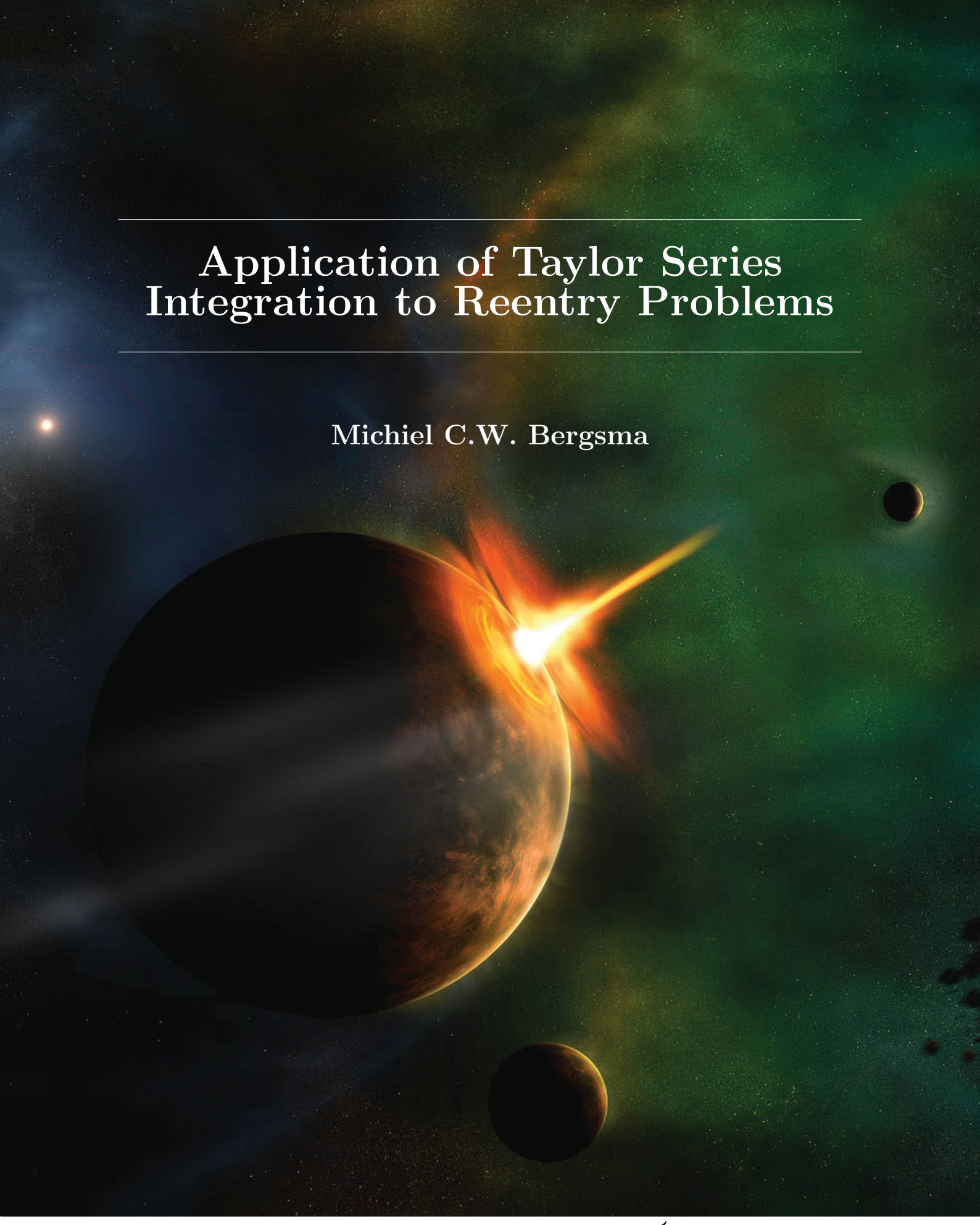


Application of Taylor Series Integration to Reentry Problems

Michiel C.W. Bergsma

Faculty of Aerospace Engineering

Delft University of Technology

Application of Taylor Series Integration to Reentry Problems

Master of Science Thesis

Michiel C.W. Bergsma

Thesis supervisor:
Dr. Ir. E. Mooij

TU Delft Delft University of Technology

Faculty of Aerospace Engineering
December 23, 2015

Preface

This is it. The final report I will be writing for my study in Aerospace Engineering. It has been a long road, with about as many highs as lows. Fortunately, I could always count on the support of my supervisor, Erwin Mooij, who always guided me in the right direction, provided me with new ideas and has pointed out my every spelling mistake. I would like to thank him for all this and for giving me the opportunity to work on this topic.

I would also like to thank the other students in rooms 903 and 906 for the support they have given me, both mentally and spiritually, but also by giving me the much need distractions. Without them, I would have probably gone insane from all the error messages my computer has been throwing at me over the past months. Last, but not least, I would like to thank my friends, family and band members for also supporting me and keeping me motivated.

I hope that the reader will enjoy reading this report and I would like to apologize in advance for Chapter 6. This chapter, consisting of 27 pages of almost purely equations, is a necessary evil and I completely understand it if the reader would like to skip this chapter. No hard feelings.

Michiel Bergsma

PS: I am completely aware of the fact that all Kepler orbits simulated in this report would crash into the surface of the Earth.

Abstract

The determination of optimal reentry trajectories and the analysis of the sensitivity of these trajectories to various disturbances often requires large numbers of simulations, making a fast and reliable reentry propagation tool a valuable asset to reduce computational times. Taylor Series Integration (TSI) is a numerical integration technique, which generates the Taylor series expansions of the state variables to propagate them in time. The generation of the Taylor series is done through automatic differentiation, which rewrites the equations of motion into recurrence relations, with which the Taylor coefficients, i.e., the coefficients that make up a Taylor series expansion, can be obtained up to an arbitrary order. This allows TSI to adapt its order to the accuracy requirements, allowing it to propagate with high efficiency. For TSI, the step size can be computed after the Taylor coefficients are obtained, which means that no time steps have to be rejected during the step size selection. These advantages allowed TSI to be on average 15.8 times faster for celestial mechanics than the Runge-Kutta-Fehlberg integrators that are typically used for this type of problems. The purpose of this MSc. Thesis is to determine whether TSI is also faster for reentry trajectory propagation.

