity pattern.

Parameter Constraints	x_0	p_0	u_0	t_{00}	t_{f0}	x_1	p_1	u_1	t_{01}	t_{f1}
Initial boundary	x	x	x	x	x					
Parameter constr.		x		x	x					
Path constraints	x	x	x	x	x					
State connect	x	x	x	x	x	x				
Parameter connect		x					x			
Control connect			x					x		
t_0 connect				x					x	
t_f connect					x					x
Path constraints						x	x	x	x	x
Terminal constr.						x	x	x	x	x

Fig. 2.3: Sparsity pattern of the Jacobian matrix in a multiple shooting phase

2.1.2 Integration Methods

The core part of any multiple shooting optimization method is, besides the NLP solver, the integration method that is used to propagate the states. Special care needs to be taken in order

to be able to calculate gradients of all constraints w.r.t. variations in the optimization parameters.

PROMIS (Ref. [51] and [70]) is equipped with a special Runge-Kutta 4/5 integration method which is using a so-called “interior differentiation” technique (see Ref. [70], p. 10, and Ref. [45], p. 183). Such an implementation is highly specialized and complicated in data management, since all derivatives must be calculated in parallel to a new reference trajectory.

Ref. [45], p. 183, shows an easy way to simplify the construction of derivatives by using an integration method with step-size control to compute a new reference trajectory and saving the generated step-size sequence for later use. Once parameters are perturbed to compute the cost and constraint derivatives, the trajectories are re-integrated using the same integration method without step-size control, but with the same step-size sequence that is used for the reference trajectory. Ref. [45] shows a nice example for the effects caused by a step-size control algorithm during the computation of perturbed trajectories. The dynamic system used is a so-called Brusselator, modeled by the equations

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 1 + x_1^2 x_2 - (b + 1)x_1 \\ bx_1 - x_1^2 x_2 \end{bmatrix}, \quad (2.5)$$

where b can be regarded as an optimizable parameter. \mathbf{y} is the value of the states at $t = 20$ when Eq. (2.5) is integrated from $t = 0$ with the starting values $\mathbf{x}_0 = [1.3 \ b]^T$. Fig. 2.4 shows a comparison of the computed gradients $\partial \mathbf{y} / \partial b$ with and without step-size control. Note that the numerical integration accuracy is the same for both cases, and Δb is set to 10^{-4} . Freezing the step-size sequence for the gradient calculation is clearly giving better results.

The approach of freezing the step-size sequence gives the same good quality for the derivatives as the more complicated “interior differentiation”. The important factor is, that the gradient of the actually used integration method must be computed. If perturbed trajectories are also computed with a step-size control, this may lead to a completely different step-size sequence, generating inconsistent gradients.

Three different integration methods have been implemented for CAMTOS in the framework of this thesis: a Runge-Kutta 2/3 method, a Runge-Kutta 4/5 method, and an integration method according to Paus (Ref. [61]). A detailed description for each of the methods can be found in Appendix A.

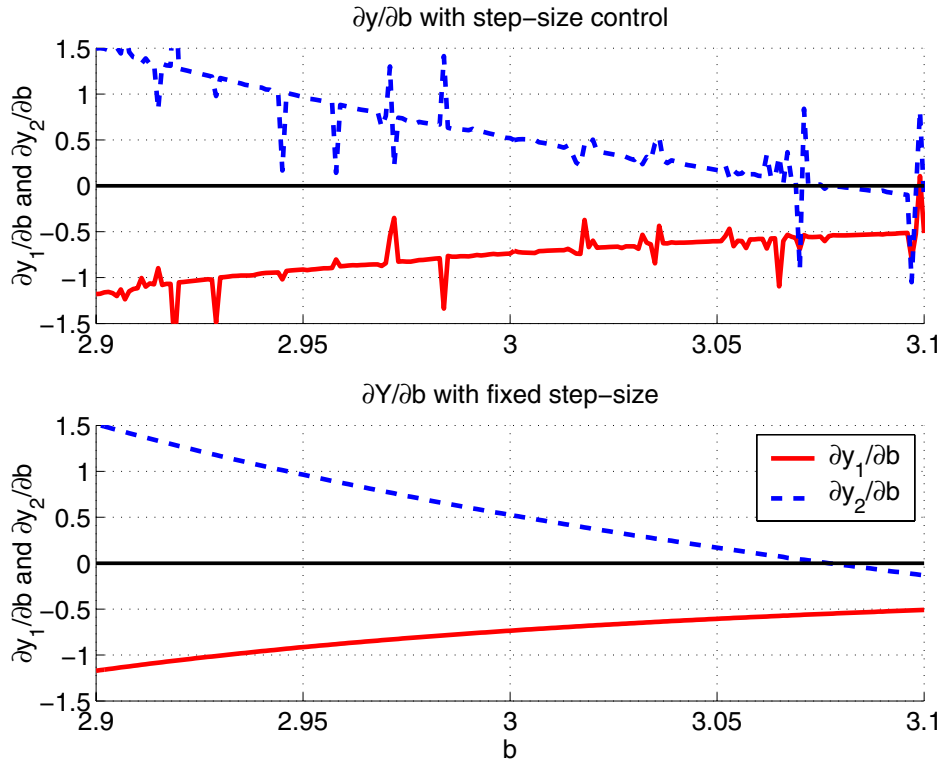


Fig. 2.4: Gradient computation with and without stepsize control

2.1.3 Comparison of the Integration Methods

A detailed analysis of the computational performance of the Paus integration method can be found in Ref. [35]. It turns out that the integration methods Runge-Kutta 2/3 and 4/5 are clearly superior when they are compared to the Paus method in a general setup like CAMTOS. However, Paus has shown in Ref. [61] that the algorithm can be tuned to very specific models by exploiting sparsity patterns in the dynamics and writing specialized code (including e.g. loop unrolling) for a specific application. It should also be noted that, for some specific applications, the required system matrices A and B can be computed along with the system dynamics with only slightly added labor. Although this is useful for a special guidance application where only small perturbations occur and the same mission only needs to be re-optimized, starting from a very good initial guess, such a setup is not applicable to a general optimization algorithm like CAMTOS.

Another major drawback of the method of Paus is a fairly large memory consumption due to the storage of many matrices at each integration step. For dynamic systems with many states such an approach will easily run into problems with memory limitations and the sparsity patterns in the linearized dynamics should be exploited.

The best performance for all of the problems investigated in the framework of this thesis is shown by the Runge-Kutta 4/5 method. Since this method is also able to handle mildly-stiff problems efficiently, it is the default integration method for multiple shooting phases.

2.2 The Direct Collocation Method

The direct collocation method implemented in CAMTOS is based on Refs. [11], [12], [48], [50], and [51]. Slight modifications have been performed to the discretization of the optimal control problem in order to allow for a piecewise constant control approximation. However, it turns out that these modifications also improve the convergence behavior of the algorithm significantly when a piecewise linear control approximation is used. This section gives a detailed discussion on the different elements of the direct collocation method and discuss the advantages compared to earlier implementations.

2.2.1 Discretization of a Collocation Phase

When using direct collocation, no control refinement points can be defined. The reason for this is the assumption of a piecewise linear or constant control within one collocation interval. Since the state time histories are approximated by using polynomials within one interval, it is necessary to split the whole phase into several smaller collocation intervals.

In order to allow for discontinuities in the control between the collocation intervals, each of the boundary points includes two optimization parameters for each control: a left-hand and a right-hand control value (see Fig. 2.5). If a control function has to be continuous, an additional control continuity constraint is introduced to the problem. This approach increases the number of parameters and constraints, but it is found that this also improves the convergence behavior of the optimization method significantly (see Ref. [38] and Section 2.7.2).

Assuming that a collocation phase consists of three collocation intervals, the sparsity pattern of the corresponding Jacobian matrix is shown in Fig. 2.6. While other approaches usually arrange path constraints such that they create a horizontal block after all the defects (see e.g. Ref. [48]), CAMTOS groups the path constraints together with their corresponding collocation intervals. Such an approach preserves a block diagonal structure of the Jacobian matrix. In addition, it also simplifies the generation of the corresponding sparsity pattern data.

In contrast to the multiple shooting method discussed in Chapter 2.1, optimizable phase times and real parameters are not duplicated for each collocation interval. Although such an approach would lead to the same type of block diagonal structures as it was already constructed for the multiple shooting method, it can easily produce hundreds of additional parameters and constraints. This would easily lead to problems with the memory consumption of SNOPT, since the internal approximation of the Hessian matrix always produces a dense matrix. If this matrix becomes too large, hardware memory limits are easily exceeded.

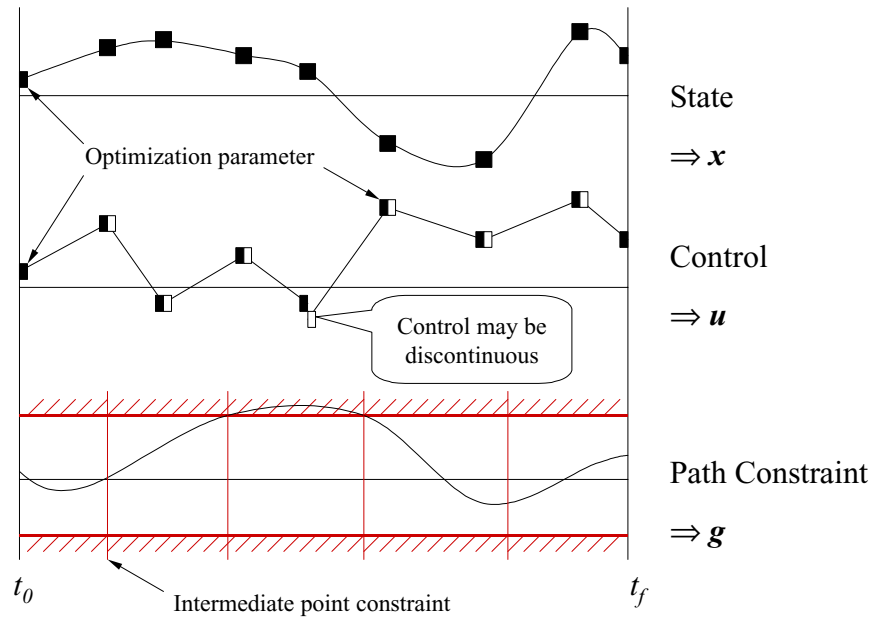


Fig. 2.5: Discretization of a collocation phase

Parameter Constraints	x_1	u_1	x_2	u_2	x_3	u_3	x_4	u_4	p_1	t_0	t_f
Initial boundary	x	x							x	x	x
Parameter constr.									x	x	x
Path constraints	x	x	x	x					x	x	x
Defect 12	x	x	x	x					x	x	x
Control continuity		x		x							
Path constraints			x	x	x	x			x	x	x
Defect 23			x	x	x	x			x	x	x
Control continuity				x		x					
Path constraints					x	x	x	x	x	x	x
Defect 34					x	x	x	x	x	x	x
Control continuity						x		x			
Terminal constr.							x	x	x	x	x

Fig. 2.6: Sparsity pattern of Jacobian matrix in a collocation phase

2.2.2 State Approximation Using Hermite-Simpson Polynomials

The method of collocation using Hermite-Simpson polynomials was first described in the framework of the solution of optimal control problems with indirect methods by Dickmanns and Well (Ref. [32]). Later, Renes (Ref. [67]) proposed a direct collocation method which used

the spline coefficients as optimization variables. However, he did not give any numerical results. Hargraves and Paris (Ref. [48]) were the first to publish numerical results on a direct collocation method using Hermite-Simpson polynomials for approximating the state time histories. Their implementation is called OTIS (Ref. [60]).

The basic idea behind this collocation method is to approximate the state time histories using a polynomial representation for the states such as

$$x_c = C_0 + C_1 t + C_2 t^2 + C_3 t^3. \quad (2.6)$$

In case of a Hermite-Simpson polynomial and a scalar for x_c , the coefficients C_0 through C_3 can be calculated with the state values and the state derivatives at the boundaries of each collocation interval (see Fig. 2.7) as

$$\begin{bmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -3/h^2 & -2/h & 3/h^2 & -1/h \\ 2/h^3 & 1/h^2 & -2/h^3 & 1/h^2 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ f_1 \\ x_2 \\ f_2 \end{bmatrix} \quad (2.7)$$

where $h = t_2 - t_1$ is the length of the current collocation interval. Note that

$$f_1 = f(x_1, u_1, t_1) \text{ and } f_2 = f(x_2, u_2, t_2). \quad (2.8)$$

Now, the state in the middle of the collocation interval can be calculated as

$$x_c(h/2) = \frac{x_1 + x_2}{2} + \frac{f_1 - f_2}{8} h \quad (2.9)$$

and the slope is

$$\dot{x}_c(h/2) = \frac{3(x_2 - x_1)}{2h} - \frac{f_1 + f_2}{4}. \quad (2.10)$$

Using Eq. (2.10), the defect Δ_{12} (see Fig. 2.7) can be calculated as

$$\Delta_{12} = hf(x_c) - \frac{3}{2}(x_2 - x_1) + \frac{h}{4}(f_1 + f_2). \quad (2.11)$$

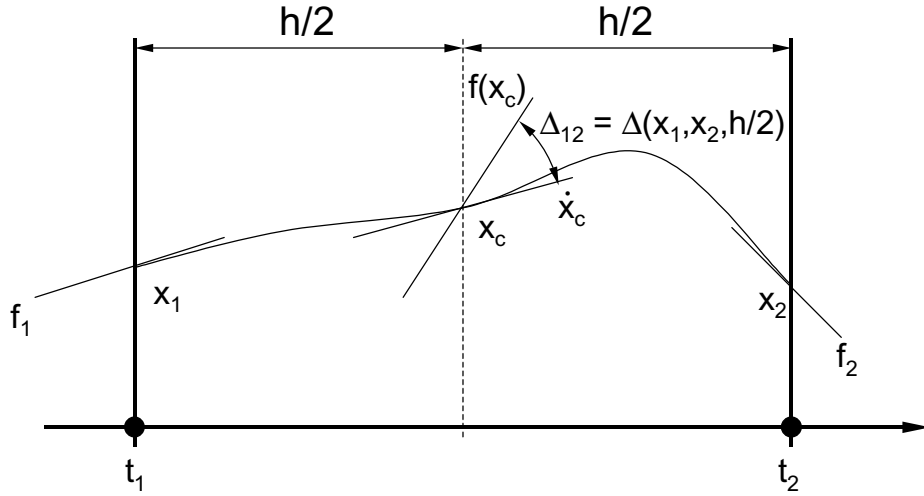


Fig. 2.7: State approximation using Hermite-Simpson polynomials

Note that Eq. (2.11) is obtained from Eq. (2.10) by multiplying Eq. (2.10) with h in order to cancel out h in the denominator of Eq. (2.10). This procedure increases the numerical stability for small values of h . Also note that Eq. (2.11) can be rewritten as

$$\Delta_{12} = x_2 - x_1 - \frac{h}{6}[f_1 + 4f(x_c) + f_2] \quad (2.12)$$

which is the same expression as given in Ref. [12] and can also be regarded as an implicit integration formula in case Δ_{12} is set to zero (also see e.g. Ref. [65], p. 132):

$$x_2 = x_1 + \frac{h}{6}[f_1 + 4f(x_c) + f_2]. \quad (2.13)$$

The order of this method is four, i.e. $O(h^4)$. Note that all necessary equations derived above can easily be extended from a scalar form to a vector form by simply applying the scalar equations to each element of the state vector.

2.2.3 Other Collocation Schemes

The basic strategy presented in Section 2.2.2 can be used to add other collocation schemes of different accuracy. Several possibilities are listed in Refs. [11] and [12]. Besides the Hermite-Simpson approximation, two other methods are implemented in CAMTOS: a trapezoidal rule and a Runge-Kutta 4 scheme.

Additional collocation schemes can easily be implemented. The only limitation is that any collocation scheme may only depend on x_1, x_2, u_1, u_2, t_1 , and t_2 , since only these variables are passed over to the corresponding routines.

2.2.3.1 Trapezoidal Scheme

The trapezoidal rule can be regarded as an implicit integration, too. The defect is calculated as

$$\Delta_{12} = (x_2 - x_1) - \frac{h}{2}(f_1 + f_2). \quad (2.14)$$

The order of the trapezoidal method is two, i.e. $O(h^2)$.

2.2.3.2 Runge-Kutta 4 Scheme

The Runge-Kutta 4 scheme can be regarded as a shooting method with exactly one integration step per interval. The defects are calculated as

$$\Delta_{12} = (x_2 - x_1) - \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4), \quad (2.15)$$

where

$$\begin{aligned} k_1 &= hf(x_1, u_1, t_1) \\ k_2 &= hf(x_1 + k_1/2, \bar{u}, t_1 + h/2) \\ k_3 &= hf(x_1 + k_2/2, \bar{u}, t_1 + h/2) \\ k_4 &= hf(x_1 + k_3, u_2, t_2) \end{aligned} \quad (2.16)$$

and

$$\bar{u} = \frac{u_2 - u_1}{2}. \quad (2.17)$$

The order of the Runge-Kutta 4 method is four, i.e. $O(h^4)$, as for the Hermite-Simpson method. However, computational experience has shown that the Hermite-Simpson method is numerically more stable which lies in the nature of implicit integration methods.

2.2.4 Handling Path Constraints

As for the multiple shooting methods, path constraints are only enforced at path constraint evaluation points. In the framework of collocation methods, such points can be placed exactly at the locations of an evaluation of the dynamics. However, in order to supply the same flexibility already present in the GESOP interface, a user is allowed to manually select the position of path constraint evaluation points. Therefore, any collocation scheme must supply a routine which returns the values of the state variables at any intermediate point of a collocation interval.

In case of the implemented collocation schemes it is assumed that the state time histories can be approximated with third order polynomials as shown in Eq. (2.6). The coefficients are calculated based on Eq. (2.7). The numerically most efficient way to evaluate the cubic polynomial is to use the Horner scheme. The principle is outlined in Ref. [65], Chapter 5.3:

$$x(\Delta t) = C_0 + \Delta t[C_1 + \Delta t(C_2 + \Delta t C_3)] \quad (2.18)$$

Special care needs to be taken in order to ensure numerical stability when the distance Δt of the evaluation point to the collocation interval boundaries is small. Evaluating Eq. (2.18) for very small values of Δt may lead to larger truncation errors. Therefore, a flag is passed to the corresponding routine which supplies information whether an evaluation of the states should be performed from the left side or from the right side of the collocation interval. An evaluation should be performed from the left side if $t - t_0 > h/2$ while it should be performed from the right side when $t - t_0 < h/2$. This logic always yields a $\Delta t \geq h/2$ in Eq. (2.18).

2.3 Software Interfaces

A general purpose trajectory optimization software like CAMTOS must provide several different interfaces. Since CAMTOS is embedded in the GESOP environment, it also has to reflect several different interfaces already defined within this environment. Fig. 2.8 shows a general overview of all relevant software interfaces. The end user is communicating with CAMTOS through the graphical user interface only, and he also provides the mathematical optimization model in a general format outlined in Section 2.3.1 and discussed in more detail in Ref. [40].

CAMTOS is also using the TOPS data format (Trajectory Optimization Specification). This data structure contains all current values of the optimization parameters as well as program specific settings. It allows the user to save any optimization result or an intermediate solution and reload it later to completely restore all program settings.

The interfaces to the user model and to the NLP solver are discussed in Section 2.3.1 and 2.3.2. The generic interfaces to the integration methods and collocation schemes are discussed in Section 2.3.3 and 2.3.4, respectively.

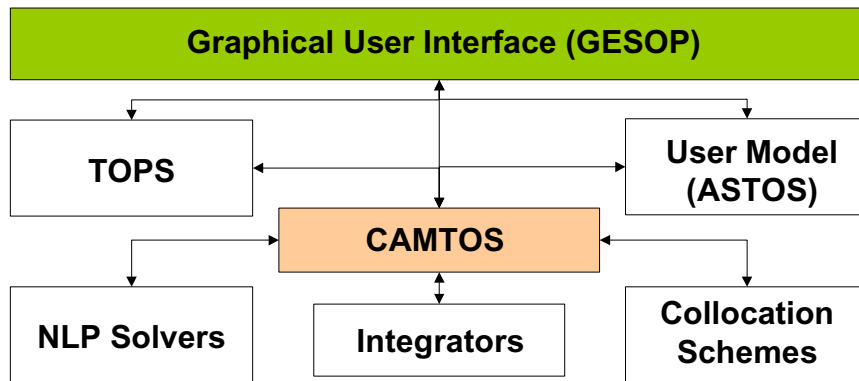


Fig. 2.8: Interfaces of CAMTOS

2.3.1 General User Model Interface

A general purpose trajectory optimization software like CAMTOS must supply a well defined user model interface. Such an interface must allow the user to formulate the optimization problem with as little additional information as possible. An ideal setup would be an interface which only requires the user to code the set of equations shown in Section 1.2, Eqs. (1.6)-(1.12). The whole transcription of the problem into a nonlinear parameter optimization problem has to be performed automatically without any user interaction.

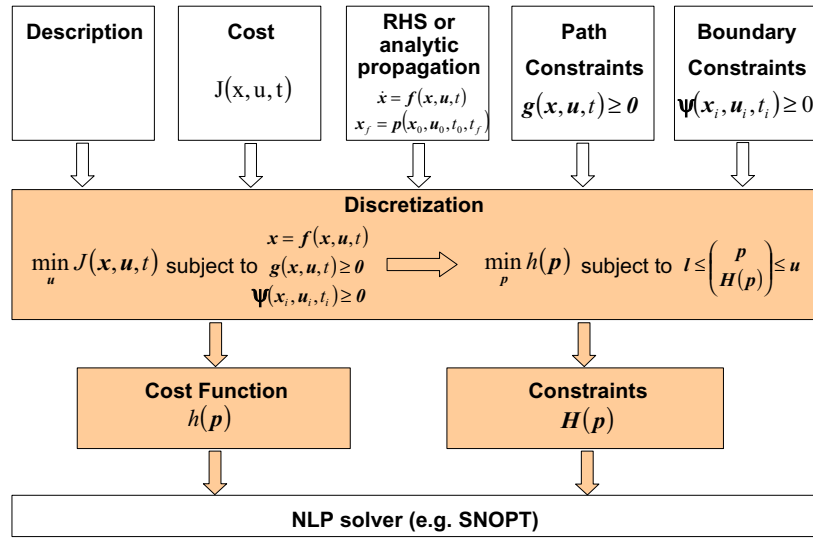


Fig. 2.9: User model interface and internal data flow

Since CAMTOS is embedded in the GESOP environment, it makes use of the already existing GESOP model interface. This interface basically consists of a set of routines which supply the set of (physical) equations shown in Section 1.2 plus a small overhead called “Description” to communicate structural information to the underlying algorithms (e.g. number of phases, states, controls, etc.). In addition to the structural information, the user is required to supply upper and lower bounds for all optimization variables as well as an initial guess. The initial guess can either be computed by using a control law in connection with a simulator (see e.g. Ref. [57]), or by reading existing trajectory data from a file.

2.3.2 General Interface to NLP Solvers

Designing a general interface for an NLP solver is not as easy as for an integration method or a collocation scheme. In order to maximize the flexibility of this interface, NLP solvers are coded as sub-packages of the main package that is responsible for all trajectory calculations. Therefore, NLP solvers can access any structural information that is required.

The main program is calling three routines within the NLP solver interface. The first routine, “Initialize”, is responsible for the correct initialization of internal NLP solver data. It has access to the TOPS data structure in order to get the user supplied values for all program parameters, e.g. the optimization tolerance. It also gets several filenames such that output

channels for the iteration log or additional, user supplied configuration files can be read. In case the user has requested a warm start, a previously stored restart record is read and compared to the current problem structure. If there is a mismatch between the restart record and the current problem (e.g. a different number of parameters or constraints), the program automatically switches back to the cold start mode.

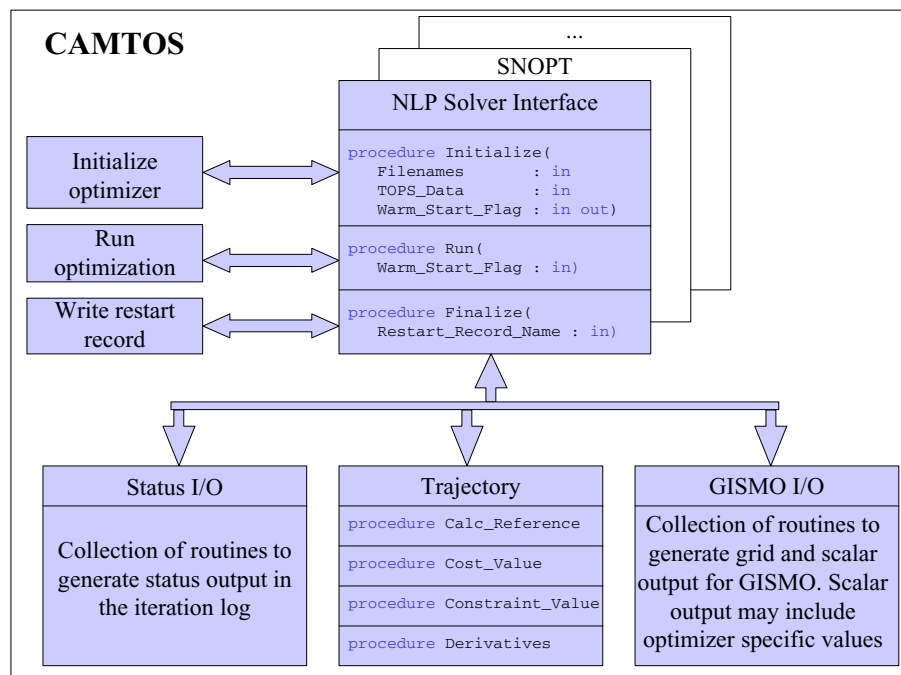


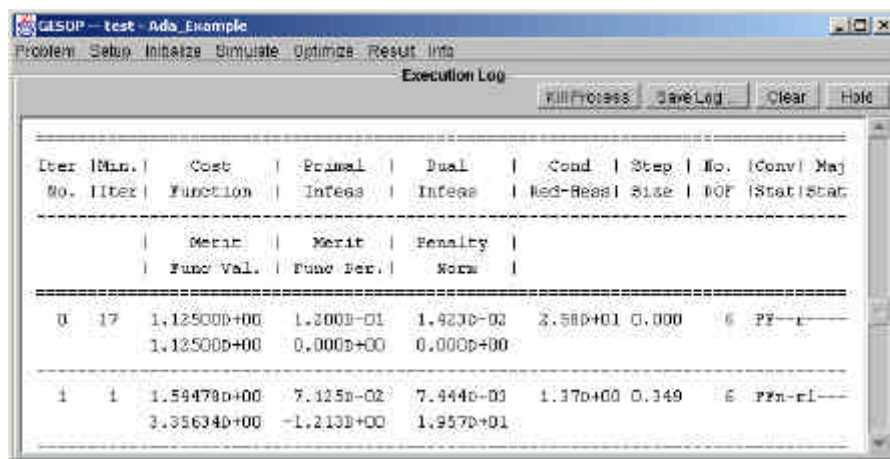
Fig. 2.10: Structure of the NLP solver interface

The second routine, “Run”, starts the optimization process. In case of SNOPT, it calls SNOPT’s main routine and supplies function pointers to the SNOPT-specific routines for evaluating the cost function and the constraint violation vector. These latter routines make use of an additional support package called “Trajectory”. This package supplies high level routines to perform all tasks that are required to calculate a reference trajectory and evaluate the cost function gradient and the Jacobian matrix using sparse finite differencing. The parameter vector as well as its bounds and additional structural information, such as the sparsity pattern of the Jacobian matrix, are obtained directly from the main program.

The “Status I/O” package is used to generate output information for the graphical user interface of GESOP. The status output in the user’s iteration log is very similar to the output of PROMIS and TROPIC. This way, the user can easily get familiar with the software (see Fig. 2.11). The output level can be adjusted such that more or less details are shown.

An extremely useful output to evaluate the progress of the optimizer is generated by the “GISMO I/O” package. The routines within this package generate output files for the Graphical Iteration Sequence Monitor (GISMO) that is integrated into the GESOP software (Fig. 2.12). The information is split into two different kinds of output: scalar and grid information. The scalar data contains all values that appear once during each iteration, e.g. boundary

constraint violations, phase times or design parameters. Each NLP solver can add its own set of additional status values to this set of standard output. In case of SNOPT, all information that is available within the routine “s8user” is added to this data file (e.g. primal and dual infeasibility, see also the “Show Item” list in Fig. 2.12). The grid data contains information about all values at the different grid points, e.g. states, controls and collocation defects. The data is written during the optimization process and CAMTOS makes sure that any intermediate cache is emptied and the data is completely written to the harddisk. Therefore, the display can frequently be updated by GISMO, even while the optimizer is still running.



Iter	Min	Cost	Primal	Dual	Cond	Step	No	(Conv)	Maj
No.	Iter	Function	Infeas	Infeas	Red-Resid	Size	DOF	Stat	Stat
0	17	1.125000+00	1.3000-01	1.9230-03	3.580+01	0.000	6	PF	---
		1.125000+00	0.0000+00	0.0000+00					
1	1	1.594790+00	7.1250-02	7.4440-03	1.370+00	0.349	6	PFn-ri	---
		3.356340+00	-1.2100+00	1.9570+01					

Fig. 2.11: Iteration log output

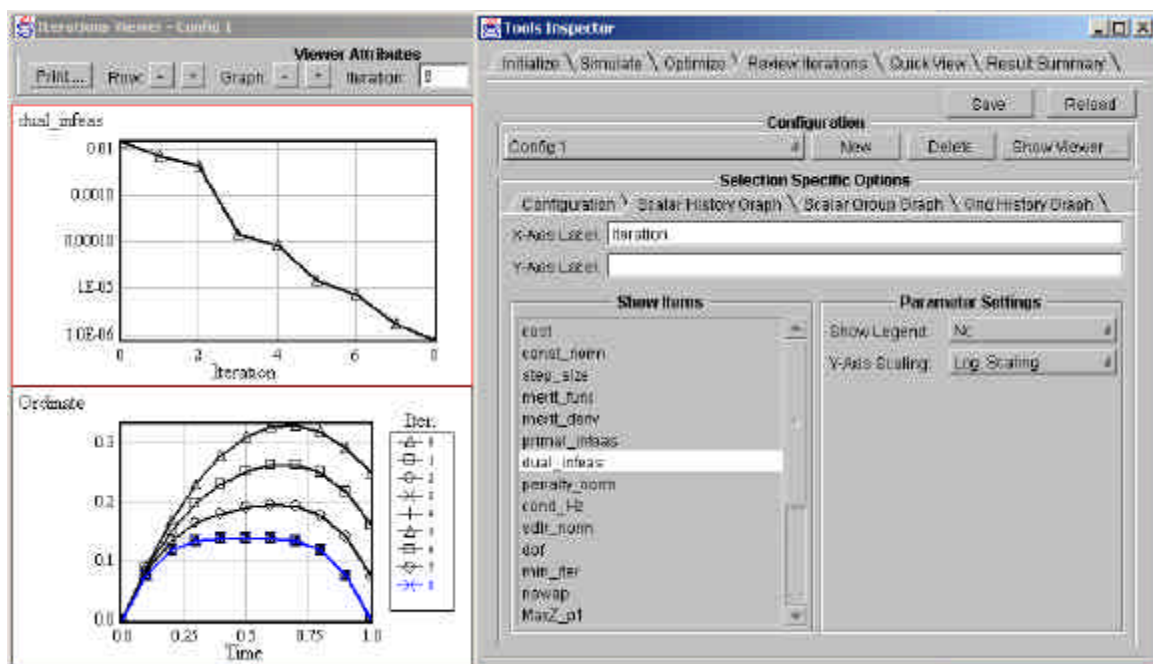


Fig. 2.12: Graphical Iteration Sequence Monitor (GISMO)

Another very important feature of SNOPT is that the optimization process can be stopped after each major iteration if a “stop-flag” is set to true. This feature enables the user to stop an ongoing optimization process before convergence is achieved or the maximum number of iterations is reached. Such a feature is often very useful to prevent the optimizer from completely destroying intermediate results in case of numerical problems, or to stop the optimization process in case the required numerical tolerances cannot be achieved. Especially in connection with GISMO this feature can save a lot of time during an optimization process.

In order to support such a “soft stop”, the GESOP user interface creates a file called “stop_iteration” in the current problem directory whenever the user asks GESOP to stop the optimization after the next major iteration. Within the “s8user” routine, which is called after each major iteration from SNOPT, CAMTOS checks if this file is present and sets the stop-flag of SNOPT accordingly. This approach may not be the most elegant way to perform this task, but it ensures a source-code compatibility across many target platforms.

2.3.3 General Interface to Integration Methods

As it is mentioned above, a variety of different integration methods is implemented in the CAMTOS software. However, it is easy to include additional methods due to a simple software interface which is described in more detail in this section.

The basis for the implementation of any integration method is a generic definition of an integrator interface. This interface consists of two subroutines and one integrator specific data record. In Ada95, this interface is specified with a specification file including so-called abstract record types and subroutines (see Ref. [7], chapter 13.3). In terms of object-oriented programming, this can be seen as the definition of an abstract root-class.

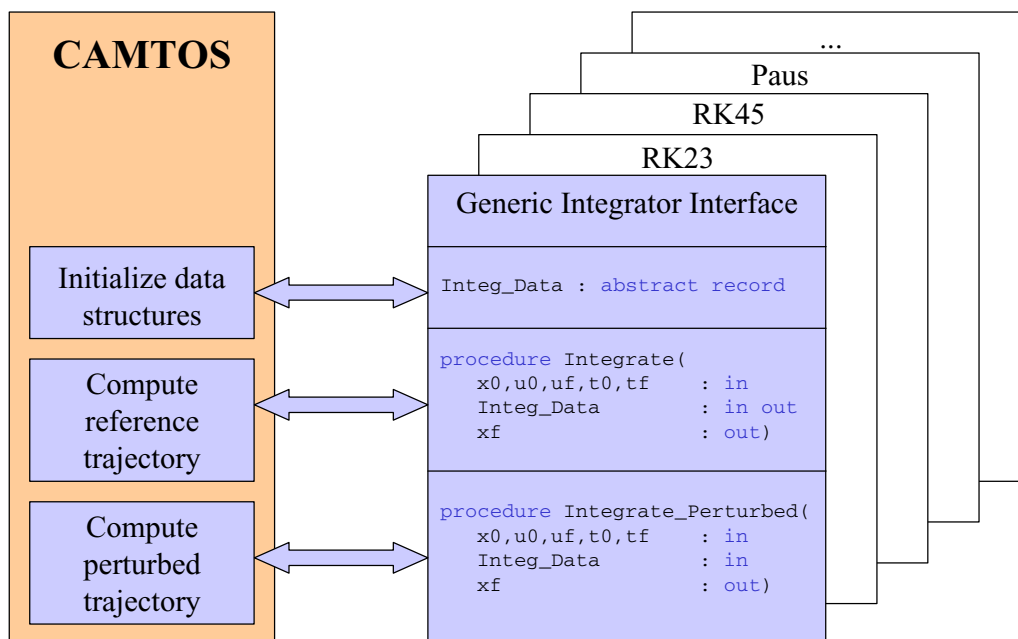


Fig. 2.13: Structure of the general integrator interface

The beauty of this approach is that the main program need not know anything about the details of the available integration methods. When new methods are introduced, only a few small changes have to be made in the initialization routine of the main program to include the new integration method. All other subroutines are using the generic integrator interface only, and the Ada95 compiler will automatically make sure that the correct routines are called during run-time (see Fig. 2.13).

Each integration method is built on top of this basic declaration and is allowed to extend the basic integrator data record with any additional information required. Usually, it becomes necessary to include a dynamic list in this data structure in order to store the step-size sequence. However, in case of the Paus integration method it also becomes necessary to store the required matrices.

As it is mentioned above, two routines need to be coded for each integration method. The first routine, “Integrate”, is called when a new reference trajectory is generated. It is expected that this routine initializes the integrator specific data structure and integrates the trajectory using a step-size control algorithm. Therefore, the following parameters are passed to this routine:

1. initial state vector
2. initial control vector
3. final control vector
4. real parameter vector
5. array of flags marking optimizable real parameters
6. initial (normalized) time of integration interval
7. final (normalized) time of integration interval
8. initial phase time
9. final phase time
10. flags indicating optimizable phase times
11. integrator specific data structure
12. values for required relative and absolute tolerance

The routine is expected to return

1. the state vector at final time and
2. an initialized (or updated) integrator specific data structure.

Note that CAMTOS is calling the integration routines with normalized times (see Section 2.1), i.e. that these normalized times will not change during the optimization process. However, the initial and/or final phase times may change and should be treated as additional optimization parameters if necessary. Such an approach simplifies the construction of a routine for calculating perturbed trajectories, because the integration grid generated for the reference trajectory does not shift in (normalized) time when the phase times are perturbed. It is also the reason for the introduction of the additional derivatives $\mathbf{T}_i = \partial \mathbf{f} / \partial t_i|_0$, $\mathbf{T}_f = \partial \mathbf{f} / \partial t_f|_0$ in the Paus integration method (see Section A.3).

All integrator routines have access to a package that includes routines to evaluate the right-hand side of the dynamics as well as its derivatives (“Right_Hand_Side” and “RHS_Deri”, respectively). The method of supplying a package instead of passing function

pointers is advantageous, because future enhancements to this interface can easily be performed without effecting existing integration methods. If new integration methods, e.g. an integrator that includes event trapping, require additional routines from the user, these routines can easily be included in the integrator model package in addition to the already existing routines. This process does not effect any other areas of the code.

The second routine, “Integrate_Perturbed”, is called during the calculation of the numerical derivatives for the Jacobian matrix and the performance index gradient. Therefore, it only gets a reduced set of inputs:

1. initial state vector
2. initial control vector
3. final control vector
4. real parameter vector
5. initial (normalized) time of integration interval
6. final (normalized) time of integration interval
7. initial phase time
8. final phase time
9. integrator specific data structure

The only output expected from this routine is the final state vector. No changes must be made to the integrator specific data structure. Any implementation must ensure that the second routine performs exactly like the first routine if it is called with the same input values. In addition, it should use the same algorithmic steps as were used for generating the reference trajectory when it is called with slightly perturbed input values. I.e., it should serve as a “consistent function generator” as it is required for the computation of numerical gradients (see Ref. [11] and [45] and Section 2.1.2).

2.3.4 General Interface to Collocation Schemes

The general structure of the collocation interface is very similar to the interface for integration methods discussed in Section 2.3.3. The basic collocation interface consists of four subroutines and a method-specific data structure. Again, an abstract specification of this interface is implemented, and any real implementation of a collocation scheme is derived from this package (see Fig. 2.14 and Ref. [7], Chapter 13.3 for details).

The method-specific data array should hold any data items which are required to compute information on perturbed trajectories or intermediate states. The procedure “Init_Colloc_Data” is used for the initialization of this data structure and is called once for each collocation interval before the optimization starts.

A second subroutine, “Copy_Colloc_Data”, has to be supplied to copy this data structure to a corresponding backup. This is required during the gradient calculation, and since the main program does not have any knowledge on the type of data contained in any method-specific data structure, each collocation method must supply a corresponding “copy” subroutine. Such a routine is not required for the integration methods since only the initial states need to be stored in a backup for the multiple shooting methods. Collocation methods, however, often require additional information to be stored during the calculation of perturbed trajectories.

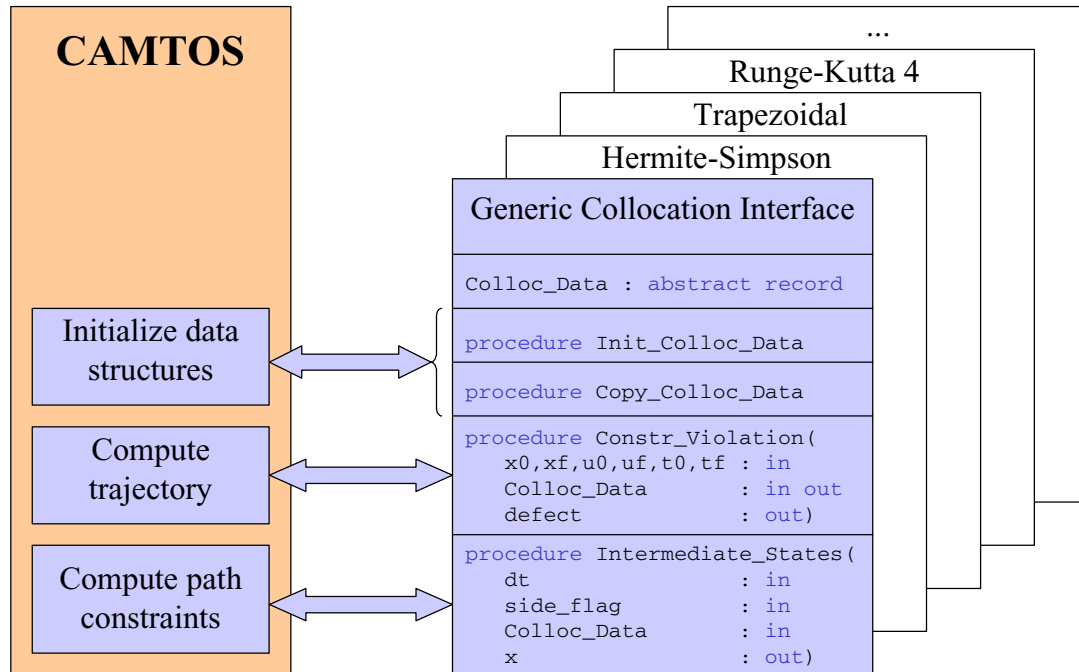


Fig. 2.14: Structure of the general collocation method interface

The third subroutine, “Constr_Violation”, is responsible for calculating the defects. Therefore, it gets a number of input arguments as there are:

1. initial state vector
2. initial control vector
3. initial time of collocation interval
4. final state vector
5. final control vector
6. final time of collocation interval
7. real parameter vector
8. length of collocation interval
9. flag indicating the existence of Lagrange cost
10. flag indicating if changes on left, right, or both sides of the interval have occurred
11. method-specific data structure

The routine is expected to return

1. an updated method-specific data structure (if required),
2. Lagrange cost (if present), and
3. vector of defects.

Note that for collocation phases, the interval boundaries are *not* normalized in time. Computational experience with this approach has shown that such a normalization will in general cause a degraded performance of the overall algorithm for phases with a long duration. This is caused by a significant increase in the numerical round-off errors, because the length of the

collocation intervals can get very small in terms of normalized time. Similar effects have also been observed in the framework of branching with normalized times (see Ref. [36]).