TSI requires complete knowledge of the equations of motion, and all the equations that make up the environment and aerodynamic models. Furthermore, this method can only handle purely mathematical expressions, so the discrete data of these models are fitted with regression lines to obtain mathematical expression that approximate the data. Since the tables of the atmosphere model and the tabular form of the aerodynamic coefficients of the reentry vehicle, HORUS-2B, were known before integration, regression could be applied to them before integration. Another important feature added to TSI is the step-size reduction in case it integrated over a discontinuity. This way, the integrator does not unnecessarily accumulate errors by integrating an equation set that is no longer valid after the discontinuity. Discontinuities for reentry include the layer boundaries of the atmosphere model, the limits in terms of Mach number of the aerodynamics regression model, the energy levels of the nodes that define the control profiles, bank reversals, and the Terminal Area Energy Management interface. The step-size reductions are done using root-finding methods that were specially selected for each type of discontinuity, most of which make use of the Taylor coefficients of the variables of which the root is to be found. Furthermore, a number of root-finding methods were added to determine when constraints are violated.

In previous applications, it was found that the Runge-Kutta-Fehlberg 5(6) (RKF5(6)) method has the best performance for reentry of the traditional integration methods. The state-variable set yielding the lowest computational times for TSI is the spherical set for problems both with and without wind, whereas for RKF5(6), it is the Cartesian set in $\mathcal{F}_R$ without wind and the spherical set with wind. When comparing the computational times of TSI and this integrator, TSI is faster for all cases, ranging from 1.24 to 4.48 times for trajectories with wind and 3.28 to 11.61 times for trajectories without wind. During optimization with high error tolerance, it was found that TSI also has better accuracy than RKF5(6) and is able to detect all constraint violations. During sensitivity analysis with low error tolerance, it was found that TSI is again more accurate and that the cause for the better accuracy is the fact that TSI adapts its step sizes to avoid integrating over discontinuities.

Nomenclature

Notation

$\mathcal{F}_i$	Reference frame i
f'	Derivative of function f
$\dot{x}$	Derivative of variable x
$\boldsymbol{x}$	Vector
$\boldsymbol{X}$	Matrix
$\boldsymbol{X}^T$	Transpose of $\boldsymbol{X}$
$\boldsymbol{C}$	Transformation matrix

Latin Symbols

a	Speed of sound	[m/s]
B_l	Lower boundary	
B_u	Upper boundary	
C	Aerodynamic force coefficient	[-]
C_p	Specific heat at constant pressure	[J/kg/K]
CR	Crossover constant	[-]
C_v	Specific heat at constant volume	[J/kg/K]
D	Aerodynamic Drag	[N]
c	Cosine	[-]
d	Distance	[rad]
e	Eccentricity	[-]
E	Energy	[J]
F	Force	[N]
F	Scaling factor	[-]
f	Flattening	[-]
f	Aerodynamic acceleration	[m/s^2]
f	Auxiliary variable	
g	Gravity acceleration	[m/s^2]
H	Altitude	[m]
H_{ang}	Angular momentum	[m^2/s]
H_S	Scale height	[m]
h	Time-step size	[s]
J_n	J-coefficient of Earth's gravity field of degree n and order 0	[-]
$J_{n,m}$	J-coefficient of Earth's gravity field of degree n and order m	[-]
L	Lapse rate	[K/m]
L	Aerodynamic lift	[N]
L_M	Molecular scale lapse rate	[K/m]
M	Planet mass	[kg]
M_M	Molecular mass	[kg/mol]
M	Mach number	[-]
N_A	Avogadro constant	[mol^{-1}]
N_p	Population size	[-]
N_r	Maximum number of replacements	[-]
m	Mass	[kg]
p	Pressure	[N/m^2]

p	Mutation chance	[-]
$\dot{q}_c$	Convective heating rate	[W/m^2]
q_{dyn}	Dynamic pressure	[N/m^2]
R	Radius	[m]
R	Specific gas constant	[J/kg/K]
R_c	Cross-range	[rad]
R_d	Downrange	[rad]
$\mathcal{R}^*$	Absolute gas constant	[J/mol/K]
Re	Reynolds Number	[-]
r	Radial distance	[m]
S	Aerodynamic side-force	[N]
S_{ref}	Aerodynamic reference area	[m^2]
s	Sine	[-]
T	Temperature	[K]
T	Neighborhood size	[-]
T_M	Molecular scale temperature	[K]
t	Time	[s]
U	Gravitational potential	[m^2/s^2]
u	x-component of a velocity vector	[m/s]
u	Control variable	
V	Velocity	[m/s]
v	y-component of a velocity vector	[m/s]
w	z-component of a velocity vector	[m/s]
x	Cartesian coordinate of position	[m]
y	Cartesian coordinate of position	[m]
z	Cartesian coordinate of position	[m]

Greek Symbols

α	Angle of attack	[rad]
β	Side-slip angle	[rad]
γ	Specific heat ratio	[-]
γ	Flight-path angle	[rad]
δ	Geocentric latitude or declination	[rad]
δ	Control-surface deflection	[rad]
ϵ	Error tolerance	
η	Mutation distribution index	[-]
θ	Arbitrary angle	[rad]
μ	Gravitational parameter	[m^3/s^2]
ρ	Density	[kg/m^3]
ρ	Radius of convergence	
σ	Bank angle	[rad]
τ	Longitude	[rad]
ϕ	Geodetic latitude	[rad]
χ	Heading angle	[rad]
ω	Rotational rate	[rad/s]

Indices

∞	Free stream
0	Initial
A	Airspeed-based

AA	Defined in the aerodynamic reference frame, airspeed-based
AG	Defined in the aerodynamic reference frame, groundspeed-based
abs	Absolute
B	Defined in the body reference frame
b	body flap
D	Aerodynamic drag
db	Dead band
E	Earth
e	Error
e	Elevon
G	Gravity
G	Groundspeed-based
I	Defined in the inertial planetocentric reference frame
L	Aerodynamic lift
l	Left
R	Defined in the rotating planetocentric reference frame
rel	Relative
r	Right
S	Aerodynamic side-force
T	Target
V	Defined in the vertical reference frame
W	Wind
x	Component in x-direction
y	Component in y-direction
z	Component in z-direction
δ	Geocentric latitude or declination
γ	Flight-path angle
σ	Bank angle
τ	Longitude
χ	Heading angle