On the other hand, the use of “real” time is causing an “accordion”-like grid-point movement in case that the initial and/or final phase times are optimizable (see Ref. [11], p. 92). Therefore, it is often useful to artificially insert phases with fixed initial and final times to reduce the number of moving grid-points during the optimization process (see Fig. 2.15).

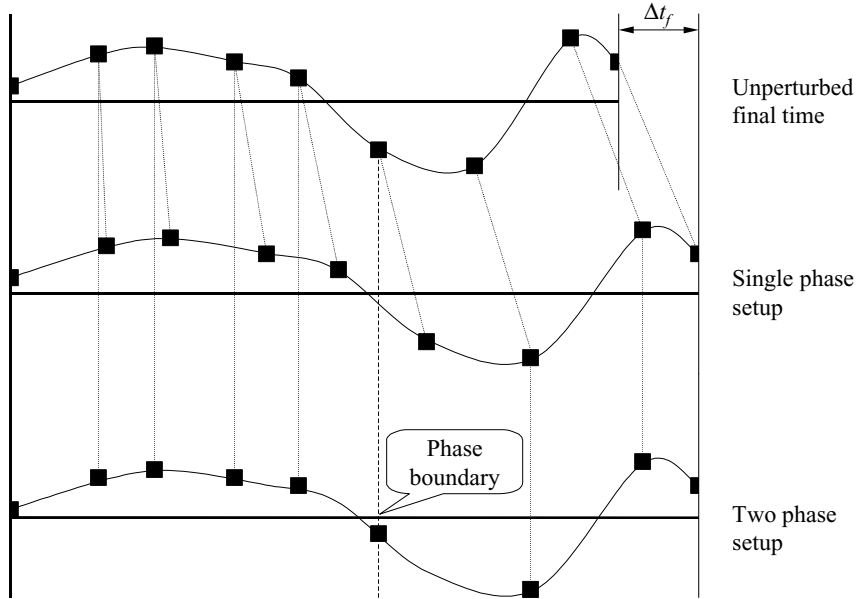


Fig. 2.15: Accordion-like grid-point movement when final time is changed

In case the current phase also involves the computation of a Lagrange cost (integral cost), the collocation method is also expected to return the integrated value of this function over the current interval. The initial value can be set to zero since the main algorithm will later compute the sum over all collocation intervals. Computation of the Lagrange cost within a collocation interval should be based on a simple quadrature formula, such as Simpson’s method which is used by all fourth order collocation schemes currently implemented in CAMTOS:

$$\int_{t_1}^{t_2} L dt = \frac{t_2 - t_1}{6} \left[L(t_1) + 4L\left(\frac{t_1 + t_2}{2}\right) + L(t_2) \right] \quad (2.19)$$

Note that L usually does not only depend on time, but also on the state, control, and real parameter vectors. Especially the state and control vectors must be computed properly in order to evaluate $L[(t_1 + t_2)/2]$ correctly. If such an intermediate state is not available (e.g. in case of a trapezoidal discretization), a lower order quadrature formula, such as the trapezoidal rule, is used:

$$\int_{t_1}^{t_2} L dt = \frac{t_2 - t_1}{2} [L(t_1) + L(t_2)] \quad (2.20)$$

An important performance issue for computing the defects is the fact that the main program also supplies a flag which indicates on which side of the collocation interval changes have been performed. During the generation of a reference trajectory, this flag is always set to “both”. However, during the calculation of the Jacobian matrix and the performance index gradient, it may be set to “left” or “right”, according to the current perturbations. Therefore, several evaluations of the dynamics can be omitted if such values are stored in the method-specific data structure during the calculation of a reference trajectory (or whenever the flag is set to “both”), and if they are reused whenever possible.

The last routine, “Intermediate_States”, is expected to return the state vector at a specific time within a collocation interval. As an input it gets

1. distance to the interval boundary
2. flag indicating if the evaluation has to be performed from left or right side of the collocation interval
3. the method-specific data structure

and it is expected to return the state vector. “Intermediate_States” is called for evaluating path constraints within a collocation interval.

2.4 Implementation Considerations

When implementing any complex software it is obvious that the resulting source code should be well commented and as readable as possible. Especially in highly numerical applications such as a trajectory optimization method, it is very beneficial to use a code profiler in order to investigate the run-time performance of the different code sections. It is a well known fact that the run-time performance of a software code can be significantly improved by improving only a few routines which are frequently called during the execution.

In case of CAMTOS, these “bottle-necks” are the integrators in case of the multiple shooting method, and the matrix and vector operations in case of the collocation method. In addition, the calls to the user supplied model routines, such as the dynamics and constraint evaluation routines, must be minimized. Especially when the ASTOS model is used and extensive data tables must be interpolated, most of the run-time is spent for evaluations inside the user model.

In order to improve the maintainability and enhanceability of the software, a straight-forward implementation is usually preferred in those parts of the code that are not time critical. Routines that involve some “numerical tricks” are equipped with extensive comments and references to the original sources of the applied methods and algorithms.

2.4.1 Roundoff and Truncation Errors

One of the major problems in numerical computations are roundoff errors. Roundoff errors are associated with the computer hardware and the way floating-point numbers are represented. In Ref. [65], Chapter 1.3, a short but very comprehensive overview on this topic can be found. It also gives some formulas to estimate the overall roundoff error after N arithmetic operations:

$$\varepsilon \approx \varepsilon_m \sqrt{N} \quad (2.21)$$

where ε_m is the machine accuracy. However, it is also mentioned that roundoff errors often tend to accumulate preferentially in one direction, leading to an overall error of the order

$$\varepsilon \approx \varepsilon_m N. \quad (2.22)$$

It is important to understand that ε_m is *not* the smallest floating-point number that can be represented on a machine. However, it is the smallest value that can be added to the floating-point value of one that still yields a change in the least significant bit of the mantissa. Any arithmetic operation can be thought of introducing at least an error of ε_m . However, some operations tend to introduce much larger errors, especially subtractions of two numbers in the same order of magnitude. During such operations, the result will include only the (few) low-order significant bits in which the two operands differ. It is important to avoid such situations in an optimization code as much as possible (see also 2.4.3 for a possible workaround).

The second class of error encountered in numerical codes is the truncation error. This error is *not* associated with any hardware and will also occur on a perfect computer with an infinite floating-point accuracy. Furthermore, the truncation error is a characteristic of the algorithm used. This error especially applies for numerical integration methods or any other algorithm using a “discrete” approximation to some desired “continuous” quantity.

It is a general rule that there is not much a programmer can do about roundoff errors, other than choosing algorithms that do not magnify them unnecessarily. On the other hand, truncation errors can directly be influenced by choosing appropriate algorithms.

In some cases, roundoff and truncation errors can interact. In these cases, an otherwise attractive algorithm becomes numerically unstable. This means that any roundoff error that is introduced into the calculation at an early stage is being magnified until it comes to swamp the true answer. Ref. [65] gives a very illustrative example for an unstable algorithm to calculate powers of the so-called “Golden-Mean” given as

$$\kappa \equiv \frac{\sqrt{5}-1}{2} \approx 0.61803398. \quad (2.23)$$

Powers of this value can be calculated as

$$\kappa^{N+1} = \kappa^{N-1} - \kappa^N. \quad (2.24)$$

Thus, knowing the first two values $\kappa^0 = 1$ and κ^1 given in Eq. (2.23), the powers of κ can be calculated by using a simple subtraction instead of a more time consuming multiplication at each stage. Unfortunately, the recurrence in Eq. (2.24) has a second solution, namely $-(\sqrt{5}+1)/2$. Since the magnitude of this undesired solution is greater than one, any small roundoff error introduced into the calculation will grow exponentially. Fig. 2.16 shows that the recursion works fine for the first 40 iterations on a machine with 64-bit wordlength, and it starts giving completely wrong results afterwards.

Such effects need to be avoided especially by the NLP solver that is used. SNOPT, for example, can be configured such that the approximated Hessian matrix is reset after a certain number of iterations. In case of CAMTOS, such a reset of the Hessian matrix is forced after every ten major iterations. If the user supplied optimization model exhibits a lot of numerical noise, it may be required to reset the Hessian matrix more frequently, e.g. after every five major iterations. However, the default value of ten is sufficient for all example problems discussed in this thesis.

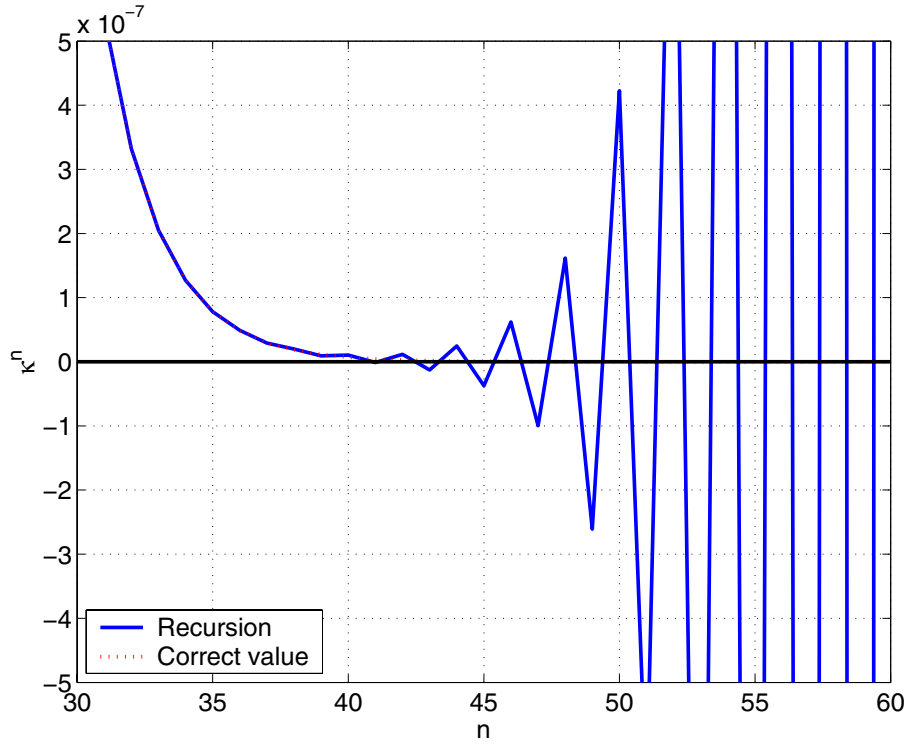


Fig. 2.16: Power series of Golden Mean value evaluated with different algorithms

2.4.2 Parameter and Function Scaling

Scaling is essential for solving any practical optimization problem. Even though there exist some algorithms that are theoretically invariant to scaling, any practical implementation is more or less sensitive to the scaling of the problem. A very comprehensive discussion about scaling can be found in Ref. [41], Chapter 8.5.

In the framework of CAMTOS, several “automatic” scaling methods are implemented which have been found to work very well for the problems encountered during this thesis. However, the user always has the option of turning off the automatic scaling, thereby supplying a well-scaled problem by himself.

2.4.2.1 Parameter Scaling

The major goal of the automatic scaling routines implemented in CAMTOS is to transform all optimization parameters to the same order of magnitude. Two basic schemes are implemented. First of all, a simple, linear scaling where a scaling factor is computed as

$$s_c = \frac{1}{\max(1, |p_0|)} \quad (2.25)$$

and p_0 is the initial value of the parameter to be scaled. In case of a control node at the beginning or at the end of a phase, this scaling factor is multiplied with $1/\sqrt{2}$ in order to reflect the fact that these control nodes only influence one side of a collocation or shooting interval, while control nodes within a phase influence the two bordering intervals. The scaled parameter is calculated as

$$p_c = s_c p_0. \quad (2.26)$$

The second option for automatic scaling is called “affine” and involves the user-supplied upper and lower bounds for each parameter. All parameters are scaled such that their scaled values lie in a range between plus and minus one. Such a scaling method involves a scaling factor

$$s_c = \frac{2}{p_u - p_l} \quad (2.27)$$

as well as an offset calculated as

$$o_c = \frac{p_u + p_l}{p_l - p_u} \quad (2.28)$$

where p_l and p_u are the parameter’s lower and upper bound, respectively. Again, the scaling factor computed with Eq. (2.27) is multiplied with $1/\sqrt{2}$ in case of control nodes at the phase boundaries. The upper and lower bounds are set to plus and minus one, respectively. The scaled parameter is calculated as

$$p_c = s_c p_0 + o_c \quad (2.29)$$

Practical experience has shown that the simple linear scaling works very well for all optimization problems encountered in the framework of this thesis. The affine approach requires the user to supply reasonable bounds for all parameters which is sometimes difficult to do. In addition, the affine approach easily leads to larger round-off errors if there is a large difference in the magnitudes of the bounds and the parameter value. Therefore, warnings are issued whenever the absolute difference between a bound and the original parameter value is greater than 1000.

Both scaling methods are also discussed in Ref. [41], Chapter 7.5.

2.4.2.2 Scaling of the Performance Index

The performance index is scaled by using its initial value for calculating a scaling factor according to Eq. (2.30). Note that s_c is always set to one if the magnitude of J_0 is less than 100.

$$s_c = \frac{1}{|J_0|} \quad (2.30)$$

2.4.2.3 Default Scaling of Constraints

The default scaling for all constraints is based on the initial constraint violation C_0 . The scaling factor is computed as

$$s_c = \frac{1}{|C_0|}. \quad (2.31)$$

Note that s_c is always set to one if the initial constraint violation C_0 is less than 100. A special treatment is applied for the continuity conditions at multiple shooting points, the defects in collocation intervals, and for phase connect conditions.

2.4.2.4 Scaling of Continuity Conditions in Multiple Shooting Phases

For the continuity conditions in a multiple shooting phase the scaling factor is calculated as

$$s_c = \frac{2}{|x_l| + |C_0| + |x_r|}, \quad (2.32)$$

where x_l and x_r are the state values “just left” (the integrated state at the end of the current shooting interval) and “just right” (the initial state of the next shooting interval) of the continuity constraint, and C_0 is the initial constraint violation. However, this scaling factor is only applied if the denominator in Eq. (2.32) is greater than one. Otherwise, s_c is set to one.

2.4.2.5 Scaling of Defects in Collocation Phases

For the collocation method, a similar approach for the scaling is chosen as for the multiple shooting method connect conditions:

$$s_c = \frac{2}{h[|x_1| + |\Delta_{12}| + |x_2|]} \quad (2.33)$$

Again, this scaling factor is only applied when the denominator in Eq. (2.33) is greater than one. Otherwise, s_c is set to one.

2.4.2.6 Scaling of Phase Connect Conditions

In case of the phase connect conditions, a scaling factor of

$$s_c = \frac{2}{|C_0| + 2|x_r|} \quad (2.34)$$

is applied in case the denominator of Eq. (2.34) is greater than one. Otherwise, s_c is set to one. x_r is the initial value of the connected state at the beginning of the next phase,

2.4.3 Selecting the Parameter Perturbation Size for Gradient Calculation

Selecting a parameter perturbation size during the calculation of the Jacobian matrix and the cost function gradient is based on the simple heuristic

$$\Delta p = \sqrt{\epsilon_m} \cdot \max(|p|, 1), \quad (2.35)$$

where ϵ_m is the machine accuracy (approximately 2.22×10^{-16} for a Pentium-III processor using double precision). Before applying this perturbation, it is made sure that the perturbed parameter remains within its bounds. In case the upper bound is violated after applying the parameter perturbation, the perturbation is applied in the opposite direction.

In addition, a numerical trick is employed in order to reduce truncation errors. After the perturbed parameter is calculated as

$$\tilde{p} = p + \Delta p, \quad (2.36)$$

the perturbation used to calculate the gradients is calculated as

$$\Delta \tilde{p} = \tilde{p} - p. \quad (2.37)$$

Mathematically, Eq. (2.37) yields the same value as Eq. (2.35), but it is discussed in Ref. [65], p. 388, that this procedure reduces the finite precision error introduced by the summation in Eq. (2.36).

In Chapter 8.6.2 of Ref. [41], another procedure for an “Automatic Estimation of Finite-Difference Intervals” is discussed. However, since the method shown above works very well in practice, the much more complicated algorithm in Ref. [41] is not implemented.

2.5 Setup of Multi-Phase Problems

For multi-phase problems as defined in Section 1.2, Eqs. (1.6)-(1.12), it is possible to choose a combination of direct multiple shooting and direct collocation. Each phase may use a different transcription method. The different phases are connected by using the phase connect conditions in Eq. (1.11) as well as the time connect condition in Eq. (1.12). The different sparsity patterns are simply patched together, which yields an overall sparsity pattern of the Jacobian matrix which looks similar to Fig. 2.17.

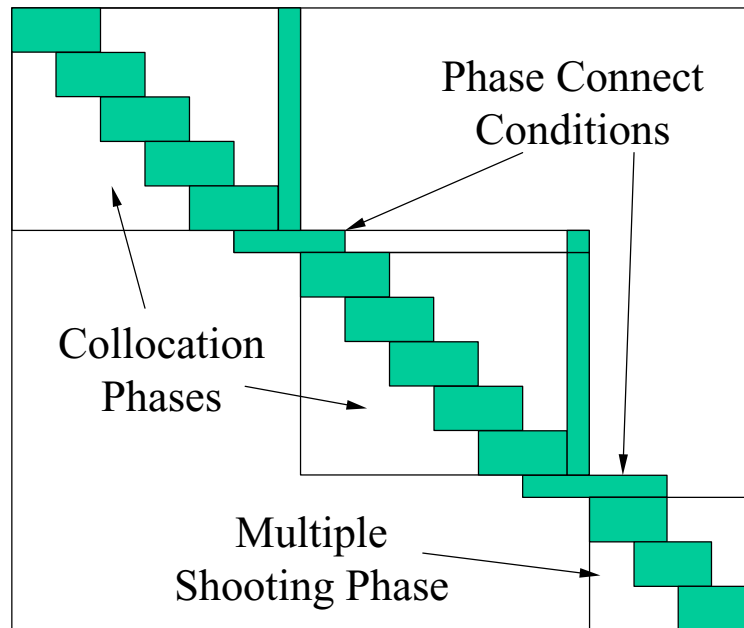


Fig. 2.17: Sparsity pattern of Jacobian matrix in a multi-phase problem

2.6 NLP Solvers

Currently, SNOPT 6.1 is the only NLP solver completely integrated in the CAMTOS software. However, initial experiments have been performed with a constrained simulated annealing algorithm. Other NLP solvers can be integrated, as long as the general interface defined in Section 2.3.2 provides enough functionality.

2.6.1 SNOPT 6.1

SNOPT (Ref. [42]) is an SQP method using a BFGS update formula to approximate the Hessian matrix of the problem to be solved. Therefore, SNOPT requires the user to supply routines that compute the performance index, the constraint vector, and the first derivatives of each of them. While SNOPT does not exploit any sparsity in the gradient of the performance index, it does exploit sparsity in the Jacobian matrix. It finds solutions that are locally optimal.

Besides nonlinear constraints, it is also possible to communicate a set of linear constraints to SNOPT. However, this feature is not used in the framework of CAMTOS since it would significantly increase the implementation complexity, reduce generality, and hardly increases the performance of the SQP method.

According to the SNOPT user manual, the algorithm is suitable for nonlinear problems with thousands of parameters and constraints, but is not suited for problems with thousands of degrees of freedom. That means, that for large problems, many constraints and parameter bounds should be active at a solution. The latter fact makes SNOPT very useful in the framework of trajectory optimization problems, since the continuity conditions and defects form a

large set of equality constraints that are active at a solution. However, problems may occur when a problem involves many path constraints that do not become active at the solution.

One important implementation specific detail of SNOPT is that in the final result, the optimization parameters may be outside their bounds as much as the so-called “Minor feasibility tolerance”. This fact is important to know and must also be communicated to the end-user of CAMTOS, since it prohibits a problem formulation, where the parameter bounds reflect the boundaries of the permissible parameter domain. E.g., the lower state bound for eccentricity when using Kepler elements for orbit propagation must not be specified as zero since negative eccentricities do not exist by definition. Before saving a final result, CAMTOS makes sure that all parameters are located within or at least at their bounds. Corresponding warning messages are issued if some parameters have been found to be out of bounds.

2.6.2 Other Methods

Initial experiments have been performed with NLP solvers that are able to find a globally optimal solution. Recent developments in the framework of simulated annealing and genetic algorithms have resulted in codes that are able to handle constrained problems as well (see Refs. [79]-[83]). The beauty of these methods is that they are derivative free and, in addition, do not require the constraints and/or objective function to be continuous and/or differentiable.

However, both, simulated annealing and genetic algorithms, are computationally expensive. Especially when they are applied in a framework where one evaluation of the performance index and constraint vector is dominating any other operation, the specific parameters of such methods must be tuned in order to reduce the number of function evaluations to a minimum.

Due to the serial nature of simulated annealing, it does not seem very promising to use such an algorithm in the framework of direct trajectory optimization methods for performance reasons. However, a genetic algorithm may be used if the computation of one population can be performed in parallel on different processors or computers (see e.g. Ref. [10]).

2.7 Performance of the Direct Optimization Method

In this section, the direct optimization method CAMTOS is compared with the two existing optimization methods within the GESOP environment: PROMIS and TROPIC. In addition, the benefits of using a combination of direct multiple shooting and direct collocation is demonstrated.

2.7.1 Performance of the Direct Multiple Shooting Method

For the performance evaluation of the direct multiple shooting method, a dual payload mission of the Ariane 5 launcher is chosen. A detailed description of the test case setup is given in Ref. [6], Appendix A.5. The same test case also serves as a basis for a more detailed investigation in Section 4 of this thesis.

Starting from the same initial guess, the mission is optimized using PROMIS/SNOPT and CAMTOS in a direct multiple shooting setup using the Runge-Kutta 4/5 integration method. The resulting trajectories show no visible difference (see Fig. 2.18) and the computed payloads are comparable within the numerical accuracy of 10^{-5} for the scaled performance index that is used for this optimization. As shown in Table 2.1, CAMTOS requires a slightly smaller number of iterations than PROMIS/SNOPT. However, this can easily change when a different initial guess is used. When looking at a larger variety of test cases, it can be concluded that the performance of PROMIS/SNOPT and CAMTOS using the Runge-Kutta 4/5 method is virtually the same. Since both methods are using the same discretization method and the same SQP solver, this result shows that the new implementation achieves the same, good performance as the existing method.

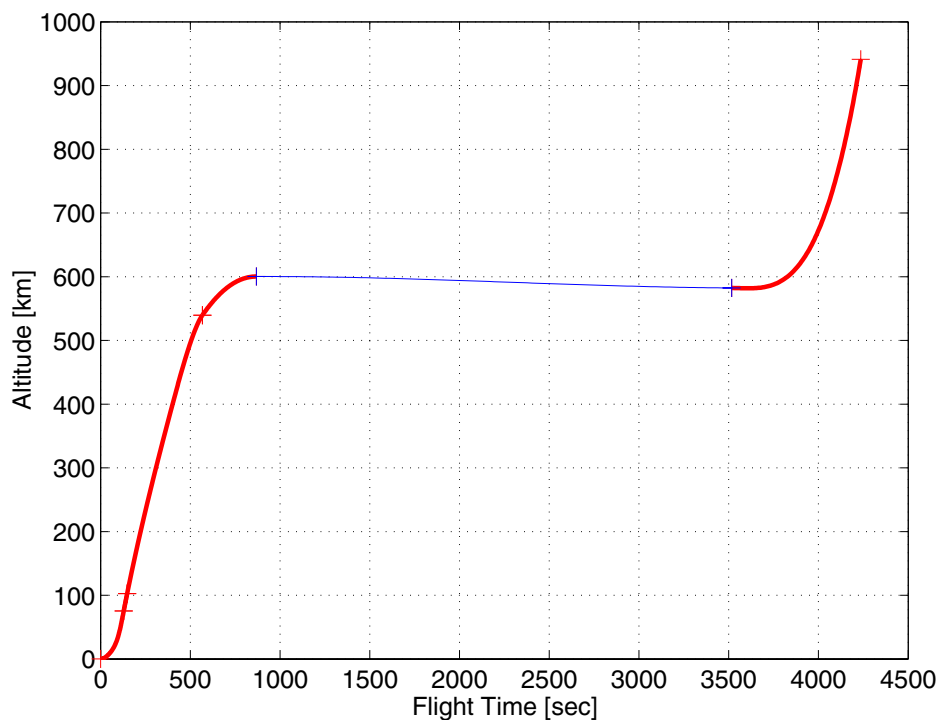


Fig. 2.18: Altitude profile for Ariane 5 dual payload mission

Method	Iterations	Payload [kg]
CAMTOS/RK45	42	1,986.7
PROMIS/SNOPT	56	1,985.5

Table 2.1: Computational performance of CAMTOS versus PROMIS

2.7.2 Performance of the Direct Collocation Method and its Combination with Direct Multiple Shooting

In order to evaluate the performance of the direct collocation method, an Ariane 5 mission to a Molnija orbit is selected from Ref. [6], Appendix A.2. This testcase is fine-tuned for an optimization with the existing, direct collocation method TROPIC. TROPIC is also using SNOPT for solving the discretized problem, and the transcription is based on Hermite-Simpson polynomials.

The mission is optimized with TROPIC and CAMTOS using direct collocation with Hermite-Simpson-Polynomials, starting from the same initial guess. The resulting trajectories show no difference (see Fig. 2.19). The mission involves a rather long coast arc which makes it highly sensitive to even small differences in the solution.

Table 2.2 shows that the computed payloads show a small deviation in the range of 5 kg. This difference is caused by the slightly different transcription methods and could be reduced by introducing additional collocation intervals into the problem. Note that CAMTOS is used in a pure direct collocation mode using Hermite-Simpson polynomials (CAMTOS HS), and, in a second optimization run, the first phase is discretized using multiple shooting with a Runge-Kutta 4/5 integrator while all other phases are using Hermite-Simpson polynomials (CAMTOS RK45/HS). The latter setup requires some additional iterations, but it significantly improves the solution accuracy compared to a pure collocation setup. The reason is that the Ariane 5 boosters follow a predefined thrust profile during the first 123 seconds (see Fig. 2.20). For the collocation setup, collocation nodes are placed exactly at the data points in order to achieve a good approximation of the state time histories. When using a multiple shooting method during this phase, there is no necessity of placing nodes exactly at the data points. The step-size control within the integration method ensures that this phase is accurately propagated.

Both, the pure collocation setup and the mixed setup, require a significantly smaller number of iterations than the optimization with TROPIC. The same performance is achieved when running several other test cases using the direct collocation method of CAMTOS and comparing it to TROPIC. Since both algorithms are using SNOPT to solve the underlying NLP problems, it can be concluded that the slightly different transcription of the optimal control problem that also involves control continuity conditions at the boundaries of each collocation interval significantly improves the convergence behavior of SNOPT. Even though additional variables and constraints are introduced into the problem, the run-time performance of CAMTOS is between 1.5 and 2 times better than the performance of TROPIC.

Method	Iterations	Payload [kg]
CAMTOS HS	95	6,916.5
CAMTOS RK45/HS	149	6,921.3
TROPIC	177	6,920.9

Table 2.2: Computational performance of CAMTOS versus TROPIC

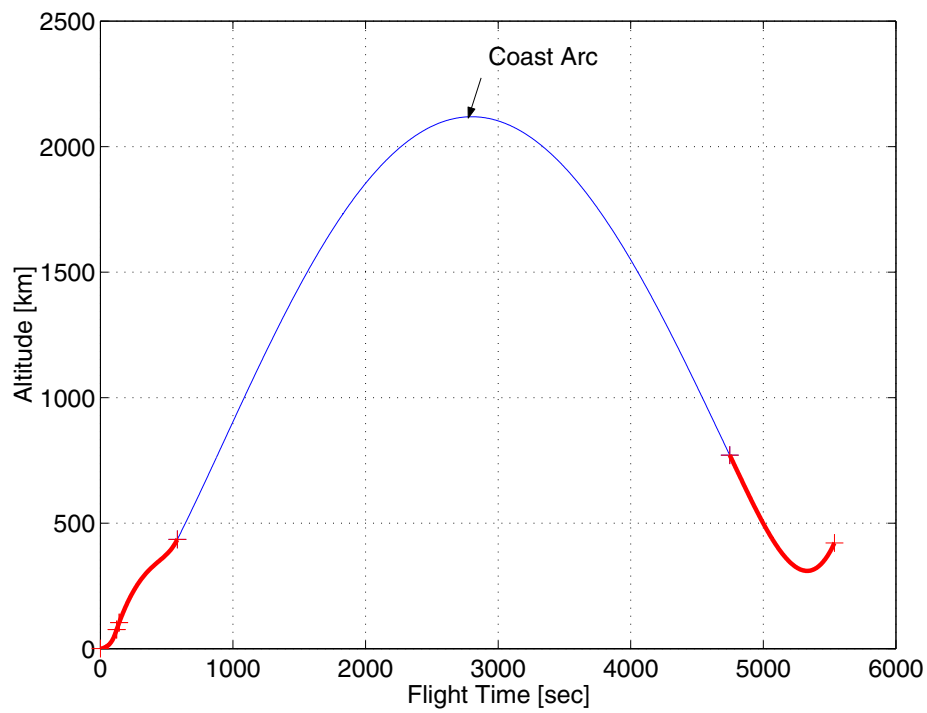


Fig. 2.19: Altitude profile of Ariane 5 mission to a Molnija orbit

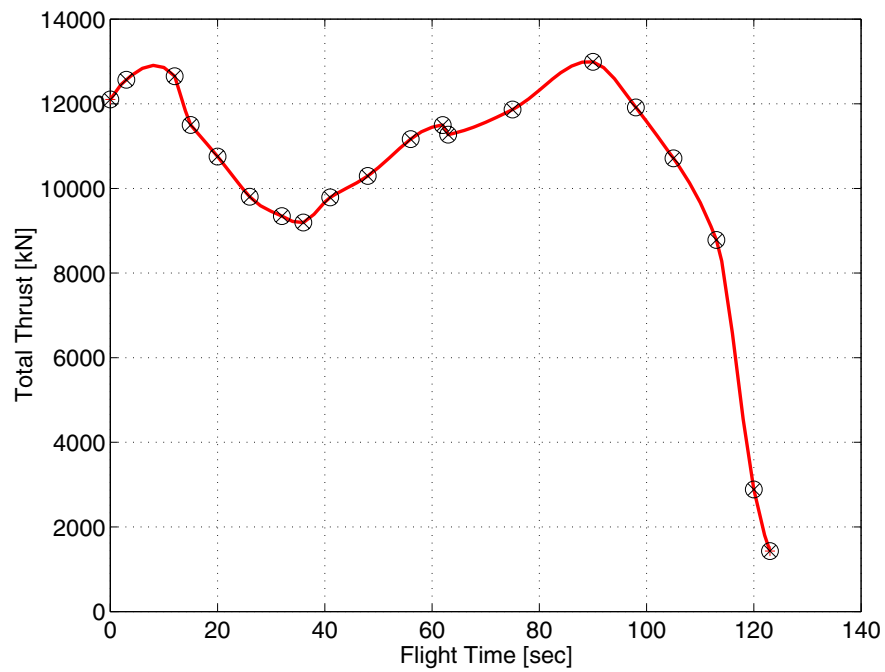


Fig. 2.20: Thrust profile of during the first phase of the rocket ascent

3 Indirect Method for Optimal Thrust Arcs in Vacuum

The main difficulty with these methods is “getting started”, i.e. finding a first estimate of the unspecified conditions at one end that produces a solution reasonably close to the specified conditions at the other end.

Arthur E. Bryson, Jr., Yu-Chi Ho in “Applied Optimal Control”, Chapter 7.3

One of the most serious drawbacks for the application of indirect optimization methods is the requirement for an accurate estimate of the initial costates. Once the dynamics involve dissipative terms such as friction or drag, small errors in the initial costates may produce enormous errors at final time. It is not unusual that the values at final time, or even earlier during the trajectory integration, exceed the numerical range of the computer (see Ref. [23], p. 214f).

However, it turns out that it is possible to compute the initial costates for optimal thrust arcs in vacuum fairly easy using almost arbitrary initial guesses. The method applied within the framework of this thesis is based on Refs. [17], [26], [27], [33], [34], and [37].

3.1 Formulation of Dynamics

The dynamics of a rocket in vacuum can be represented in an inertial, Cartesian coordinate frame (Fig. 3.1) as

$$\begin{bmatrix} \dot{V} \\ \dot{\mathbf{R}} \\ \dot{m} \end{bmatrix} = \begin{bmatrix} \mathbf{G} + \frac{T}{m} \mathbf{i}_b \\ V \\ -q \end{bmatrix}, \quad (3.1)$$

where \mathbf{G} is the gravity acceleration vector, and \mathbf{i}_b is a unit vector along thrust direction which is the control vector in this problem. Vacuum thrust T and massflow q may be given as constant

values or, in case of solid rocket boosters, as fixed functions of time. For a simple inverse square gravity law, \mathbf{G} can be calculated as

$$\mathbf{G} = -\frac{\mu}{r^2} \mathbf{i}_R = \mathbf{G}_0 \quad (3.2)$$

where \mathbf{i}_R is a unit vector along \mathbf{R} .

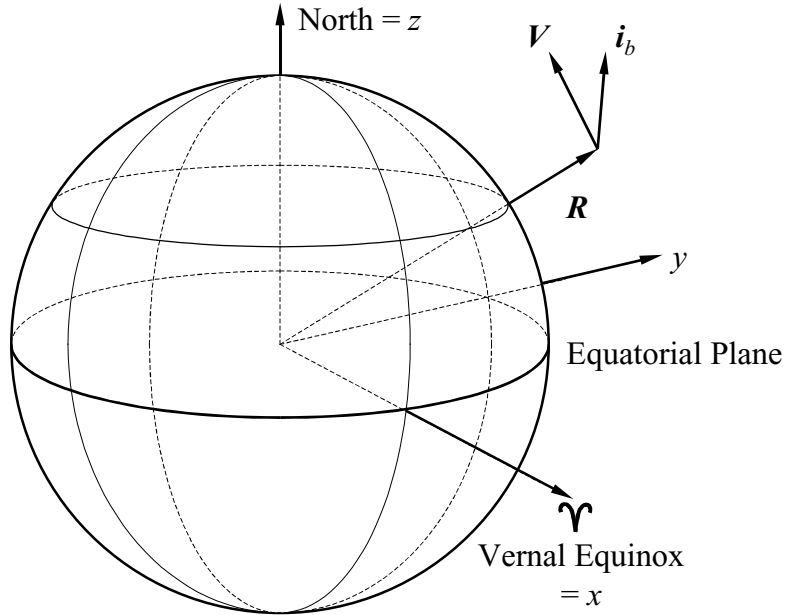


Fig. 3.1: Definition of the inertial coordinate frame

3.1.1 Gravity Field with J_2 terms

Due to the oblateness of the Earth, the simple inverse square gravity law shown in Eq. (3.2) is often not accurate enough. Therefore, J_2 terms are added to the problem formulation, which yields

$$\mathbf{G} = \mathbf{G}_0 - \frac{\mu}{r^2} \begin{bmatrix} \frac{3}{2} J_2 \left(\frac{r_E}{r} \right)^2 (1 - 3 \sin^2 \delta) \\ 0 \\ 3 J_2 \left(\frac{r_E}{r} \right)^2 \sin \delta \cos \delta \end{bmatrix}_L \quad (3.3)$$

in a local horizontal system L (Ref. [9], p. 407 and Ref. [30], p. 204ff). Other zonal harmonics such as J_3 or J_4 are neglected since they hardly have any influence for most central bodies and missions of a rather short duration of up to a few hours. Eq. (3.3) can be transformed to the inertial, Cartesian coordinate frame which yields

$$\mathbf{G} = \mathbf{G}_0 - \frac{\mu}{r^2} \left[\frac{3}{2} J_2 \left(\frac{r_E}{r} \right)^2 (1 - 3 \sin^2 \delta) \right] \mathbf{i}_R - \frac{\mu}{r^2} \left[3 J_2 \left(\frac{r_E}{r} \right)^2 \sin \delta \cos \delta \right] \cdot \frac{\mathbf{i}_R \times (\mathbf{i}_N \times \mathbf{i}_R)}{|\mathbf{i}_N \times \mathbf{i}_R|}, \quad (3.4)$$

where \mathbf{i}_N is a unit vector pointing north and δ is the declination (see Fig. 3.2). The terms $\sin \delta$ and $\cos \delta$ can be replaced with

$$\sin \delta = \mathbf{i}_R^T \mathbf{i}_N \text{ and } \cos \delta = |\mathbf{i}_R \times \mathbf{i}_N|. \quad (3.5)$$

This also cancels out the potential singularity for $\delta = 90$ deg in the last term of Eq. (3.4). Note that the gravity acceleration only depends on the radius vector. Also note that \mathbf{G} has one component along the radius vector and one component perpendicular to the radius vector within the plane of radius and north vector. Using Eqs. (3.4) and (3.5), the gravity acceleration vector can be rewritten as

$$\mathbf{G} = \mathbf{G}_0 + \mathbf{G}_R + \mathbf{G}_N \quad (3.6)$$

where \mathbf{G}_R is the acceleration component caused by J_2 along \mathbf{i}_R and \mathbf{G}_N is the acceleration component caused by J_2 along $\mathbf{i}_R \times (\mathbf{i}_N \times \mathbf{i}_R)$. \mathbf{G}_R and \mathbf{G}_N are defined as

$$\mathbf{G}_R = \frac{C_1}{2} [1 - 3(\mathbf{i}_R^T \mathbf{i}_N)^2] \mathbf{i}_R \text{ and } \mathbf{G}_N = C_1 (\mathbf{i}_R^T \mathbf{i}_N) [\mathbf{i}_R \times (\mathbf{i}_N \times \mathbf{i}_R)] \quad (3.7)$$

$$\text{with } C_1 = \frac{-3\mu J_2 \left(\frac{r_E}{r} \right)^2}{r^2}. \quad (3.8)$$

3.1.2 Altitude over an Oblate Planet

It turns out that the effect caused by the additional J_2 accelerations on a launcher trajectory is fairly small. However, the effect of the Earth's oblateness on the altitude is much larger. Therefore, it is important to incorporate the correct calculation of the altitude in any output generated by such a method. As it is shown in Ref. [30], p. 75, the local Earth radius can be calculated as

$$r_E = \frac{r_{eq}(1-f)}{\sqrt{1-f(2-f)\cos^2 \phi'}} \quad (3.9)$$

where

$$f = 1 - \frac{r_{pol}}{r_{eq}} \quad (3.10)$$

is the flattening factor, r_{eq} is the equatorial radius, and r_{pol} is the polar radius. The geometry is shown in Fig. 3.2.

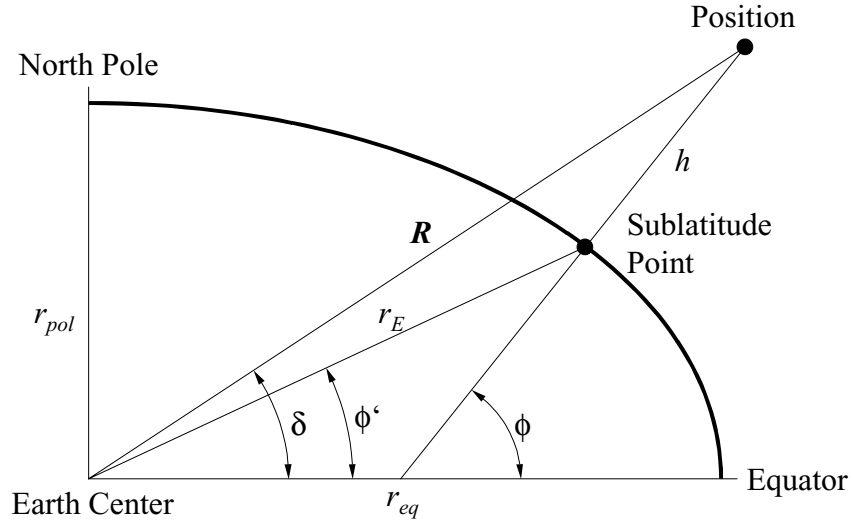


Fig. 3.2: Altitude over an oblate planet

Because the difference between ϕ' and δ is usually very small for low Earth orbits, ϕ' in Eq. (3.9) can be replaced with δ which can easily be calculated by using Eq. (3.5). Substituting $\cos\phi$ in Eq. (3.9) yields:

$$r_E \approx \frac{r_{eq}(1-f)}{\sqrt{1-f(2-f)|\mathbf{i}_R \times \mathbf{i}_N|^2}} \quad (3.11)$$

An approximation for the altitude can now be computed as

$$h \approx |\mathbf{R}| - r_E. \quad (3.12)$$

3.1.3 Including Backpressure in the Thrust Calculation

A small correction to incorporate atmospheric effects on thrust can easily be included in Eq. (3.1) by changing the thrust calculation to

$$T = T_{vac} - A_e p_a \quad (3.13)$$

where T_{vac} is the vacuum thrust calculated as $T_{vac} = I_{sp} g_0 \dot{m}$ (see Ref. [44], p. 165), A_e is the nozzle exit area, and p_a is the ambient atmospheric pressure.

A similar approach can be used to incorporate the aerodynamic effects. The required equations are shown in Ref. [34]. However, such dissipative terms tend to destabilize the solution process of the two-point boundary value problem and a special solution process such as a homotopy method is required to compute the initial costates. Since in the following only thrust maneuvers in vacuum are discussed, the interested reader is referred to Refs. [33], [34], and [37] for a further discussion on this topic.

3.2 Adjoint Differential Equations

Based on Section 3.1, the Hamiltonian can be formulated as

$$H = L + \mathbf{P}^T \left(\mathbf{G} + \frac{T}{m} \mathbf{i}_b \right) + \mathbf{Q}^T \mathbf{V} - \lambda_m q, \quad (3.14)$$

where L represents the integral part of the performance index

$$J = \phi(\mathbf{x}_f, t_f) + \int_{t_0}^{t_f} L dt \quad (3.15)$$

which is minimized. By setting $L = q$ and $\phi = 0$ to minimize the fuel consumption, the adjoint differential equations can be calculated as

$$\begin{bmatrix} \dot{\mathbf{P}} \\ \dot{\mathbf{Q}} \\ \dot{\lambda}_m \end{bmatrix} = \begin{bmatrix} -\mathbf{Q} \\ -(\partial \mathbf{G} / \partial \mathbf{R})^T \mathbf{P} \\ T(\mathbf{i}_b^T \mathbf{P}) / m^2 \end{bmatrix}. \quad (3.16)$$

For the simple inverse square law gravity field shown in Eq. (3.2), $(\partial \mathbf{G} / \partial \mathbf{R})^T$ yields

$$\left(\frac{\partial \mathbf{G}}{\partial \mathbf{R}} \right)^T = \left(\frac{\partial \mathbf{G}_0}{\partial \mathbf{R}} \right)^T = \frac{\mu}{r^3} \left[3 \mathbf{i}_R \mathbf{i}_R^T - \mathbf{I}_3 \right] \quad (3.17)$$

where \mathbf{I}_3 is the 3×3 identity matrix.

3.2.1 Adjoint Differential Equations including J_2 Terms

After applying some tedious algebra, it can be shown that the derivatives of Eqs. (3.7) w.r.t. the radius vector can be computed as

$$\left(\frac{\partial \mathbf{G}_R}{\partial \mathbf{R}} \right)^T = C_2 \left\{ \frac{1}{2} \left[1 - 3(\mathbf{i}_R^T \mathbf{i}_N)^2 \right] \mathbf{I}_3 - \frac{7}{2} \left[1 - 3(\mathbf{i}_R^T \mathbf{i}_N)^2 \right] \mathbf{i}_R \mathbf{i}_R^T + \mathbf{i}_R \left[\mathbf{i}_R - 3(\mathbf{i}_R^T \mathbf{i}_N) \mathbf{i}_N \right]^T \right\} \quad (3.18)$$

and

$$\begin{aligned} \left(\frac{\partial \mathbf{G}_N}{\partial \mathbf{R}} \right)^T = C_2 \left\{ \mathbf{i}_R^T \mathbf{i}_N \left[2 \mathbf{i}_N \mathbf{i}_R^T - \mathbf{i}_R \mathbf{i}_N^T - (\mathbf{i}_R^T \mathbf{i}_N) \mathbf{I}_3 \right] - 7(\mathbf{i}_R^T \mathbf{i}_N) [\mathbf{i}_R \times (\mathbf{i}_N \times \mathbf{i}_R)] \mathbf{i}_R^T \right. \\ \left. + [\mathbf{i}_R \times (\mathbf{i}_N \times \mathbf{i}_R)] \mathbf{i}_N^T \right\} \end{aligned} \quad (3.19)$$

where

$$C_2 = \frac{-3\mu J_2 \left(\frac{r_E}{r}\right)^2}{r^3} = \frac{C_1}{r}. \quad (3.20)$$

3.2.2 Correction for Backpressure Term

If thrust is computed including a backpressure term, the adjoint equations need to be corrected with additional terms for $\dot{\mathbf{Q}}$. Note that the Hamiltonian changes to

$$H = L + \mathbf{P}^T \left(\mathbf{G} + \frac{T_{vac}}{m} \mathbf{i}_b - \frac{A_e p_a}{m} \mathbf{i}_b \right) + \mathbf{Q}^T \mathbf{V} - \lambda_m q \quad (3.21)$$

where the ambient pressure p_a is assumed to depend on altitude only. Applying the chain rule to construct the derivative for p_a w.r.t. the radius vector yields

$$\dot{\mathbf{Q}} = - \left(\frac{\partial \mathbf{G}}{\partial \mathbf{R}} \right)^T \mathbf{P} + \frac{A_e}{m} \cdot \frac{\partial p_a}{\partial h} \left(\frac{\partial h}{\partial \mathbf{R}} \right)^T (\mathbf{P}^T \mathbf{i}_b). \quad (3.22)$$

Note that

$$\left(\frac{\partial h}{\partial \mathbf{R}} \right)^T = \mathbf{i}_R - \left(\frac{\partial r_E}{\partial \mathbf{R}} \right)^T \quad (3.23)$$

where $(\partial r_E / \partial \mathbf{R})^T$ is zero for a spherical Earth. In case an oblate Earth model is used, it can be calculated from Eq. (3.11) as

$$\left(\frac{\partial r_E}{\partial \mathbf{R}} \right)^T = \frac{r_E}{r} \left[\mathbf{i}_R - \frac{\mathbf{i}_R + f(2-f)\mathbf{i}_N \times (\mathbf{i}_N \times \mathbf{i}_R)}{1 - f(2-f)|\mathbf{i}_R \times \mathbf{i}_N|^2} \right]. \quad (3.24)$$

The corresponding values for $\partial p_a / \partial h$ must be supplied by the atmospheric model.

3.3 Optimal Control

For calculating the optimal control \mathbf{i}_b it must be noted that \mathbf{i}_b is constrained to be a unit vector. Therefore, a path constraint

$$1 - \mathbf{i}_b^T \mathbf{i}_b = 0 \quad (3.25)$$

can be formulated which is adjoined to the Hamiltonian with an associated Lagrange multiplier λ_b such that

$$H = L + \mathbf{P}^T \left(\mathbf{G} + \frac{T}{m} \mathbf{i}_b \right) + \mathbf{Q}^T \mathbf{V} - \lambda_m q + \lambda_b (1 - \mathbf{i}_b^T \mathbf{i}_b). \quad (3.26)$$

Formulating the optimality condition yields

$$\left(\frac{\partial H}{\partial \mathbf{i}_b} \right)^T = \frac{T}{m} \mathbf{P} - 2\lambda_b \mathbf{i}_b = \mathbf{0} \quad \text{or} \quad \mathbf{i}_b = \frac{T}{2m\lambda_b} \mathbf{P}. \quad (3.27)$$

Noting that the Legendre-Clebsch condition is

$$\frac{\partial^2 H}{\partial \mathbf{i}_b^2} = -2\lambda_b \cdot I_3 \geq \mathbf{0}, \quad (3.28)$$

the Lagrange multiplier λ_b must be less or equal to zero. Therefore, the control vector \mathbf{i}_b must point in the opposite direction of the unit vector along \mathbf{P} :

$$\mathbf{i}_b = \frac{-\mathbf{P}}{|\mathbf{P}|} \quad (3.29)$$

This is a well-known result in the classical literature (see e.g. Refs. [17], [18]). It means that the body axis (thrust vector) has to point in the opposite direction of the costate vector \mathbf{P} , which is also known as the “primer-vector”. It can easily be seen that the correction for the backpressure term does not change anything in the derivation of the optimal control vector, since the magnitude of T does not depend on the controls \mathbf{i}_b . In Ref. [56] a detailed mathematical proof for the optimality of Eq. (3.29) is presented.

3.4 Boundary Conditions

In order to correctly formulate an optimal control problem using the indirect approach, the transversality conditions must be formulated in addition to the initial and final boundary conditions. In the following, we expect the initial position, velocity, and mass to be fixed, and the final conditions to be specified by a set of terminal constraints. Based on the choice of the terminal constraints, the corresponding sets of transversality conditions are derived.

All transversality conditions are derived for a free final time problem, where the overall fuel consumption is minimized ($L = q$, $\phi = 0$). Therefore, as long as neither ϕ nor any of the terminal constraints Ψ depends explicitly on time, the final value of the Hamiltonian must be zero

$$H(t_f) = 0. \quad (3.30)$$

Note that Eq. (3.30) only holds as long as neither any terminal constraint nor the Mayer cost term explicitly depend on time (see Ref. [23], Chapter 2.8 for a detailed discussion). When no terminal constraint depends on mass, the final value for λ_m must be zero.

It is interesting to note that Eq. (3.30) is a very nonlinear equation. Computational experience has shown that the resulting two-point boundary value problem is hard to solve. However, Eq. (3.30) is basically only scaling the magnitude of the adjoint vector. Noting that all transversality conditions (see Sections 3.4.1ff and Appendix C) depend linear on the adjoint variables, the terminal constraint

$$\|\mathbf{P}\| - 1 = 0 \quad (3.31)$$

can be introduced instead. A similar approach is suggested in Ref. [29], where the first element of the primer vector is set to one. However, Eq. (3.31) will also work in a very general setup where the optimal solution may require the first element of the primer vector to be zero. Using Eq. (3.31) as a constraint yields a two-point boundary value problem which is much easier to solve numerically. It only implies a positive scaling factor k for the Lagrange cost such that the Hamiltonian becomes

$$H = kL + \mathbf{P}^T \left(\mathbf{G} + \frac{T}{m} \mathbf{i}_b \right) + \mathbf{Q}^T \mathbf{V} - \lambda_m q. \quad (3.32)$$

For $H = 0$, the scaling factor k can easily be calculated as

$$k = \frac{-1}{L} \left[\mathbf{P}^T \left(\mathbf{G} + \frac{T}{m} \mathbf{i}_b \right) + \mathbf{Q}^T \mathbf{V} - \lambda_m q \right] \quad (3.33)$$

in case it is required at all. However, using Eq. (3.31) instead of Eq. (3.30) will not yield the correct values for the switching function discussed below because the adjoint variables are scaled differently (in order to compute corrected values for the switching function, the scaling factor k needs to be included in the derivation of the switching function). However, the SQP method will adjust the final time such that the performance index is minimized.

3.4.1 Boundary Conditions for Circular Orbits with a Fixed Inclination

A constraint for the orbital inclination can be formulated as

$$\mathbf{i}_f^T \mathbf{i}_N - \cos i = 0 \quad (3.34)$$

where

$$\mathbf{i}_f = \frac{\mathbf{R} \times \mathbf{V}}{|\mathbf{R} \times \mathbf{V}|} \quad (3.35)$$

is a vector perpendicular to the orbit plane. The geometry is defined in Fig. 3.3 and Fig. 3.4. The right ascension of the ascending node is regarded as an optimizable parameter.

In order to specify a circular orbit with the radius r_p and an orbital, circular velocity of v_p , three more constraints are introduced as there are

$$\text{the magnitude of the final radius vector } \|R\| - r_p = 0, \quad (3.36)$$

$$\text{the magnitude of the final velocity vector } \|V\| - v_p = 0, \quad (3.37)$$

and the fact that final radius and velocity vector are perpendicular to each other:

$$R^T V = 0 \quad (3.38)$$

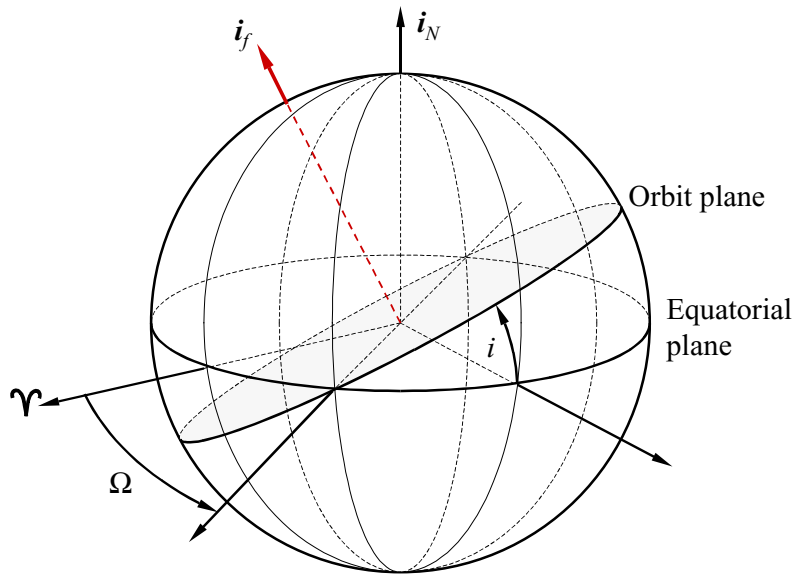


Fig. 3.3: Definition of RAAN and inclination

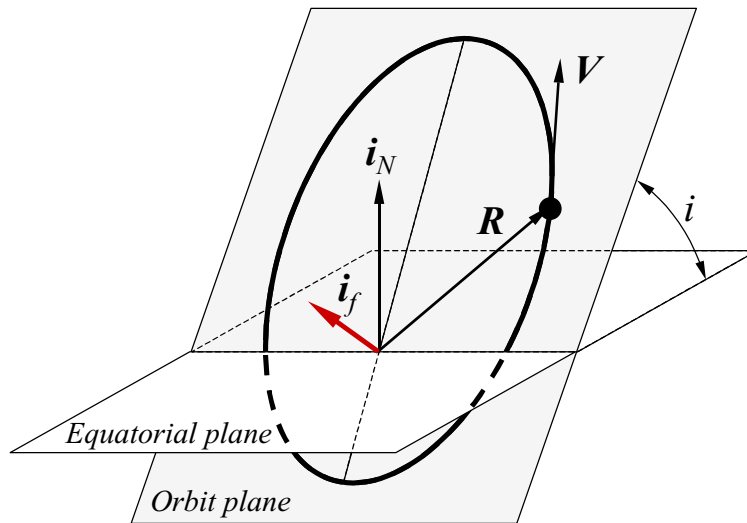


Fig. 3.4: Definition of important vectors

The latter constraint can be introduced, because a circular orbit does not have a well-defined argument of perigee. Therefore, an additional constraint can be introduced to completely define the orbit. Eq. (3.38) can also be seen as a constraint for the final flight-path angle ($\gamma = 0$). Note that Eqs. (3.36) - (3.38) can also be used for missions where the target orbit is elliptical and the attachment point is either apogee or perigee. If Eq. (3.38) is not set to zero but to a value reflecting the cosine between the final radius and velocity vector, any attachment point location on an elliptical orbit can be modelled as well. Another option to model any attachment point location is to insert a coast arc to the perigee (or apogee) point with an optimizable duration.

The transversality conditions to the set of (physical) terminal constraints formed by Eqs. (3.34) - (3.38) can be derived as

$$\left(\frac{a_r}{a_v} \mathbf{P} + \mathbf{Q} \right)^T \mathbf{i}_f = 0 \text{ and} \quad (3.39)$$

$$\mathbf{Q}^T \mathbf{i}_V - \frac{v}{r} \mathbf{P}^T \mathbf{i}_R = 0 \quad (3.40)$$

where

$$\frac{a_r}{a_v} = \frac{\mathbf{i}_f^T (\mathbf{i}_N \times \mathbf{V})}{\mathbf{i}_f^T (\mathbf{i}_N \times \mathbf{R})} \quad (3.41)$$

Further details on the derivation of Eqs. (3.39)-(3.41) are given in Appendix C.2 and Refs. [33], [34], and [37].

3.4.2 Boundary Conditions for Equatorial Orbits

It turns out that Eq. (3.34) cannot be used in connection with equatorial orbits, because the associated transversality condition (Eq. (3.39)) shows a singularity in the denominator of a_r/a_v when the inclination becomes zero and \mathbf{R} and \mathbf{V} are perpendicular to \mathbf{i}_N . In this case, $\mathbf{i}_N \times \mathbf{R}$ is perpendicular to \mathbf{i}_f . However, Eq. (3.34) and the transversality condition (3.39) can be replaced with the two physical constraints

$$\mathbf{R}^T \mathbf{i}_N = 0 \text{ and} \quad (3.42)$$

$$\mathbf{V}^T \mathbf{i}_N = 0. \quad (3.43)$$

The reason for the problem with $i = 0$ can be explained by the fact that an equatorial orbit does not have a well-defined right ascension of the ascending node.

3.4.3 Boundary Conditions for Elliptical Orbits with a Fixed Argument of Perigee

In case of an attachment to an elliptic orbit at either apogee or perigee, r_p and v_p have to be replaced with the corresponding values at perigee or apogee. In this kind of setup, an additional constraint required for geostationary transfer-orbits can easily be incorporated. The argument of perigee must be constrained to be $\omega = 0^\circ$ or 180° . For these special values of ω , the final radius vector must be perpendicular to the unit vector pointing north (Eq. (3.42)). When using Eqs. (3.34), (3.36), (3.37), (3.38), and (3.42) as the (physical) terminal constraints, the following transversality condition can be derived (see also Ref. [34]):

$$\frac{v}{r} \mathbf{P}^T \mathbf{i}_R - \mathbf{Q}^T \mathbf{i}_V + \frac{(\mathbf{i}_N^T \mathbf{i}_V)(\mathbf{Q}^T \mathbf{i}_f)}{\mathbf{i}_N^T \mathbf{i}_f} = 0 \quad \text{or} \quad \frac{v}{r} \mathbf{P}^T \mathbf{i}_R - \mathbf{Q}^T \mathbf{i}_V + \tan i (\mathbf{Q}^T \mathbf{i}_f) = 0. \quad (3.44)$$

Details are given in Appendix C.3. A more general constraint for the argument of perigee can be found in Ref. [29]:

$$\frac{\mathbf{i}_R^T (\mathbf{i}_N \times \mathbf{i}_f)}{|\mathbf{i}_N \times \mathbf{i}_f|} - \cos \omega = 0 \quad \text{for } 0^\circ < \omega < 180^\circ \quad \text{and} \quad (3.45)$$

$$\frac{\mathbf{i}_V^T (\mathbf{i}_N \times \mathbf{i}_f)}{|\mathbf{i}_N \times \mathbf{i}_f|} - \sin \omega = 0 \quad \text{for } \omega \text{ bordering } 0^\circ \text{ and } 180^\circ. \quad (3.46)$$

These constraints can be simplified further, if the fact that $|\mathbf{i}_N \times \mathbf{i}_f| = \sin i$ is taken into account, since this is ensured when Eq. (3.34) is simultaneously satisfied. When applying this constraint in addition to the other four terminal constraints (Eqs. (3.34), (3.36), (3.37), (3.38)), only one additional transversality condition is required:

$$0 = \frac{v}{r} \mathbf{P}^T \mathbf{i}_R - \mathbf{Q}^T \mathbf{i}_V + \tan i \left(\frac{v}{r} \mathbf{P} \sin \omega + \mathbf{Q} \cos \omega \right)^T \mathbf{i}_f. \quad (3.47)$$

The derivation of this equation is shown in Appendix C.4.

3.4.4 Boundary Conditions for Elliptical Orbits with a Fixed Inclination

For elliptical orbits, a more general set of equations can be derived. As far as the inclination constraint is concerned, Eqs. (3.34)/(3.39) or Eqs. (3.42)/(3.43) can be used as well. However, a more general set of constraints must be incorporated for constraining perigee radius and orbital energy of the final orbit. The constraint on perigee radius can be formulated as

$$a(1 - e) - r_p = 0 \quad (3.48)$$

where expressions for a and e as functions of \mathbf{R} and \mathbf{V} are given in Appendix B.3 and B.5, respectively, and r_p is the desired perigee radius of the target orbit. The constraint on orbital energy can be computed as

$$\frac{v^2}{2} - \frac{\mu}{r} - \varepsilon = 0 \quad (3.49)$$

where

$$\varepsilon = \frac{-\mu}{2a_d} \quad (3.50)$$

is the desired orbital energy (see Ref. [30], p. 27) and a_d is the desired semimajor axis. Note that it is numerically not beneficial to substitute Eq. (3.50) into Eq. (3.48) because this would yield a constraint for eccentricity. In Ref. [33] it is found that such an approach does not exhibit a good convergence behavior for orbits with small eccentricities. The same problems are encountered when using the square of the eccentricity or the magnitude of the orbital momentum vector while the terminal constraint on perigee radius achieves the best convergence behavior for elliptical orbits with a free attachment point.

Deriving the required transversality conditions and eliminating the corresponding Lagrange multipliers is more complicated in this case. After applying some painful algebra, the following set of terminal constraints can be derived (for details, see Appendix C.5):

$$\left(\frac{a_r}{a_v} \mathbf{P} + \mathbf{Q} \right)^T \mathbf{i}_f = 0 \quad (3.51)$$

$$\mathbf{P}^T \mathbf{i}_R - \tilde{v}_2 \tilde{c}_{2V} (\mathbf{i}_V^T \mathbf{i}_R) - \tilde{v}_3 (\mathbf{i}_V^T \mathbf{i}_R) = 0 \quad (3.52)$$

$$\mathbf{Q}^T \mathbf{i}_R - \tilde{v}_2 \tilde{c}_{2R} (\mathbf{i}_V^T \mathbf{i}_R) - \tilde{v}_3 \frac{\mu}{vr^2} (\mathbf{i}_V^T \mathbf{i}_R) = 0 \quad (3.53)$$

where the values \tilde{v}_2 , \tilde{v}_3 , \tilde{c}_{2V} and \tilde{c}_{2R} are defined in Appendix C.5.

In Ref. [29] another alternative of formulating the terminal constraints for elliptical orbits with a free attachment point is mentioned. In that approach, an additional coast arc is appended after the last burn arc. This coast arc involves an optimizable final time and the terminal conditions are formulated at the end of this coast arc such that the coast arc is forced to end either at apogee or perigee of the target orbit. Even though this approach exhibits an excellent convergence behavior, it sometimes produces results where the duration of the final coast arc is negative. This means that the burn out occurs “after” the perigee point. Since negative phase times are not permitted in the framework of ASTOS, this approach is not investigated any further. Still, it would easily be possible to append a coast arc after the last burn phase in order to let the vehicle coast to apogee. However, especially when the target orbit is a GTO,

this easily introduces a coast arc of approximately six hours duration to the problem, which is likely to decrease the convergence behavior significantly.

3.5 Optimal Coast Arcs and Switching Function

In order to consider optimal coast arcs, an additional control η is introduced to the system. It can be interpreted as a throttle setting for the engines and can take values between zero and one. The Hamiltonian changes to

$$H = \eta L + \mathbf{P}^T \left(\mathbf{G} + \eta \frac{T}{m} \mathbf{i}_b \right) + \mathbf{Q}^T \mathbf{V} - \lambda_m \eta q. \quad (3.54)$$

The switching function is defined as the partial derivative of the Hamiltonian with respect to η :

$$S = L + \frac{T}{m} \mathbf{P}^T \mathbf{i}_b - \lambda_m q \quad (3.55)$$

In order to minimize the value of the Hamiltonian with respect to η , S must be less than zero during thrust arcs. On the other hand, η must be zero when S becomes greater than zero, which basically means that the engine must be turned off. Both requirements for S can be used to detect the desirability of further burn and coast arcs. The special situation of singular arcs with $S = 0$, where a throttle setting between zero and one is the optimal solution, is not considered in the framework of this thesis.

If the system is autonomous, i.e. neither q nor T depend explicitly on time, the Hamiltonian is a constant. In addition, H must be zero at final time for a free final time problem. In that case, the switching function during burn arcs can be rewritten as

$$S_{burn} = -\mathbf{P}^T \mathbf{G} - \mathbf{Q}^T \mathbf{V}. \quad (3.56)$$

Note that λ_m cancels out and need not be determined during the solution of the optimal control problem. In Ref. [17], Eq. (3.56) is referred to as the “transversality variable” which has to be zero at each switching point. It is important to note that Eq. (3.56) is zero during a coast arc since H and η are zero.

It is shown in the classical literature (e.g. Ref. [17]) that S must be zero at the beginning and at the end of each coast arc. This yields two additional constraints for the two additional variables (initial and final time of each coast arc). Note that Eq. (3.56) is not sufficient to determine the magnitude of the adjoint vectors \mathbf{P} and \mathbf{Q} , since both appear linear in this equation. Therefore, it is necessary to use Eq. (3.54) or Eq. (3.55) instead of Eq. (3.56) at least at one switching point of the trajectory, preferably at the end of a burn arc. It is interesting to note that Eq. (3.55) can be rewritten as

$$S = q - \frac{T}{m} \|\mathbf{P}\| - \lambda_m q \quad (3.57)$$

and since \dot{m} and $\dot{\lambda}_m$ are zero during a coast arc, the magnitude of the primer vector \mathbf{P} at the beginning and at the end of each coast arc must be equal.

Another important remark concerning the number of burns in an optimal trajectory is made in Ref. [17] in the framework of a realistic problem statement: once a vehicle has reached orbital velocity, it is possible to insert a coast arc at arbitrary points during a burn arc. The vehicle can coast around the Earth once and ignite the engine again after one revolution with no change in the cost functional at all (assuming a perfect two-body motion). Therefore, either a limit on the number of burns must be imposed, a constraint on final time, or a non-zero Lagrange cost must be introduced during coast arcs. In the framework of this thesis it is assumed that the user limits the number of burns.

3.6 Generating an Initial Guess for the Initial Costates

As it is mentioned above, the two-point boundary value problem resulting for a thrust maneuver in vacuum can be solved for almost arbitrary guesses for the initial costates. One of the few restrictions is that the initial primer vector \mathbf{P} must not be set to a zero vector because the optimal control law relies on the existence of a unit vector along \mathbf{P} . Besides the option that the user supplies his own initial costate guesses, two different costate estimates can be generated automatically.

3.6.1 Ad-Hoc Initial Guess for \mathbf{P} and \mathbf{Q}

As an ad-hoc initial guess for the initial costates, \mathbf{P}_0 is estimated to point along the initial, inertial velocity vector such that

$$\mathbf{P}_0 = \frac{-\mathbf{V}}{v}. \quad (3.58)$$

Such an approach is also reported to be successful in Ref. [47]. The initial \mathbf{Q} vector is initialized to be perpendicular to \mathbf{P}_0 within the initial orbit plane. Its magnitude is set to 0.01:

$$\mathbf{Q}_0 = 0.01 \cdot \frac{\mathbf{P}_0 \times (\mathbf{P}_0 \times \mathbf{R})}{\|\mathbf{P}_0 \times (\mathbf{P}_0 \times \mathbf{R})\|} \quad (3.59)$$

3.6.2 Estimate based on Thrust-Acceleration and Target Radius

In Ref. [78] an approach is presented to calculate the initial costates for an orbital transfer maneuver between two circular orbits. The approach is using several simplifying assumptions and yields fairly simple equations for the initial costate estimates. In Appendix D, this approach is generalized for a non-circular initial orbit. Based on an estimated (constant) thrust-acceleration \bar{a} and a target radius r_f , the estimates for the initial costates are calculated as

$$p_{0,x} = \sqrt{\frac{r-1}{\bar{a}}} \quad (3.60)$$

$$p_{0,y} = v_0 \frac{\bar{r} - 1}{\bar{a}} \quad (3.61)$$

$$q_{0,x} = 1 \quad (3.62)$$

$$q_{0,y} = v_0 \sqrt{\frac{\bar{r} - 1}{\bar{a}}} \quad (3.63)$$

where $\bar{r} = r_f/r_0$ is the ratio between final and initial radius and \bar{a} is a constant acceleration given in multiplies of g_0 . Note that Eqs. (3.60)-(3.63) are given in a coordinate frame where the x -axis is aligned with \mathbf{i}_R and the y -axis is aligned with $\mathbf{i}_f \times \mathbf{i}_R$. In order to calculate the corresponding vectors \mathbf{P} and \mathbf{Q} in the inertial coordinate frame, a coordinate transformations must be performed:

$$\mathbf{P}_0 = p_{0,x} \mathbf{i}_R + p_{0,y} (\mathbf{i}_f \times \mathbf{i}_R) \quad (3.64)$$

$$\mathbf{Q}_0 = q_{0,x} \mathbf{i}_R + q_{0,y} (\mathbf{i}_f \times \mathbf{i}_R) \quad (3.65)$$

Note that an approximation for the flight time can be calculated as

$$t_b \approx 2 \sqrt{\frac{\bar{r} - 1}{\bar{a}}}. \quad (3.66)$$

It turns out that Eq. (3.66) is a good indicator for showing if the calculated costates are reasonable enough.

3.7 Solution of the Two-Point Boundary Value Problem

Solving a two-point boundary value problem requires a multi-dimensional zero finding algorithm. One of the most sophisticated, general methods is the code BNDSCO (Ref. [59]), which is based on a multiple-shooting method. However, the special problem encountered in this setup can easily be solved using a slightly modified Newton method as described in Refs. [33], [34], and [37]. The method described there also relies on the almost analytical propagation of the trajectory using a closed form solution for the propagation of states and costates during coast arcs, and a closed form solution with a small numerical correction during burn arcs. In addition, the required Jacobian matrix becomes available in an analytical form as a by-product of the trajectory propagation and some slightly added labor in the calculation of the terminal constraints. Details on this approach are given in the references mentioned above.

The same quality of solution can be achieved when the trajectory is integrated with an ordinary Runge-Kutta 4/5 method, as presented in Section A.2, and applying the same principle of freezing the corresponding step-size sequence for calculating a numerical Jacobian matrix (see Section 2.1.2). This greatly simplifies the implementation and the flexibility of the

whole procedure, and it ensures an accurate trajectory calculation while it hardly decreases the computational performance of the algorithm.

It also means that the problem can be implemented as a simple GESOP model, and CAMTOS, or any other optimization method available in this environment, can be used as a solver for the two-point boundary value problem. The only elements that have to be supplied by the user are the differential equations (states and costates), terminal constraints, and transversality conditions. The performance index for the SQP method is simply set to zero, since it is implicitly included in the adjoint differential equations (Lagrange term) and the transversality conditions (Mayer term).

A nice side-effect of using GESOP is that the user can easily switch between using a shooting method or a collocation method for solving the boundary value problem. Although both methods are originally coded as direct optimization methods, they can also be used for the solution of an indirect problem.

Numerical experiments have been performed in such a setup to verify the performance of CAMTOS in the framework of solving boundary value problems. It turns out that the performance of CAMTOS when solving a boundary value problem is not as good as an algorithm especially designed for such problems. However, several simple vacuum ascent cases can be solved.

3.8 Solution in the Framework of an SQP Method

The beauty of having an SQP method available for the solution of a trajectory optimization problem is that the user can avoid the formulation of transversality conditions if the performance index of the SQP method is used instead. A detailed mathematical analysis for this is given in Ref. [24] with a short summary in Ref. [25].

In such a setup, both, the state and costate differential equations are coded. In addition, the initial values of the costates λ_0 are introduced as optimization parameters for the NLP method. The value of the cost functional is used as a cost function for the NLP solver. The constraint vector ψ only consists of the physical terminal constraints.

The basic approach to prove the equivalence of this approach and the method discussed in Section 3.7 is to carefully construct the Karush-Kuhn-Tucker conditions for the resulting NLP problem. First of all, note that the Mayer term ϕ and the terminal constraints ψ only depend on the final states. The final states x_f are introduced as additional variables to the NLP problem. Noting that x_f is computed by a nonlinear function $h(\lambda_0)$, an additional set of terminal constraints $h(\lambda_0) - x_f = 0$ can be introduced, and the Lagrangian for the parameter optimization problem can be formulated as

$$L = \phi(x_f) + v^T \psi(x_f) + \lambda_f^T [h(\lambda_0) - x_f]. \quad (3.67)$$

$h(\lambda_0)$ represents the numerical integration of the state and costate differential equations including the optimality condition. In this setup, x_f and λ_0 are the parameters to be optimized by the SQP method. The corresponding Karush-Kuhn-Tucker conditions for this problem are

$$\frac{\partial L}{\partial \mathbf{x}_f} = \frac{\partial \phi}{\partial \mathbf{x}_f} + \mathbf{v}^T \frac{\partial \Psi}{\partial \mathbf{x}_f} - \boldsymbol{\lambda}_f^T = \mathbf{0} \text{ and} \quad (3.68)$$

$$\frac{\partial L}{\partial \boldsymbol{\lambda}_0} = \boldsymbol{\lambda}_f^T \frac{\partial \mathbf{h}}{\partial \boldsymbol{\lambda}_0} = \mathbf{0}. \quad (3.69)$$

Comparing Eq. (3.68) with the transversality conditions for the optimal control problem (see Ref. [23], p. 89)

$$\boldsymbol{\lambda}_f^T = \left(\frac{\partial \phi}{\partial \mathbf{x}} + \mathbf{v}^T \frac{\partial \Psi}{\partial \mathbf{x}} \right)_{t=t_f} \quad (3.70)$$

shows the equivalence of the obtained solution.

The discussion can be extended to a problem with interior point constraints $N(\mathbf{x}_1)$ at t_1 . In such a case, jumps may occur in the costate variables of the optimal control problem. Again, these jump conditions are already included in the Karush-Kuhn-Tucker conditions when the problem is solved with an SQP method. Assuming that the intermediate state \mathbf{x}_1 is calculated by a nonlinear function $\mathbf{h}_1(\boldsymbol{\lambda}_0)$ and the final state \mathbf{x}_f is calculated by another nonlinear function $\mathbf{h}_f[\mathbf{x}_1, \boldsymbol{\lambda}_+(t_1)]$. The Lagrangian can now be formulated as

$$L = \phi(\mathbf{x}_f) + \mathbf{v}^T \Psi(\mathbf{x}_f) + \boldsymbol{\lambda}_f^T \{ \mathbf{h}_f[\mathbf{x}_1, \boldsymbol{\lambda}_+(t_1)] - \mathbf{x}_f \} + \boldsymbol{\lambda}_-^T [\mathbf{h}_1(\boldsymbol{\lambda}_0) - \mathbf{x}_1] + \boldsymbol{\pi}^T N(\mathbf{x}_1). \quad (3.71)$$

The Karush-Kuhn-Tucker conditions are readily derived as

$$\frac{\partial L}{\partial \mathbf{x}_f} = \frac{\partial \phi}{\partial \mathbf{x}_f} + \mathbf{v}^T \frac{\partial \Psi}{\partial \mathbf{x}_f} - \boldsymbol{\lambda}_f^T = \mathbf{0}, \quad (3.72)$$

$$\frac{\partial L}{\partial \mathbf{x}_1} = \boldsymbol{\lambda}_f^T \frac{\partial \mathbf{h}_f}{\partial \mathbf{x}_1} - \boldsymbol{\lambda}_-^T + \boldsymbol{\pi}^T \frac{\partial N}{\partial \mathbf{x}_1} = \mathbf{0}, \quad (3.73)$$

$$\frac{\partial L}{\partial \boldsymbol{\lambda}_+} = \boldsymbol{\lambda}_f^T \frac{\partial \mathbf{h}_f}{\partial \boldsymbol{\lambda}_+} = \mathbf{0} \text{ and} \quad (3.74)$$

$$\frac{\partial L}{\partial \boldsymbol{\lambda}_0} = \boldsymbol{\lambda}_-^T \frac{\partial \mathbf{h}_1}{\partial \boldsymbol{\lambda}_0} = \mathbf{0}. \quad (3.75)$$

Eq. (3.72) represents the final transversality conditions. If this expression is used for $\boldsymbol{\lambda}_f^T$ and \mathbf{h}_f is replaced by \mathbf{x}_f , Eq. (3.73) can be rewritten as

$$\left(\frac{\partial \phi}{\partial \mathbf{x}_f} + \mathbf{v}^T \frac{\partial \Psi}{\partial \mathbf{x}_f} \right) \frac{\partial \mathbf{x}_f}{\partial \mathbf{x}_1} - \boldsymbol{\lambda}_-^T + \boldsymbol{\pi}^T \frac{\partial N}{\partial \mathbf{x}_1} = \frac{\partial \phi}{\partial \mathbf{x}_1} + \mathbf{v}^T \frac{\partial \Psi}{\partial \mathbf{x}_1} - \boldsymbol{\lambda}_-^T + \boldsymbol{\pi}^T \frac{\partial N}{\partial \mathbf{x}_1} = \mathbf{0}. \quad (3.76)$$

Noting that $\lambda_+(t_1)$ is the gradient of $\phi(\mathbf{x}_f) + \mathbf{v}^T \boldsymbol{\psi}(\mathbf{x}_f)$ with respect to changes in \mathbf{x}_1 (see Ref. [23], p. 49), Eq. (3.76) can be written as

$$\lambda_+(t_1) - \lambda_-^T + \boldsymbol{\pi}^T \frac{\partial N}{\partial \mathbf{x}_1} = 0 \quad (3.77)$$

which is equivalent to the jump condition in the costates of the optimal control problem (see Ref. [23], p. 101). This proof can easily be extended to problems with more interior point constraints at different times.

If the approach described in this Section is used to formulate a problem including the adjoint differential equations, it is much easier for CAMTOS to obtain a solution compared to a setup as a pure boundary value problem. In addition, it becomes possible to construct problems involving both, direct and indirect phases. An example for this approach is discussed in Section 4.

3.9 Verification of the Indirect Method

In order to verify the correct derivation of the adjoint differential equations, the optimality condition, and the transversality conditions, all derivatives are checked numerically using MATLAB. In addition, a simple GESOP model is coded which models a one-stage rocket ascent from 30 deg northern latitude to a 200 km, circular orbit over an oblate Earth according to Appendix E.1 and E.2. The launch mass of the vehicle is 270,000 kg, vacuum thrust is 3,000 kN with a massflow of 600 kg/s. In this setup, a small amount of backpressure, calculated with a nozzle exit area of 0.9 m², is also included in order to verify the equations derived for the backpressure and the altitude calculation over an oblate planet.

The problem is formulated as a two-point boundary value problem including the transversality conditions for circular orbits. The resulting altitude profile is shown in Fig. 3.5. The trajectory ends at an altitude higher than 200 km because the terminal constraints are formulated such that the magnitude of the final radius vector is the sum of the equatorial radius plus 200 km, while Fig. 3.5 shows the altitude computed according to Eq. (3.12).

The correct implementation of the optimality condition and the adjoint differential equations can be verified by checking the Hamiltonian. Since the example dynamics is an autonomous system, the value of the Hamiltonian must be constant along the trajectory. Fig. 3.6 shows some very small oscillations which are well within the numerical accuracy of the integration method. In addition, some small jumps can be observed which occur exactly at the multiple shooting nodes. Therefore, it can be concluded that the implemented equations are correct. The solution is also verified by comparing it with an equivalent setup where a direct optimization method is used for the solution. No visible differences can be observed.

It is interesting to note that CAMTOS exhibits a quadratic convergence rate as soon as the parameter vector comes close to the solution (see Fig. 3.7). Since this example includes a small amount of backpressure (see thrust profile in Fig. 3.8), the problem becomes very sensitive to the initial guess for the costate vector. When the same problem is optimized without the backpressure, CAMTOS converges in eight iterations (Fig. 3.9). Therefore, the final imple-

mentation of the indirect method in ASTOS does not include the backpressure term and is only made available for thrust arcs in vacuum.

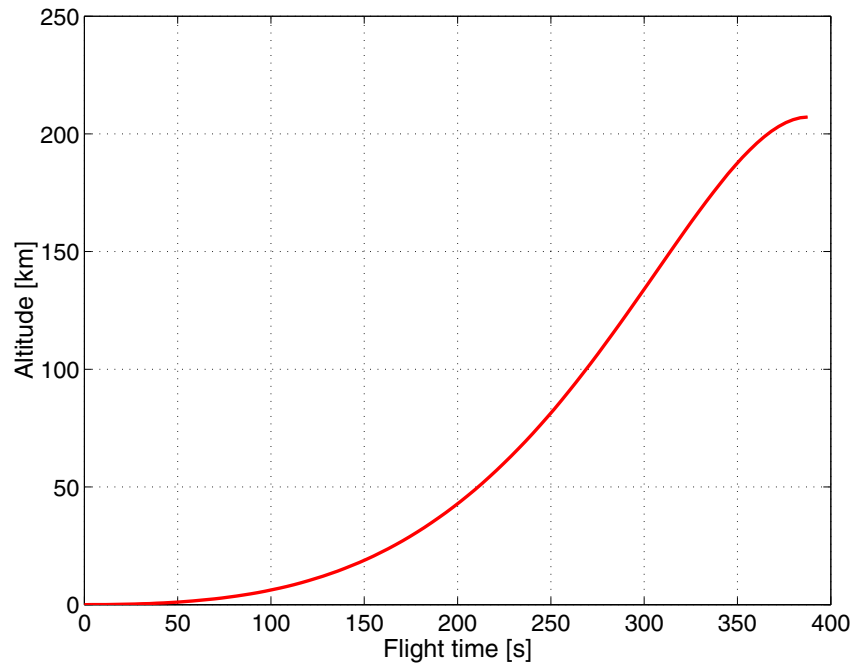


Fig. 3.5: Altitude profile of indirect rocket ascent example

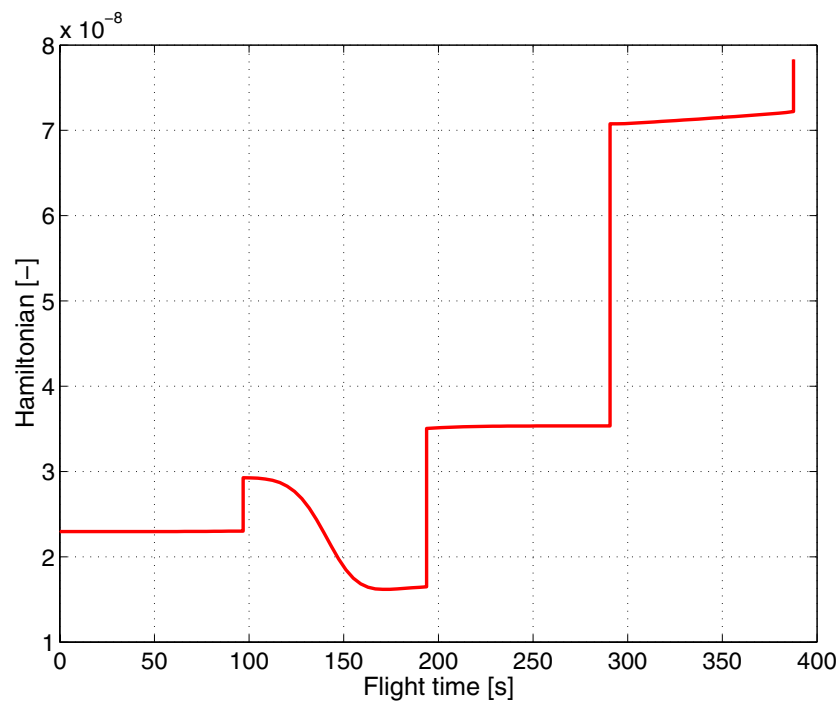


Fig. 3.6: Hamiltonian for rocket ascent example

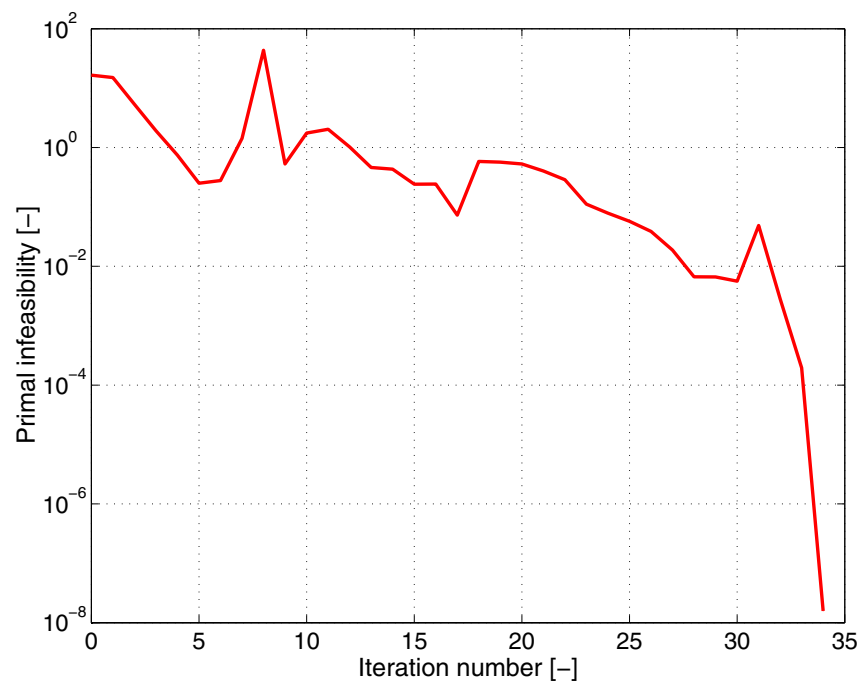


Fig. 3.7: Convergence behavior of CAMTOS for solving a two-point boundary value problem that involves dissipative terms

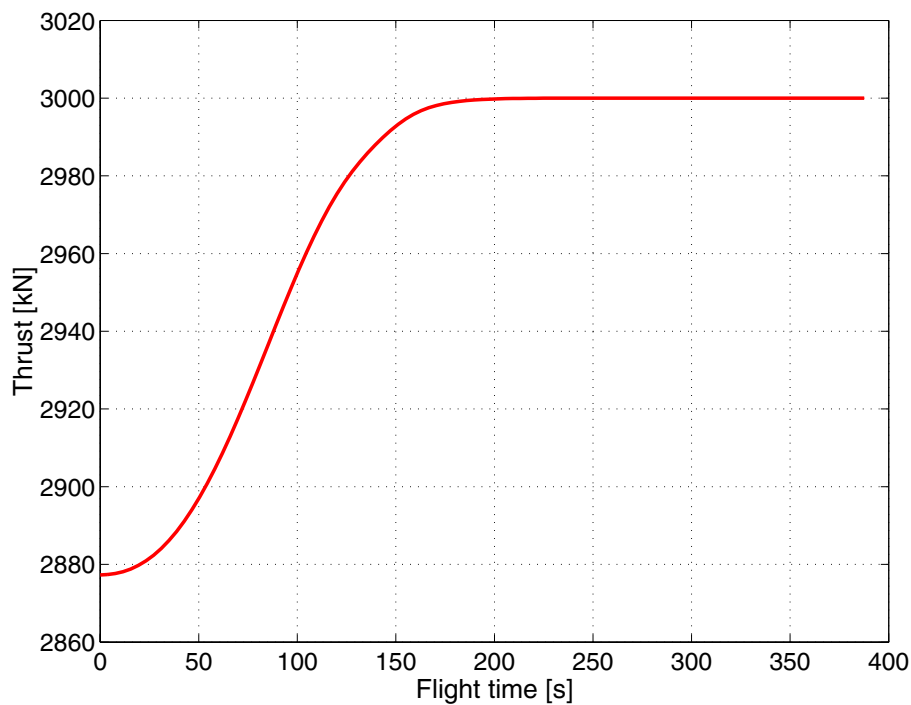


Fig. 3.8: Thrust profile of example rocket ascent including backpressure

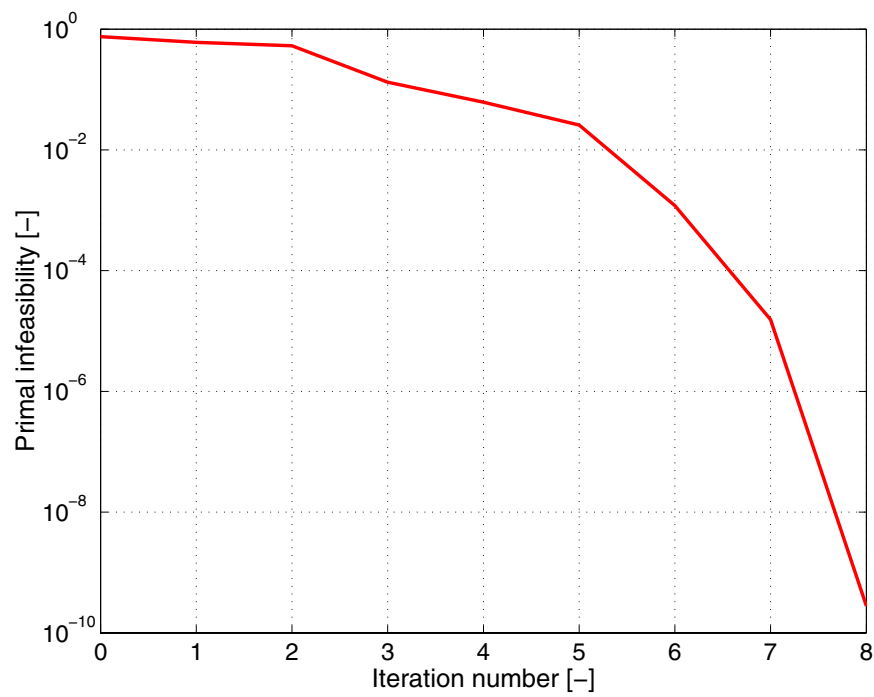


Fig. 3.9: Convergence behavior of CAMTOS for a rocket ascent without backpressure

4 Example 1: Ariane 5 Dual Payload Mission Analysis

It is true that, on account of the complex nature of the majority of practical problems in this field, recourse to numerical techniques is usually inevitable before an acceptable solution is forthcoming, but the analytical approach is none the less valuable for the following reasons: Experience teaches us that the form of an optimal trajectory is rarely, if ever, very critically dependent upon the data of a problem and consequently if, by making suitable simplifying approximations, the actual problem can be transformed into an idealized problem whose solution is analytically tractable, then this latter solution will often provide an excellent substitute for the optimal motor thrust programme in the actual situation. All that then remains to be done is to recompute the trajectory employing this programme and taking account of the real circumstances.