Acronyms and Abbreviations

DE	Differential Evolution
DFPM	Double-false position method
FPI	Fixed-point iteration
GA	Genetic Algorithms
GRAM	Global Reference Atmospheric Model
IVP	Initial Value Problem
MOEA/D	Multi-Objective Evolutionary Algorithm based on Decomposition
NSGA-II	Non-dominated Sorting Genetic Algorithm II
ODE	Ordinary Differential Equation
PaGMO	Parallel Global Multi-objective Optimizer
RK	Runge-Kutta
RKF	Runge-Kutta-Fehlberg
SPCV	Spherical Position with Cartesian Velocity
TAEM	Terminal Area Energy Management
TSI	Taylor Series Integration
Tudat	TU Delft Astrodynamics Toolbox
US76	United States Standard Atmosphere 1976
WGS84	World Geodetic System 1984

Contents

1 Introduction **1**

1.1 Reentry Simulation . . . 1
1.2 Taylor Series Integration . . . 2
1.3 Research Questions and Tasks . . . 4
1.4 Layout of This Report . . . 5

2 Flight Mechanics **7**

2.1 Reference Frames . . . 7
2.2 Frame Transformations . . . 9
2.3 State Variables . . . 14
2.3.1 Cartesian Components . . . 15
2.3.2 Spherical Components . . . 15
2.3.3 Spherical Position with Cartesian Velocity . . . 15
2.3.4 Conversions between State-Variable Sets . . . 15
2.4 Environment Models . . . 17
2.4.1 Planet Shape . . . 17
2.4.2 Gravity Field . . . 18
2.4.3 Atmosphere . . . 19
2.5 External Forces . . . 23
2.5.1 Aerodynamic Forces . . . 23
2.5.2 Gravity . . . 24
2.6 Translational Equations of Motion . . . 25
2.6.1 Cartesian State Variables . . . 25
2.6.2 Spherical State Variables . . . 27
2.6.3 Spherical Position with Cartesian Velocity . . . 28
2.7 The Influence of Wind . . . 28
2.7.1 Cartesian State Variables . . . 28
2.7.2 Spherical State Variables . . . 30
2.7.3 Spherical Position with Cartesian Velocity . . . 30

3 Mission Characteristics **33**

3.1 HORUS-2B . . . 33
3.1.1 Reference Mission . . . 34
3.1.2 Aerodynamics . . . 35
3.2 Controls . . . 36
3.2.1 Control Profile . . . 36
3.2.2 Bank Reversals . . . 38
3.3 Trajectory Constraints . . . 39
3.4 Downrange and Cross-Range . . . 40
3.5 Heat Load . . . 41

4 Numerical Methods **43**

4.1 Root-Finding . . . 43
4.1.1 Fixed-Point Iteration . . . 43

4.1.2 Newton's Method . . . 44
4.1.3 Double-False Position Method . . . 44
4.2 Interpolation . . . 45
4.2.1 Linear Interpolation . . . 45
4.2.2 Single Polynomial . . . 45
4.2.3 Cubic Splines . . . 47
4.3 Least-Squares Regression . . . 49
4.4 Integration . . . 50
4.4.1 Integrating Ordinary Differential Equations . . . 51
4.4.2 Errors . . . 51
4.4.3 Runge-Kutta . . . 53
4.5 Optimization . . . 54
4.5.1 Basics . . . 55
4.5.2 Constraint Handling . . . 55
4.5.3 Differential Evolution . . . 56
4.5.4 Multi-Objective Optimization . . . 57
4.5.5 Non-dominated Sorting Genetic Algorithm II . . . 58
4.5.6 Multi-Objective Evolutionary Algorithm based on Decomposition . . . 58

5 Taylor Series Integration 61
5.1 Basics . . . 62
5.2 Recurrence relations for Basic Mathematical Operations . . . 63
5.3 Solving an Ordinary Differential Equation . . . 64
5.4 Order and Time-Step Size . . . 66
5.4.1 Time-Step Size . . . 67
5.4.2 Order . . . 68
5.5 Discontinuities . . . 69
5.5.1 Altitude, Mach Number and Energy . . . 71
5.5.2 Bank Reversals . . . 72
5.5.3 Terminal Distance . . . 73
5.5.4 Constraint Violations . . . 75
5.6 Discrete Data . . . 77
5.6.1 US76 Tables . . . 77
5.6.2 Aerodynamics . . . 79
5.6.3 Wind Model . . . 81

6 Reentry Recurrence Relations 83
6.1 Cartesian State Variables . . . 83
6.1.1 Gravity . . . 84
6.1.2 Aerodynamic Forces without Wind . . . 86
6.1.3 Aerodynamic Forces with Wind . . . 90
6.1.4 Rotating Planetocentric Frame . . . 93
6.2 Spherical State Variables . . . 93
6.2.1 Gravity . . . 96
6.2.2 Aerodynamic Forces without Wind . . . 97
6.2.3 Aerodynamic Forces with Wind . . . 98
6.3 Spherical Position with Cartesian Velocity . . . 99
6.3.1 Gravity . . . 100
6.3.2 Aerodynamic Forces without Wind . . . 100
6.3.3 Aerodynamic Forces with Wind . . . 101

6.4 Environment Models . . . 101
6.4.1 Planet Shape . . . 102
6.4.2 Exponential Atmosphere . . . 102
6.4.3 United States Standard Atmosphere 1976 . . . 103
6.5 Aerodynamics . . . 106
6.6 Constraints . . . 108

7 Software **109**
7.1 External Software . . . 109
7.1.1 Tudat, Eigen and Boost . . . 109
7.1.2 PaGMO . . . 110
7.1.3 GRAM 1999 . . . 110
7.2 Custom Software . . . 110
7.2.1 Runge-Kutta . . . 111
7.2.2 Taylor Series Integration . . . 112
7.2.3 Wind Model . . . 113
7.2.4 Optimization . . . 115
7.3 Verification and Validation . . . 115
7.3.1 Verification of the Numerical Methods . . . 115
7.3.2 Verification of the US76 Fits . . . 119
7.3.3 Validation of the Equations of Motion . . . 120

8 Simulation Results **123**
8.1 Optimal Settings for Taylor Series Integration . . . 123
8.1.1 Step-size Control . . . 123
8.1.2 Order Control . . . 127
8.1.3 State-Variable Set . . . 130
8.1.4 Wind . . . 131
8.2 Integration . . . 133
8.2.1 Equations of Motion without Wind . . . 133
8.2.2 Equations of Motion with Wind . . . 134
8.3 Optimization . . . 135
8.3.1 Flight to TAEM . . . 135
8.3.2 Downrange and Integrated Heat Load . . . 137
8.4 Sensitivity Analysis . . . 140
8.4.1 Control Perturbation . . . 140
8.4.2 Wind Perturbation . . . 146