D.F. Lawden, Optimal Trajectories for Space Navigation, 1963

The first example problem shows the advantages of using an indirect optimization method for the identification of possible improvements in the switching structure. In this setup, an Ariane 5 dual payload mission involving a re-ignitable upper stage is analyzed. While the lower stage is modelled using the direct multiple shooting method and includes all atmospheric effects, the indirect method discussed in Section 3 is used for the upper stage burns. The initial guess is generated by using a sequence of guidance laws for the lower stage, and the ad-hoc initial guess discussed in Section 3.6.1 is used for the costates in indirect phases. By analyzing the switching function during the upper stage burns for the original burn-coast-burn sequence, an improved structure is identified which leads to a significant increase in payload performance.

4.1 Reference Mission Scenario

The objective of this mission is to deliver two payloads into their target orbits. The first payload has a fixed mass of 3,000 kg and is to be delivered into a 600 km circular orbit with 7 deg inclination. The mass of the second payload is optimizable and is delivered into a modified geostationary transfer orbit with an apogee altitude of 35,800 km, 7 deg inclination, and 270 deg argument of perigee. The launch site is Kourou. The environment models used for this mission are given in Appendix E, the data used for the Ariane 5 launcher is given in Ref. [6], Appendix A.5.

In addition to the payloads' target orbits, several path constraints must be included into the scenario. The payload fairing is jettisoned once the heatflux falls below $1,135 \text{ W/m}^2$. In order to demonstrate the applicability of the method in the presence of path constraints, a path constraint on dynamic pressure of 30 kPa is also included. Finally, a splash-down constraint for the main tank is required. The main tank is forced to splash-down in a safe distance from the African coastline into the Indian Ocean. This constraint is approximated by propagating the orbital elements at burnout of the main tank until this intermediate orbit intersects the Earth surface. No aerodynamic forces are included for the return path of the main tank.

	Time [sec]	Active Prop.	Path Constr.	Boundary Constr.	Remark
1	$0 - t_{f,1}$	2 x P230 1 x H155	-	$h_0 = 1 \text{ m}$ $\lambda_0 = -52.7686 \text{ deg}$ $\delta_0 = 5.2434 \text{ deg}$ $V_{North,0} = 0 \text{ m/s}$ $V_{Rel,East,0} = 0 \text{ m/s}$ $V_{Rel,0} = 5 \text{ m/s}$	vertical ascent
2	$t_{f,1} - 20$	2 x P230 1 x H155	-	-	pitch over pitch plane optimized
3	$20 - 129.1$	2 x P230 1 x H155	$q \leq 30 \text{ kPa}$	-	boosters are jettisoned
4	$129.1 - t_{f,4}$	1 x H155	-	$\dot{Q} \leq 1135 \text{ W/m}^2$	fairing jettisoned
5	$t_{f,4} - 567.764$	1 x H155	-	H155 splash-down	H155 jettisoned
6	$567.764 - t_{f,6}$	1 x L9	-	payload 1 target orbit	payload 1 jettisoned
7	$t_{f,6} - t_{f,7}$	-	-	-	coast arc
8	$t_{f,7} - t_{f,8}$	1 x L9	-	payload 2 target orbit	-

Table 4.1: Phase structure of the dual payload reference mission scenario

The first setup for the mission is split into 8 phases as shown in Table 4.1. In order to avoid a singularity in the initial yaw angle, the mission starts with a vertical ascent phase where pitch is fixed to 90 deg, followed by a pitch over phase with a linear control law for

pitch. The final pitch value at the end of phase two as well as the plane of the pitch-over maneuver are optimizable.

Since the target orbit of payload one is a circular orbit, a special set of terminal conditions is applied:

$$\begin{aligned} i &= 7 \text{ deg} \\ r &= 6,978.135 \text{ km} \\ v &= \sqrt{\mu/r} = 7.529 \text{ km/s} \\ \gamma &= 0 \text{ deg} \end{aligned}$$

This special set of terminal constraints avoids singularities in the constraint derivatives which would otherwise appear when eccentricity or apogee and perigee radius are used instead (see Appendix B.6 and Ref. [19], p. 503).

4.1.1 Generating the Initial Guess

The initial guess for this mission is constructed by applying a sequence of guidance laws as described in Ref. [6] and [57]. The phase structure for the initial guess is shown in Table 4.2.

	Time [sec]	Active Prop.	Guidance Law	Boundary Constr.	Remark
1	0 - 5	2 x P230 1 x H155	vertical ascent	$h_0 = 1 \text{ m}$ $\lambda_0 = -52.7686 \text{ deg}$ $\delta_0 = 5.2434 \text{ deg}$ $V_{North,0} = 0 \text{ m/s}$ $V_{Rel,East,0} = 0 \text{ m/s}$ $V_{Rel,0} = 5 \text{ m/s}$	vertical ascent
2	5 - 15	2 x P230 1 x H155	final pitch 84 deg	-	pitch over pitch plane optimized
3	15 - 20	2 x P230 1 x H155	constant pitch	-	pitch constant
3	15 - 129.1	2 x P230 1 x H155	gravity turn	-	boosters are jettisoned
4	129.1 - 142	1 x H155	gravity turn	$\dot{Q} \leq 1135 \text{ W/m}^2$	fairing jettisoned
5	142 - 567.764	1 x H155	required velocity	H155 splash-down	H155 jettisoned
6	567.764 - 857	1 x L9	required velocity	payload 1 target orbit	payload 1 jettisoned
7	857 - 3710	-	-	-	coast arc
8	3710 - 4400	1 x L9	required velocity	payload 2 target orbit	-

Table 4.2: Phase structure for the initial guess of the dual payload reference mission scenario

As an alternative, an initial guess could also be generated by using the indirect method in connection with a multi-dimensional zero finding algorithm to generate a vacuum ascent. A detailed discussion of this approach can be found in Ref. [39].

4.1.2 Optimization using CAMTOS

Using the control law sequence of the initial guess phase structure shown above, an optimal solution including all constraints and atmospheric effects can quickly be generated. All phases are discretized using multiple shooting with a Runge-Kutta 4/5 integration method due to the thrust profiles of the solid rocket boosters in the first phases and in order to compute a trajectory of high accuracy. It is necessary to model the upper stages with a shooting method in order to model the behavior of the costates accurately enough. The switching function as well as the Hamiltonian are highly sensitive to even small inaccuracies in the numerical integration. Therefore, many collocation nodes would be required to obtain exact results while the automatic step-size adaption in the shooting method always delivers results that are accurate within the numerical integration accuracy.

The resulting altitude profile is shown in Fig. 4.1 and Fig. 4.2. The slight drop in altitude is caused by the oblateness of the Earth. This already has been shown in Ref. [85]. The crosses in Fig. 4.1 show the phase boundaries while the dashed line is representing the coast arc of phase 7 and the thin, continuous line is the trajectory of the empty H155 stage until splash-down.

In order to obtain the switching function, the mission is optimized in a hybrid setup. In this setup, phases 6 and 8 are using an indirect problem formulation involving the adjoint differential equations (see Section 3.2). All the remaining phases are using the direct approach. Since both indirect phases are outside the atmosphere, the neglect of the atmospheric effects does not produce any errors. However, the switching function is now available for analysis. After phase 6, the constraints for the 600 km circular orbit are applied. Such constraints yield additional jump conditions for the adjoint variables. The formulation of these jump conditions can be omitted, because the initial values of the adjoints in phase 8 are treated as additional optimization parameters. The SQP method selects all adjoint variables such that the second payload mass is maximized and the corresponding jump conditions and transversality conditions are satisfied implicitly due to the Karush-Kuhn-Tucker conditions (see discussion in Section 3.8).

Fig. 4.3 depicts the value of the switching function during the two upper stage burns. It can clearly be seen that the first upper stage burn is not optimal since the switching function is greater than zero. However, the coast arc and the second upper stage burn exhibit an optimal behavior of the switching function: it is positive during the coast arc and negative during the burn arc. The large jump in the switching function at the beginning of the coast arc is caused by the interior point constraints for the circular orbit of the first payload. Even though the corresponding jump and transversality conditions are not explicitly enforced during the optimization process, they are satisfied in the framework of the Karush-Kuhn-Tucker conditions very accurately within a numerical accuracy of 10^{-5} (see Section 3.8).

Fig. 4.4 shows the control time histories during the ascent mission. It can be observed that the vehicle is thrusting with negative pitch during the first upper stage burn. This explains the behavior of the switching function discussed above. In order to drive the flight-path angle to zero for the intermediate circular orbit, it would be better to insert a coast arc after the burn out of the main stage. In the next section, an improved mission profile with such an additional coast arc is presented in order to obtain an optimal switching function profile. Fig. 4.4 also shows a small discontinuity in yaw and pitch during the first burn phase at the beginning of phase 6. This discontinuity occurs when the main tank is jettisoned and is caused by the associated boundary constraint for the splash-down.

The payload delivered to GTO is calculated as 1,938 kg for this reference mission profile. In order to verify the correct implementation of the costate differential equations and the optimality condition, the Hamiltonian is plotted during the two indirect phases. Fig. 4.5 shows a perfectly constant Hamiltonian which only exhibits some small jumps at the multiple shooting points. These could further be reduced by lowering the constraint tolerance or removing the multiple shooting points in subsequent optimization runs.

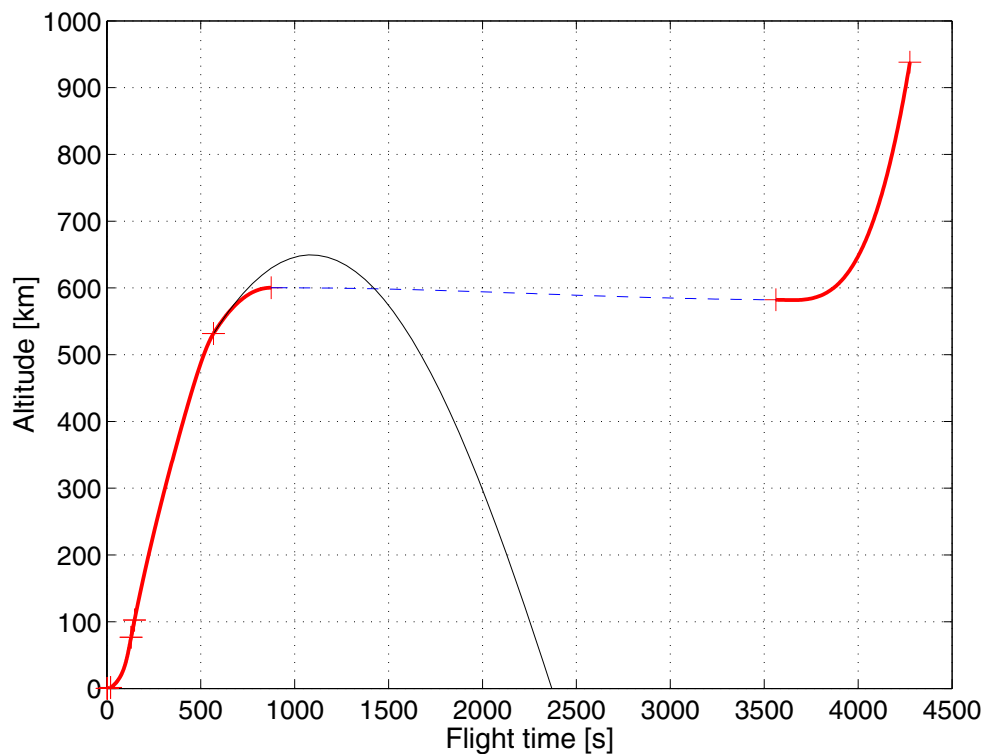


Fig. 4.1: Altitude profile for the dual payload reference mission



Fig. 4.2: 3D trajectory representation of the dual payload reference mission

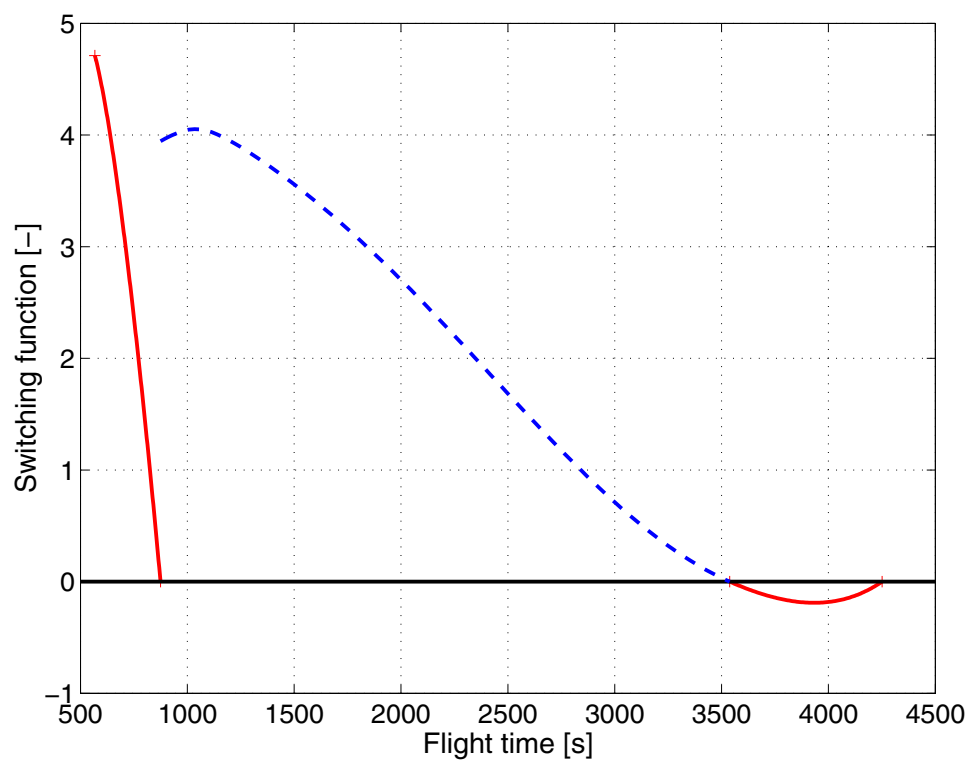


Fig. 4.3: Switching function during the upper stage burns of the dual payload reference mission

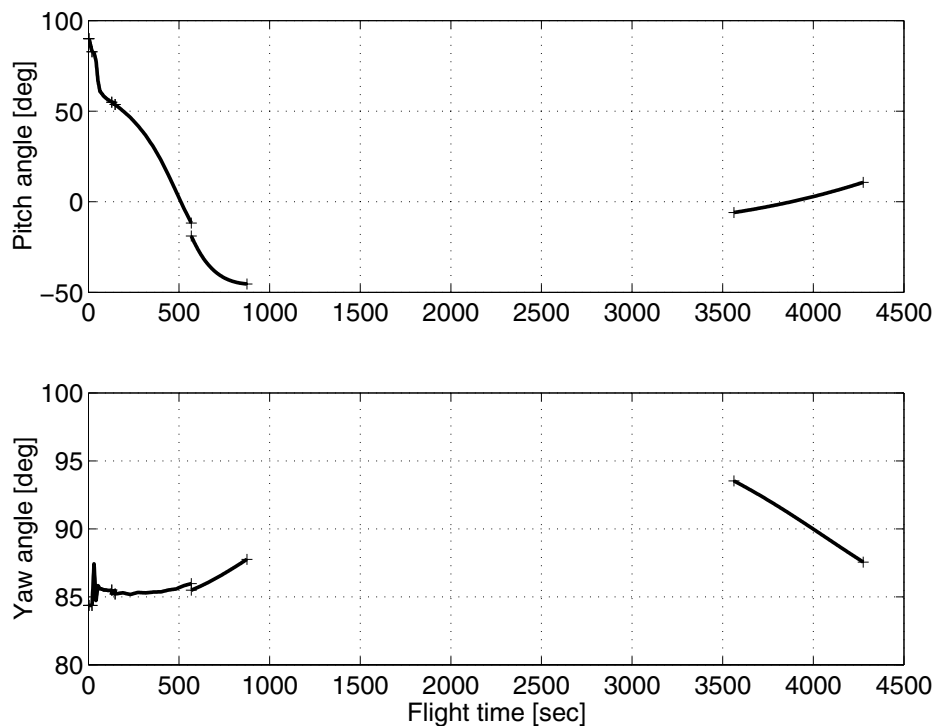


Fig. 4.4: Control time histories for the dual payload reference mission

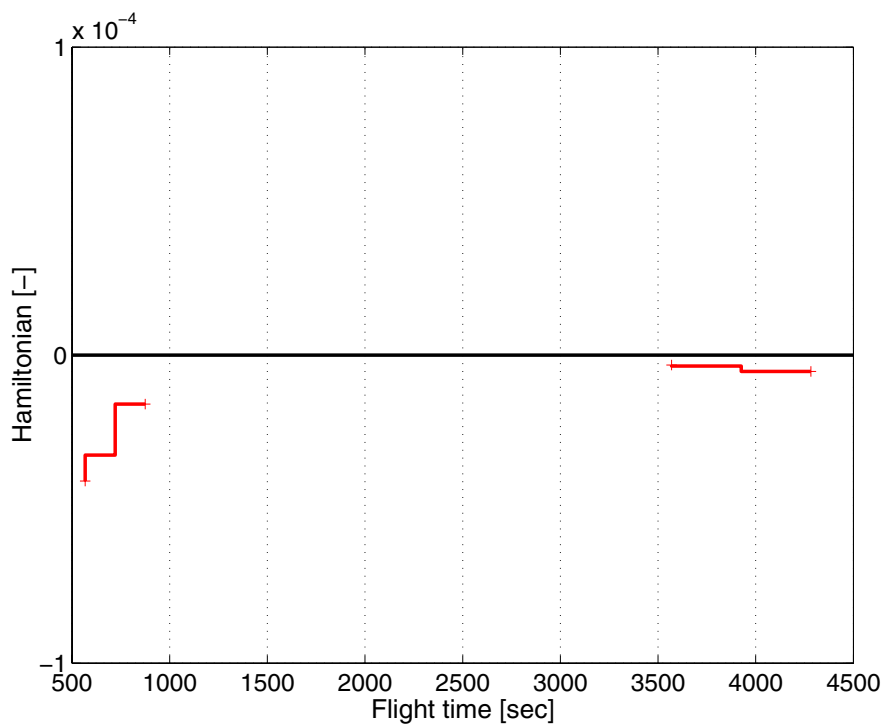


Fig. 4.5: Hamiltonian during upper stage burns

4.2 Improved Mission Profile with Two Coast Arcs

The reference mission profile clearly shows a potential for a further payload improvement. As it is mentioned above, the structure of the switching function during the first burn arc suggests that an additional coast arc should be inserted into the mission in order to increase the payload performance. This additional coast arc splits the first upper stage burn into two burns: The first burn is required to avoid a re-entry of the upper stage due to the splash-down constraint of the main tank and raises the perigee altitude to at least 200 km. The second burn is used to circularize the orbit for the first payload. Table 4.3 shows the phase structure of the modified mission.

Since the second upper stage burn is very short, the whole setup is using “normalized times”. I.e., the flight time is scaled such that each phase has a normalized duration of one. In addition, real parameters for initial and final time as well as for the phase duration are introduced in each phase. These parameters are associated with constraints such that they hold a consistent set of time information. A detailed discussion on this approach can be found in Ref. [36]. Experience has shown that the normalized time approach works best in connection with multiple shooting methods.

	Time [sec]	Active Prop.	Path Constr.	Boundary Constr.	Remark
1	0 - $t_{f,1}$	2 x P230 1 x H155	-	$h_0 = 1 \text{ m}$ $\lambda_0 = -52.7686 \text{ deg}$ $\delta_0 = 5.2434 \text{ deg}$ $V_{North,0} = 0 \text{ m/s}$ $V_{Rel,East,0} = 0 \text{ m/s}$ $V_{Rel,0} = 5 \text{ m/s}$	vertical ascent
2	$t_{f,1} - 20$	2 x P230 1 x H155	-	-	pitch over pitch plane optimized
3	20 - 129.1	2 x P230 1 x H155	$q \leq 30 \text{ kPa}$	-	boosters are jettisoned
4	129.1 - $t_{f,4}$	1 x H155	-	$\dot{Q} \leq 1135 \text{ W/m}^2$	fairing jettisoned
5	$t_{f,4} - 567.764$	1 x H155	-	H155 splash-down	H155 jettisoned
6	567.764 - $t_{f,6}$	1 x L9	-	$h_{peri} > 200 \text{ km}$	target for first orbit
7	$t_{f,6} - t_{f,7}$	-	-	-	coast arc
8	$t_{f,7} - t_{f,8}$	1 x L9	-	payload 1 target orbit	payload 1 jettisoned
9	$t_{f,8} - t_{f,9}$	-	-	-	coast arc
10	$t_{f,9} - t_{f,10}$	1 x L9	-	payload 2 target orbit	-

Table 4.3: Phase structure of the modified dual payload mission scenario

4.2.1 Generating the Initial Guess

Just as for the reference mission, the initial guess for the modified mission scenario is generated by using a sequence of guidance laws. It can be observed that generating such a sequence from scratch becomes more and more difficult once additional coast arcs are introduced. A good approach for such cases is to optimize submissions first and patch the results together for initializing the overall mission scenario. Such an approach is demonstrated in Section 5.

The guidance law sequence used to generate the initial guess for the modified dual payload mission is shown in Table 4.4. The upper stage is now using indirect phases in connection with the ad-hoc initial guess for the costates. By changing the burn times of these phases the trajectory can easily be shaped such that the upper stage does not reenter the atmosphere, especially during the first coast arc.

	Time [sec]	Active Prop.	Guidance Law	Boundary Constr.	Remark
1	0 - 5	2 x P230 1 x H155	vertical ascent	$h_0 = 1 \text{ m}$ $\lambda_0 = -52.7686 \text{ deg}$ $\delta_0 = 5.2434 \text{ deg}$ $V_{North,0} = 0 \text{ m/s}$ $V_{Rel,East,0} = 0 \text{ m/s}$ $V_{Rel,0} = 5 \text{ m/s}$	vertical ascent
2	5 - 15	2 x P230 1 x H155	final pitch 84 deg	-	pitch over pitch plane optimized
3	15 - 20	2 x P230 1 x H155	constant pitch	-	pitch constant
4	15 - 129.1	2 x P230 1 x H155	gravity turn	-	boosters are jettisoned
5	129.1 - 142	1 x H155	gravity turn	$\dot{Q} \leq 1135 \text{ W/m}^2$	fairing jettisoned
6	142 - 567.764	1 x H155	required velocity	H155 splash-down	H155 jettisoned
7	567.764 - 836	1 x L9	indirect		raise perigee
8	836 - 4500	-	-	-	coast arc
9	4500 - 4570	1 x L9	indirect	payload 1 target orbit	payload 1 jettisoned
10	4570 - 8950	-	-	-	coast arc
11	8950 - 9830	1 x L9	indirect	payload 2 target orbit	-

Table 4.4: Phase structure for the initial guess of the modified dual payload reference mission scenario

4.2.2 Optimization using CAMTOS

It turns out that the optimization process for this mission is far more complicated than for the reference mission. This is a well-known problem for missions involving many coast arcs. Such coast arcs create a problem which is highly sensitive to even small changes in the initial condi-

tions of a coast. However, the multiple shooting approach, compared to a single shooting approach, greatly improves the convergence behavior.

Fig. 4.6 and Fig. 4.7 show the altitude profile of the modified mission. It can be observed that the second upper stage burn is very short. The controls in Fig. 4.8 show that the pitch angle is now positive during the first burn of the upper stage. It can also be observed that there is a discontinuity in the controls at the beginning of the upper stage burn. This discontinuity is caused by the splash-down constraint for the H155 stage.

Again, the upper stage uses the indirect method during the optimization. Fig. 4.9 and Fig. 4.10 show the resulting switching function structure. It can clearly be seen that the value of the switching function is now always negative during the burn arcs and positive during coast arcs. Again, the jumps in the switching function are caused by the interior point constraints. During the second upper stage burn, the value of the switching function is close to zero which means that this burn is close to being a singular arc (see Fig. 4.10). In order to obtain the correct structure of the switching function it is necessary to incorporate additional constraints into the optimization problem to make sure that the scaling of the adjoint variables is correct. According to the analysis presented in Chapter 3.5, the value of the switching function should be zero at the beginning and at the end of each (optimal) burn arc. These additional constraints are introduced to the problem as well, in addition to a constraint that forces the value of the Hamiltonian to zero at the end of a burn arc. The switching function at the beginning of the first upper stage burn must not be forced to zero because this burn must be started right after the burn out of the main stage. Otherwise, the upper stage would re-enter the atmosphere. Such situations are also mentioned in Ref. [17].

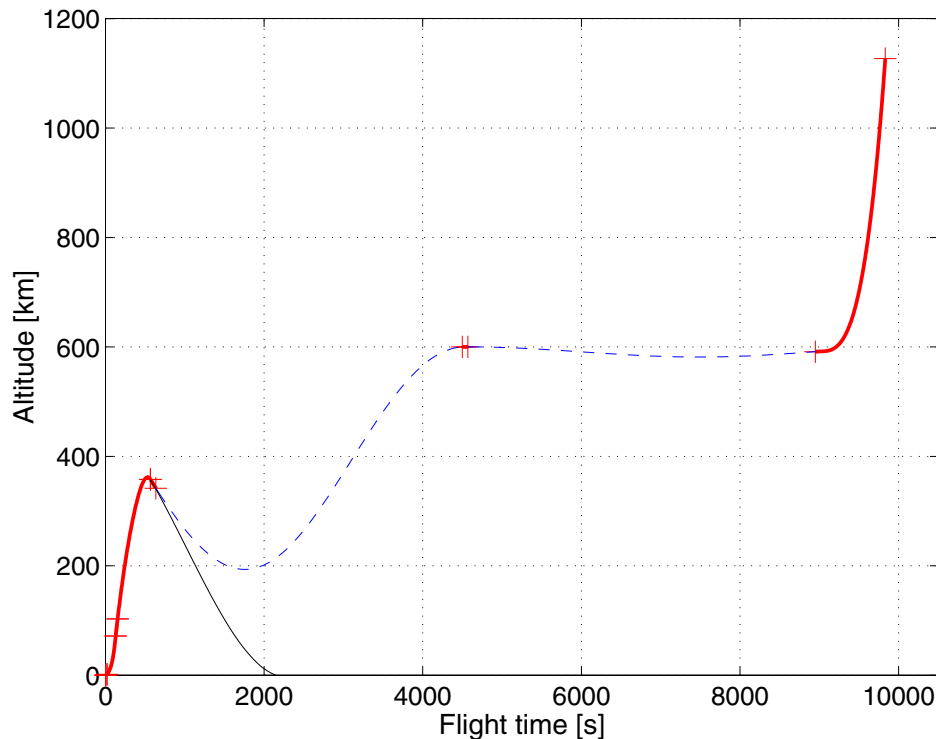


Fig. 4.6: Altitude profile for the modified dual payload mission

The payload to GTO is calculated as 3,224 kg for the modified mission profile. This is 66% more than for the reference mission. At the same time, the mission duration is more than doubled from 1 h 11 min 16 s (4,276 s) in the reference mission to 2 h 43 min 51 s (9,831 s) in the modified mission scenario. Such a significant increase in the mission duration must be considered in the design of the electronic equipment that is used for the upper stage, since it is usually not designed for such a long exposure to the space environment (see also Ref. [17]).

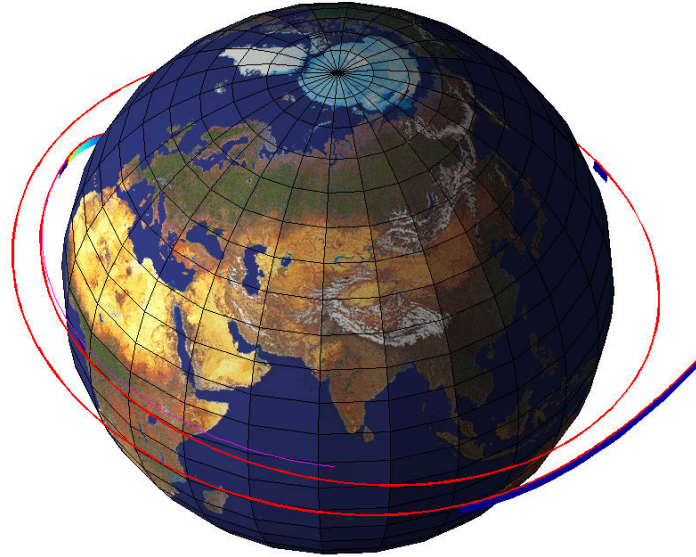


Fig. 4.7: 3D trajectory representation of the modified dual payload mission

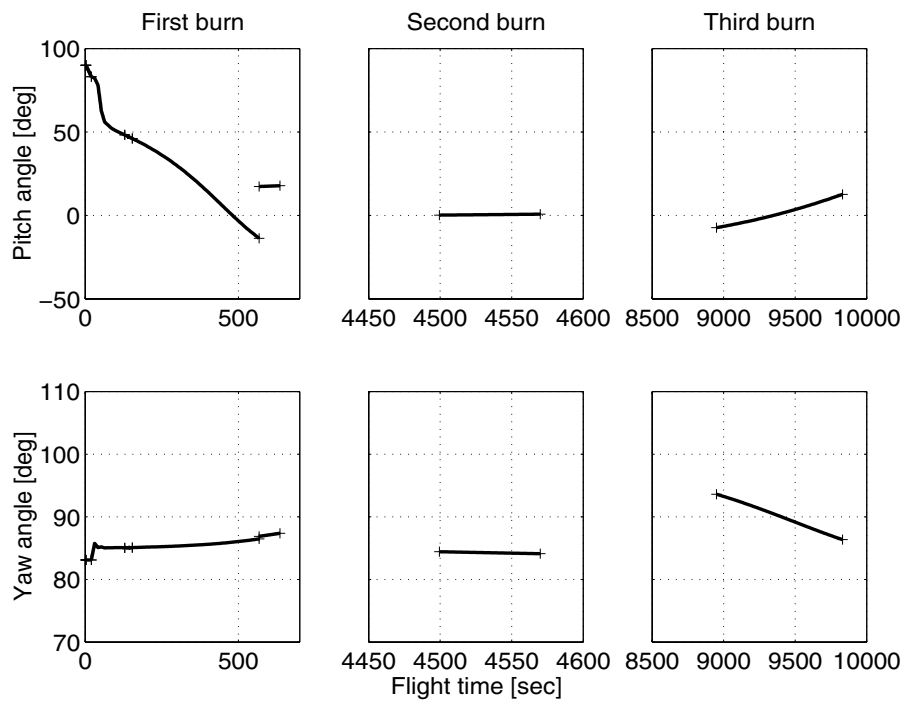


Fig. 4.8: Control time histories for the modified dual payload mission

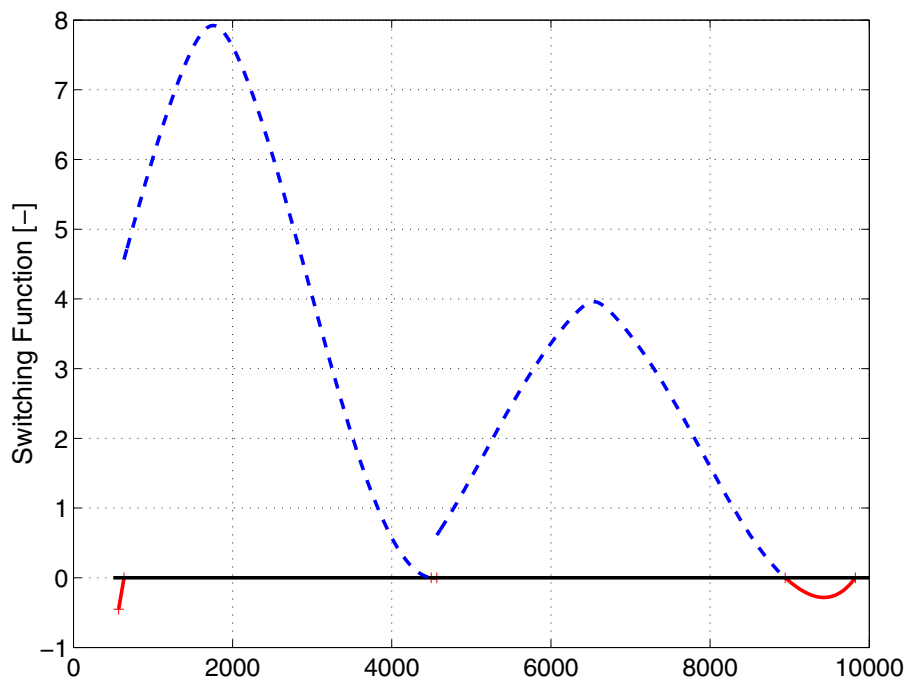


Fig. 4.9: Switching function for the modified dual payload mission

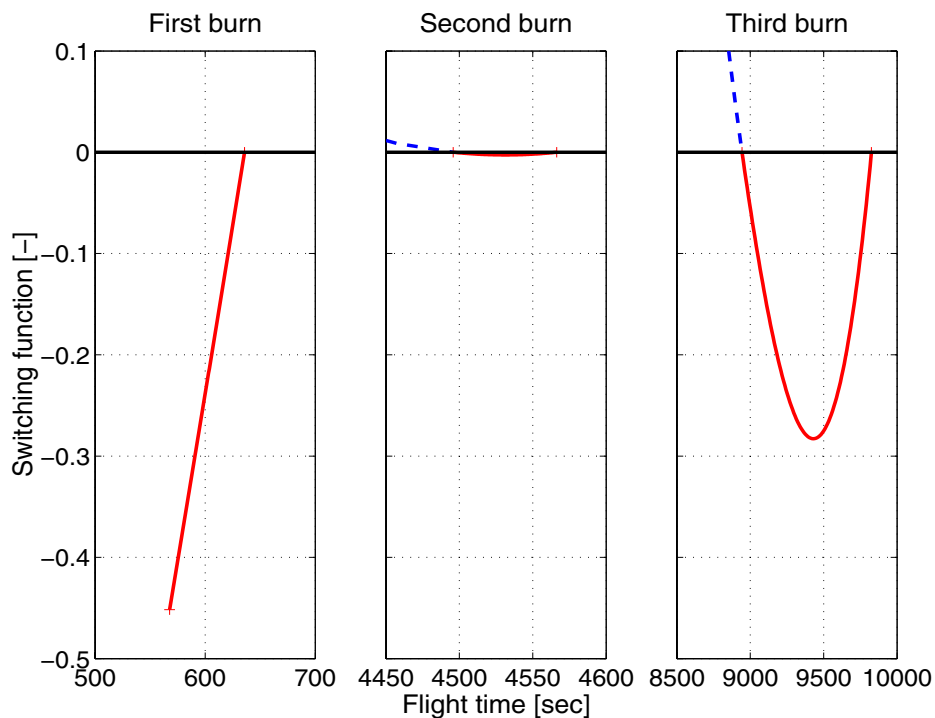


Fig. 4.10: Switching function during burn arcs for the modified dual payload mission

4.3 Summary

The hybrid approach of using direct and indirect optimization methods in one setup leads to optimal solutions for both mission scenarios. An analysis of the first mission revealed that the payload mass can be increased by using a different switching structure. Such a modification is presented in the second mission.

Even though it can be observed that the solution of a problem involving both, direct and indirect phases, is more challenging, the selection of initial guesses for the adjoint variables is not critical to achieve convergence. In both setups, the “ad-hoc” approach discussed in Section 3.6.1 is used to calculate initial guesses for the adjoints. This approach is reported to be successful in other similar applications as well (see Ref. [47]).

As for any trajectory optimization problem, it is very important to use smoothly interpolated data during the optimization process. In this case, the data points given for thrust and aerodynamics are interpolated using Akima splines (Ref. [1]).

In addition, it is necessary to carefully avoid any potential singularity in the constraints or in the differential equations. This is the reason for the special set of terminal constraints used to define the circular target orbit of the first payload, and for the vertical ascent and pitch over phase. During the vertical ascent with a pitch angle of 90 deg, the yaw angle is undefined. Therefore, it is modelled by using a constant real parameter which is connected to the control variable yaw once the pitch angle gets less than 90 deg after the pitch over phase and the yaw angle becomes defined.

Using the indirect method during the upper stage burns greatly enhances the possibilities of a post-optimality analysis. Especially the information provided by the switching function can significantly support the user in finding possible improvements to the selected burn-coast-burn structure. Even though a certain set of transversality conditions has been implemented into the software, it is not necessary to explicitly enforce these constraints during the optimization. Therefore, other combinations of terminal constraints could easily be used in connection with the indirect phases as well if the problem is solved in the framework of an SQP method. However, in most cases, the insertion of indirect phases decreases the convergence performance compared to a purely direct formulation of the problem. This is usually caused by the great sensitivity of the trajectory w.r.t. changes in the adjoint variables. In addition, in cases where the switching function is getting close to zero during burn arcs, it becomes necessary to introduce the switching conditions at the beginning and end of each coast arc in order to obtain the correct switching function values. If these conditions are not introduced, the accuracy obtained with the SQP method may not be sufficient to obtain exact switching function values.

Fig. 4.11 shows the development of the performance index and the merit function during the optimization of the dual payload reference mission. Around iteration number 40, both have almost reached their final value and the optimizer can only slightly improve the performance index afterwards. Fig. 4.12 shows the development of primal and dual infeasibility during the course of the optimization process. It can be observed that the primal infeasibility is reaching fairly low values very rapidly during the first 20 iterations. Afterwards, the optimizer only slightly improves the performance index as well as the constraint satisfaction until it finally achieves convergence after 70 iterations.

The reference mission is optimized using a mixture of multiple shooting and collocation phases, too. Using the collocation method for the coast arc and the two upper stage burns does not improve the convergence behavior compared to the multiple shooting solution in this case. However, it is shown in the next example that collocation phases can significantly improve the convergence behavior if only bad initial guesses are available.

As far as the second mission is concerned, the optimization process is much more difficult. While a first approximate solution is achieved after 98 iterations, obtaining a solution of higher accuracy requires several restarts of the optimizer. This seems to be caused by a very flat minimum. While the performance index is only slightly increased for solutions with higher accuracy demands, the shape of the trajectory may change significantly as the timing of the second upper stage burn is altered.

An indicator showing that the optimizer is having trouble converging this case is the optimization step-size shown in Fig. 4.14. While the convergence history of the reference mission shows the optimal step of one almost during the whole optimization process (see Fig. 4.12), this is not the case for the modified mission scenario. Again, the final value of the performance index is achieved fairly fast as shown in Fig. 4.13.

Both problems can be considered as medium sized for the NLP solver SNOPT. The reference mission involves 286 optimization parameters and 228 constraints (215 equality constraints). The sparsity of the Jacobian matrix (Fig. 4.15) is 8.72%. The modified mission involves 420 optimization parameters and 360 constraints (345 equality constraints). The sparsity of the Jacobian matrix (Fig. 4.16) is 6.15%.

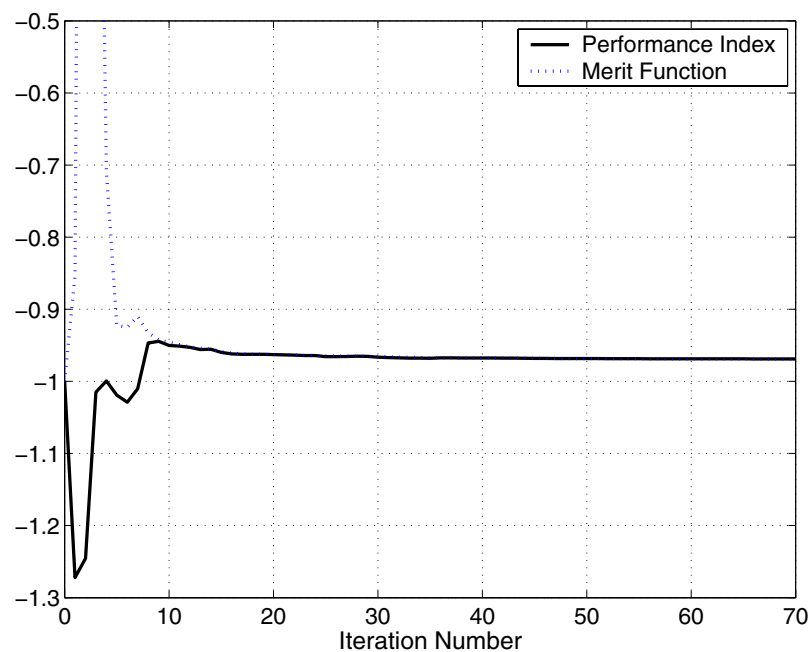


Fig. 4.11: Performance index and merit function value during optimization of the dual payload reference mission

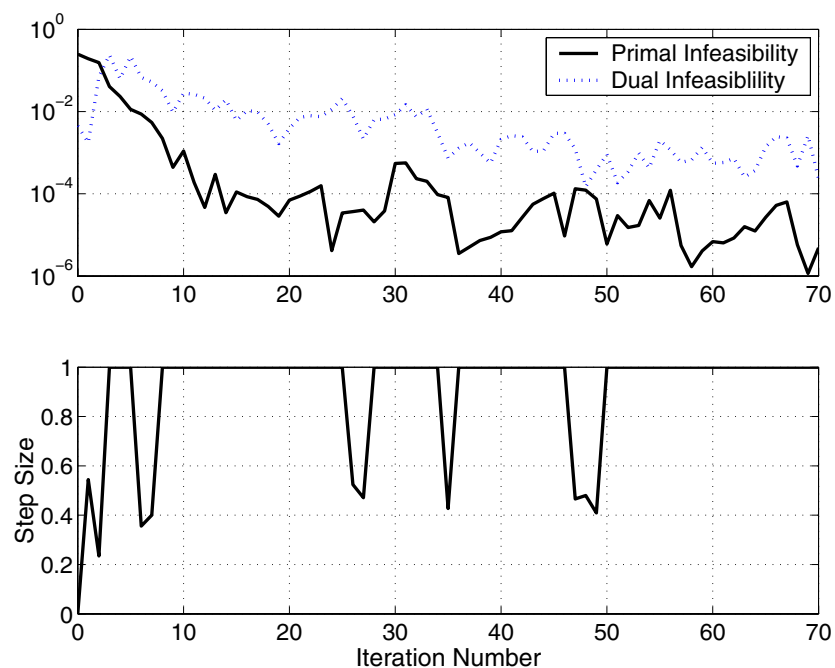


Fig. 4.12: Primal and dual infeasibility and optimization step size during the optimization of the dual payload reference mission.

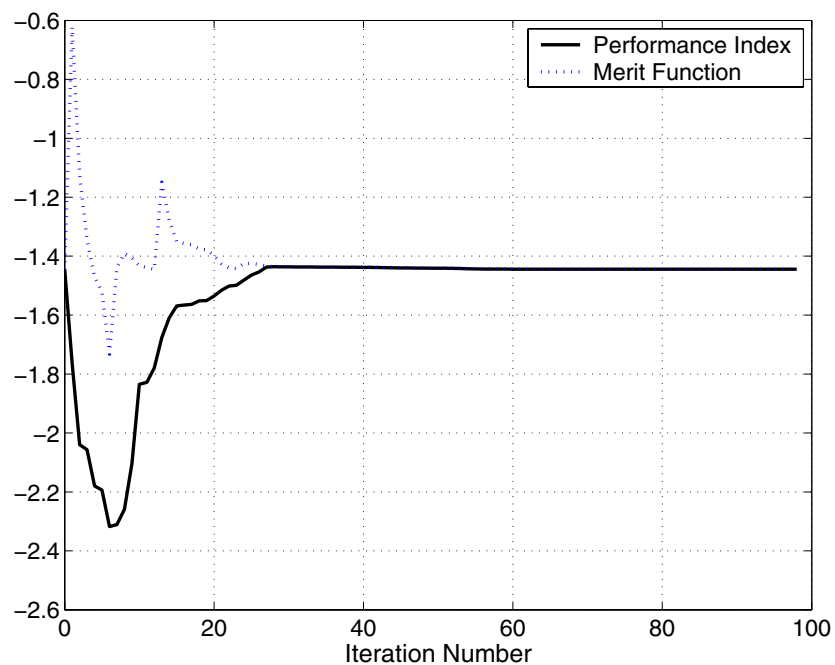


Fig. 4.13: Performance index and merit function value during optimization of the modified dual payload mission

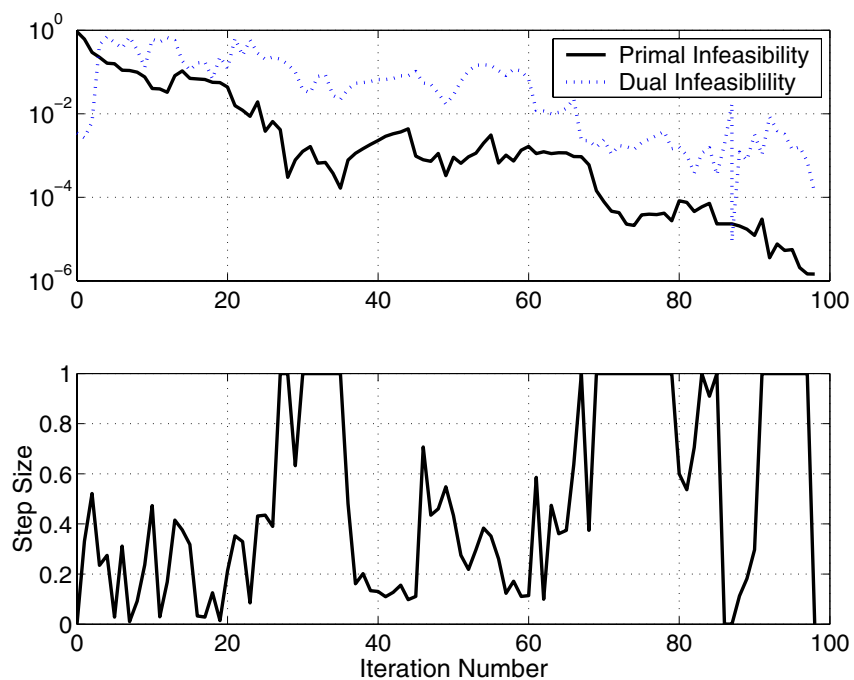


Fig. 4.14: Primal and dual infeasibility and optimization step size during the optimization of the modified dual payload mission

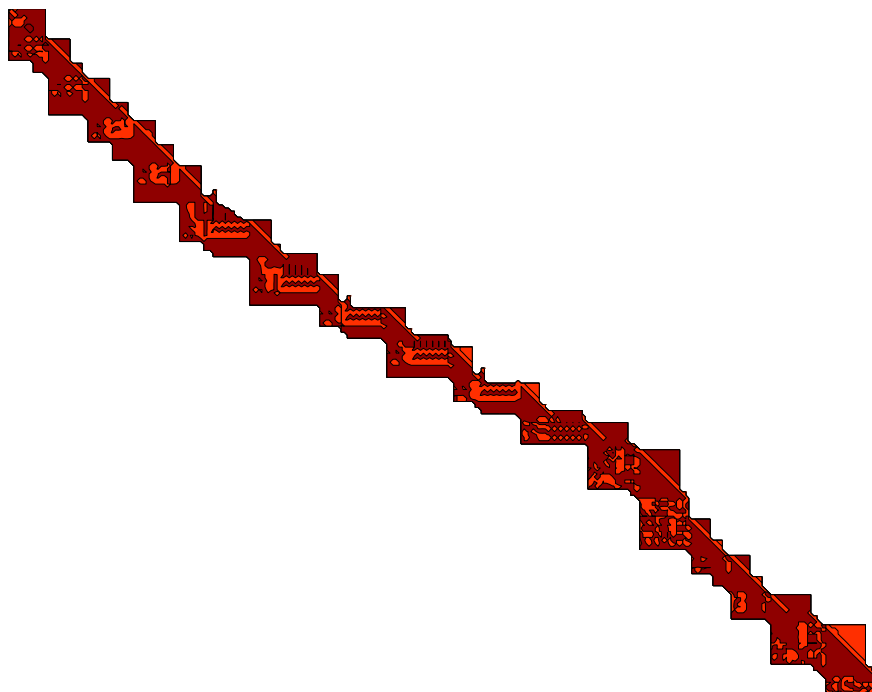


Fig. 4.15: Sparsity structure of the Jacobian matrix for the dual payload reference mission

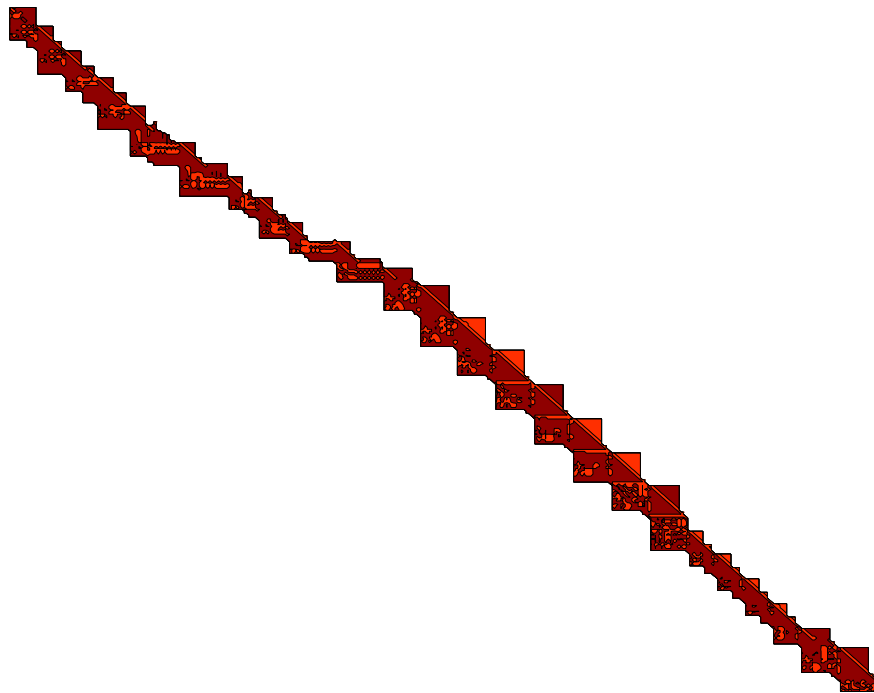


Fig. 4.16: Sparsity structure of the Jacobian matrix for the modified dual payload mission

5 Example 2: Branched Mission Scenario

*May the wind under your wings bear you
where the sun sails and the moon walks.*

Gandalf in J.R.R. Tolkien, “The Hobbit”

The second example problem is based on the Hopper concept from the Astrium GmbH. Preliminary data for this mission and for the vehicle was kindly provided by J. Spies from the Astrium GmbH, Bremen. The mission is optimized using a mixture of direct collocation and direct multiple shooting.



Fig. 5.1: The Hopper concept [89]

5.1 Mission Scenario

The Hopper is a suborbital, winged launch and re-entry vehicle (Fig. 5.1). It will be launched horizontally from a slide at Kourou spaceport and perform a rocket-powered ascent using the three Vulcain-2 engines up to an altitude of about 110 km. An upper stage using one Vinci engine will then be jettisoned and delivers the payload into its target orbit. The Hopper vehicle itself returns in an un-powered descent to a landing site (see Fig. 5.2). Propellant and structural masses used for this example mission are shown in Table 5.1, the propulsion system data is given in Table 5.2. The environment models are defined in Appendix E. The Objective of the optimization problem in this example is to maximize the payload delivered into the target orbit.

Component	Structure [kg]	Propellant [kg]
Hopper	50,130.0	272,250.0
Upper stage	2,714.65	15,000.0
Payload	maximized	-

Table 5.1: Vehicle components of the Hopper mission

	Vulcain-2	Vinci
Vacuum ISP	440.0 s	460.0 s
Thrust	1,350 kN	150.0 kN
Nozzle exit area	3.22724 m ²	0.0 m ²
Number of engines	3	1

Table 5.2: Propulsion system properties

The problem must be treated as a branched trajectory optimization problem, since the requirement of a lower stage return significantly influences the trajectory shape and the payload performance. In addition, the return leg cannot be modeled by a simple constraint due to the aerodynamic maneuvering capabilities of the Hopper vehicle and the requirement of a safe return without violating the vehicle's limitations on maximum heatflux, dynamic pressure, and load factor.

In this example, a mission is optimized where the payload is delivered to a geostationary transfer orbit with 200 km perigee altitude, 7 deg inclination, and 180 deg argument of perigee. The apogee altitude is 35,800 km. The Hopper vehicle returns to Ascension Island.

In order to allow for a smooth interpolation of the aerodynamic data for the ascent, the original data set is split into two Mach-domains. The first domain is for Mach numbers less than two, the second one for Mach numbers greater than two. Splitting the original data set at

Mach two yields the best interpolation results when applying the minimum curvature B-spline method shown in Ref. [11], Chapter 5.2.3. The interpolated aerodynamic data is shown in Fig. 5.3 and Fig. 5.4.

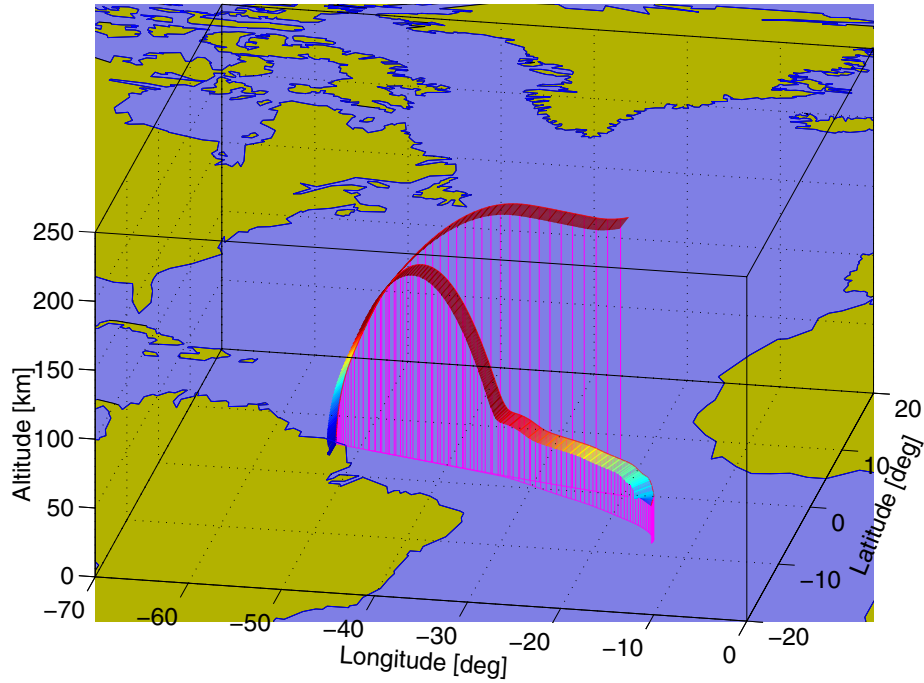


Fig. 5.2: 3D trajectory representation of the Hopper mission scenario

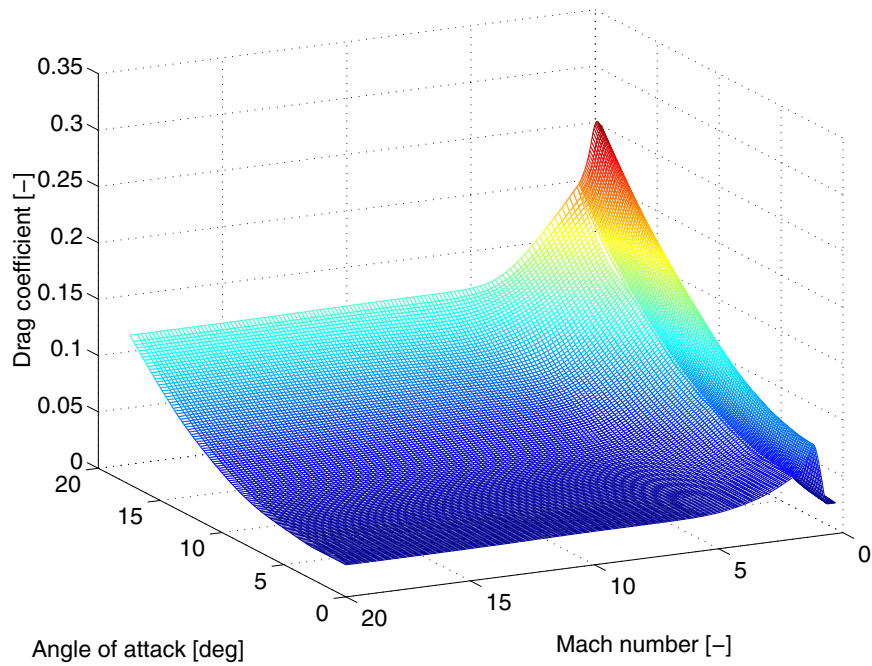


Fig. 5.3: Drag coefficient for Hopper vehicle

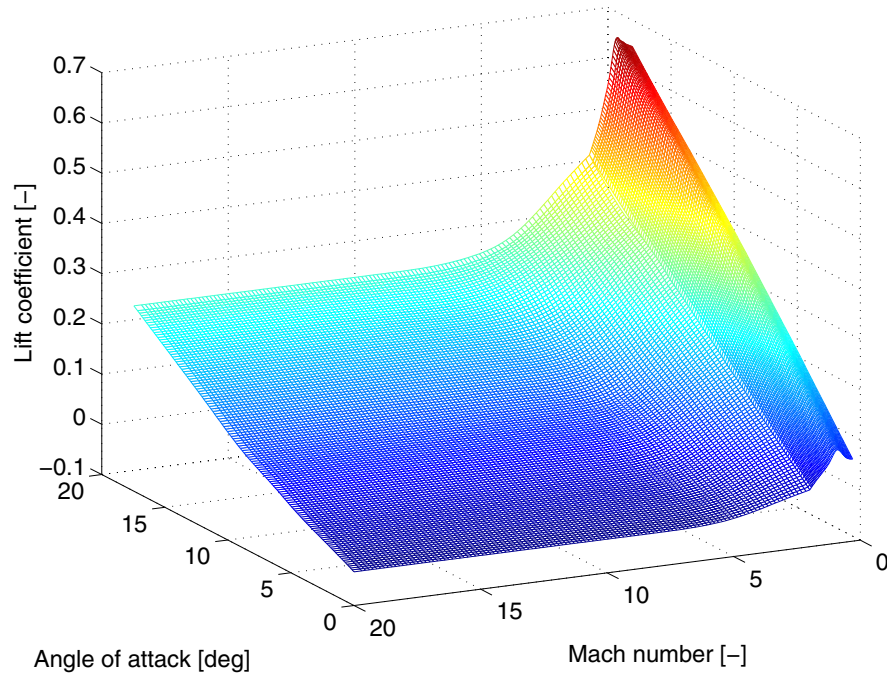


Fig. 5.4: Lift coefficient for Hopper vehicle

The phase structure used for the optimization is shown in Table 5.3 while Table 5.4 depicts the initial conditions. The first four phases could also be merged into one large phase. However, smaller phases are used to improve the handling in the graphical user interface. It also reflects the structure of the control law sequence that is used for the initial guess generation.

	Time [sec]	Active Prop.	Path Constr.	Boundary Constr.	Remark
1	0 - 20	3 x Vulcain 2	-	Kourou launch site	
2	20 - 30	3 x Vulcain 2	-	-	
3	30 - 54	3 x Vulcain 2	$q \leq 40kPa$	-	
4	54 - $t_{f,4}$	3 x Vulcain 2	$q \leq 40kPa$	$M = 2$	
5	$t_{f,4}$ - 290	3 x Vulcain 2	-	lower stage burnout	branch upper stage
6	290 - $t_{f,6}$	-	-	$\dot{Q} \leq 1135W/m^2$	coast arc
7	$t_{f,6}$ - $t_{f,7}$	1 x L9	-	GTO conditions	payload jettisoned
8	$t_{f,7}$ - $t_{f,8}$	-	-	$h_f = 120$ km	coast arc
9	$t_{f,8}$ - $t_{f,9}$	-	$\dot{Q} \leq 1135W/m^2$ $n \leq 3.5$ $\gamma \leq 0deg$ $\alpha(M) \leq f(M)$	TAEM (Ascension Island)	-

Table 5.3: Phase structure of the branched Hopper mission scenario

	Value
Altitude	1 m
Longitude	-52.775 deg
Declination	5.232 deg
Flight-path velocity	100 m/s
Flight-path angle	0 deg
Flight-path azimuth	90 deg

Table 5.4: Initial conditions of the branched Hopper mission scenario

5.2 Initial Guess Generation for the Hopper Mission

Before the complete branching case is optimized, a simple ascent to GTO without the lower stage return path is optimized using a direct collocation setup. The initial guess for this ascent mission is constructed by using a sequence of phases with constant controls for α and μ . Values and timing are found interactively by performing simulations in the GESOP environment and subsequently adding additional phases when the vehicle seems to fly a reasonable trajectory. During the upper stage burn, yaw and pitch are used as controls. For the initial guess they are simply kept constant; their initial values automatically correspond to the final values of α and μ (see Table 5.5). No effort was taken to perform any fine tuning of this initial guess since the whole setup required approximately half an hour, and it is sufficient to get the optimizer started.

Flight time [sec]	Angle of attack [deg]	Bank angle [deg]
0 - 20	20	0
20 - 30	13	0
30 - 54	0	30
54 - 70	2	10
70 - 290	5	0
290 - 310	-	-
310 - 760	pitch = constant	yaw = constant

Table 5.5: Initial guess for the ascent branch of the Hopper mission

In order to achieve convergence for this submission, a complete direct collocation setup is used. This greatly accelerates the computation compared to a direct multiple shooting setup. Since the Hopper vehicle does not involve any solid rocket boosters with thrust profiles, the trajectory can be discretized with fairly few collocation intervals in order to achieve a sufficient accuracy. The resulting NLP problem has 1,457 optimization parameters and 1,244 con-

straints (1,213 equality constraints). The density of the Jacobian matrix is 1.7%. All constraints are enforced right from the beginning of the optimization process, and the optimizer rapidly converges to an approximate solution. This solution is only an approximate solution, since additional collocation nodes would have to be inserted in order to accurately represent the dynamics.

However, such an approximate solution is sufficient to initialize a new optimization problem which includes the return leg for the lower stage. For this mission, phases 8 and 9 need to be added to the scenario. Phase 8 is a coast arc to the entry interface in 120 km altitude, and phase 9 represents a guided re-entry phase to the target area energy management interface (TAEM). The latter one is defined as:

$$27 \text{ km} < h < 40 \text{ km}$$

$$0.8 \text{ km/s} < V_{rel} < 1.2 \text{ km/s}$$

$$\lambda = -14.388 \text{ deg}$$

$$\delta = 7.937 \text{ deg}$$

For the initial guess of the guided re-entry phase, the bank angle is set to a linear function increasing from 20 to 30 deg. The angle of attack is held constant at 20 deg. The control time history of the remaining phases is initialized by using the results from the simple GTO mission obtained previously. These settings produce an initial guess trajectory that involves several skips, but guide the lower stage towards Ascension Island (see Fig. 5.5).

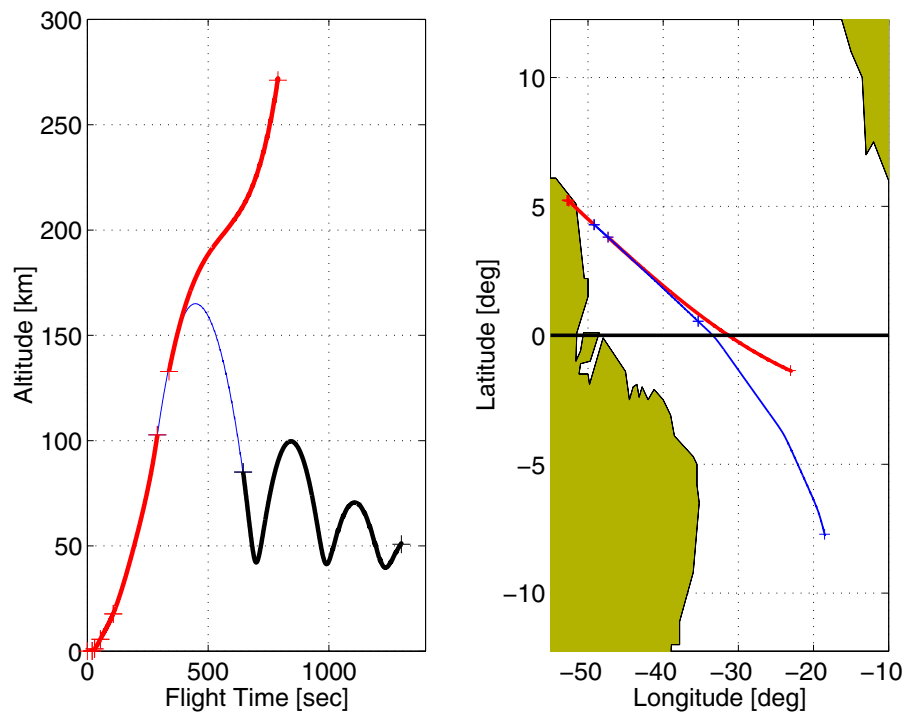


Fig. 5.5: Initial guess trajectory for branched Hopper mission

5.3 Optimization of the Branched Mission Scenario

The optimization of the branched mission scenario is started with a complete collocation setup as well. This setup results in an NLP problem with 2,052 optimization parameters and 1,869 constraints (1,749 equality constraints). The density of the Jacobian matrix is 1.35%. The typical sparsity pattern of this matrix is shown in Fig. 5.7. Initially, the path constraint on the flight-path angle are not enforced, which yields an intermediate trajectory with several skips. In a subsequent optimization run, the missing constraint are enforced.

Finally, the bank angle during the first 10 seconds of the ascent is fixed to zero and the angle of attack is fixed to its maximum value of 20 deg in order to make sure that the vehicle does not drop too far below ground level at the beginning. Fixing the controls in this manner leads to a much better convergence behavior as compared to the introduction of a path constraint on altitude. Unfortunately, the vehicle still drops several meters below its launch altitude which suggests that either the payload mass has to be reduced or the launch area and procedure must permit such a drop, e.g. by using an elevated slide, a higher take-off speed, or permitting an initial flight-path angle greater than zero.

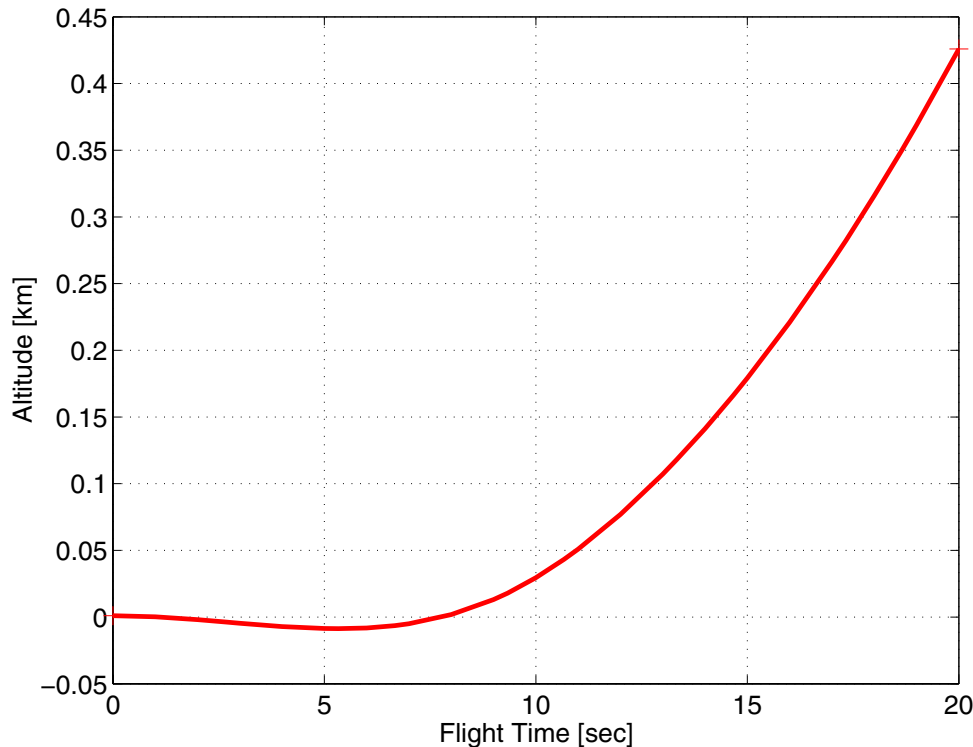


Fig. 5.6: Altitude drop right after launch

In order to maintain a reserve in the control authority during the re-entry part of the trajectory, the bank angle is also constrained to be a minimum of 30 deg in the region where the maximum heatflux and load factor are encountered.

By activating the additional constraints, the problem size quickly grows to 2,052 optimization parameters and 1,926 constraints (1,749 equality constraints). Such a problem requires about 600 MB of RAM during the solution process. In order to reduce the number of param-

ters and constraints, and therefore also memory consumption, the first three phases are changed to multiple shooting phases with no multiple shooting points. This reduces the number of optimization parameters to 1,715 and the number of constraints to 1,620 (1,422 equality constraints). Due to the multiple shooting method, the density of the Jacobian slightly increases to 1.7%.

Using the collocation method during the re-entry phase has shown to be superior compared to the multiple shooting method. The latter one significantly increases the required computation time and is much more sensitive to the path constraints, especially to the flight-path angle path constraint. The direct collocation method is much more robust with respect to the initial path constraint violations during the optimization process. With the direct collocation method, it is even possible to find a solution from a fairly bad initial guess with which it is not possible to use the direct multiple shooting approach.

Once the optimization process is converged, the re-entry phase is changed to a multiple shooting phase, too, in order to obtain a final solution of high accuracy. The final NLP problem includes 1,471 optimization parameters and 1,351 constraints (1,170 equality constraints). The density of the Jacobian matrix increases to 2.05%. Run-time also significantly increases by a factor of approximately four compared to the previous setup. However, the final result ensures a flyable trajectory which does not violate the critical path constraints during re-entry.

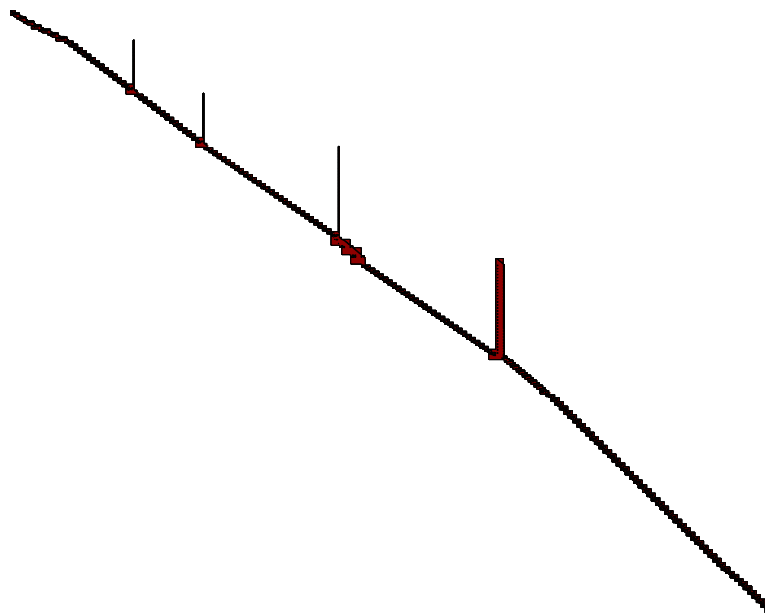


Fig. 5.7: Sparsity patten of the Jacobian matrix for the branched Hopper mission

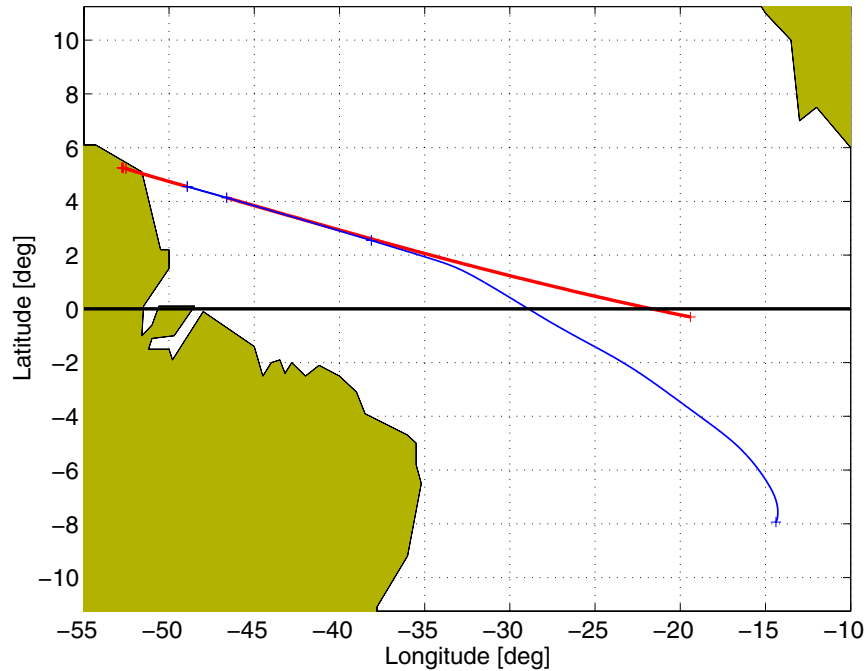


Fig. 5.8: Ground track of the Hopper vehicle and the upper stage

5.4 Optimization Results

The optimization yields a payload of 5,055 kg to GTO. In order to obtain this result, the optimization is run until a tight convergence is achieved. The final primal infeasibility obtained is 5×10^{-8} and the dual feasibility is 8×10^{-6} .

The resulting trajectory shows a long gliding phase for the lower stage to Ascension Island (Fig. 5.8 and Fig. 5.9). The entry interface in 120 km altitude is encountered at 2.3 deg northern latitude. The total duration of the gliding phase from the entry interface to the TAEM interface is 963 s. At the very beginning of the guided re-entry phase, control saturation during the maximum acceleration phase is avoided by constraining the bank angle to a minimum of 30 deg (see Fig. 5.10 and Fig. 5.11). For a reference trajectory, such a constraint in the bank angle needs to be included in the setup in order to have a sufficient control margin available for the correction of any disturbances.

The heatflux limit is encountered shortly after the maximum acceleration. However, this is not as critical since there is a sufficient control margin in μ . The remainder of the trajectory is far away from any of the constraints and also shows sufficient reserves in lift and cross-range capabilities (see Fig. 5.10, Fig. 5.11 and Fig. 5.12). The drag-velocity diagram in Fig. 5.12 shows a very small violation of the load-factor limit which can also be observed in Fig. 5.11. These violations could be removed by adding further path constraint refinement points.

During the ascent phase, the Hopper vehicle performs some small roll maneuvers during the first few seconds, followed by a larger roll maneuver at around 50 s. The larger maneuver is used to guide the vehicle in a south-eastern direction. The small bank maneuvers at the very beginning are used to maintain an optimal angle of attack. Similar results have been found in Ref. [31] (p. 1187). The small bank angle slightly reduces the vertical component of the lift force in order to maintain an optimal climb path. The heading change is corrected by a roll maneuver in the opposite direction shortly afterwards. Note that for low Mach numbers, the vehicle has a positive lift even for an angle of attack of zero (see Fig. 5.4).

The upper stage controls are depicted in Fig. 5.13, the resulting thrust vector is illustrated in Fig. 5.14. It can be observed that the pitch angle does not follow a simple bilinear tangent guidance law but changes its behavior at 600 s into the flight. Fig. 5.14 shows that the thrust vector is almost tangential to the trajectory at the end of the upper stage burn while it follows the typical bilinear tangent guidance law in the remainder of the burn arc. This behavior can be explained by considering the additional constraint on the argument of perigee for the GTO orbit in connection with the lower stage return to Ascension Island. The geographic location at the beginning of the upper stage burn is not optimal in terms of orbital mechanics. Therefore, the vehicle does not perform a simple guidance scheme during the whole upper stage burn. In addition, the vehicle first raises the perigee radius to the required 200 km altitude. Beginning at 600 s during the flight, the required perigee altitude is achieved and the apogee altitude is raised (see Fig. 5.15).