9 Conclusion and Recommendations **149**
9.1 Conclusions . . . 149
9.2 Recommendations . . . 151

Bibliography **153**

A Tables of US76 **157**

B Aerodynamic Tables of HORUS-2B **159**

C 2D Kepler Orbit **161**

D Fixed Order Timing **165**

E Sensitivity Analysis Graphs **169**

Chapter 1

Introduction

Reentry is the step from space to the surface of a planet (or another large object in space) in the possession of an atmosphere. When entering the atmosphere, an object, for instance a reentry vehicle, has a large amount of kinetic and potential energy. Most of this energy will be dissipated by the atmosphere through drag. For reentry vehicles, it is important to know what path through the atmosphere it can take without burning up or getting pulverized by the aerodynamic forces. Unfortunately, the mathematical characteristics of reentry are very complex, as the vehicle travels through all layers of the atmosphere at high speed. A trajectory can also be very sensitive to small changes in the motion high up in the atmosphere, leading to a completely different landing/impact zone. This non-linearity is the reason that reentry trajectories are typically computed using numerical methods. Numerically optimizing a reentry mission is very hard due to the complex dynamics, meaning that up to 100,000 or more trajectories may have to be simulated before an optimal result is obtained. Similarly, determining the impact of different types of disturbances on a trajectory may also take a large number of simulations. Therefore, one will benefit from having a fast and reliable trajectory propagation tool. In this MSc. thesis research, it is investigated whether Taylor Series Integration (TSI) could be such a tool.

In the following sections, the history of reentry simulation is briefly discussed, followed by a description of TSI and its history, and the research questions. The last two sections of this chapter are dedicated to the research questions and the layout of this report, respectively.

1.1 Reentry Simulation

Atmospheric entry dates back much further than human history, as meteors and other objects from space have been impacting on Earth since its birth. Hence the picture on the cover of this report, which depicts an artists impression of a large meteor hitting an unfortunate distant planet. The first man-made object to reenter at hypersonic speed would be a modified version of Von Braun's V-2 rocket, launched from White Sands Missile Range in 1946. The first human reentering the Earth's atmosphere was the Russian Yuri Gagarin in the Vostok-1 in 1961. This reentry involved the cosmonaut ejecting from the craft at 7 km altitude, while it was still falling at supersonic velocity, and parachuting separately to the surface.

Reentry trajectories can be divided into three different types, being ballistic, gliding and skipping entry, which are shown in Figure 1.1. Ballistic entry features zero or close-to-zero aerodynamic lift, and applies mainly to capsules and objects that fall down without guidance, such as space debris or meteors. Gliding entry does feature lift, which allows for lower aerodynamic and thermal loads on the vehicle and greatly extends the flight time and range in the atmosphere. This applies mostly to winged reentry vehicles, with a popular example being the Space Shuttle, although capsules can also generate lift by moving the center of gravity to the side, as this tilts the capsule a little during flight. An additional advantage of having lift (and having control

over it) is that the reentry vehicle can be steered, which in turn gives control over the landing location. Skipping flight involves generating enough lift to skip out of the atmosphere, possibly even multiple times, greatly extending the range of the vehicle.

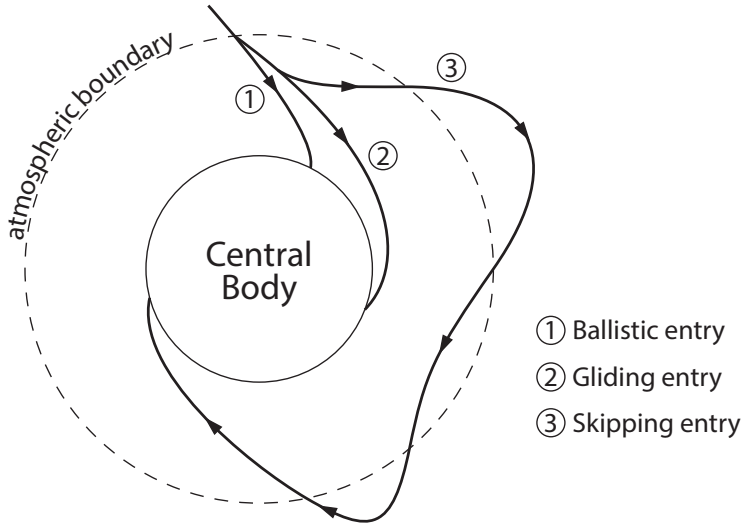


Figure 1.1: The three different types of reentry trajectories.

One of the first to recognize the long range that could be reached by hypersonic vehicles was Eugen Sänger (1905-1964). He exploited this feature in the 1930s and early 1940s in his design for a long-range bomber. This Silbervogel would fly a skipping trajectory from Germany to the Japanese held Pacific, along the way releasing a large bomb above the United States [61]. However, the project was canceled before the end of World War II due to budgetary reasons.

Before computers became available that were fast enough to do reentry simulation, reentry trajectories were analyzed using analytical models. The first major breakthroughs on these models after World War II were due to Allen and Eggers in 1958 [2], analyzing the aerodynamic heating of intercontinental ballistic missiles, and Chapman in 1959 [8], who developed an approximate methods for studying the entry into planetary atmospheres. The early analytical models were all first-order methods that only apply to parts of the reentry. These would form the basis for the more accurate second-order solution developed by Loh in the late 1960s [39, 40]. The analytic approximations have later been summarized in different books, for instance by Vinh et al. in 1980 [66].

Nowadays, the equations of motion for reentry can be simulated numerically and this is typically done using numerical integrators such as the methods of Runge-Kutta, Adams-Bashforth or Adams-Moulton. These traditional methods have in common that they sample the equations of motion at different points in time to predict the state at the end of a time step. The advantage of such methods is that the equations of motion can easily be replaced by other differential equations.

1.2 Taylor Series Integration

TSI differs from the traditional integration methods in that it propagates the state by setting up its Taylor series expansion about the current point in time. A Taylor series consists of the

derivatives of the variables up to a specific order and these derivatives are determined using automatic differentiation. Automatic differentiation can obtain the derivatives of a mathematical expression up to any order by rewriting these expressions into recurrence relations. These relations compute the higher-order terms of a Taylor series from the lower ones.