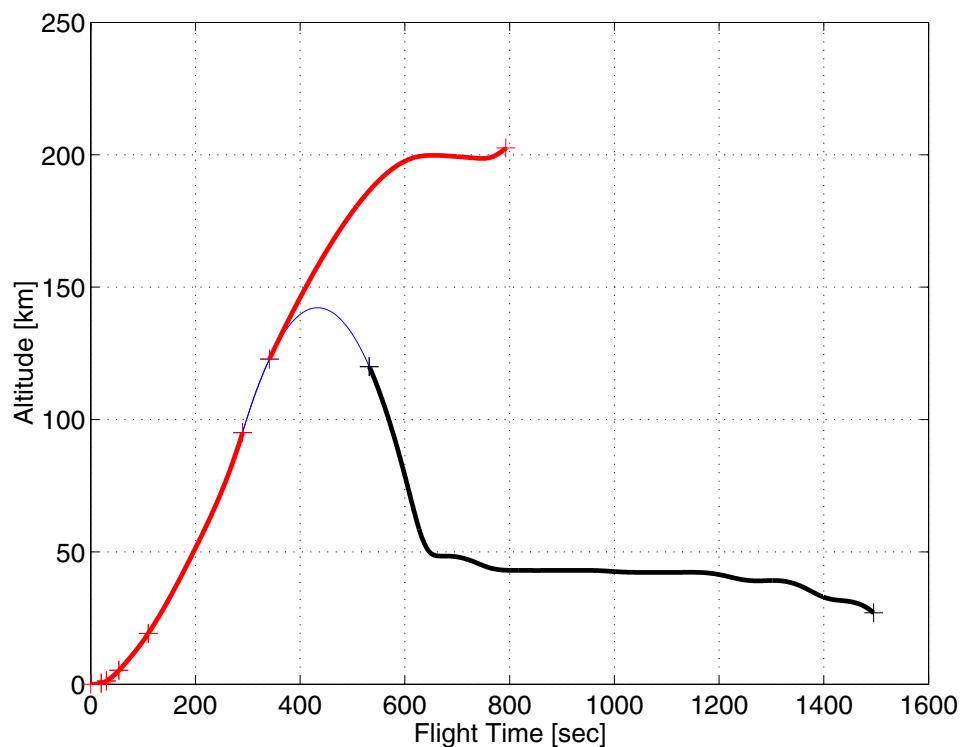


Fig. 5.9: Altitude profile of the Hopper mission

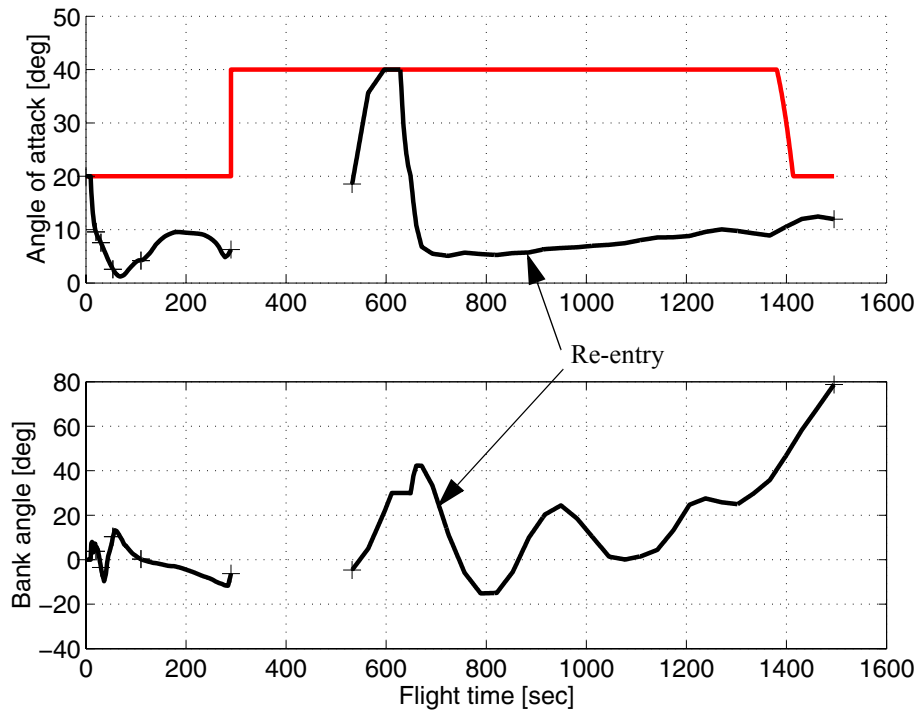


Fig. 5.10: Control time histories for the Hopper vehicle

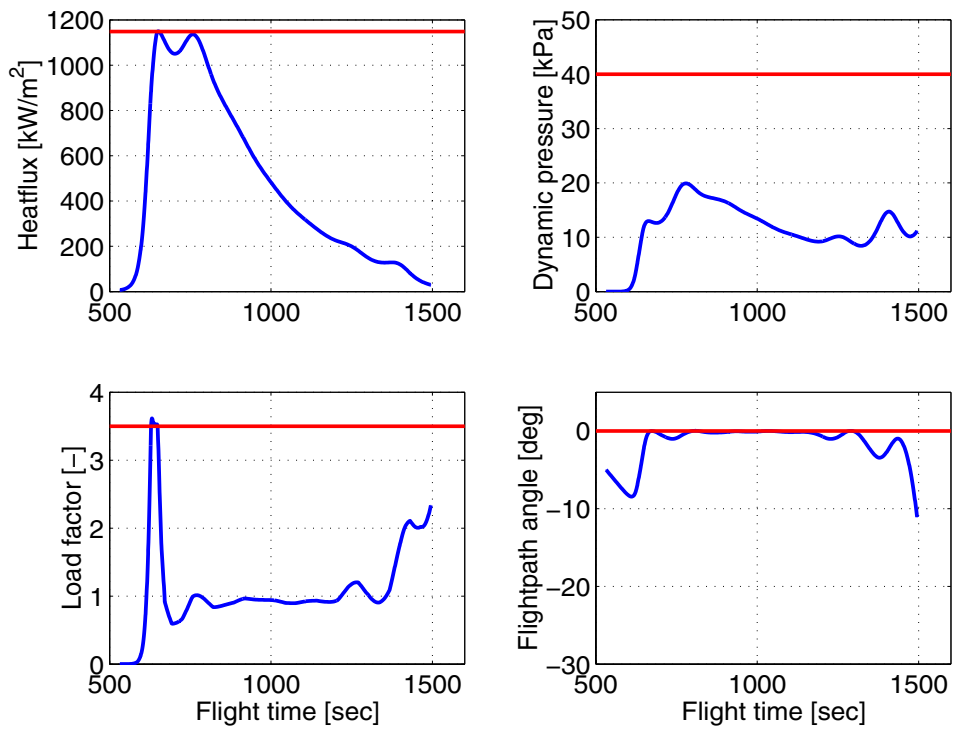


Fig. 5.11: Path constraints during the Hopper re-entry

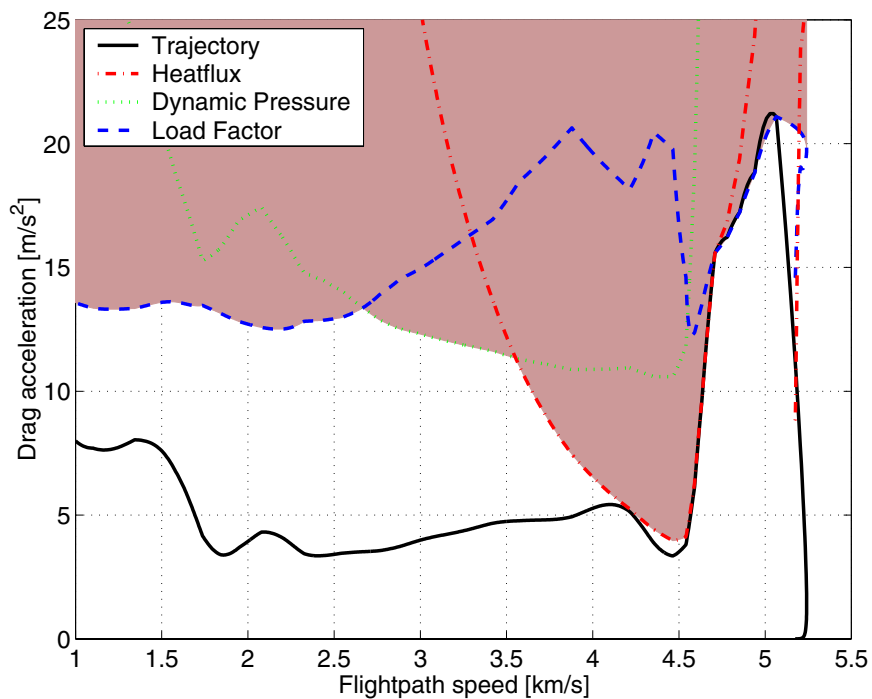


Fig. 5.12: Drag velocity diagram for Hopper re-entry

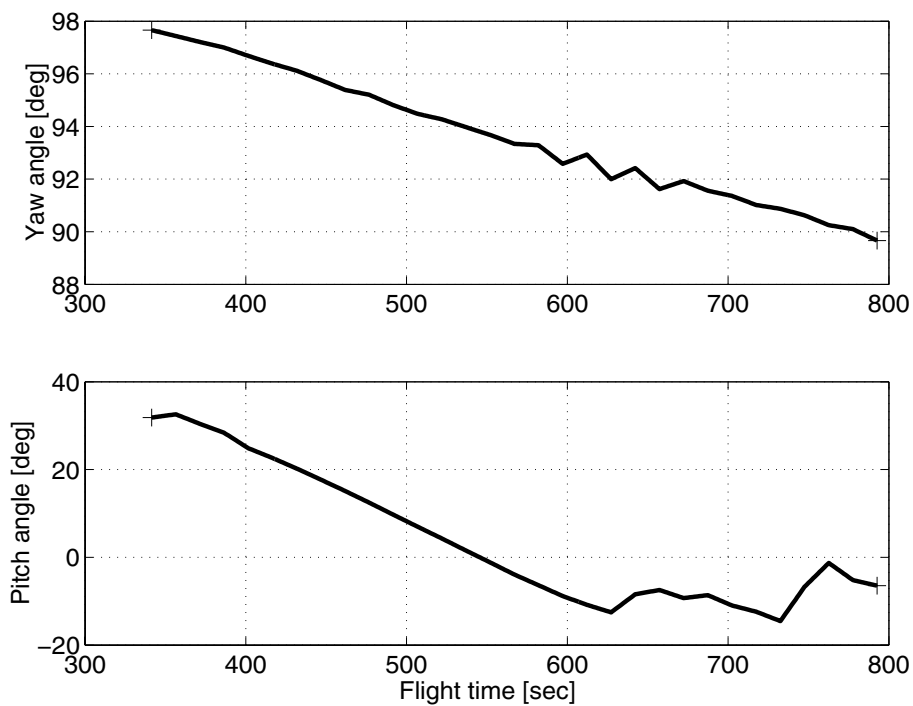


Fig. 5.13: Control time histories of upper stage

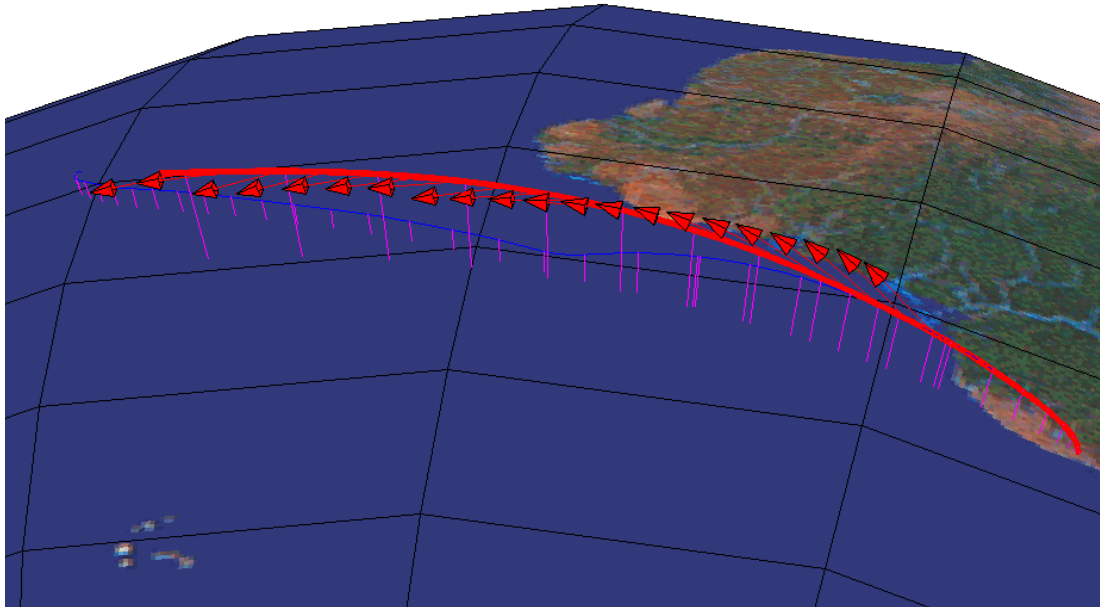


Fig. 5.14: Thrust direction during upper stage ascent

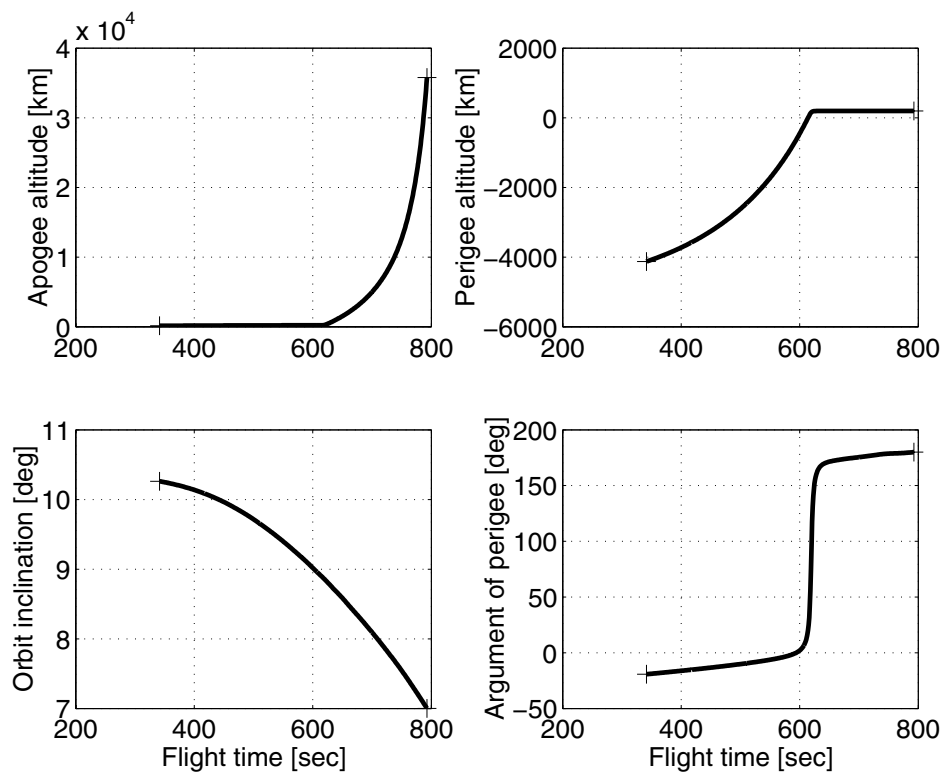


Fig. 5.15: Orbital elements during the upper stage ascent

6 Conclusions and Future Research

The trend in the numerical treatment of optimal control problems in aerospace engineering points towards problems of increasing complexity in order to approximate reality as close as possible. These problems lead rather to an increasing number of constraints than to an increasing number of unknowns involved in the problem. Thus, engineers and mathematicians who are involved in the design and development of future aerospace enterprises have a strong demand for reliable and efficient software that can handle optimal control problems with different types of constraints.

H. J. Pesch, A practical guide to the solution of real-life optimal control problems, Ref. [64]

6.1 Conclusions

The objective of this thesis is the development of a hybrid trajectory optimization method that combines the benefits of existing direct and indirect methods. This new method called CAMTOS allows for a phase-wise discretization using either direct multiple shooting or direct collocation. If the user also supplies the adjoint differential equations, CAMTOS is also able to compute the required values of the initial costates. It can even be used to solve the corresponding boundary value problem when the transversality conditions are available as well. However, the derivation of the transversality conditions can be avoided if the cost function of the underlying NLP solver is used. In these cases, the NLP solver will select the initial adjoint variables such that they minimize the performance index. The latter approach works very well for computing rocket thrust arcs in vacuum.

Using the combination of direct and indirect methods also allows for an analysis of the switching function. This method gives a lot of insight into the problem and the optimality of the chosen burn-coast-burn structure in case of rocket ascent trajectories. At the same time, it requires a minimum amount of effort from the user's side since the transversality conditions need not be coded since they are computed numerically in the Karush-Kuhn-Tucker conditions

within the NLP solver. Therefore, a large variety of terminal conditions becomes readily available without any additional effort. As far as the computation time is concerned, the indirect method sometimes achieves a better performance in this setup when it is compared to an equivalent setup that uses the direct method. However, it usually requires more computation time since states and costates need to be integrated numerically. In the Ariane 5 dual payload example, the setup with the indirect method in the upper stage requires eight times more computation time for the reference mission and 1.3 times more computation time for the modified mission scenario. Therefore, the main advantage of this method is the availability of additional information about the optimality of the burn-coast-burn sequence.

The combination of direct multiple shooting and direct collocation simplifies the overall process of obtaining optimal results for new problems. While initial solutions can be obtained quickly by using the fast and robust direct collocation method with only a few collocation intervals, a final solution of high accuracy can readily be achieved by switching critical phases to use a multiple shooting discretization.

The general applicability is shown on two complex mission scenarios that involve several Ariane 5 launcher scenarios. These scenarios include missions with up to two coast arcs and a set of realistic path and boundary constraints. By using the combination of direct and indirect phases, an improved burn-coast-burn sequence is identified for an Ariane 5 dual-payload mission. This improvement leads to an increase in the payload mass of approximately 66%.

In addition, a branched trajectory scenario is optimized, where the rocket-powered lower stage returns in a controlled, gliding flight to a landing site while the upper stage delivers a payload into GTO. This optimization problem is solved by starting from a fairly bad initial guess using the direct collocation method. Once an approximate solution is obtained, the critical phases within the atmosphere are changed to the direct multiple shooting method in order to obtain a result with higher accuracy.

The investigation of several different integration methods for the direct multiple shooting method shows that for all problems that are encountered in the framework of this thesis, the Runge-Kutta 4/5 method exhibits the best convergence behavior. The integration method suggested by Paus (Ref. [61]) is not able to achieve the same performance as a simple Runge-Kutta 2/3 method with the same order of accuracy in such a general setup. It can therefore be concluded that an application of the Paus method requires a significant amount of fine-tuning between the integration method and the optimization model. An exploitation of sparsity patterns in the dynamics as well as the computation of the linearized system matrices A and B should be performed in an efficient manner.

The performance of CAMTOS is compared to an existing direct multiple shooting method as well as to an existing direct collocation method. All codes are using SNOPT to solve the underlying NLP problem. The direct multiple shooting method implemented in CAMTOS provides the same computational performance as the existing PROMIS code, while the direct collocation method exhibits an improved convergence behavior and run-time performance compared to the direct collocation method TROPIC. This improvement is achieved by introducing additional control variables and constraints at the boundaries of the collocation intervals. These additional variables do not only permit for a piece-wise constant control

approximation, which is not possible with TROPIC, but they also relax the transcribed optimal control problem such that it can be solved more efficiently by the SQP solver.

The overall design of CAMTOS allows for an easy implementation of future enhancements. Such enhancements can be additional integration methods, collocation schemes, or even NLP solvers. Some of these possibilities are discussed in the next Section.

6.2 Future Research

There are several interesting possibilities for future enhancements of the developed trajectory optimization algorithms.

For phases that are discretized with the direct collocation method, it is beneficial to have an algorithm that performs an automatic mesh refinement. Such an algorithm requires an estimator for the discretization error in each interval. Based on this information, a strategy can be developed that adds additional grid points in regions that exhibit a large discretization error. It may also be interesting to include a logic that removes grid points from regions in which the discretization error is much smaller than the required accuracy. Such an approach might avoid situations, in which many additional grid points are inserted in early stages of the mesh refinement in regions, where these points may not be required anymore in later stages.

Especially for engineering design problems, it is also very beneficial to perform a post-optimality and sensitivity analysis. However, additional research is required in order to find a general approach for such analyses that covers many different kinds of questions that arise in the framework of vehicle design and system identification. The Lagrange multipliers calculated by the SQP method should be exploited in order to compute some of the required derivative information.

Many real-world problems involve not only one (global) minimum, but include several different local minima. The current NLP solver included in CAMTOS is only able to find a local solution. It depends on the initial guess that is provided, which local solution is found. By using the method of multiple restarts with different initial guesses, it is possible to find other local solutions and, possibly, also the global solution. However, such a process is usually very tedious to do and often requires a lot of system knowledge by the user. Recent developments in the area of genetic algorithms has produced methods that are not only able to handle unconstrained minimization problems. With the new methods, it is also possible to find globally optimal solutions for problems that involve nonlinear constraints. Such methods are also interesting for applications that involve constraints or performance indices that are not continuous or differentiable.

While the results obtained with constrained genetic algorithms are often very impressive, they also require a lot of computation time. It is common that such algorithms need several thousand times longer to obtain a solution than SQP methods. Therefore, it will most likely become necessary to perform a distributed and/or parallel computation for the whole population of one generation. Such a distributed computation could be performed on a cluster of fairly cheap standard PCs that are communicating via TCP/IP sockets. Also, the computation of the Jacobian matrix when using the direct multiple shooting or direct collocation method

could be performed on several computers at once. Additional research needs to be performed in order to find an easy way to establish such a parallel computation with a minimum amount of overhead for the communication protocol. It is also necessary to incorporate provisions to handle a failure of one or more computers during the optimization process. In such cases, the corresponding work packages should be sent out again to a different computer so that the overall optimization can continue and does not stall only because one computer has failed.

Another enhancement for the multiple shooting method is a special integration method that allows for event handling. While special events can also be modelled by using the multi-phase approach, many problems exist where the required number of phases cannot be pre-determined. Example are a linearly interpolated data table or an integrator that involves upper and lower limits. While a Runge-Kutta 4/5 method is often able to handle such discontinuities in the dynamics, it may deliver results that are not accurate enough because the exact position of an event is not determined. If the user supplies an additional function that changes sign at such events, these can be trapped during the computation of a reference trajectory. Such an approach can also save computation time in the step-size control algorithm that may otherwise be spent by adjusting the step-size to a very small value in order to reduce the local error introduced by a local discontinuity in the dynamics.

References

- [1] Akima, H., “A New Method of Interpolation and Smooth Curve Fitting Based on Local Procedures”, *Journal of the Association for Computing Machinery*, Vol. 17, No. 4, October 1970, 589-602
- [2] Aksu, A., “Analyse von Eintrittsbahnen einer Raumkapsel auf dem Mars”, Studienarbeit, IFR_SR_00_006, Institute of Flight Mechanics and Control, University of Stuttgart, Germany, 2000 (in German)
- [3] Aksu, E., “Optimierung von Aufstiegsbahnen mit mehreren Freiflugphasen”, Studienarbeit, IFR_SR_00_007, Institute of Flight Mechanics and Control, University of Stuttgart, Germany, 2000 (in German)
- [4] Ardema, M.D., Windhorst, R., Phillips, J., “Optimization of Supersonic Transport Trajectories”, NASA-TM-1998-112223, NASA Technical Memorandum, Ames Research Center, Moffett Field, CA, March 1998
- [5] ASTOS Model Library Reference Manual, ASTOS MLR 4.3, Institute of Flight Mechanics and Control, University of Stuttgart, Germany, March 2001
- [6] ASTOS Conventional Launcher Application Manual, ASTOS CLA 4.3, Institute of Flight Mechanics and Control, University of Stuttgart, Germany, March 2001
- [7] Barnes, J., *Programming in Ada95, Second Edition*, Addison-Wesley Longman Limited, Harlow, England, 1998
- [8] Bate, R., Muller, D., White, J., *Fundamentals of Astrodynamics*, Dover, New York, 1971, pp. 40-43
- [9] Battin, R.H., *An Introduction to the Mathematics and Methods of Astrodynamics*, AIAA Education Series, American Institute of Aeronautics and Astronautics, Inc., New York, NY, 1987

- [10] Betts, H., Huffman, W.P., "Trajectory Optimization on a Parallel Processor", *Journal of Guidance, Control, and Dynamics*, Vol. 14, 1991, pp. 431-439
- [11] Betts, J.T., *Practical Methods for Optimal Control Using Nonlinear Programming*, Advances in Design and Control, Society for Industrial and Applied Mathematics, 2001
- [12] Betts, J.T., Huffman, W.P., "Mesh Refinement in Direct Transcription Methods for Optimal Control", *Optimal Control Applications & Methods*, Vol. 19, pp. 1-21, 1998
- [13] Bock, H.G., Plitt, K.J., A Multiple Shooting Algorithm for Direct Solution of Optimal Control Problems, Institut für Angewandte Mathematik, Report 09.2/A-5, SFB 72, University of Bonn, in Preprints of the 9th IFAC World Congress, Budapest, Hungary, Vol. 9, 1984, pp. 242-247
- [14] Brändle, T., "Bahntransfermission eines Satelliten mit elektrischem Antrieb", Studienarbeit, IFR_SR_01_009, July 2001 (in German)
- [15] Breitner, M.H., Koslik, B., Stryk, O.v., Pesch, H.J., "Optimal Control of Investment, Level of Employment and Stockkeeping", in A. Bachem, U. Derigs, M. Jünger, R. Schrader (Eds.): *Operations Research 93*, Heidelberg, Physica Verlag, 1994, pp. 60-63
- [16] Bronstein, I.N., Semendjajew, K.A., *Taschenbuch der Mathematik*, 25. Auflage, B.G. Teubner Verlagsgesellschaft Stuttgart, Leipzig und Verlag Nauka, Moskau, 1991 (in German)
- [17] Brown, K.R., Harrold, E.F., Johnson, G.W., "Rapid Optimization of Multiple-Burn Rocket Flights", IBM/NASA, NASA Contractor Report CR-1430, Marshall Space Flight Center, AL, September 1969
- [18] Brown, K.R., Johnson, G.W., "Rapid Computation of Optimal Trajectories", *IBM Journal*, July 1967
- [19] Brown, K.R., Johnson, G.W., "Real-Time Optimal Guidance", *IEEE Transactions on Automatic Control*, Vol. AC-12, No. 5, October 1967, pp. 501-506
- [20] Brusch, R.G., Schappelle, R.H., "Solution of Highly Constrained Optimal Control Problems Using Nonlinear Programming", AIAA 70-964, AIAA Guidance, Control and Flight Mechanics Conference, Santa Barbara, CA, 17-19 August 1970
- [21] Brusch, R.G., Peltier, J.P., "Gradient Generation for Parametric Control Models", *Acta Astronautica*, Vol. 1, pp. 1453-1466, 1974
- [22] Brusch, R.G., Peltier, J.P., "Parametric Control Models and Gradient Generation by Impulsive Response", XXIV Congress of the International Astronautical Federation, Baku, USSR, 8-13 October 1973
- [23] Bryson, A.E., Jr., Ho, Y.-C., *Applied Optimal Control*, Revised Printing, Hemisphere Publishing Corporation, 1975
- [24] Büskens, Ch., "Optimierungsmethoden und Sensitivitätsanalyse für optimale Steuerprozesse mit Steuer- und Zustands-Beschränkungen", Dissertation, Institute for Numerical Mathematics, University of Münster, Germany, 1998

- [25] Büskens, Ch., Maurer, H., “SQP-Methods for Solving Optimal Control Problems with Control and State Constraints: Adjoint Variables, Sensitivity Analysis and Real-Time Control”, in: *Journal of Computational and Applied Mathematics on SQP-based Direct Discretization Methods for Practical Optimal Control problems*, Vol. 120, 2000, pp. 85-108
- [26] Calise, A.J., Leung, M.S.K., “Hybrid Approach to Near-Optimal Launch Vehicle Guidance”, *Journal of Guidance, Control, and Dynamics*, Vol. 17, No. 5, September-October 1994, pp. 881-888
- [27] Calise, A.J., Leung, M.S.K., “Hybrid Approach to Solution of Optimal Control Problems”, *Journal of Guidance, Control, and Dynamics*, Vol. 17, No. 5, September-October 1994, pp. 966-974
- [28] Calise, A.J., Melamed, N., Lee, S., “Design and Evaluation of a Three-Dimensional Optimal Ascent Guidance Algorithm”, *Journal of Guidance, Control, and Dynamics*, Vol. 21, No. 6, November-December 1998, pp. 867-875
- [29] Calise, A.J., Tandon, S., Young, D.H., Kim, S., “Further Improvements to a Hybrid Method for Launch Vehicle Ascent Trajectory Optimization”, AIAA-2000-4261, AIAA Guidance, Navigation, and Control Conference, 14-17 August 2000
- [30] Chobotov, V.A., et al., *Orbital Mechanics*, Second Edition, AIAA Education Series, American Institute of Aeronautics and Astronautics, Inc., Reston, Virginia, 1996
- [31] Corban, J.E., Calise, A.J., Flandro, G.A., “Rapid Near-Optimal Aerospace Plane Trajectory Generation and Guidance”, *Journal of Guidance, Control, and Dynamics*, Vol. 14, No. 6, November-December 1991, pp. 1181-1190
- [32] Dickmanns, E.D., Well, K.H., “Parametrization of Optimal Control Problems Using Piecewise Polynomial Approximation”, AIAA 74-822, AIAA Mechanics and Control of Flight Conference, 5-9 August, Anaheim, California, USA, 1974
- [33] Gath, P.F., “Improvements to a Hybrid Algorithm for Rapid Generation of 3-D Optimal Launch Vehicle Ascent Trajectories”, Institute of Flight Mechanics and Control, Diplomarbeit IFR_SR_98_016, University of Stuttgart, December 1998
- [34] Gath, P.F., Well, K.H., “HISTOS Technical Report 1”, TR 99-002, Institute of Flight Mechanics and Control, University of Stuttgart, June 1999
- [35] Gath, P.F., Well, K.H., “HISTOS Technical Report 2 - Direct Method”, HISTOS TR 00-002, Institute of Flight Mechanics and Control, University of Stuttgart, June 2000
- [36] Gath, P.F., Wiegand, A., Markl, A., Well, K.H., “Recent Improvements in the Trajectory Optimization Software ASTOS”, FORTWIHR 2001 Conference, Erlangen-Nürnberg, 12-14 March 2001 (proceedings to appear)
- [37] Gath, P.F., Calise, A.J., “Optimization of Launch Vehicle Ascent Trajectories with Path Constraints and Coast Arcs”, *Journal of Guidance, Control, and Dynamics*, Vol. 24, No. 2, March-April 2001, pp. 296-304

- [38] Gath, P.F., Well, K.H., "Trajectory Optimization Using a Combination of Direct Multiple Shooting and Collocation", AIAA 2001-4047, AIAA Guidance, Navigation, and Control Conference, Montréal, Québec, Canada, 6-9 August 2001
- [39] Gath, P.F., Well, K.H., Mehlem, K., "Automatic Initial Guess Generation for Ariane 5 Dual Payload Ascent Trajectory Optimization", AIAA 2000-4589, AIAA Guidance, Navigation, and Control Conference, Denver, CO, 14-17 August 2000
- [40] GESOP Software User Manual - The User Interface Tutorial, GESOP SUM 4.3, Institute of Flight Mechanics and Control, University of Stuttgart, Germany, April 2001
- [41] Gill, P.E., Murray, W., Wright, M.H., *Practical Optimization*, Academic Press, Inc., San Diego, CA, 1988
- [42] Gill, P.E., Murray, W., Saunders, M.A., *User's Guide for SNOPT 5.3: A FORTRAN Package for Large-Scale Nonlinear Programming*, Report NA 97-5, Department of Mathematics, University of California, San Diego, February 1999
- [43] Goodson, T., Chuang, J.C.H., Hanson, J., "Optimal Finite Thrust Orbit Transfers with Large Number of Burns", *Journal of Guidance, Control, and Dynamics*, Vol. 22, No. 1, January-February 1999, pp. 139-148
- [44] Griffin, M.D., French, J.R., *Space Vehicle Design*, AIAA Education Series, American Institute of Aeronautics and Astronautics, Inc., Washington DC, 1991
- [45] Hairer, E., Norsett, S.P., Wanner, G., *Solving Ordinary Differential Equations, Nonstiff Problems*, Springer Verlag, Berlin, Heidelberg, 1987
- [46] Hull, D.G., "Application of Parameter Optimization Methods to Trajectory Optimization", AIAA 74-825, AIAA Mechanics and Control of Flight Conference, Anaheim, CA, 5-9 August 1974
- [47] Hanson, J.M., Dukeman, G.A., "Optimization of Many-Burn Orbital Transfers", *The Journal of the Astronautical Sciences*, Vol. 45, No. 1, January-March 1997, pp. 1-40
- [48] Hargraves, C.R., Paris, S.W., "Direct Trajectory Optimization Using Nonlinear Programming and Collocation", *Journal of Guidance, Control, and Dynamics*, Vol. 10, No. 4, July-August 1987, pp. 338-342
- [49] Horn, M.K., Solution of the optimal control problem using the software package STOMP, 8th IFAC Workshop on Control Applications of Nonlinear Programming and Optimization, Paris, June 7-9, 1989 (cited in Ref. [76])
- [50] Jänsch, C., Schnepfer, K., Well, K.H., "Multiphase Trajectory Optimization Methods with Applications to Hypersonic Vehicles", *Applied Mathematics in Aerospace Science and Engineering*, Miele, A., Salvetti, A., ed., Plenum Press, New York, Chapter 8, 1994
- [51] Jänsch, C., Schnepfer, K., Well, K.H., Kraft, D., "ALTOS - Technical Report TR1, Survey Paper on Trajectory Optimization Methods", Deutsche Forschungsanstalt für Luft- und Raumfahrt, Institute for Flight System Dynamics, and Fachhochschule München, Fachbereich Maschinenbau und Fahrzeugtechnik, March 1990

- [52] Kindler, J.T., Schöttle, U.M., Well, K.H., “Entry Interface Window of Landing Site Coober Pedy for the Experimental Vehicle X-38 V201”, AIAA 2000-4117, AIAA Atmospheric Flight Mechanics Conference, Denver, CO, 14-17 August 2000
- [53] Koslik, B., Breitner, M.H., “An Optimal Control Problem in Economics with Four Linear Controls”, *Journal of Optimization Theory and Applications*, Vol. 94, No. 3, 1997, pp. 619-634
- [54] Kraft, D., “On converting optimal control problems into nonlinear programming problems”, in NATO ASI Series, Vol. F15, *Computational Mathematical Programming*, ed. by K. Schittkowski, Springer Verlag, 1985, pp. 261-280 (cited in Ref. [76])
- [55] Kraft, D., *TOMP - FORTRAN Modules for Optimal Control Calculations*, Fortschritt-Berichte VDI, Reihe 8, Nr. 254, 1991 (cited in Ref. [76])
- [56] Lawden, D.F., *Optimal Trajectories for Space Navigation*, Butterworth Mathematical Texts, Butterworth, London, 1963
- [57] Markl, A.W., “An Initial Guess Generator for Launch and Reentry Vehicle Trajectory Optimization”, Dissertation, Institute of Flight Mechanics and Control, University of Stuttgart, June 2001
- [58] Nucera, B., “Special Optimization Problems on Capsule Reentry”, Diplomarbeit, Institute of Flight Mechanics and Control, University of Stuttgart, Germany, 1999
- [59] Oberle, H.J., Grimm, W., “BNDSCO - A Program for the Numerical Solution of Optimal Control Problems”, DLR IB 515-89/22, July 1989
- [60] Paris, S., Nelson, R., “Trajectory Optimization at the Boeing Phantom Works”, Workshop on Trajectory Optimization Methods and Applications, AIAA Atmospheric Flight Mechanics Conference, Boston, MA, 10 August 1998
- [61] Paus, M., “Ein numerisches Verfahren für die optimale Echtzeitsteuerung von Luft- und Raumfahrzeugen”, Dissertation, Institute of Flight Mechanics and Control, University of Stuttgart, 1996
- [62] Paus, M., Grimm, W., Well, K.H., “Real-Time Optimization for the Guidance of Dynamic Systems”, 10th IFAC Workshop on Control Applications of Optimization, Haifa, Israel, 19-21 December, 1995 (paper prepared; proceedings did not appear). Institute of Flight Mechanics and Control, University of Stuttgart, Pfaffenwaldring 7a, 70550 Stuttgart, Germany
- [63] Pesch, H.J., Bulirsch, R., The Maximum Principle, Bellman’s Equation, and Carathéodory’s Work, Report No. 396, Mathematical Institute, Technical University of Munich, Germany, 1992
- [64] Pesch, H.J., “A practical guide to the solution of real-life optimal control problems”, *Control and Cybernetics*, Vol. 23, No. 1/2, 1994
- [65] Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P., *Numerical Recipes in C, The Art of Scientific Computing*, Second Edition, Cambridge University Press, New York, 1992

- [66] Raif, M., "Bahnoptimierung eines Atmosphärischen Transfervehikels", Studienarbeit, IFR_SR_00_031, January 2001 (in German)
- [67] Renes, J.J., "On the Use of Splines and Collocation in a Trajectory Optimization Algorithm Based on Mathematical Programming", National Aerospace Lab., Amsterdam, the Netherlands, NLR-TR-78016 U, September 1978
- [68] Roenneke, A.J., "Ein bordautonomes Verfahren zur Flugführung und Regelung von Rückkehrfahrzeugen", Dissertation, Institute of Flight Mechanics and Control, University of Stuttgart, Germany, May 2001 (in German)
- [69] Ruby, V., "Numerische Optimierung verfahrenstechnischer Prozesse hoher Ordnung", Fortschrittsbericht der VDI, Reihe 8, Nr. 31, May 1980 (in German)
- [70] Schnepfer, K., "ALTOS - Architectural Design Document - PROMIS: Optimization Program", Institute of Flight System Dynamics, DLR, Oberpfaffenhofen, Germany, ESA Contract No. 8046/88/NL/MAC, 1992
- [71] Seywald, H., Kumar, R.R., "Method for Automatic Costate Calculation", *Journal of Guidance, Control, and Dynamics*, Vol. 19, No. 6, November-December 1996, pp. 1252-1261
- [72] Shampine, L.F., Reichelt, M.W., "The MATLAB ODE Suite", The MathWorks, Inc., 1994
- [73] Spägle, Th., "Modellierung, Simulation und Optimierung menschlicher Bewegungen", Dissertation, Institute A for Mechanics, University of Stuttgart, 1998
- [74] Stoer, J., Bulirsch, R., *Numerische Mathematik 2*, Third Edition, Springer-Verlag, New York, 1990 (in German)
- [75] Stryk, O.v., "Numerical Solution of Optimal Control Problems by Direct Collocation", *Optimal Control - Calculus of Variations*, Optimal Control Theory and Numerical Methods, International Series of Numerical Mathematics 111, 1993, pp. 129-143
- [76] Stryk, O.v., "Numerische Lösung optimaler Steuerungsprobleme: Diskretisierung, Parameteroptimierung und Berechnung der adjungierten Variablen", Fortschritt-Berichte VDI, Reihe 8, Nr. 441, VDI-Verlag, Düsseldorf, 1995
- [77] Stryk, O.v., Schlemmer, M., "Optimal Control of the Industrial Robot Manutec r3", *Computational Optimal Control*, International Series of Numerical Mathematics 115, 1994, pp. 367-382
- [78] Thorne, J.D., Hall, C.D., "Approximate Initial Lagrange Costates for Continuous-Thrust Spacecraft", *Journal of Guidance, Control, and Dynamics*, Vol. 19, No. 2, March-April 1996, pp. 283-288
- [79] Wah, B.W., Wang, T., "Constrained Simulated Annealing with Applications in Nonlinear Continuous Constrained Global Optimization", *Proceedings of the 11th IEEE International Conference on Tools with Artificial Intelligence*, November 1999, pp. 381-388

- [80] Wah, B.W., Wang, T., "Tuning Strategies in Constrained Simulated Annealing for Non-linear Global Optimization", *International Journal on Artificial Intelligence Tools*, Vol. 9, No. 1, June 2000, 3-25
- [81] Wah, B.W., Chen, Y., "Hybrid Constrained Simulated Annealing and Genetic Algorithms for Nonlinear Constrained Optimization", *Proceedings of the IEEE Congress on Evolutionary Computation*, May 2001
- [82] Wah, B.W., Chen, Y., "Constrained Genetic Algorithms and their Applications in Non-linear Constrained Optimization", in Yao, X., Starker, R. (ed.), *Evolutionary Computation*, Kluwer Academic Publishers, 2001
- [83] Wah, B.W., Chen, Y., "Optimal Anytime Constrained Simulated Annealing for Constrained Global Optimization", *Proceedings of the 6th International Conference on Principles and Practice of Constraint Programming*, Springer-Verlag, September 2000
- [84] Weigel, N., Well, K.H., "Dual Payload Ascent Trajectory Optimization with a Splash-Down Constraint", *Journal of Guidance, Control, and Dynamics*, Vol. 23, No. 1, January-February 2000, pp. 45-52
- [85] Weigel, N., "Dual Payload Trajectories for the ARIANE5 Launch Vehicle", Studienarbeit, IFR_SR_95_004, Institute of Flight Mechanics and Control, University of Stuttgart, Germany, 1995
- [86] Well, K.H., "Neighboring Vehicle Design for a Two-Stage Launch-Vehicle", accepted for publication in *Applied Mathematics in Aerospace Science and Engineering* (Ed.: A. Miele), Plenum Publishing Corporation, New York, 2000
- [87] Wiegand, A., Markl, A., Well, K.H., Mehlem, K., Ortega, G., Steinkopf, M., "ALTOS - ESA's Trajectory Optimization Tool Applied to Reentry Vehicle Trajectory Design", IAF-99-A6.09, 50th International Astronautical Congress, Amsterdam, The Netherlands, 4-8 October 1999
- [88] Windhorst, R., Ardema, M., Kinney, D., "Fixed-Range Optimal Trajectories of Supersonic Aircraft by First-Order Expansions", *Journal of Guidance, Control, and Dynamics*, Vol. 24, No. 4, July-August 2001, pp. 700-709
- [89] Zaun, H., "Europa startet 2015 verwendbaren Raumtransporter ins All", Telepolis - Magazin der Netzkultur, Verlag Heinz Heise, Hannover, 2001, <http://www.heise.de/tp/deutsch/special/raum/7317/1.html>

A Integration Methods for Direct Multiple Shooting

This appendix describes the integration methods that are implemented for the direct multiple shooting method.

A.1 Runge-Kutta 2/3 Method

A lower-order integration method is implemented for the fast computation of approximate results to trajectory optimization problems. It is also used for a performance comparison with the Paus integration algorithm presented in Section A.3.

One integration step of size h involves three evaluations of the dynamics.

$$\mathbf{k}_1 = h \cdot \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{p}, t) \quad (\text{A.1})$$

$$\mathbf{k}_2 = h \cdot \mathbf{f}\left(\mathbf{x} + \frac{1}{2}\mathbf{k}_1, \mathbf{u} + \frac{h}{2}\dot{\mathbf{u}}, \mathbf{p}, t + \frac{h}{2}\right)$$

$$\mathbf{k}_3 = h \cdot \mathbf{f}\left(\mathbf{x} + \frac{3}{4}\mathbf{k}_1, \mathbf{u} + \frac{3h}{4}\dot{\mathbf{u}}, \mathbf{p}, t + \frac{3h}{4}\right)$$

The propagated state at $t + h$ is computed as

$$\mathbf{x}(t + h) = \mathbf{x} + \frac{1}{9}(2\mathbf{k}_1 + 3\mathbf{k}_2 + 4\mathbf{k}_3). \quad (\text{A.2})$$

In order to get an error estimate for the current integration step, an additional evaluation of the dynamics is performed at $\mathbf{x}(t+h)$:

$$\mathbf{k}_4 = h \cdot \mathbf{f}(\mathbf{x}(t + h), \mathbf{u} + h\dot{\mathbf{u}}, \mathbf{p}, t + h) \quad (\text{A.3})$$

An error approximation for $\mathbf{x}(t+h)$ is calculated as

$$\mathbf{e} = -\frac{5}{72}\mathbf{k}_1 + \frac{1}{12}\mathbf{k}_2 + \frac{1}{9}\mathbf{k}_3 - \frac{1}{8}\mathbf{k}_4. \quad (\text{A.4})$$

The error is now normalized for each component such that

$$\bar{e}_i = \frac{e_i}{d} \quad (\text{A.5})$$

with

$$d = \max \left\{ |x_i|, |x_i(t+h)|, \frac{tol_{abs}}{tol_{rel}}, \frac{2\varepsilon_m}{tol_{rel}} \right\}, \quad (\text{A.6})$$

where ε_m is the machine accuracy, representing the smallest value which can be added to one (see Ref. [65] Section 1.3 and 20.1). tol_{abs} and tol_{rel} are the required absolute and relative integration tolerances.

The error in the current integration step is then calculated as

$$\varepsilon = \frac{1}{n} \sum_{i=1}^n |\bar{e}_i|. \quad (\text{A.7})$$

If ε is less or equal to tol_{rel} , the current step is accepted, the stepsize for the next integration step is estimated by calculating

$$s = 1.25 \cdot \left(\frac{\varepsilon}{tol_{rel}} \right)^{1/3}. \quad (\text{A.8})$$

The factor s is modified if it meets one of the following conditions:

1. If $s > 5$ then s is set to 5.
2. If $s < 1/10$ then s is set to $1/10$.

The new stepsize is then calculated as

$$h_{new} = \frac{h}{s}. \quad (\text{A.9})$$

In case the error in the current integration step is larger than tol_{rel} , the step is repeated with the stepsize (A.8), (A.9). This applies to the first failure. Otherwise, s is simply set to two and the integration step is repeated again until it meets the error criterion. In case h becomes less than eps , an integration error is raised. As an initial estimate for h in the first integration step, the formula

$$r_h = \frac{\max_{1 \leq i \leq n} \frac{f_i(\mathbf{x}, \mathbf{u}, \mathbf{p}, t)}{\max(|x_{0i}|, tol_{abs}/tol_{rel})}}{\frac{8}{10} tol_{rel}^p} \quad (\text{A.10})$$

where $p = 1/3$ and k_1 is computed with the first equation in Eq. (A.1). The initial step-size is now estimated as

$$h = \frac{1}{r_h} \text{ if } (t_f - t_0)r_h > 1, \text{ and} \quad (\text{A.11})$$

$$h = (t_f - t_0) \text{ if } (t_f - t_0)r_h \leq 1. \quad (\text{A.12})$$

This heuristic is based on Ref. [45], p. 182.