TSI has a long history. For instance, Barrio [3] states that Newton already used recursive computation of Taylor series for the solution of differential equations, which must have been in the late 17^{th} or early 18^{th} century. The art of recursive generation of Taylor series was first seen in the twentieth century when Airey [1] attempted to incorporate recursion into the Taylor series in 1932. In 1946, Miller [41] had developed fully recursive schemes for Taylor series, so that he could numerically compute the so-called Airy integral for the British Association Mathematical Table. Despite the fact that one could now compute Taylor series up to arbitrary order, doing so by hand is a very tedious and error-prone process. In the 1960s and 1970s, the first translator programs were written that could translate user-defined differential equations to a computer code that generates the Taylor series and could then be used to solve the system of equations [22, 36, 56]. Barton and his colleagues were the first to write a program that would accept a system of differential equations without the user having to manipulate it in some way [4, 5].

Among the first to use TSI was Moore [48], who used it for his invention, interval analysis, which allows one to obtain bounds on the error of integration. Moore was in that era also the main promoter of TSI as replacement for the, in his opinion, less efficient traditional numerical integrators. He argued that the main objection to TSI, namely the need to evaluate derivatives up to a high order, is overrated, as a computer can do this recursively and Runge-Kutta integrators of the same order require more arithmetic operations than TSI [49].

Comparing TSI with other integrators for efficiency was not done much, as it required comparing the number of arithmetic operations rather than the more conventional comparison of the number of function evaluations. The conclusion of Hull et al. was that TSI, which can be of higher order than its competitors, fares better than Runga-Kutta and variable-order Adams methods when the accuracy requirements are more stringent [27]. Similar results were found in [28], where it was stated that for high accuracy, i.e., about 16 significant digits, TSI could be up to twenty times faster.

From the 1970s onwards, TSI was analyzed thoroughly, especially by Chang and Corliss, focusing on elements such as step-size control and expanding the range of problems TSI could be applied to (see for instance [7, 9–12]). In the 1990s it was first used as a method to solve differential algebraic equations, i.e., systems of differential equations and additional constraint equations [10, 59]. From there on out it was also used for the determination and computation of periodic orbits in dynamical systems [25, 58], i.e., repeating behaviors in these systems, such as walking motions, where the same motion is performed every n number of steps, and orbits of objects in space. At the same time, the combination of TSI and interval analysis was rediscovered as means to obtain validated solutions for both differential equations and differential algebraic equations [26, 51, 60]. TSI was also already in its early days used for celestial mechanics [18], but was later rediscovered as being a particularly fast method [62], yielding on average 15.8 times lower computational times than the Runge-Kutta-Fehlberg integrators that are typically used for this type of problems.

1.3 Research Questions and Tasks

The strength of TSI is that it is very adaptive in terms of order and step size (as will be explained in Chapter 5). Changing these settings requires very little computational (and programming) effort, and thus TSI can be customized for any problem, allowing it, in theory, to solve any problem with the highest efficiency. This has given it an edge over the traditional methods for celestial mechanics and now it is of interest whether this is also the case for reentry, which has different equations of motion. The main research question that is to be answered by this thesis is:

Does Taylor Series Integration with automatic differentiation have a better performance for reentry trajectory propagation than traditional numerical integrators?

Better performance is here defined as having a lower computational time, while maintaining the same level of accuracy of the results.

In Section 5.4, different strategies for the control of order and step size of TSI will be given and to maximize the performance of TSI, the best combination of strategies has to be determined. Furthermore, the location of the vehicle and its velocity and velocity direction can be described by different state-variable sets, each having different equations of motion, as will be discussed in Chapter 2. The choice of state-variable set could impact the performance of TSI, in which case the optimal set should be determined. Overall, this can be summarized by the first sub-question of this research:

Which combination of order control, step-size control and state-variable set optimizes the performance of Taylor Series Integration?

Once TSI is optimized for reentry, its performance can be compared to that of the best traditional integrator for reentry, which was found to be Runge-Kutta-Fehlberg 5(6) (RKF5(6)) [45]. The two integrators will first be compared for reentry integration, to answer the second sub-question:

Does Taylor Series Integration have better performance than traditional integrators for reentry integration?

After that, the two integrators can be compared for optimization and sensitivity analysis, which are two common fields of reentry research:

Does Taylor Series Integration have better performance than traditional integrators for reentry optimization and sensitivity analysis, while yielding the same solutions?

To answer these questions, a TSI program will be coded that can handle the reentry equations of motion. This program and the pieces of code needed for the comparison between integrators will have to be thoroughly tested and verified to make sure that the results are reliable. In the comparison for integration, computational time will mainly be of interest, to determine which integrator is faster while integrating the same trajectories. For the optimization and sensitivity analysis, the focus will be on the correctness of the results. TSI has to achieve the same (or better) results if it is to outperform the other integrators.

The vehicle that will be used for the reentry simulations is the HORUS-2B, as there is a complete aerodynamic database available for it [47**?**]. HORUS is an Earth reentry vehicle, so Earth environment models will be used. The planet shape will be approximated by an

ellipsoid, flattened at the poles and bulging at the equator. The Earth's gravity field will be approximated by spherical harmonics truncated at degree 2 and order 0. Including higher degree and order terms is considered to add too little improvement in accuracy for the amount of work this requires. The exponential atmosphere and the United States Standard Atmosphere 1976 (US76) will be used as atmosphere models. The first is a simple model that can be used to test TSI without having to address discontinuities or discrete data (it will be explained in Chapter 5 that these features cannot be integrated directly by TSI). The second is a commonly used model that does contain these features. The 1999 version of Global Reference Atmospheric Model (GRAM) will supply the wind model, which is mainly because this model is directly available.

1.4 Layout of This Report

Chapter 2 introduces reentry mechanics. This chapter contains the equations of motion, a description of the used environment models and explains the concepts of state-variable sets, reference frames and frame transformations. Chapter 3 introduces the used reentry vehicle, HORUS-2B, and its reference mission. Furthermore, this chapter explains how the controls of the vehicle are defined, together with the trajectory constraints and different optimization goals. Chapter 4 lists the different numerical methods used, which include root-finding, interpolation, least-squares regression and numerical optimization. This chapter also explains the concept of the numerical integration of ordinary differential equations and the Runge-Kutta method of numerical integration. TSI will then be explained in Chapter 5, together with the concepts of automatic differentiation and recurrence relations. It will also explain TSI's application to differential equations, strategies for step-size and order control and the application to piece-wise functions and discrete data. Next, Chapter 6 will list all recurrence relations for reentry. Chapter 7 will list the software used during this research, including both external and home-made software. It will also show the results of the verification and validation of the software. In Chapter 8, the results of the simulations will be given. This chapter is divided into four parts: determining the optimal settings for TSI and the comparison between the integrators for integration, optimization and sensitivity analysis. In Chapter 9, the conclusions of this research and recommendations for future research are given.

Chapter 2

Flight Mechanics

In the present chapter, the mechanics of reentry are presented, with the goal to derive the equations of motion for a Shuttle-like reentry vehicle. Mechanics can be divided into statics, dynamics and kinematics. Statics addresses the state of a body under influence of an equilibrium of forces. Dynamics, on the other hand deals with the inequilibrium of forces that cause a change in the motion or attitude of the body. Kinematics then solely focuses on the motion of that body, without reference to mass or forces [46].

The basis of the mechanics in this chapter is formed by Newton's Laws of Motion for a single particle (or point-mass) [46]:

I In the absence of forces, a particle is either at rest or moves in a straight line with constant speed.

II A particle experiencing a force F experiences an acceleration a related to F by $F = ma$, where m is the mass of the particle. Alternatively, force is proportional to the time derivative of momentum.

III Whenever a first particle exerts a force F_{12} on a second particle, the second particle exerts a force F_{21} on the first. F_{12} and F_{21} are equal in magnitude and opposite in direction.

Note that these laws only hold in an inertial reference frame.

The reentry vehicle will be simulated as a point-mass and the rotational dynamics and kinematics are not taken into account. Thus the equations have three degrees of freedom and only the translational equations of motion will be given in this chapter. Two additional assumptions for these equations are: the vehicle's mass is constant during the entire reentry and the Earth rotates with constant angular velocity about the Z-axis of the inertial/rotating reference frame.

This chapter starts with the specification of a number of reference frames, followed by the transformations between these frames. Next, the different sets of state variables that will be used are specified. The environment models and the external forces acting on the vehicle are given in Sections 2.4 and 2.5, respectively. As the rotational dynamics are not taken into account, the external moments are not given. In Section 2.6, the translational equations of motion are given, followed by a discussion on how wind will change these equations in Section 2.7.

2.1 Reference Frames

This section describes the different reference frames that will be used in the equations of motion. The reference frames described here are all right-handed, have three orthonormal axes and were

obtained from [46]. Each reference frame has its own index and vectors defined in a particular reference frame will also get this index. If a vector has two indices separated by a comma, the second one will denote the reference frame. The next section will provide a description of the transformations between the different reference frames.

Note that for the aerodynamic and trajectory reference frames, a distinction is made between groundspeed, i.e., the velocity of the vehicle w.r.t. the rotating planet and the airspeed, which is the velocity w.r.t. the local atmosphere. These velocities are the same when wind is absent.

Inertial Planetocentric Reference Frame

This frame is denoted by $\mathcal{F}_I$. The origin of this frame is located at the center of mass of the Earth. The Z_I-axis is oriented along the spin-axis of the planet and points North. The X_I-axis points at the Greenwich meridian at some reference point in time, in this case the start of the simulation. The Y_I-axis points to the equator, east of the X_I and is perpendicular to the X_I and Z_I-axis.

Rotating Planetocentric Reference Frame

This frame is denoted by $\mathcal{F}_R$. The origin of this frame is also located at the center of mass of the Earth. The Z_R-axis is aligned with the spin-axis of the planet and points North. X_R will point to the intersection of the Greenwich meridian with the equator. Y_R points to the equator at 90° longitude. At $t = 0$, $\mathcal{F}_I$ and $\mathcal{F}_R$ are perfectly aligned.

Vertical Reference Frame

This frame is denoted by $\mathcal{F}_V$. The origin of the vertical frame is the center of mass of the reentry vehicle. The Z_V-axis points down towards the center of mass of the Earth. X_V and Y_V span a plane perpendicular to the line connecting the centers of mass of the vehicle and the Earth, i.e., the local horizontal plane. X_V always points north and Y_V points east.

Body Reference Frame

This frame is denoted by $\mathcal{F}_B$. This frame also has the origin at the center of mass of the vehicle. The axes are aligned with the vehicle, with X_B and Z_B spanning the plane of symmetry. In case of a Shuttle-like vehicle, X_B points towards the nose of the vehicle, Z_B points downwards and Y_B points in the direction of the right wing and is perpendicular to the plane of symmetry. In case of a capsule, it can be convention to have X_B pointing backwards, Z_B pointing up and Y_B pointing to the right side, meaning that capsule enters the atmosphere rear first.

Wind Reference Frame

This frame is denoted by $\mathcal{F}_W$. The X_W-axis points in the same direction as the wind-velocity vector (wind velocity w.r.t. the rotating planet). The Z_W-axis is located in the local vertical plane, points downwards in case of horizontal wind. Y_W completes the right-handed system.

Trajectory Reference Frame

This frame is denoted by $\mathcal{F}_{TG}$ for the groundspeed based frame and $\mathcal{F}_{TA}$ for the airspeed based frame. The origin is in both cases located at the center of mass of the vehicle.

The X_{TG}-axis points in the direction of the velocity of the vehicle w.r.t. the rotating planet surface. Z_{TG} is, just like X_{TG}, located in the vertical plane, but rotated by 90° and pointing downward. Y_{TG} completes the right-handed system (pointing towards the right wing when the vehicle is not side-slipping or banking).

The X_{TA}-axis points in the direction of the velocity of the vehicle w.r.t. the atmosphere. Z_{TA} is, just like X_{TA}, located in the vertical plane, but rotated by 90° and pointing downward. Y_{TA} again completes the right-handed system (pointing towards the right wing when the vehicle is not side-slipping or banking).

Aerodynamic Reference Frame

This frame is denoted by $\mathcal{F}_{AG}$ for the groundspeed based frame and $\mathcal{F}_{AA}$ for the airspeed based frame. The origin is located at the center of mass of the vehicle.

The X_{AG}-axis points in the direction of the velocity of the vehicle w.r.t. the rotating planet surface. Z_{AG} is in the plane of symmetry of the vehicle, pointing in the opposite direction of the lift-force (based on groundspeed). Y_{AG} completes the right-handed system (pointing towards the right wing when the vehicle is not side-slipping).

The X_{AA}-axis points in the direction of the velocity of the vehicle w.r.t. the atmosphere. Z_{AA} is in the plane of symmetry, pointing in the opposite direction of the lift-force (based on airspeed). Y_{AA} again completes the right-handed system (pointing towards the right wing when the vehicle is not side-slipping).