At the beginning of each integration step it is checked if the final time t_f can be reached if the current step-length is increased by a maximum of 10%. This often increases the numerical stability of the algorithm and also leads to an increase in performance since very small final steps are avoided and several evaluations of the dynamics can be omitted.

For calculating the perturbed trajectories, only Eqs. (A.1) are used in conjunction with the previously generated step-size sequence.

A.2 Runge-Kutta 4/5 Method

The implementation of the Runge-Kutta 4/5 method is based on Ref. [72] and the integration method implemented in PROMIS. It is very similar to the Runge-Kutta 2/3 method presented in Section A.1. However, a slightly different error estimator is used. For the sake of completeness, the methods coefficients are given below.

$$k_1 = h \cdot f(\mathbf{x}, \mathbf{u}, \mathbf{p}, t) \quad (\text{A.13})$$

$$k_2 = h \cdot f\left(\mathbf{x} + \frac{1}{5}k_1, \mathbf{u} + \frac{h}{5}\dot{\mathbf{u}}, \mathbf{p}, t + \frac{h}{5}\right)$$

$$k_3 = h \cdot f\left(\mathbf{x} + \frac{3}{40}k_1 + \frac{9}{40}k_2, \mathbf{u} + \frac{3h}{10}\dot{\mathbf{u}}, \mathbf{p}, t + \frac{3h}{10}\right)$$

$$k_4 = h \cdot f\left(\mathbf{x} + \frac{44}{45}k_1 - \frac{56}{15}k_2 + \frac{32}{9}k_3, \mathbf{u} + \frac{8h}{10}\dot{\mathbf{u}}, \mathbf{p}, t + \frac{8h}{10}\right)$$

$$k_5 = h \cdot f\left(\mathbf{x} + \frac{19372}{6561}k_1 - \frac{25360}{2187}k_2 + \frac{64448}{6561}k_3 - \frac{212}{729}k_4, \mathbf{u} + \frac{8h}{9}\dot{\mathbf{u}}, \mathbf{p}, t + \frac{8h}{9}\right)$$

$$k_6 = h \cdot f\left(\mathbf{x} + \frac{9017}{3168}k_1 - \frac{355}{33}k_2 + \frac{46732}{5247}k_3 + \frac{49}{176}k_4 - \frac{5103}{18656}k_5, \mathbf{u} + h\dot{\mathbf{u}}, \mathbf{p}, t + h\right)$$

The propagated state at $t + h$ is computed as

$$\mathbf{x}(t + h) = \mathbf{x} + \frac{35}{384}\mathbf{k}_1 + \frac{500}{1113}\mathbf{k}_3 + \frac{125}{192}\mathbf{k}_4 - \frac{2187}{6784}\mathbf{k}_5 + \frac{11}{84}\mathbf{k}_6. \quad (\text{A.14})$$

In order to get an error estimate for the current integration step, an additional evaluation of the dynamics is performed at $\mathbf{x}(t+h)$

$$\mathbf{k}_7 = h \cdot \mathbf{f}(\mathbf{x}(t + h), \mathbf{u} + h\dot{\mathbf{u}}, \mathbf{p}, t + h). \quad (\text{A.15})$$

and an error approximation for $\mathbf{x}(t+h)$ is calculated as

$$\mathbf{e} = \frac{71}{57600}\mathbf{k}_1 - \frac{71}{16695}\mathbf{k}_3 + \frac{71}{1920}\mathbf{k}_4 - \frac{17253}{339200}\mathbf{k}_5 + \frac{22}{525}\mathbf{k}_6 - \frac{1}{40}\mathbf{k}_7. \quad (\text{A.16})$$

Again, the error estimate is normalized for each component based on Eqs. (A.5) and (A.6). The error in the current integration step is then calculated as

$$\varepsilon = \sqrt{\frac{1}{n}\mathbf{e}^T \mathbf{e}}. \quad (\text{A.17})$$

If ε is less or equal to tol_{rel} , the current step is accepted. The stepsize for the next integration step is estimated by calculating

$$s = 1.25 \cdot \left(\frac{\varepsilon}{tol_{rel}} \right)^{1/5}. \quad (\text{A.18})$$

The factor s is modified in the same manner as for the Runge-Kutta 2/3 method in Section A.1 and the same logic for a step-size update or correction is applied. The same applies for estimating an initial step-size, only the factor ρ in Eq. (A.10) is replaced with $1/5$.

For calculating the perturbed trajectories, only Eqs. (A.13) are used in conjunction with the previously generated step-size sequence.

A.3 Integration Method of Paus

In Ref. [61], Paus presents an integration method which is especially suitable for applications in the framework of trajectory optimization. The beauty of this method is that the computation of gradients does not involve any evaluation of the dynamics, but only some matrix computations. A brief summary of this method is presented in Ref. [62].

The method was originally only applied in the framework of optimal guidance, i.e. that updated trajectories were computed based on an already available, almost optimal solution. For CAMTOS, this method is generalized and a new step-size control algorithm is developed which significantly improves the performance of the algorithm by a factor of about two compared to the approach suggested in Ref. [62]. The new step-size control algorithm is mainly

based on the strategy used for the Runge-Kutta 2/3 method, because the Paus algorithm is a second order integration method as well.

Based on Refs. [61] and [62], the integration algorithm is extended in order to allow for optimizable phase times as well. The nonlinear dynamics

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{p}, t_i, t_f, t), \quad (\text{A.19})$$

where t_i and t_f are the initial and final phase time, respectively, is linearized about $(\mathbf{x}, \mathbf{u}, \mathbf{p}, t_i, t_f)_0$:

$$\mathbf{A} = \left. \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right|_0, \mathbf{B} = \left. \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \right|_0, \mathbf{Q} = \left. \frac{\partial \mathbf{f}}{\partial \mathbf{p}} \right|_0, \mathbf{T}_i = \left. \frac{\partial \mathbf{f}}{\partial t_i} \right|_0, \mathbf{T}_f = \left. \frac{\partial \mathbf{f}}{\partial t_f} \right|_0 \quad (\text{A.20})$$

A linearization of Eq. (A.19) about a reference point yields:

$$\begin{aligned} \dot{\mathbf{x}} = & \mathbf{f}(\mathbf{x}_0, \mathbf{u}_0, \mathbf{p}_0, t_{i0}, t_{f0}) \\ & + \mathbf{A}(\mathbf{x} - \mathbf{x}_0) + \mathbf{B}(\mathbf{u} - \mathbf{u}_0) + \mathbf{Q}(\mathbf{p} - \mathbf{p}_0) + \mathbf{T}_i(t_i - t_{i0}) + \mathbf{T}_f(t_f - t_{f0}) \end{aligned} \quad (\text{A.21})$$

Using the definitions

$$\Delta \mathbf{x} = \mathbf{x} - \mathbf{x}_0, \Delta \mathbf{u} = \mathbf{u} - \mathbf{u}_0, \Delta \mathbf{p} = \mathbf{p} - \mathbf{p}_0, \Delta t_i = t_i - t_{i0}, \Delta t_f = t_f - t_{f0}, \quad (\text{A.22})$$

the linear system in Eq. (A.21) can be integrated using the assumption of a linear control:

$$\begin{aligned} \mathbf{x}(t+h) = & \mathbf{x} + \int_0^h e^{A(h-\tau)} d\tau \cdot \mathbf{f}(\mathbf{x}_0, \mathbf{u}_0, \mathbf{p}_0, t_{i0}, t_{f0}) + e^{Ah} \Delta \mathbf{x} \\ & + \left[\int_0^h e^{A(h-\tau)} d\tau - \frac{1}{h} \int_0^h \tau e^{A(h-\tau)} d\tau \right] \mathbf{B} \Delta \mathbf{u}(t_0) \\ & + \frac{1}{h} \int_0^h \tau e^{A(h-\tau)} d\tau \cdot \mathbf{B} \Delta \mathbf{u}(t_0 + h) \\ & + \int_0^h e^{A(h-\tau)} d\tau \cdot \mathbf{Q} \Delta \mathbf{p} \\ & + \int_0^h e^{A(h-\tau)} d\tau \cdot \mathbf{T}_i \Delta t_i + \int_0^h e^{A(h-\tau)} d\tau \cdot \mathbf{T}_f \Delta t_f \end{aligned} \quad (\text{A.23})$$

Note that $\Delta \mathbf{u}(t_0 + h) = \mathbf{u}(t_0 + h) - \mathbf{u}_0(t_0)$ becomes $\Delta \mathbf{u}(t_0)$ for a piecewise constant control and cancels out with the second term in line two of Eq. (A.23). Also note that the last three terms in Eq. (A.23) only need to be evaluated if optimizable real parameters or phase times are present.

As shown in Refs. [61] and [62], the integrals in Eq. (A.23) can be approximated by using a Taylor series expansion. After applying some additional algebraic relations, it can easily be shown that the following equalities hold:

$$F_2 = \int_0^h \tau e^{A(h-\tau)} d\tau = \frac{h^2}{2!} I_n + \frac{h^3}{3!} A + \frac{h^4}{4!} A^2 + \dots \quad (\text{A.24})$$

$$F_1 = \int_0^h e^{A(h-\tau)} d\tau = AF_2 + hI_n \quad (\text{A.25})$$

$$F_0 = e^{Ah} = AF_1 + I_n \quad (\text{A.26})$$

I_n is an $n \times n$ unity matrix. It is shown in Ref. [62] that the power series only need to be evaluated up to the second term. Evaluating additional terms does not decrease the local discretization error. Using these definitions, Eq. (A.23) can be rewritten as

$$\begin{aligned} x(t+h) &= x + F_0 \Delta x + F_1 f(x_0, u_0, p_0, t_{i0}, t_{f0}) \\ &+ \left[F_1 - \frac{1}{h} F_2 \right] B \Delta u(t_0) + \frac{F_2}{h} B \Delta u(t_0 + h) \\ &+ F_1 Q \Delta p + F_1 T_i \Delta t_i + F_1 T_f \Delta t_f \end{aligned} \quad (\text{A.27})$$

Note that once the system matrices (Eqs. (A.20)) are evaluated, F_0 through F_2 can be computed and stored for a later use during the calculation of perturbed trajectories. In order to reduce the number of matrix operations, all constant matrix products in Eq. (A.27) should be calculated and stored during the generation of a new reference trajectory.

The basic procedure for calculating the local discretization error is taken from Ref. [62]. It requires the derivatives \dot{f}_x and \dot{f}_u . These derivatives are estimated by calculating finite differences:

$$\dot{f}_x \approx \frac{A(t+h) - A(t)}{h}, \quad \dot{f}_u \approx \frac{B(t+h) - B(t)}{h} \quad (\text{A.28})$$

Now, the local error can be estimated as

$$e = \frac{h}{6} [\dot{f}_x \cdot f(x_0, u_0, p_0, t_{i0}, t_{f0}) + \dot{f}_u \cdot \dot{u}]. \quad (\text{A.29})$$

The components of the local error estimate are normalized again using the same procedure described in Eqs. (A.5) and (A.6), and an overall error is computed with Eq. (A.17). An update or correction of the step-size h is performed as described in Section A.1.

It is essential for this algorithm to avoid unnecessary matrix operations and especially linearizations of the dynamics. Therefore, the matrix Q and the vectors T_i and T_f are only calculated once an integration step is successful.

In addition, matrix operations during the calculation of perturbed trajectories are only performed for those components in Eq. (A.27) where changes have occurred. If a parameter did not change compared to the reference trajectory, the corresponding term is completely omitted.

Experiments are performed in which the reference trajectory is not updated in each major iteration of the SQP method. An update of the corresponding matrices is only performed when the deviation from the previously generated reference trajectory is larger than a certain trigger value. However, this method does not improve the computational performance of the overall algorithm, since a matrix update is triggered for each major iteration when small trigger values (e.g. 10^{-6}) are used, while larger trigger values (e.g. 10^{-3}) significantly degrade the accuracy of the trajectory and cause serious convergence problems during the optimization process.

B Derivation of Important Derivatives

This appendix gives the derivation of several important derivatives which are required for the computation of transversality conditions in the framework of the indirect optimization method. The derivations of the transversality conditions are given in Appendix C.

B.1 Derivatives of Inclination w.r.t. Radius and Velocity Vectors

The orbit inclination can be calculated based on the current inertial radius and velocity vectors as

$$\cos i = \frac{\mathbf{i}_N^T (\mathbf{R} \times \mathbf{V})}{|\mathbf{R} \times \mathbf{V}|}. \quad (\text{B.1})$$

The derivative w.r.t. \mathbf{V} can now be calculated as

$$\begin{aligned} \left(\frac{\partial \cos i}{\partial \mathbf{V}} \right)^T &= \frac{\mathbf{i}_N^T \times \mathbf{R}}{|\mathbf{R} \times \mathbf{V}|} + \frac{\mathbf{i}_N^T (\mathbf{R} \times \mathbf{V})}{|\mathbf{R} \times \mathbf{V}|^3} \cdot \mathbf{R} \times (\mathbf{R} \times \mathbf{V}) \\ &= \frac{r}{|\mathbf{R} \times \mathbf{V}|} \cdot \left[\mathbf{i}_N^T (\mathbf{i}_R \times \mathbf{i}_V) \right] \cdot \mathbf{i}_R \times (\mathbf{i}_R \times \mathbf{i}_V) - \frac{\mathbf{R} \times \mathbf{i}_N}{|\mathbf{R} \times \mathbf{V}|} \end{aligned} \quad (\text{B.2})$$

Note that the second term in Eq. (B.2) has components along $\mathbf{i}_R \times \mathbf{i}_V$ and $\mathbf{i}_R \times (\mathbf{i}_R \times \mathbf{i}_V)$, and the first term in Eq. (B.2) is only a component along $\mathbf{i}_R \times (\mathbf{i}_R \times \mathbf{i}_V)$. It can now be shown that $\partial i / \partial \mathbf{V}$ only has a component along $\mathbf{i}_R \times \mathbf{i}_V$. That means, the projection of $\mathbf{R} \times \mathbf{i}_N / |\mathbf{R} \times \mathbf{V}|$ on $\mathbf{i}_R \times (\mathbf{i}_R \times \mathbf{i}_V)$ has to cancel out the first term in Eq. (B.2). The projection can be calculated as

$$\frac{(\mathbf{R} \times \mathbf{i}_N)^T}{|\mathbf{R} \times \mathbf{V}|} \cdot \mathbf{i}_R \times (\mathbf{i}_R \times \mathbf{i}_V) = \frac{r}{|\mathbf{R} \times \mathbf{V}|} \cdot (\mathbf{i}_R \times \mathbf{i}_N)^T [\mathbf{i}_R \times (\mathbf{i}_R \times \mathbf{i}_V)]. \quad (\text{B.3})$$

It can clearly be seen that the scalar term $r/|\mathbf{R} \times \mathbf{V}|$ can be canceled out immediately. It only remains to be shown that

$$(\mathbf{i}_R \times \mathbf{i}_N)^T [\mathbf{i}_R \times (\mathbf{i}_R \times \mathbf{i}_V)] = \mathbf{i}_N^T (\mathbf{i}_R \times \mathbf{i}_V). \quad (\text{B.4})$$

Ref. [16], p. 556 shows that $\mathbf{a} \times (\mathbf{b} \times \mathbf{c}) = \mathbf{b}(\mathbf{a}^T \mathbf{c}) - \mathbf{c}(\mathbf{a}^T \mathbf{b})$, so the left side of Eq. (B.4) can be rewritten as

$$(\mathbf{i}_R \times \mathbf{i}_N)^T \cdot [\mathbf{i}_R(\mathbf{i}_R^T \mathbf{i}_V) - \mathbf{i}_V(\mathbf{i}_R^T \mathbf{i}_R)] = -(\mathbf{i}_R \times \mathbf{i}_N)^T \mathbf{i}_V \quad (\text{B.5})$$

Note that $(\mathbf{i}_R \times \mathbf{i}_N)$ is perpendicular to \mathbf{i}_R and therefore the first product on the left side in Eq. (B.5) is zero. Also note that $(\mathbf{i}_R^T \mathbf{i}_R) = 1$.

Ref. [16], p. 556 also shows that $(\mathbf{a} \times \mathbf{b})^T \mathbf{c} = -(\mathbf{a} \times \mathbf{c})^T \mathbf{b}$ which directly yields that

$$-(\mathbf{i}_R \times \mathbf{i}_N)^T \mathbf{i}_V = \mathbf{i}_N^T (\mathbf{i}_R \times \mathbf{i}_V) \text{ q.e.d.} \quad (\text{B.6})$$

Therefore, $\partial i / \partial \mathbf{V}$ can be computed as

$$\left(\frac{\partial \cos i}{\partial \mathbf{V}} \right)^T = \frac{(\mathbf{i}_N \times \mathbf{R})^T \mathbf{i}_f}{|\mathbf{R} \times \mathbf{V}|} \cdot \mathbf{i}_f \text{ where } \mathbf{i}_f = \frac{\mathbf{R} \times \mathbf{V}}{|\mathbf{R} \times \mathbf{V}|} \quad (\text{B.7})$$

A similar procedure can be applied to the derivative of inclination w.r.t. the radius vector \mathbf{R} , which finally yields

$$\left(\frac{\partial \cos i}{\partial \mathbf{R}} \right)^T = \frac{(\mathbf{V} \times \mathbf{i}_N)^T \mathbf{i}_f}{|\mathbf{R} \times \mathbf{V}|} \cdot \mathbf{i}_f. \quad (\text{B.8})$$

B.2 Derivatives of Orbital Momentum w.r.t. Radius and Velocity Vectors

The orbital momentum vector is defined as

$$\mathbf{h} = \mathbf{R} \times \mathbf{V}. \quad (\text{B.9})$$

The derivatives of \mathbf{h} w.r.t. \mathbf{R} and \mathbf{V} yields

$$\frac{\partial \mathbf{h}}{\partial \mathbf{R}} = -\tilde{\mathbf{V}} \text{ and } \frac{\partial \mathbf{h}}{\partial \mathbf{V}} = \tilde{\mathbf{R}} \quad (\text{B.10})$$

$$\text{where } \tilde{\mathbf{V}} = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix} \text{ and } \tilde{\mathbf{R}} = \begin{bmatrix} 0 & -r_3 & r_2 \\ r_3 & 0 & -r_1 \\ -r_2 & r_1 & 0 \end{bmatrix} \quad (\text{B.11})$$

The magnitude of the orbital momentum vector is

$$h = |\mathbf{R} \times \mathbf{V}| = \sqrt{(\mathbf{R} \times \mathbf{V})^T (\mathbf{R} \times \mathbf{V})}. \quad (\text{B.12})$$

Applying the chain rule during the construction of the derivatives of Eq. (B.12) w.r.t. \mathbf{R} and \mathbf{V} finally yields the following expressions:

$$\left(\frac{\partial h}{\partial \mathbf{R}}\right)^T = \frac{-\mathbf{V} \times (\mathbf{V} \times \mathbf{R})}{h} \text{ and } \left(\frac{\partial h}{\partial \mathbf{V}}\right)^T = \frac{-\mathbf{R} \times (\mathbf{R} \times \mathbf{V})}{h} \quad (\text{B.13})$$

B.3 Derivatives of Semimajor Axis w.r.t. Radius and Velocity Vectors

Based on the vis-viva equation, the semimajor axis a can be calculated as

$$a = \frac{1}{\frac{2}{r} - \frac{v^2}{\mu}} \quad (\text{B.14})$$

The derivatives of a w.r.t. \mathbf{R} and \mathbf{V} can now be calculated as

$$\left(\frac{\partial a}{\partial \mathbf{R}}\right)^T = \frac{2a^2}{r^2} \mathbf{i}_R \text{ and } \left(\frac{\partial a}{\partial \mathbf{V}}\right)^T = \frac{2a^2 v}{\mu} \mathbf{i}_V. \quad (\text{B.15})$$

B.4 Derivatives of Orbital Energy w.r.t. Radius and Velocity Vectors

Based on radius and velocity vector, the orbital energy can be computed as

$$\varepsilon = \frac{v^2}{2} - \frac{\mu}{r}. \quad (\text{B.16})$$

The derivatives of ε w.r.t. \mathbf{R} and \mathbf{V} can now be computed as

$$\left(\frac{\partial \varepsilon}{\partial \mathbf{R}}\right)^T = \frac{\mu}{r^2} \mathbf{i}_R \text{ and } \left(\frac{\partial \varepsilon}{\partial \mathbf{V}}\right)^T = \mathbf{V}. \quad (\text{B.17})$$

B.5 Derivatives of Eccentricity w.r.t. Radius and Velocity Vectors

Before calculating the derivatives of eccentricity e w.r.t. radius and velocity, an expression for eccentricity must be derived which depends on \mathbf{R} and \mathbf{V} . The orbital momentum can be calculated as

$$h = |\mathbf{R} \times \mathbf{V}| = r_p v_p \quad (\text{B.18})$$

where r_p is the perigee radius and v_p is the orbital velocity at perigee of the current orbit. Perigee radius can be calculated as (see Ref. [30])

$$r_p = a(1 - e). \quad (\text{B.19})$$

The orbital velocity at perigee is given by

$$v_p = \sqrt{\frac{\mu}{p}}(1 + e) \text{ where } p = \frac{h^2}{\mu}. \quad (\text{B.20})$$

Using Eqs. (B.19) and (B.20) in (B.18) and solving for e finally yields

$$e = \sqrt{1 - \frac{h^2}{\mu a}}. \quad (\text{B.21})$$

Note that e is always greater than zero by definition. Applying the chain rule and using the results from Sections B.2 and B.3 finally yields

$$\left(\frac{\partial e}{\partial \mathbf{R}}\right)^T = \frac{1}{e\mu} \left[\frac{h^2}{r^3} \mathbf{R} + \frac{\mathbf{V} \times (\mathbf{V} \times \mathbf{R})}{a} \right] \quad (\text{B.22})$$

$$\left(\frac{\partial e}{\partial \mathbf{V}}\right)^T = \frac{1}{e\mu} \left[\frac{h^2}{\mu} \mathbf{V} + \frac{\mathbf{R} \times (\mathbf{R} \times \mathbf{V})}{a} \right] \quad (\text{B.23})$$

Note that e appears in the denominator of the derivatives which means that these expressions must not be used in connection with circular or almost circular orbits.

B.6 Derivatives of Apogee and Perigee Radius w.r.t. Radius and Velocity Vectors

Apogee and perigee radius can be calculated as (Ref. [30])

$$r_a = a(1 + e) \text{ and} \quad (\text{B.24})$$

$$r_p = a(1 - e). \quad (\text{B.25})$$

Using the chain rule and the results from Section B.3 and B.5 yields

$$\left(\frac{\partial r_a}{\partial \mathbf{R}}\right)^T = \left[\frac{2a^2(1+e)}{r^2} + \frac{ah^2}{\mu e r^2} \right] \mathbf{i}_R + \frac{1}{\mu e} \mathbf{V} \times (\mathbf{V} \times \mathbf{R}) \quad (\text{B.26})$$

$$\left(\frac{\partial r_a}{\partial \mathbf{V}}\right)^T = \left[\frac{2a^2 v(1+e)}{\mu} + \frac{ah^2 v}{\mu^2 e} \right] \mathbf{i}_V + \frac{1}{\mu e} \mathbf{R} \times (\mathbf{R} \times \mathbf{V}) \quad (\text{B.27})$$

$$\left(\frac{\partial r_p}{\partial \mathbf{R}}\right)^T = \left[\frac{2a^2(1-e)}{r^2} - \frac{ah^2}{\mu e r^2} \right] \mathbf{i}_R - \frac{1}{\mu e} \mathbf{V} \times (\mathbf{V} \times \mathbf{R}) \quad (\text{B.28})$$

$$\left(\frac{\partial r_p}{\partial \mathbf{V}}\right)^T = \left[\frac{2a^2 v(1-e)}{\mu} - \frac{ah^2 v}{\mu^2 e} \right] \mathbf{i}_V - \frac{1}{\mu e} \mathbf{R} \times (\mathbf{R} \times \mathbf{V}) \quad (\text{B.29})$$

Note that e appears in the denominator of several terms which means that Eqs. (B.26) through (B.29) must not be used in connection with circular or almost circular orbits.

B.7 Derivatives of the Argument of Perigee w.r.t. Radius and Velocity Vectors

When the final position vector is located at perigee, the argument of perigee can be calculated as

$$\cos \omega = \frac{\mathbf{i}_R^T (\mathbf{i}_N \times \mathbf{i}_f)}{|\mathbf{i}_N \times \mathbf{i}_f|} \text{ for } 0^\circ < \omega < 180^\circ \text{ and} \quad (\text{B.30})$$

$$\sin \omega = \frac{\mathbf{i}_V^T (\mathbf{i}_N \times \mathbf{i}_f)}{|\mathbf{i}_N \times \mathbf{i}_f|} \text{ for } \omega \approx 0^\circ \text{ or } \omega \approx 180^\circ \quad (\text{B.31})$$

If an inclination constraint is satisfied at the same time, the following equality holds

$$\sin i = |\mathbf{i}_N \times \mathbf{i}_f|. \quad (\text{B.32})$$

Therefore, it becomes necessary to construct the derivatives of $\mathbf{i}_R^T (\mathbf{i}_N \times \mathbf{i}_f)$ and $\mathbf{i}_V^T (\mathbf{i}_N \times \mathbf{i}_f)$ w.r.t. \mathbf{R} and \mathbf{V} . Noting that

$$\frac{\partial |\mathbf{R} \times \mathbf{V}|}{\partial \mathbf{R}} = \frac{-\mathbf{V} \times (\mathbf{V} \times \mathbf{R})}{|\mathbf{R} \times \mathbf{V}|} \text{ and } \frac{\partial |\mathbf{R} \times \mathbf{V}|}{\partial \mathbf{V}} = \frac{-\mathbf{R} \times (\mathbf{R} \times \mathbf{V})}{|\mathbf{R} \times \mathbf{V}|}, \quad (\text{B.33})$$

the required derivatives of $(\mathbf{i}_N \times \mathbf{i}_f)$ can be calculated as

$$\frac{\partial (\mathbf{i}_N \times \mathbf{i}_f)}{\partial \mathbf{R}} = \mathbf{M}_R = \frac{-\tilde{\mathbf{i}}_N}{|\mathbf{R} \times \mathbf{V}|} [\mathbf{i}_f (\mathbf{V} \times \mathbf{i}_f)^T + \tilde{\mathbf{V}}] \text{ and} \quad (\text{B.34})$$

$$\frac{\partial (\mathbf{i}_N \times \mathbf{i}_f)}{\partial \mathbf{V}} = \mathbf{M}_V = \frac{\tilde{\mathbf{i}}_N}{|\mathbf{R} \times \mathbf{V}|} [\mathbf{i}_f (\mathbf{R} \times \mathbf{i}_f)^T + \tilde{\mathbf{R}}] \quad (\text{B.35})$$

where

$$\tilde{\mathbf{x}} = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}. \quad (\text{B.36})$$

Now, the derivatives of $\mathbf{i}_R^T (\mathbf{i}_N \times \mathbf{i}_f)$ can be calculated as

$$\left[\frac{\partial \mathbf{i}_R^T (\mathbf{i}_N \times \mathbf{i}_f)}{\partial \mathbf{R}} \right]^T = \mathbf{H}_R^T (\mathbf{i}_N \times \mathbf{i}_f) + \mathbf{M}_R^T \mathbf{i}_R \text{ and} \quad (\text{B.37})$$

$$\left[\frac{\partial \mathbf{i}_R^T (\mathbf{i}_N \times \mathbf{i}_f)}{\partial \mathbf{V}} \right]^T = \mathbf{M}_V^T \mathbf{i}_R \quad (\text{B.38})$$

where $\partial \mathbf{i}_R / \partial \mathbf{R} = \mathbf{H}_R = (\mathbf{I}_3 - \mathbf{i}_R \mathbf{i}_R^T) / r$. By evaluating the matrix-vector products, Eqs. (B.37) and (B.38) can be rewritten as

$$\left[\frac{\partial \mathbf{i}_R^T (\mathbf{i}_N \times \mathbf{i}_f)}{\partial \mathbf{R}} \right]^T = \frac{1}{r} \{ \mathbf{i}_N \times \mathbf{i}_f - [\mathbf{i}_R^T (\mathbf{i}_N \times \mathbf{i}_f)] \mathbf{i}_R \} \quad (\text{B.39})$$

$$+ \frac{v}{|\mathbf{R} \times \mathbf{V}|} \{ [\mathbf{i}_f^T (\mathbf{i}_N \times \mathbf{i}_R)] (\mathbf{i}_V \times \mathbf{i}_f) - \mathbf{i}_V \times (\mathbf{i}_N \times \mathbf{i}_R) \}$$

$$\text{and } \left[\frac{\partial \mathbf{i}_R^T (\mathbf{i}_N \times \mathbf{i}_f)}{\partial \mathbf{V}} \right]^T = \frac{r}{|\mathbf{R} \times \mathbf{V}|} \{ [\mathbf{i}_f^T (\mathbf{i}_N \times \mathbf{i}_R)] (\mathbf{i}_f \times \mathbf{i}_R) + \mathbf{i}_R \times (\mathbf{i}_N \times \mathbf{i}_R) \}. \quad (\text{B.40})$$

Because the derivation is performed at perigee (or apogee), radius and velocity vector are perpendicular to each other and Eqs. (B.39) and (B.40) can be rewritten as

$$\left[\frac{\partial \mathbf{i}_R^T (\mathbf{i}_N \times \mathbf{i}_f)}{\partial \mathbf{R}} \right]^T = \frac{\mathbf{i}_N \times \mathbf{i}_f + (\mathbf{i}_V^T \mathbf{i}_N) \mathbf{i}_R}{r} \quad (\text{B.41})$$

$$\text{and } \left[\frac{\partial \mathbf{i}_R^T (\mathbf{i}_N \times \mathbf{i}_f)}{\partial \mathbf{V}} \right]^T = \frac{\mathbf{i}_N - \mathbf{i}_R (\mathbf{i}_R^T \mathbf{i}_N) - [\mathbf{i}_f^T (\mathbf{i}_N \times \mathbf{i}_R)] \mathbf{i}_V}{v}. \quad (\text{B.42})$$

Note that Eq. (B.41) only has components in the $\mathbf{i}_R/\mathbf{i}_V$ plane while Eq. (B.42) also has a component along \mathbf{i}_N .

C Derivation of Transversality Conditions

This appendix gives the derivation of the transversality conditions which are required for the indirect optimization method. Note that all values are evaluated at final time.

C.1 Preliminaries

In the framework of the indirect optimization method presented, the transversality conditions can be formulated as

$$\mathbf{P} = \frac{\partial \phi}{\partial \mathbf{V}} + \sum_{i=1}^k v_i \left(\frac{\partial \psi_i}{\partial \mathbf{V}} \right)^T \quad (\text{C.1})$$

$$\mathbf{Q} = \frac{\partial \phi}{\partial \mathbf{R}} + \sum_{i=1}^k v_i \left(\frac{\partial \psi_i}{\partial \mathbf{R}} \right)^T \quad (\text{C.2})$$

$$\lambda_m = \frac{\partial \phi}{\partial m} + \sum_{i=1}^k v_i \left(\frac{\partial \psi_i}{\partial m} \right)^T \quad (\text{C.3})$$

where k is the number of terminal constraints. Note that all equations are evaluated at final time.

In order to reduce the number of unknown parameters, Eqs. (C.1) and (C.2) are multiplied with different vectors and the resulting scalar equations are used to eliminate the unknown parameters v_i .

C.2 Transversality Conditions for Circular Orbit with Free Argument of Perigee

The set of constraints for a circular orbit with a free argument of perigee is

$$\boldsymbol{\Psi} = \begin{bmatrix} \mathbf{i}_f^T \mathbf{i}_N - \cos i \\ r - r_p \\ v - v_p \\ \mathbf{R}^T \mathbf{V} \end{bmatrix} = \mathbf{0} \quad (\text{C.4})$$

Assuming that ϕ does not depend on \mathbf{R} or \mathbf{V} , the transversality conditions for \mathbf{P} and \mathbf{Q} can be written as

$$\mathbf{P} = v_1 \left[\frac{(\mathbf{i}_N \times \mathbf{i}_R)^T \mathbf{i}_f}{v} \right] \mathbf{i}_f + v_3 \mathbf{i}_V + v_4 \mathbf{R} \quad \text{and} \quad (\text{C.5})$$

$$\mathbf{Q} = -v_1 \left[\frac{(\mathbf{i}_N \times \mathbf{i}_V)^T \mathbf{i}_f}{r} \right] \mathbf{i}_f + v_2 \mathbf{i}_R + v_4 \mathbf{V}. \quad (\text{C.6})$$

Multiplying Eqs. (C.5) and (C.6) with \mathbf{i}_f^T yields

$$\left\{ \left[\frac{(\mathbf{i}_N \times \mathbf{i}_V)^T \mathbf{i}_f}{r} \right] \mathbf{P} + \mathbf{Q} \right\}^T \mathbf{i}_f = 0 \quad (\text{C.7})$$

as a first transversality condition. Multiplying (C.5) with \mathbf{i}_R^T and (C.6) with \mathbf{i}_V^T yields

$$\mathbf{Q}^T \mathbf{i}_V - \frac{v}{r} \mathbf{P}^T \mathbf{i}_R = 0 \quad (\text{C.8})$$

as the second transversality condition. Note that \mathbf{i}_f is perpendicular to \mathbf{i}_R and \mathbf{i}_V , and due to the last constraint in Eq. (C.4) $\mathbf{i}_R^T \mathbf{i}_V = 0$.

C.3 Transversality Conditions for Circular Orbit with Zero Argument of Perigee

The set of constraints for a circular orbit with a fixed argument of perigee is

$$\boldsymbol{\Psi} = \begin{bmatrix} \mathbf{i}_f^T \mathbf{i}_N - \cos i \\ r - r_p \\ v - v_p \\ \mathbf{R}^T \mathbf{V} \\ \mathbf{R}^T \mathbf{i}_N \end{bmatrix} = \mathbf{0} \quad (\text{C.9})$$

Assuming that ϕ does not depend on \mathbf{R} or \mathbf{V} , the transversality conditions for \mathbf{P} and \mathbf{Q} can be written as

$$\mathbf{P} = v_1 \left[\frac{(\mathbf{i}_N \times \mathbf{i}_R)^T \mathbf{i}_f}{v} \right] \mathbf{i}_f + v_3 \mathbf{i}_V + v_4 \mathbf{R} \text{ and} \quad (\text{C.10})$$

$$\mathbf{Q} = v_1 \left[\frac{(\mathbf{i}_V \times \mathbf{i}_N)^T \mathbf{i}_f}{r} \right] \mathbf{i}_f + v_2 \mathbf{i}_R + v_4 \mathbf{V} + v_5 \mathbf{i}_N. \quad (\text{C.11})$$

Noting that

$$(\mathbf{i}_V \times \mathbf{i}_N)^T \mathbf{i}_f = (\mathbf{i}_V \times \mathbf{i}_f)^T \mathbf{i}_N = [\mathbf{i}_V \times (\mathbf{i}_R \times \mathbf{i}_V)]^T \mathbf{i}_N = [\mathbf{i}_R - \mathbf{i}_V(\mathbf{i}_R^T \mathbf{i}_V)]^T \mathbf{i}_N = 0, \quad (\text{C.12})$$

Eq. (C.11) simplifies to

$$\mathbf{Q} = v_2 \mathbf{i}_R + v_4 \mathbf{V} + v_5 \mathbf{i}_N. \quad (\text{C.13})$$

Multiplying Eq. (C.13) with \mathbf{i}_f^T yields

$$v_5 = \frac{\mathbf{Q}^T \mathbf{i}_f}{\mathbf{i}_N^T \mathbf{i}_f} \quad (\text{C.14})$$

Multiplying Eq. (C.10) with \mathbf{i}_R^T yields

$$v_4 = \frac{\mathbf{P}^T \mathbf{i}_R}{r}. \quad (\text{C.15})$$

Now, Eq. (C.13) can be multiplied with \mathbf{i}_V^T which yields the transversality condition

$$\frac{v}{r} \mathbf{P}^T \mathbf{i}_R - \mathbf{Q}^T \mathbf{i}_V + \frac{(\mathbf{i}_N^T \mathbf{i}_V)(\mathbf{Q}^T \mathbf{i}_f)}{\mathbf{i}_N^T \mathbf{i}_f} = 0. \quad (\text{C.16})$$

C.4 Transversality Conditions for Circular Orbit with Fixed Argument of Perigee

The set of constraints for a circular orbit with a fixed argument of perigee is

$$\Psi = \begin{bmatrix} \mathbf{i}_f^T \mathbf{i}_N - \cos i \\ r - r_p \\ v - v_p \\ \mathbf{R}^T \mathbf{V} \\ \mathbf{i}_R^T (\mathbf{i}_N \times \mathbf{i}_f) - \sin i \cos \omega \end{bmatrix} = \mathbf{0}. \quad (\text{C.17})$$

Assuming that ϕ does not depend on \mathbf{R} or \mathbf{V} , the transversality conditions for \mathbf{P} and \mathbf{Q} can be written as

$$\mathbf{P} = v_1 \left[\frac{(\mathbf{i}_N \times \mathbf{i}_R)^T \mathbf{i}_f}{v} \right] \mathbf{i}_f + v_3 \mathbf{i}_V + v_4 \mathbf{R} + v_5 \frac{\mathbf{i}_N - (\mathbf{i}_R^T \mathbf{i}_N) \mathbf{i}_R - [\mathbf{i}_f^T (\mathbf{i}_N \times \mathbf{i}_R)] \mathbf{i}_V}{v} \quad \text{and} \quad (\text{C.18})$$

$$\mathbf{Q} = v_1 \left[\frac{(\mathbf{i}_V \times \mathbf{i}_N)^T \mathbf{i}_f}{r} \right] \mathbf{i}_f + v_2 \mathbf{i}_R + v_4 \mathbf{V} + v_5 \frac{\mathbf{i}_N \times \mathbf{i}_f + (\mathbf{i}_V^T \mathbf{i}_N) \mathbf{i}_R}{r}. \quad (\text{C.19})$$

Multiplying Eq. (C.18) with \mathbf{i}_f^T yields

$$v_1 = \frac{r}{(\mathbf{i}_V \times \mathbf{i}_N)^T \mathbf{i}_f} \mathbf{Q}^T \mathbf{i}_f. \quad (\text{C.20})$$

Multiplying Eq. (C.19) with \mathbf{i}_f^T yields and substituting v_1 with Eq. (C.20) yields

$$v_5 = \frac{v}{\mathbf{i}_N^T \mathbf{i}_f} \left[\mathbf{P} + \frac{r}{v} \cdot \frac{(\mathbf{i}_N \times \mathbf{i}_R)^T \mathbf{i}_f}{(\mathbf{i}_N \times \mathbf{i}_V)^T \mathbf{i}_f} \mathbf{Q} \right]^T \mathbf{i}_f \quad (\text{C.21})$$

Multiplying Eq. (C.19) with \mathbf{i}_R^T yields

$$v_4 = \frac{\mathbf{P}^T \mathbf{i}_R}{r}. \quad (\text{C.22})$$

Finally, multiplying Eq. (C.20) with \mathbf{i}_V and substituting v_4 and v_5 with Eqs. (C.21) and (C.22) yields the transversality condition

$$0 = \frac{v}{r} \mathbf{P}^T \mathbf{i}_R - \mathbf{Q}^T \mathbf{i}_V + \frac{1}{\mathbf{i}_N^T \mathbf{i}_f} \left\{ \frac{v}{r} [(\mathbf{i}_N \times \mathbf{i}_V)^T \mathbf{i}_f] \mathbf{P} - [(\mathbf{i}_N \times \mathbf{i}_R)^T \mathbf{i}_f] \mathbf{Q} \right\}^T \mathbf{i}_f \quad (\text{C.23})$$

Note that Eq. (C.23) can be rewritten as

$$0 = \frac{v}{r} \mathbf{P}^T \mathbf{i}_R - \mathbf{Q}^T \mathbf{i}_V + \tan i \left(\frac{v}{r} \mathbf{P} \sin \omega + \mathbf{Q} \cos \omega \right)^T \mathbf{i}_f. \quad (\text{C.24})$$

Note that Eq. (C.16) is the limiting expression of Eq. (C.24) for $\omega = 0^\circ$.

A similar procedure can be applied for the when using an alternative set of constraints for ω around 0° or 180° (see Appendix B.7):

$$\boldsymbol{\Psi} = \begin{bmatrix} \mathbf{i}_f^T \mathbf{i}_N - \cos i \\ r - r_p \\ v - v_p \\ \mathbf{R}^T \mathbf{V} \\ \mathbf{i}_V^T (\mathbf{i}_N \times \mathbf{i}_f) - \sin i \sin \omega \end{bmatrix} = \mathbf{0}. \quad (\text{C.25})$$

The set of transversality conditions can now be written as

$$\mathbf{P} = v_1 \left[\frac{(\mathbf{i}_N \times \mathbf{i}_R)^T \mathbf{i}_f}{v} \right] \mathbf{i}_f + v_3 \mathbf{i}_V + v_4 \mathbf{R} + v_5 \frac{\mathbf{i}_N - (\mathbf{i}_R^T \mathbf{i}_N) \mathbf{i}_R - [\mathbf{i}_f^T (\mathbf{i}_N \times \mathbf{i}_R)] \mathbf{i}_V}{v} \text{ and} \quad (\text{C.26})$$

$$\mathbf{Q} = v_1 \left[\frac{(\mathbf{i}_V \times \mathbf{i}_N)^T \mathbf{i}_f}{r} \right] \mathbf{i}_f + v_2 \mathbf{i}_R + v_4 \mathbf{V} + v_5 \frac{\mathbf{i}_N \times \mathbf{i}_f + (\mathbf{i}_V^T \mathbf{i}_N) \mathbf{i}_R}{r} \quad (\text{C.27})$$

Applying the same strategy as above ultimately yields Eq. (C.24) again.