2.2 Frame Transformations

This section will present the basic transformations between the different frames of the previous section. The definition of these transformations were obtained from [46] and [43]. Note that multiple of these transformations can be combined to form the transformation between any two frames.

Before the actual transformations are presented, the underlying math is given. There are different ways of expressing the transformations between reference frames, one of them being unit axis-rotations. These are the rotation by an arbitrary angle about one of the axes of a reference frame. In Figure 2.1 such a rotation is given (positive rotation, according to the right-hand rule). From this figure, one can derive the position in frame B as function of the position in frame A and the angle θ:

$$\begin{aligned} x_B &= x_A \\ y_B = a + b &= y_A \cos\theta + z_A \sin\theta \\ z_B = -d + (c + d) &= -y_A \sin\theta + z_A \cos\theta \end{aligned}$$

In matrix form this becomes:

$$\begin{bmatrix} x_B \\ y_B \\ z_B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_A \\ y_A \\ z_A \end{bmatrix} \tag{2.1}$$

Here, the matrix with cosine and sine terms is the transformation matrix $\boldsymbol{C}_1(\theta)$ (subscript 1 denotes a rotation about the X-axis). The three transformation matrices for rotations about

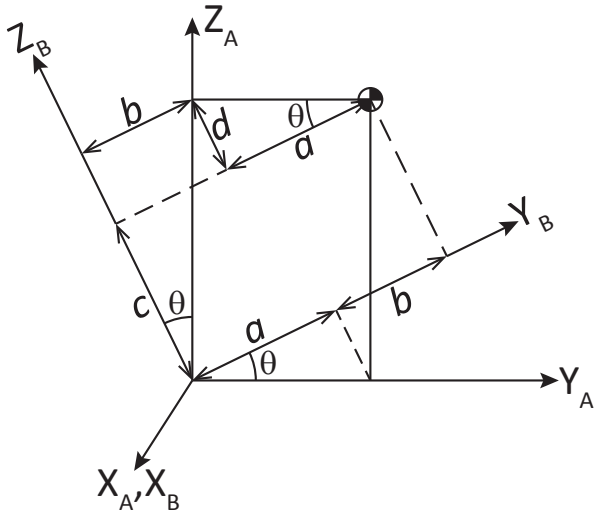


Figure 2.1: The transformation from reference frame A to reference frame B.

respectively the X, Y and Z-axis are:

$$\boldsymbol{C}_1(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix} \tag{2.2}$$

$$\boldsymbol{C}_2(\theta) = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \tag{2.3}$$

$$\boldsymbol{C}_3(\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

Any rotation can be written as a series of rotations about unit-axes, which means that any rotation can be written as the product of matrices $\boldsymbol{C}_1$, $\boldsymbol{C}_2$ and $\boldsymbol{C}_3$. Another property of these matrices is that they are orthonormal [46], which means that the inverse of these matrices is equal to the transpose. This also means that the product of a series of transformation matrices is again orthonormal.

The inverse of a series of transformations is obtained by inverting the sign of the argument of the transformations and inverting their order [46]. If, for example, the rotation from frame A to B (notation: $\boldsymbol{C}_{B,A}$) consists of first a rotation of angle θ_1 about the X-axis, followed by a rotation of angle θ_2 about the Y-axis and then a rotation of angle θ_3 about the Z-axis, the resulting transformation matrix is:

$$\boldsymbol{C}_{B,A} = \boldsymbol{C}_3(\theta_3)\boldsymbol{C}_2(\theta_2)\boldsymbol{C}_1(\theta_1) \tag{2.5}$$

and the inverse is obtained from:

$$\boldsymbol{C}_{A,B} = \boldsymbol{C}_1(-\theta_1)\boldsymbol{C}_2(-\theta_2)\boldsymbol{C}_3(-\theta_3) = \boldsymbol{C}_{B,A}^T \tag{2.6}$$

Next, the basic frame transformations are given.

Rotating Planetocentric to Inertial Planetocentric Reference Frame

The rotating frame and inertial frame share the same origin and have coinciding Z-axes. These frames are shown together in Figure 2.2. In this figure, the vector $\boldsymbol{r}$ denotes the position of the center of mass of the vehicle. The rotating frame rotates with angular velocity ω_E about the Z-axis. ω_E is the rotational velocity of Earth, which is equal to $7.292115{\cdot}10^{-5}$ rad/s [69]. If the two frames coincide at the start of the simulation and t is the amount of time that has past since then, the angle between X_I and X_R is equal to $\omega_E t$ and the transformation matrix is given by:

$$\boldsymbol{C}_{I,R} = \boldsymbol{C}_3(-\omega_E t) = \begin{bmatrix} \mathrm{c}\omega_E t & -\mathrm{s}\omega_E t & 0 \\ \mathrm{s}\omega_E t & \mathrm{c}\omega_E t & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.7}$$

From here on, cosine and sine will be denoted in transformation matrices by c and s, respectively, to reduce the size of the matrices.

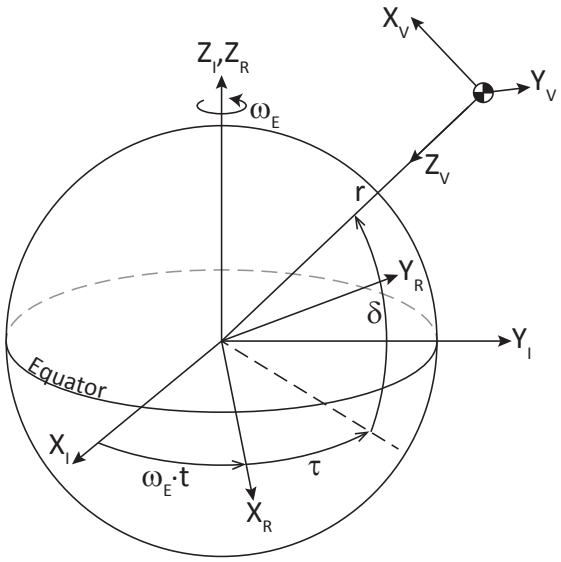


Figure 2.2: Relation between the inertial, rotating and vertical reference frame (all angles are shown positive).