C.5 Transversality Conditions for Elliptical Orbits

The set of constraints for an elliptical orbit with a free attachment point and a free argument of perigee can be formulated as

$$\boldsymbol{\Psi} = \begin{bmatrix} \mathbf{i}_f^T \mathbf{i}_N - \cos i \\ a(1-e) - r_p \\ \frac{v^2}{2} - \frac{\mu}{r} - \epsilon_p \end{bmatrix} = \mathbf{0}. \quad (\text{C.28})$$

The transversality conditions at final time can be written as

$$\mathbf{P} = v_1 \left[\frac{(\mathbf{i}_N \times \mathbf{i}_R)^T \mathbf{i}_f}{v} \right] \mathbf{i}_f + v_2 \left\{ \left[\frac{2a^2 v(1-e)}{\mu} - \frac{ah^2 v}{\mu^2 e} \right] \mathbf{i}_V - \frac{1}{\mu e} \mathbf{R} \times (\mathbf{R} \times \mathbf{V}) \right\} + v_3 \mathbf{V} \quad (\text{C.29})$$

$$\mathbf{Q} = v_1 \left[\frac{(\mathbf{i}_V \times \mathbf{i}_N)^T \mathbf{i}_f}{r} \right] \mathbf{i}_f + v_2 \left\{ \left[\frac{2a^2(1-e)}{r^2} - \frac{ah^2}{\mu e r^2} \right] \mathbf{i}_R - \frac{1}{\mu e} \mathbf{V} \times (\mathbf{V} \times \mathbf{R}) \right\} + v_3 \frac{\mu}{r^2} \mathbf{i}_R \quad (\text{C.30})$$

Noting that all terms behind v_2 and v_3 are perpendicular to \mathbf{i}_f , the first transversality condition can be computed by multiplying Eqs. (C.29) and (C.30) with \mathbf{i}_f^T and eliminating v_1 :

$$\left\{ \left[\left(\frac{(\mathbf{i}_N \times \mathbf{i}_V)^T \mathbf{i}_f}{(\mathbf{i}_N \times \mathbf{i}_R)^T \mathbf{i}_f} \right) \right] \mathbf{P} + \mathbf{Q} \right\}^T \mathbf{i}_f = 0 \quad (\text{C.31})$$

which is the same as Eq. (C.7). The remaining two transversality conditions are derived by first building the scalar products of Eqs. (C.29) and (C.30) with $(\mathbf{i}_V + \mathbf{i}_R)^T$:

$$\mathbf{P}^T(\mathbf{i}_V + \mathbf{i}_R) = v_2 \left\{ c_{2V} \mathbf{i}_V^T(\mathbf{i}_V + \mathbf{i}_R) - \frac{1}{\mu e} [\mathbf{R} \times (\mathbf{R} \times \mathbf{V})]^T(\mathbf{i}_V + \mathbf{i}_R) \right\} + v_3 (v + \mathbf{V}^T \mathbf{i}_R) \quad (\text{C.32})$$

$$\mathbf{Q}^T(\mathbf{i}_V + \mathbf{i}_R) = v_2 \left\{ c_{2R} \mathbf{i}_R^T(\mathbf{i}_V + \mathbf{i}_R) - \frac{1}{\mu e} [\mathbf{V} \times (\mathbf{V} \times \mathbf{R})]^T(\mathbf{i}_V + \mathbf{i}_R) \right\} + v_3 \frac{\mu}{r^2} (\mathbf{i}_R^T \mathbf{i}_V + 1) \quad (\text{C.33})$$

where

$$c_{2V} = \frac{2a^2 v(1-e)}{\mu} - \frac{ah^2 v}{\mu^2 e} \text{ and } c_{2R} = \frac{2a^2(1-e)}{r^2} - \frac{ah^2}{\mu e r^2} \quad (\text{C.34})$$

Defining

$$\mathbf{A} = c_{2V} \mathbf{i}_V^T(\mathbf{i}_V + \mathbf{i}_R) - \frac{1}{\mu e} [\mathbf{R} \times (\mathbf{R} \times \mathbf{V})]^T(\mathbf{i}_V + \mathbf{i}_R) \quad (\text{C.35})$$

and

$$B = c_{2R} \mathbf{i}_R^T (\mathbf{i}_V + \mathbf{i}_R) - \frac{1}{\mu e} [\mathbf{V} \times (\mathbf{V} \times \mathbf{R})]^T (\mathbf{i}_V + \mathbf{i}_R) \quad (\text{C.36})$$

and multiplying Eq. (C.32) with μ/r^2 and Eq. (C.33) with $-v$ yields

$$v_2 = \frac{\frac{\mu}{r^2} \mathbf{P}^T (\mathbf{i}_V + \mathbf{i}_R) - v \mathbf{Q}^T (\mathbf{i}_V + \mathbf{i}_R)}{\frac{\mu}{r^2} A - v B} \quad (\text{C.37})$$

and

$$v_3 = \frac{1}{v(\mathbf{i}_R^T \mathbf{i}_V + 1)} [\mathbf{P}^T (\mathbf{i}_V + \mathbf{i}_R) - v_2 A] \quad (\text{C.38})$$

Since the eccentricity e appears in the denominator of A , B , c_{2R} and c_{2V} , these values are multiplied with e to improve the stability for small values of e . The following values are defined:

$$c_{2V} = \frac{1}{e} \left[\frac{2a^2 v e (1-e)}{\mu} - \frac{a h^2 v}{\mu^2} \right] = \frac{1}{e} \tilde{c}_{2V} \quad (\text{C.39})$$

$$c_{2R} = \frac{1}{e} \left[\frac{2a^2 e (1-e)}{r^2} - \frac{a h^2}{\mu r^2} \right] = \frac{1}{e} \tilde{c}_{2R} \quad (\text{C.40})$$

$$A = \frac{1}{e} \left\{ \tilde{c}_{2V} \mathbf{i}_V^T (\mathbf{i}_V + \mathbf{i}_R) - \frac{1}{\mu} [\mathbf{R} \times (\mathbf{R} \times \mathbf{V})]^T (\mathbf{i}_V + \mathbf{i}_R) \right\} = \frac{1}{e} \tilde{A} \quad (\text{C.41})$$

$$B = \frac{1}{e} \left\{ \tilde{c}_{2R} \mathbf{i}_R^T (\mathbf{i}_V + \mathbf{i}_R) - \frac{1}{\mu} [\mathbf{V} \times (\mathbf{V} \times \mathbf{R})]^T (\mathbf{i}_V + \mathbf{i}_R) \right\} = \frac{1}{e} \tilde{B} \quad (\text{C.42})$$

$$v_2 = \frac{\frac{\mu}{r^2} \mathbf{P}^T (\mathbf{i}_V + \mathbf{i}_R) - v \mathbf{Q}^T (\mathbf{i}_V + \mathbf{i}_R)}{\frac{\mu}{r^2} \tilde{A} - v \tilde{B}} e = e \tilde{v}_2 \quad (\text{C.43})$$

$$v_3 = \frac{1}{v} \tilde{v}_3 \quad (\text{C.44})$$

Based on these definitions, the remaining two transversality conditions can be derived as

$$\mathbf{P}^T \mathbf{i}_R - \tilde{v}_2 \tilde{c}_{2V} (\mathbf{i}_V^T \mathbf{i}_R) - \tilde{v}_3 (\mathbf{i}_V^T \mathbf{i}_R) = 0 \quad (\text{C.45})$$

$$\mathbf{Q}^T \mathbf{i}_R - \tilde{v}_2 \tilde{c}_{2R} (\mathbf{i}_V^T \mathbf{i}_R) - \tilde{v}_3 \frac{\mu}{vr^2} (\mathbf{i}_V^T \mathbf{i}_R) = 0 \quad (\text{C.46})$$

D Initial Guesses for the Initial Costates

This appendix gives the derivation a method for estimating the initial costate values based on an approach presented in Ref. [78].

D.1 Preliminaries

In the framework of Ref. [78], all values are expressed in canonical units (see Ref. [8]). I.e., the gravity constant μ is normalized to one which also yields that the initial circular radius and circular velocity are set to one. All equations are specified in an orbital coordinate frame as shown in Fig. D.1.

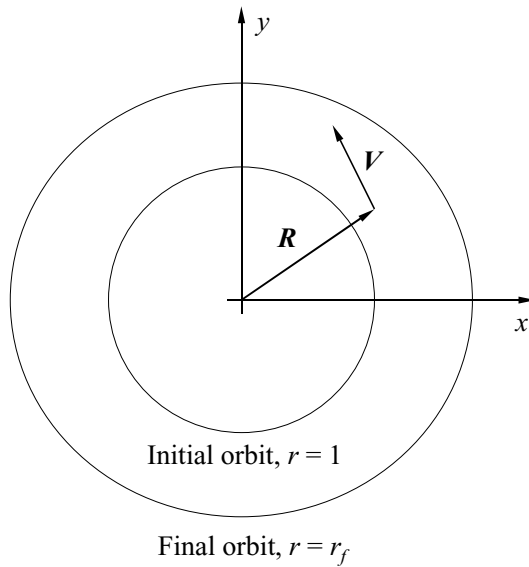


Fig. D.1: Definition of orbit coordinate frame

In the following notation, \bar{a} is the thrust acceleration expressed in canonical units and \bar{r} is the final orbit radius expressed in canonical units. Note that the required scaling factors to convert physical units to canonical units are

$$s_r = r_0, s_v = \sqrt{\mu/r_0}, s_a = \mu/r_0^2 \text{ and } s_t = \sqrt{r_0^3/\mu}. \quad (\text{D.1})$$

D.2 Derivation of Initial Costate Approximation

As a first approximation, massflow as well as the gravity constant are set to zero in the equations of motion which yields

$$\dot{x} = v_x, \dot{y} = v_y \quad (\text{D.2})$$

$$\dot{v}_x = \bar{a} \left(\frac{p_x}{\sqrt{p_x^2 + p_y^2}} \right), \dot{v}_y = \bar{a} \left(\frac{p_y}{\sqrt{p_x^2 + p_y^2}} \right) \quad (\text{D.3})$$

$$\dot{q}_x = 0, \dot{q}_y = 0 \quad (\text{D.4})$$

$$\dot{p}_x = -q_x, \dot{p}_y = -q_y \quad (\text{D.5})$$

The equations for the costates can be integrated immediately which yields:

$$q_x = a, q_y = b, p_x = -at + c, p_y = -bt + d \quad (\text{D.6})$$

Where a, b, c and d are the integration constants. Defining $q = \sqrt{q_x^2 + q_y^2}$ and $p = \sqrt{p_x^2 + p_y^2}$, the equations for velocity can be written as

$$v_x = \frac{\bar{a}}{q^3} \left[-aqp + b(bc - ad) \ln(-ac - bd + q^2 t + qp) \right] + k_1 \text{ and} \quad (\text{D.7})$$

$$v_y = \frac{\bar{a}}{q^3} \left[-bqp + a(ad - bc) \ln(-ac - bd + q^2 t + qp) \right] + k_2. \quad (\text{D.8})$$

Because the Lagrange multipliers appear as linear terms in the Hamiltonian, a is scaled to one in the following. The Hamiltonian can be expressed in polar and Cartesian coordinates. For polar coordinates, r is the scalar distance from the attracting center, u is the velocity component along r , and w is the velocity component perpendicular to u .

$$H_p = q_r u + p_u \left(\frac{w^2}{r} - \frac{1}{r} + \bar{a} \cos \phi \right) + p_w \left(-\frac{uw}{r} + \bar{a} \sin \phi \right) \quad (\text{D.9})$$

$$H_c = q_x v_x + q_y v_y + p_x \left(-\frac{x}{r^3} + \bar{a} \cos \phi \right) + p_y \left(-\frac{y}{r^3} + \bar{a} \sin \phi \right) \quad (\text{D.10})$$

Evaluating both expressions for at the initial time for the initial conditions

$$r(0) = 1, u(0) = 0, w(0) = v_0 \quad (\text{D.11})$$

$$x(0) = 1, y(0) = 0, v_x(0) = 0, v_y(0) = v_0 \quad (\text{D.12})$$

and equating both expressions yields

$$p_u(\bar{a} \cos \phi) + p_w(\bar{a} \sin \phi) + p_u(v_0^2 - 1) = q_x v_0 - p_x + p_x(\bar{a} \cos \phi) + p_y(\bar{a} \sin \phi). \quad (\text{D.13})$$

By equating the coefficients of $\cos \phi$ and $\sin \phi$ it can be determined that

$$p_u = p_x = c \text{ and } p_w = p_y = d \quad (\text{D.14})$$

and therefore

$$c(v_0^2 - 1) = b v_0 - c \text{ or } c = \frac{b}{v_0} \text{ or } b = v_0 c. \quad (\text{D.15})$$

The integration constants k_1 and k_2 can be eliminated by evaluating the expressions (D.7) and (D.8) at initial and final time which yields

$$\frac{v_{x,f} - v_{x,0}}{A} q^3 = q(p_0 - p_f) + b(bc - d) \cdot L \quad (\text{D.16})$$

$$\frac{v_{y,f} - v_{y,0}}{bA} q^3 = q(p_0 - p_f) + \frac{d - bc}{b} \cdot L \quad (\text{D.17})$$

where

$$L = \ln \left(\frac{-c - bd + q^2 t_f + qp_f}{-c - bd + qp_0} \right). \quad (\text{D.18})$$

In the special case where the initial and final velocity components are all equal to zero, the optimal trajectory is a straight line. If $\dot{m} = 0$, the thrust will switch directions midway through the trajectory to decelerate to a stop. Noting that $s = \bar{a} t^2 / 2$ for $s = r_f - r_0$ yields

$$t_f = 2 \sqrt{\frac{r_f - 1}{\bar{a}}}. \quad (\text{D.19})$$

This straight line case also allows for a simplification of Eqs. (D.16) and (D.17). They take the form

$$q(p_0 - p_f) + b(bc - d) \cdot L = 0 \text{ and} \quad (\text{D.20})$$

$$q(p_0 - p_f) + \frac{d - bc}{b} \cdot L \quad (\text{D.21})$$

The only difference in these two equations is the coefficient of the logarithmic term L . Therefore, these coefficients must be equal which yields the relationship

$$(b^2 + 1)(bc - d) = 0 \quad (\text{D.22})$$

Eq. (D.22) can only be satisfied if $bc = d$ or, using the results of Eq. (D.15):

$$d = \frac{b^2}{v_0} \quad (\text{D.23})$$

Since $q > 1$ by definition, p_0 and p_f must be equal as well. Using the definition for p yields

$$\sqrt{c^2 + d^2} = \sqrt{(c - t_f)^2 + (d - bt_f)^2}. \quad (\text{D.24})$$

After substituting c and d with the results from above finally yields

$$b = \frac{v_0 t_f}{2}. \quad (\text{D.25})$$

Therefore, the initial costates can be estimated as

$$q_{x,0} = a = 1, q_{y,0} = b = v_0 \sqrt{\frac{\bar{r} - 1}{\bar{a}}} \quad (\text{D.26})$$

$$p_{x,0} = c = \sqrt{\frac{\bar{r} - 1}{\bar{a}}}, p_{y,0} = v_0 \frac{\bar{r} - 1}{\bar{a}} \quad (\text{D.27})$$

Ref. [78] also states a correction scaling factor which should be applied if $\bar{a} > 1$. The scaling factor q is calculated as

$$q = 1 - \frac{1}{2\bar{a}} + \frac{\bar{r}}{4\bar{a}^2} \quad (\text{D.28})$$

and is applied to c and d such that

$$\tilde{c} = qc \text{ and } \tilde{d} = q^2 d. \quad (\text{D.29})$$

E Environment Models

This appendix defines the environment models used for the two example problems presented in Chapters 4 and 5.

E.1 Earth Model

The Earth model is a rotating ellipsoid with J_2 terms included in the gravity potential. All properties are given in Table E.1. The detailed equations are given in Ref. [5].

Item	Value
Equatorial radius	6378.135 km
Polar radius	6356.912 km
Angular velocity	7.2921×10^{-5} rad/s
Gravity constant	3.98602×10^5 km ³ /s ²
J_2	1.0827×10^{-3}

Table E.1: Properties of the Earth model

E.2 Atmosphere Models

The simple rocket example in Chapter 3.9 is using a smoothly interpolated US Standard 1976 atmosphere model. The following equations are used to calculate the ambient pressure and its derivative with respect to altitude:

$$p(h) = \exp \left[\sum_{i=0}^5 a_i h^i + k \cdot \exp \left(\sum_{i=1}^5 b_i h^i \right) \right] \quad (\text{E.1})$$

$$\frac{\partial p}{\partial h} = p(h) \cdot \left[\sum_{i=1}^5 i a_i h^{i-1} + k \cdot \exp \left(\sum_{i=1}^5 b_i h^i \right) \cdot \sum_{i=1}^5 i b_i h^{i-1} \right] \quad (\text{E.2})$$

where $k = 4.92934 \cdot 10^{-6}$ and the coefficients are given in Table E.2.

i	a_i	b_i
0	11.8229	-
1	-0.16026	1.43783
2	-0.000370874	-0.0657968
3	0.0000309506	0.00115937
4	-4.48838e-7	-9.09543e-6
5	1.86338e-9	2.61544e-8

Table E.2: Coefficients for the interpolated US-Standard atmosphere

The Ariane 5 example problem in Chapter 4 is using altitude dependent data tables for ambient pressure, density, and speed of sound. The tables are interpolated using Akima splines (Ref. [1]) and can be found in Ref. [6]. The Hopper example in Chapter 5 is using the US Standard atmosphere 1976 model. No winds are included.

Deutsche Kurzfassung

*Als den gelehrten Astronomen ich hörte,
Als die Beweise, die Zahlen in langen Reihen er vor mir
entwickelt',
Als die Tabellen und Diagramme er mir zeigte, sie zu
addieren, zu teilen und sie zu messen,
Als den Astronomen ich hörte, der seinen Vortrag hielt
unter großem Applaus,
Wie bald wurde ich da so sonderbar müde und krank,
Bis ich mich erhob, aus dem Saale mich schlich und
einsam wanderte
Hinaus in die feuchte, mystische Nacht, wo von Zeit zu
Zeit den Blick still ich erhob nach den Sternen...*

Walt Whitman, "Als den gelehrten Astronomen ich hörte"

Viele Probleme im Bereich der Ingenieurwissenschaften beinhalten die Berechnung "optimaler Trajektorien". Typische Beispiele aus dem Luft- und Raumfahrtsektor sind nutzlastoptimale Aufstiegsbahnen von Raketen, Wiedereintrittsmissionen mit verschiedenen Zielfunktionalen oder auch die Berechnung von Zeit-, Treibstoff- und Kostenoptimalen Flugbahnen eines Überschall-Transportflugzeuges.

Auch außerhalb der Luft- und Raumfahrttechnik treten immer wieder Probleme auf, die als Optimalsteuerungsproblem formuliert werden können. Dies kann z.B. die Berechnung energieoptimaler Roboterbewegungen, die Optimierung chemischer Prozesse oder die Optimierung von Wirtschaftsmodellen zur Gewinnmaximierung sein.

Die in dieser Arbeit entwickelte Methode zur Berechnung von Optimalsteuerungsproblemen ist ein sehr allgemeiner Ansatz zur Lösung einer umfangreichen Klasse von Problemen. Bei dem Verfahren handelt sich dabei um die Kombination zweier sogenannter "direkter Verfahren": der direkten Kollokation und dem direkten Mehrzielverfahren. Außerdem wurde für die Berechnung optimaler Schubmanöver einer Rakete im Vakuum ein sogenanntes "indirektes

Verfahren" implementiert, welches auch in Kombination mit den direkten Verfahren eingesetzt werden kann.

Motivation

Sowohl die auf den Prinzipien der Variationsrechnung beruhenden indirekten Verfahren, als auch die direkten Verfahren weisen sowohl Vor- als auch Nachteile im praktischen Einsatz auf. Im Falle der indirekten Verfahren bestehen die Hauptnachteile in der umfangreichen mathematischen Analyse des Problems und der Formulierung der notwendigen Bedingungen (Euler-Lagrange Gleichungen). Häufig sind dabei bereits Annahmen über die Struktur der erwarteten Lösung zu treffen was bei völlig neuen Problemstellungen oft nur schwer möglich ist. Zudem gestaltet sich die Lösung des entstehenden Mehrpunkt-Randwertproblems meist schwierig und die eingesetzten Verfahren weisen bei realistisch formulierten Problemen einen sehr kleinen Konvergenzradius auf. Ein großer Vorteil der indirekten Verfahren ist jedoch die Möglichkeit, zusätzliche Informationen über die Qualität der berechneten Lösung zu erhalten, beispielsweise durch die Berechnung einer Schaltfunktion.

Der Hauptnachteil bestehender direkter Verfahren ist darin zu sehen, dass diese oft sehr unstrukturiert programmiert und somit nur sehr schwer zu warten und zu erweitern sind. Außerdem zeigt sich, dass sich die Anwendung eines reinen direkten Kollokationsverfahrens bei einigen Problemstellungen schwierig gestaltet. Direkte Mehrzielverfahren hingegen liefern zwar sehr genaue Ergebnisse, weisen jedoch einen kleineren Konvergenzradius als die direkten Kollokationsverfahren auf und benötigen wesentlich größere Rechenzeiten. Im Vergleich zu den indirekten Verfahren gestaltet sich die Lösung realistischer Probleme jedoch wesentlich einfacher und der Konvergenzradius der direkten Verfahren ist wesentlich größer als bei indirekten Verfahren.

Hauptziel dieser Arbeit ist die Erstellung eines hybriden Verfahrens, welches die Vorteile vorhandener Optimierungsverfahren in sich vereint während die Nachteile weitestgehend ausgeschaltet werden. Der Ansatz ist ein Verfahren zur Lösung mehrphasiger Optimalsteuerungsprobleme, bei dem der Benutzer phasenweise zwischen den verschiedenen Lösungsverfahren wählen kann. Die dazu entwickelte Software CAMTOS (Collocation And Multiple Shooting Trajectory Optimization Software) ist vollständig in die grafische Benutzeroberfläche GESOP integriert und vollständig kompatibel zur umfangreichen ASTOS Modellbibliothek sowie sämtlichen Modellen für die GESOP Umgebung.

Bei der Entwicklung wurde das Hauptaugenmerk auf ein möglichst modulares Design der Software gelegt. Dabei wurden einfach zu bedienende Schnittstellen zu den wesentlichen Elementen eines direkten Bahnoptimierungsverfahrens entwickelt. Somit können in künftigen Entwicklungen leicht zusätzliche Integrationsverfahren für das direkte Mehrzielverfahren hinzugefügt werden, neue Schemata für direkte Kollokationsverfahren entwickelt, oder auch neue NLP-Löser zur Verfügung gestellt werden.

Das erstellte direkte Optimierungsverfahren kann auch in Verbindung mit dem indirekten Verfahren angewandt werden. Dabei müssen zusätzlich zu den System-Differentialgleichungen auch die adjungierten Differentialgleichungen zur Verfügung gestellt werden. Dies

wird für die Optimierung des Steuerungsverlaufes von Raketen-Oberstufen im Vakuum demonstriert.

Betrachtete Problemklasse

Die mit CAMTOS lösbaren, mehrphasigen Bahnoptimierungsprobleme können aus k Phasen bestehen. Jede Phase kann neben einer beliebigen Anzahl von Zuständen \mathbf{x} und Steuerungen \mathbf{u} auch einen Vektor mit konstanten Designparametern \mathbf{p} enthalten. Eine allgemeine, mathematische Formulierung ist in Gl. (1.1)-(1.7) wiedergegeben.

$$\text{Minimiere } J = \phi^0(\mathbf{x}_0^1, \mathbf{p}^1, t_0^1) + \sum_{i=1}^k \phi^i(\mathbf{x}_f^i, \mathbf{p}^i, t_f^i) + \sum_{i=1}^k \int_{t_0^i}^{t_f^i} L^i(\mathbf{x}^i, \mathbf{u}^i, \mathbf{p}^i, t) dt \quad (1.1)$$

$$\text{mit } \dot{\mathbf{x}}^i = \mathbf{f}(\mathbf{x}^i, \mathbf{u}^i, \mathbf{p}^i, t) \text{ für } t_0^i \leq t \leq t_f^i, i = 1, \dots, k, \quad (1.2)$$

$$\text{Pfadbeschränkungen } \mathbf{g}^i(\mathbf{x}^i, \mathbf{u}^i, \mathbf{p}^i, t) \geq \mathbf{0} \text{ für } t_0^i \leq t \leq t_f^i, i = 1, \dots, k, \quad (1.3)$$

$$\text{Anfangsbedingungen } \Psi_0^1(\mathbf{x}_0^1, \mathbf{p}^1, t_0^1) \geq \mathbf{0}, \quad (1.4)$$

$$\text{Endbedingungen } \Psi_f^i(\mathbf{x}_f^i, \mathbf{p}^i, t_f^i) \geq \mathbf{0}, i = 1, \dots, k, \quad (1.5)$$

$$\text{Phasen-Verknüpfungsbedingungen } \begin{bmatrix} \mathbf{x}_0^{i+1} & \mathbf{u}_0^{i+1} & \mathbf{p}^{i+1} \end{bmatrix}^T = \mathbf{h}(\mathbf{x}_f^i, \mathbf{u}_f^i, \mathbf{p}^i, t_f^i) \text{ für } i < k, \quad (1.6)$$

$$t_o^{i+1} = t_f^i \text{ für } i < k. \quad (1.7)$$

Im Rahmen des indirekten Verfahrens werden ausschließlich die Bewegungsgleichung einer Rakete im Vakuum über einem Erdellipsoid betrachtet. Pfadbeschränkungen werden dabei nicht berücksichtigt.

Das direkte Optimierungsverfahren

CAMTOS stellt sowohl direkte Kollokationsverfahren als auch direkte Mehrzielverfahren zur Verfügung. Beide Verfahren führen das ursprüngliche Optimalsteuerungsproblem durch eine Diskretisierung auf ein Parameteroptimierungsproblem mit nichtlinearen Beschränkungen zurück (NLP-Problem). Pfadbeschränkungen werden in beiden Verfahren durch eine Anzahl von punktförmigen Beschränkungen innerhalb der jeweiligen Phase erfasst. Zur Lösung des NLP-Problems wird der SQP-Löser SNOPT 6.1 verwendet. Jedoch ist die Schnittstelle zu die-

sem Lösungsverfahren so gestaltet, dass zukünftig auch andere Verfahren verwendet werden können.

Das direkte Kollokationsverfahren

Beim direkten Kollokationsverfahren werden die Steuerungsverläufe durch stückweise konstante oder stückweise lineare Verläufe approximiert. Die Zustandsverläufe werden mit Hilfe von Hermite-Simpson Polynomen, der Trapezregel oder einem Runge-Kutta Verfahren 4. Ordnung abgebildet. Letztere Approximation kann auch als Mehrzielverfahren mit einem einzigen Integrationsschritt pro Mehrzielintervall verstanden werden. Im Vergleich zu anderen direkten Kollokationsverfahren wird im Rahmen von CAMTOS an den Intervallgrenzen für jede Steuerung ein links- und ein rechtsseitiger Wert verwendet. Dies ermöglicht die Verwendung einer stückweise konstanten Steuerungsapproximation. Im Falle einer stückweisen linearen Steuerung werden zusätzliche Kontinuitätsbedingungen an den Intervallgrenzen hinzugefügt. Das dadurch entstehende nichtlineare Programm weist dann zwar mehr Parameter und Beschränkungen auf, jedoch zeigt sich, dass die benötigte Rechenzeit sowie die benötigte Anzahl an Iterationen durch diesen Ansatz deutlich reduziert wird.

Für die im Rahmen dieser Arbeit betrachteten Probleme zeigt sich, dass die Approximation der Zustandsverläufe mit Hermite-Simpson Polynomen am besten geeignet ist. Obwohl das Runge-Kutta Verfahren ebenfalls genau 4. Ordnung ist zeigt sich, dass der Konvergenzradius des Verfahrens mit Hermite-Simpson Polynomen größer ist. Dies ist zu erwarten, da die Diskretisierung mit Hermite-Simpson Polynomen als implizite Integration der Bahngleichungen angesehen werden. Implizite Integrationsverfahren weisen im Vergleich zu expliziten Runge-Kutta Verfahren im Allgemeinen eine größere Robustheit auf.

Das direkte Mehrzielverfahren

Auch beim direkten Mehrzielverfahren werden die Steuerungsverläufe durch stückweise konstante oder stückweise lineare Verläufe approximiert. Zur Berechnung der Zustandsverläufe stehen drei verschiedene numerische Integrationsverfahren mit eingebauter Schrittweitenkontrolle zur Verfügung: ein Runge-Kutta Verfahren 2./3. und 4./5. Ordnung sowie ein Integrationsverfahren nach Paus (Ref. [61]). Um eine konsistente Berechnung der vom SQP-Verfahren benötigten Gradienten der Zielfunktion sowie der Beschränkungen zu ermöglichen, stellt jeder Integrator zwei verschiedene Integrationsroutinen zur Verfügung: die erste Routine berechnet eine Referenzbahn inklusive der notwendigen Schrittweitenanpassung. Die zweite Routine verwendet die dabei generierte Schrittweitensequenz zur Berechnung leicht variierten Bahnen im Rahmen der numerischen Gradientenberechnung mit Hilfe finiter Differenzen. Dieses Verfahren ist dem der sogenannten "inneren Differenzierung" gleichwertig, ermöglicht jedoch eine sehr einfache Anwendung auf eine Vielzahl verschiedener Integrationsverfahren.

Ein Vergleich der implementierten Integrationsverfahren zeigt, dass das Runge-Kutta Verfahren 4./5. Ordnung für die im Rahmen dieser Arbeit betrachteten Probleme am besten geeignet ist. Das Paus-Verfahren entfaltet seine Vorteile erst, wenn es spezielle Problem und Modelleigenschaften ausnutzen kann. Dies war jedoch nicht Ziel dieser Arbeit, so dass bei

einer allgemeinen Anwendung mit niedrigen Genauigkeitsanforderungen das Runge-Kutta Verfahren 2./3. Ordnung dem Paus-Verfahren vorzuziehen ist.

Kombination von direkter Kollokation und direktem Mehrzielverfahren

In CAMTOS kann für jede Phase das geeignete Diskretisierungsverfahren ausgewählt werden. So ist beispielsweise bei der Optimierung von Raketenaufstiegsbahnen beim Einsatz von Feststoffboostern in den ersten Phasen das Mehrzielverfahren zu bevorzugen. Zwar ist auch eine Optimierung mit einem Kollokationsverfahren möglich, jedoch ist dabei eine präzise Platzierung der Kollokationsintervalle notwendig. Liegen die Intervallgrenzen nicht exakt auf den Datenpunkten der Schub- und Massenflussprofile der Boosterraketen, so treten durch die Approximation der Zustandsverläufe erhöhte Diskretisierungsfehler auf. Da in der Regel auch der Verlauf der aerodynamischen Koeffizienten auf Datentabellen basiert und diese nicht von der Flugzeit sondern im Wesentlichen von der Machzahl und dem Anstellwinkel abhängen, ist somit eine gute Platzierung der Kollokationsintervalle schwierig. Wird in diesen Phasen jedoch das Mehrzielverfahren eingesetzt, so sorgt die im Integrator enthaltene Schrittweitenkontrolle automatisch für eine genaue Berechnung der Zustandsverläufe.

In anderen Phasen eines Optimierungsproblem, in denen weitgehend analytische Gleichungen vorliegen, kann hingegen auf das deutlich schnellere Kollokationsverfahren zurückgegriffen werden. Es zeigt sich außerdem, dass der größere Konvergenzradius des Kollokationsverfahrens sehr gut ausgenutzt werden kann, um eine erste Lösung für ein Problem zu erhalten, welche dann durch das Umschalten auf das Mehrzielverfahren leicht verfeinert werden kann.

Zusätzlich zur phasenweisen Diskretisierung ist es in CAMTOS auch möglich, vollständig analytische Phasen zu verwenden. Dabei wird vorausgesetzt, dass der Benutzer eine Routine zur Verfügung stellen kann, die aus einem gegebenen Zustandvektor am Anfang der Phase den Zustandsvektor am Ende der Phase ohne Einsatz einer numerischen Integration berechnen kann.

Das indirekte Optimierungsverfahren

Das im Rahmen dieser Arbeit implementierte indirekte Optimierungsverfahren ist auf die Optimierung von Raketenbahnen im Vakuum spezialisiert. Dabei werden die Abplattung der Erde sowie J_2 Gravitationsstörterme berücksichtigt. Die Berücksichtigung zusätzlicher dissipativer Kräfte, wie z.B. der Einfluss des Luftdrucks auf den Triebwerksschub, führt zu einem stark verkleinerten Konvergenzradius des Verfahrens. Zwar kann in diesen Fällen auf ein Homotopieverfahren zurückgegriffen werden, jedoch sind für den atmosphärischen Teil einer Raketenbahn auch eine Vielzahl von zustandsabhängigen Pfadbeschränkungen einzuhalten, die im Rahmen des direkten Optimierungsverfahrens sehr viel eleganter erfasst werden können. Bei einer Betrachtung im Vakuum ohne jegliche Pfadbeschränkungen lässt sich das entstehende Randwertproblem jedoch relativ leicht lösen, und die Schätzung von Startwerten für den Adjungiertenvektor ist weniger kritisch.

Neben den adjungierten Differentialgleichungen und der optimalen Steuerung stehen die physikalischen Endbedingungen sowie die dazugehörigen Transversalitätsbedingungen für verschiedene Kombinationen typischer Brennschlussbedingungen zur Verfügung. Sämtliche notwendigen Bedingungen werden dabei im Hinblick auf eine Minimierung des Treibstoffbedarfs hergeleitet.

Als wichtigstes Element wird, neben den optimalen Steuerungsverläufen, auch eine Schaltfunktion berechnet. Der Wert dieser Schaltfunktion muss bei einer optimalen Lösung innerhalb einer Schubphase kleiner als Null und während einer Freiflugphase größer als Null sein. Ist diese Bedingung nicht erfüllt, so deutet dies auf die Notwendigkeit zusätzlicher Freiflugphasen hin. Somit steht eine elegante Methode zur Analyse einer zu untersuchenden Missionssequenz im Hinblick auf mögliche Treibstoffeinsparungen oder Nutzlastverbesserungen zur Verfügung.

Lösung des Randwertproblems mit einem SQP Verfahren

Das indirekte Optimierungsverfahren kann auch mit Hilfe von CAMTOS gelöst werden. Es zeigt sich dabei, dass CAMTOS als reiner Randwertlöser verwendet werden kann, wenn die Zielfunktion des SQP-Verfahrens zu Null gesetzt wird, die Transversalitätsbedingungen zur Verfügung stehen und nur die Anfangswerte der Adjungierten "optimiert" werden. In diesem Modus wird CAMTOS als reines Verfahren zur mehrdimensionalen Nullstellensuche verwendet. Erwartungsgemäß zeigt sich dabei ein quadratisches Konvergenzverhalten, sobald der Parametervektor in die Nähe der Lösung kommt.

Weitaus attraktiver ist jedoch die Möglichkeit, auf die explizite Formulierung der Transversalitätsbedingungen zu verzichten und statt dessen die Zielfunktion des SQP-Verfahrens derart einzusetzen, dass die Anfangswerte der Adjungierten den Wert der Zielfunktion minimieren. Es zeigt sich, dass die im SQP-Verfahren verwendeten Karush-Kuhn-Tucker Bedingungen den ansonsten benötigten Transversalitätsbedingungen entsprechen.

Kombination des direkten und des indirekten Verfahrens

Am Beispiel einer Ariane 5 Mission mit zwei Nutzlasten wird die Kombination aus direkten und indirekten Phasen in einem Bahnoptimierungsproblem demonstriert. In der untersuchten Mission wird eine erste Nutzlast in einem Kreisorbit in 600 km Höhe abgesetzt, während die zweite Nutzlast in einen modifiziert geostationären Transferorbit mit einem Argument des Perigäums von 270 Grad gebracht wird. Die Masse der ersten Nutzlast ist dabei konstant, die Masse der zweiten Nutzlast wird maximiert.

Der atmosphärische Teil des Aufstieges ist mit Hilfe des direkten Mehrzielverfahrens und einer linearen Steuerungsapproximation diskretisiert. Da Schubphasen der Oberstufe außerhalb der Atmosphäre liegen, wird hier das indirekte Verfahren eingesetzt. Auf eine explizite Formulierung der Transversalitätsbedingungen wird dabei verzichtet.

Für die Referenzmission mit einer Freiflugphase zeigt sich an der Schaltfunktion deutlich, dass durch Einführung einer weiteren Freiflugphase nach Brennschluss der Unterstufe

eine weitere Verbesserung der Nutzlast erreicht werden kann. Um zu gewährleisten, dass die Oberstufe innerhalb dieser ersten Freiflugphase nicht wieder in die Atmosphäre eintritt, wird zusätzlich gefordert, dass das Perigäum der ersten Freiflugphase in mindestens 200 km Höhe liegt. Für die derart modifizierte Mission ergibt sich schließlich eine Nutzlaststeigerung um etwa 66% im Vergleich zur Referenzmission mit einer Freiflugphase.

Kombination des direkten Kollokations- und direkten Mehrzielverfahrens

Die Kombination des direkten Kollokationsverfahrens mit dem direkten Mehrzielverfahren wird am Beispiel einer Mission eines zukünftigen Raumtransporters demonstriert. Dabei wird das sogenannte Hopper-Konzept der Astrium GmbH, Bremen, untersucht, in dem ein suborbitales, rückkehrfähiges Fahrzeug vom Raumfahrtbahnhof in Kourou eine Oberstufe zunächst in einer Höhe von ca. 90 km absetzt und dann zur Insel Ascension im Atlantischen Ozean zurückkehrt. Die Oberstufe bringt die Nutzlast schließlich in den geostationären Transferorbit.

Das Problem muss als verzweigtes Bahnoptimierungsproblem betrachtet werden, da die Randbedingung einer Rückkehr der Unterstufe einen Einfluss auf die Nutzlastmasse hat. Mit Hilfe des direkten Kollokationsverfahrens wird für dieses Problem zunächst eine Näherungslösung berechnet. Die atmosphärischen Phasen des Aufstieges und des Wiedereintritts der Unterstufe werden dann auf eine Diskretisierung für das direkte Mehrzielverfahren umgestellt, um in einer weiteren Rechnung eine genaue Lösung zu erhalten. Die Umstellung auf das Mehrzielverfahren ermöglicht dabei nicht nur die Berechnung einer genaueren Lösung, sondern reduziert auch die Größe des NLP-Problems.

Zusammenfassung und Ausblick

Die neuartige Kombination aus direktem Kollokations- und Mehrzielverfahren stellt eine Methode zur Lösung einer umfangreichen Klasse von Bahnoptimierungsproblemen zur Verfügung, die optimal an die Anforderungen des Problems angepasst werden kann. Für die verschiedenen Phasen kann das jeweils am besten geeignete Verfahren zur Diskretisierung verwendet werden. Außerdem ist mit dem gleichen Verfahren die Lösung von Randwertproblemen sowohl mit der indirekten Kollokation als auch mit der Mehrzielmethode möglich. Insbesondere ist dabei der Verzicht auf die explizite Formulierung der Transversalitätsbedingungen hervorzuheben, der die ansonsten notwendige analytische Vorarbeit erheblich vereinfacht.

Der modulare Aufbau von CAMTOS erlaubt eine leichte Implementierung zukünftiger Erweiterungen sowie eine einfache Wartung bereits vorhandener Komponenten. Ein Vergleich mit bereits existierender Software zeigt, dass die Leistung von CAMTOS im Bereich des direkten Mehrzielverfahrens anderen Verfahren zumindest ebenbürtig ist, im Bereich des direkten Kollokationsverfahrens wird durch eine geänderte Diskretisierung des Problems eine deutliche Leistungssteigerung erzielt.

Im Rahmen zukünftiger Entwicklungen ist insbesondere der Einsatz von Methoden zur automatischen Gitteranpassung und -verfeinerung bei Phasen mit direkter Kollokation sinn-

voll. Dabei ist sowohl das Hinzufügen zusätzlicher Intervalle zur Reduktion des Diskretisierungsfehlers, als auch das Entfernen von überflüssigen Intervallen zur Verkleinerung des NLP-Problems sinnvoll.

Für eine verstärkte Anwendung im Missionsentwurf sind zusätzliche Analysen der optimalen Lösung, insbesondere Sensitivitätsanalysen, erforderlich. Zusätzliche Forschung ist hier notwendig, um eine allgemein nutzbare Formulierung zu ermöglichen, die sowohl im Bereich des Missionsentwurfs und Fahrzeugdesigns, als auch bei Systemidentifikationen eingesetzt werden kann. Dabei sollten die vom SQP-Verfahren berechneten Lagrange Multiplikatoren ausgenutzt werden um die notwendigen Derivativa zu berechnen.

Viele Optimierungsprobleme weisen außerdem mehrere lokale Minima auf. Hierbei scheint insbesondere der Einsatz von genetischen Algorithmen vielversprechend. Im Rahmen einer solchen Entwicklung muss auch der Aspekt einer Lösung des Optimierungsproblems in einem Netzwerk von einzelnen PCs berücksichtigt werden (verteiltes Rechnen). Da bei evolutionären Algorithmen die benötigte Rechenzeit um ein vielfaches größer ist als bei SQP-Verfahren, ist es hier sinnvoll, einen solchen Algorithmus weitestgehend zu parallelisieren.

Eine Parallelisierung ist auch für die Berechnung der Jakobimatrix im Rahmen des direkten Mehrzielverfahrens interessant. Weitere Untersuchungen müssen zeigen, ob und in welchem Umfang sich eine solche Parallelisierung lohnt und welchen zusätzlichen Rechenaufwand die notwendigen Kommunikationsprotokolle erfordern.

Für spezielle Anwendungen kann die Implementierung zusätzlicher Integrationsverfahren interessant sein. Im Falle von interplanetaren Missionen käme hierbei z.B. ein Runge-Kutta Verfahren höherer Ordnung in Frage. Außerdem stellen Integrationsverfahren mit einer internen Ereignisdetektion eine interessante Verbesserungsmöglichkeit bei der Handhabung linear interpolierter Daten oder anderer Nicht-Differenzierbarkeiten in der Dynamik dar. Derartige Integrationsverfahren stellen sicher, dass der numerische Integrationsprozess genau an bestimmten Ereignissen anhält, indem neben der Integration eine Nullstellensuche in den vom Modell zur Verfügung gestellten Ereignisindikatoren durchgeführt wird. Zwar ist ein Runge-Kutta Verfahren 4./5. Ordnung in der Lage, mit derartigen Nicht-Differenzierbarkeiten umzugehen, jedoch führt dies häufig zu sehr kleinen Schrittweiten innerhalb eines Integrationsintervalls. Ereignisbasierte Integratoren können hier zu Leistungssteigerungen im Hinblick auf Rechenzeit und Genauigkeit der Lösung führen.