Vertical to Rotating Planetocentric Reference Frame

In spherical coordinates, the position of a vehicle is given by geocentric longitude τ and latitude (or declination) δ w.r.t. a rotating Earth (and distance r to the center of Earth), as shown in Figure 2.2. The transformation from $\mathcal{F}_V$ to $\mathcal{F}_R$ is a rotation $\delta + \frac{\pi}{2}$ about the Y-axis, followed by a rotation $-\tau$ about the (rotated) Z-axis:

$$\boldsymbol{C}_{R,V} = \boldsymbol{C}_3(-\tau)\boldsymbol{C}_2(\delta + \frac{\pi}{2}) = \begin{bmatrix} -\mathrm{c}\tau\mathrm{s}\delta & -\mathrm{s}\tau & -\mathrm{c}\tau\mathrm{c}\delta \\ -\mathrm{s}\tau\mathrm{s}\delta & \mathrm{c}\tau & -\mathrm{s}\tau\mathrm{c}\delta \\ \mathrm{c}\delta & 0 & -\mathrm{s}\delta \end{bmatrix} \tag{2.8}$$

Vertical to Inertial Planetocentric Reference Frame

This transformation matrix is given by the multiplication of matrices $\boldsymbol{C}_{I,R}$ and $\boldsymbol{C}_{R,V}$. Since the rotation of $\mathcal{F}_R$ to $\mathcal{F}_I$ and the second rotation of $\mathcal{F}_V$ to $\mathcal{F}_R$ are both about the Z-axis, they can be combined as a single rotation. Defining the celestial longitude as $\tilde{\tau} = \omega_E t + \tau$, which is

the longitude w.r.t. the X-axis of the inertial frame. The transformation matrix is given by:

$$\boldsymbol{C}_{I,V} = \boldsymbol{C}_3(-\tilde{\tau})\boldsymbol{C}_2(\delta + \frac{\pi}{2}) = \begin{bmatrix} -\mathrm{c}\tilde{\tau}\mathrm{s}\delta & -\mathrm{s}\tilde{\tau} & -\mathrm{c}\tilde{\tau}\mathrm{c}\delta \\ -\mathrm{s}\tilde{\tau}\mathrm{s}\delta & \mathrm{c}\tilde{\tau} & -\mathrm{s}\tilde{\tau}\mathrm{c}\delta \\ \mathrm{c}\delta & 0 & -\mathrm{s}\delta \end{bmatrix} \tag{2.9}$$

Trajectory to Vertical Reference Frame

The relation between the trajectory frames and the vertical frame is given by the flight-path angle (γ_G for groundspeed-based and γ_A for airspeed-based) and the heading (χ_G for the groundspeed-based and χ_A for the airspeed-based). In Figure 2.3, these angles are shown. Note that in this figure and in rotation matrix the subscripts A and G are not shown, but that one can use either the groundspeed variable set γ_G and χ_G or the airspeed set γ_A and χ_A. The rotation from the trajectory to the vertical frame consists of a negative flight-path angle about the Y-axis, followed by a negative heading angle about the Z-axis:

$$\boldsymbol{C}_{V,T} = \boldsymbol{C}_3(-\chi)\boldsymbol{C}_2(-\gamma) = \begin{bmatrix} \mathrm{c}\chi\mathrm{c}\gamma & -\mathrm{s}\chi & \mathrm{c}\chi\mathrm{s}\gamma \\ \mathrm{s}\chi\mathrm{c}\gamma & \mathrm{c}\chi & \mathrm{s}\chi\mathrm{s}\gamma \\ -\mathrm{s}\gamma & 0 & \mathrm{c}\gamma \end{bmatrix} \tag{2.10}$$

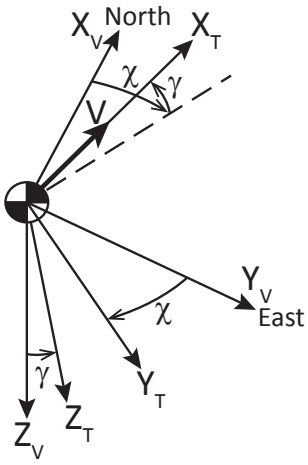


Figure 2.3: The relation between the vertical and trajectory frame (the angles are shown positive).

Trajectory to Inertial Planetocentric Reference Frame

The transformation matrix $\boldsymbol{C}_{I,T}$ can be obtained in different ways. One of them is to combine the matrices $\boldsymbol{C}_{I,V}$ and $\boldsymbol{C}_{V,T}$:

$$\begin{aligned} \boldsymbol{C}_{I,T} &= \boldsymbol{C}_{I,V}\boldsymbol{C}_{V,T} \\ &= \boldsymbol{C}_3(-\tilde{\tau})\boldsymbol{C}_2(\delta + \frac{\pi}{2})\boldsymbol{C}_3(-\chi)\boldsymbol{C}_2(-\gamma) \end{aligned} \tag{2.11}$$

This matrix is not written out in full, due to its long entries and the fact that this would give little insight.

Another option to compute $\boldsymbol{C}_{I,T}$ is by using vector calculus, which may be more efficient. For this method, either the groundspeed $\boldsymbol{V}_G$ can be used to obtain $\boldsymbol{C}_{I,TG}$ or the airspeed $\boldsymbol{V}_A$ can be used to obtain $\boldsymbol{C}_{I,TA}$. In case wind is absent, $\boldsymbol{V}_A$ is equal to $\boldsymbol{V}_G$, which is computed in Section 2.3.4. The calculation of $\boldsymbol{V}_{A,I}$ in case of wind is explained in Section 2.7. Here, the

computation of $\boldsymbol{C}_{I,TA}$ using $\boldsymbol{V}_A$ is shown. For this, the unit vectors $\boldsymbol{X}_{TA,I}$, $\boldsymbol{Y}_{TA,I}$ and $\boldsymbol{Z}_{TA,I}$, which point in the directions of the axes of $\mathcal{F}_{TA}$ in the inertial frame, have to be obtained. $\boldsymbol{X}_{TA,I}$ is found by noting that it is defined in the direction of the airspeed vector in $\mathcal{F}_I$, $\boldsymbol{V}_{A,I}$:

$$\boldsymbol{X}_{TA,I} = \frac{\boldsymbol{V}_{A,I}}{||\boldsymbol{V}_A||} \tag{2.12}$$

As can be seen in Figure 2.4, $\boldsymbol{V}_{A,I}$ (and $\boldsymbol{X}_{TA,I}$), $\boldsymbol{Z}_{TA,I}$ and position vector $\boldsymbol{r}_I$ are located in the same vertical plane. $\boldsymbol{Y}_{TA,I}$ is perpendicular to this plane, since it is perpendicular to $\boldsymbol{X}_{TA,I}$ and $\boldsymbol{Z}_{TA,I}$, and can be found with the cross-product of $\boldsymbol{V}_{A,I}$ and $\boldsymbol{r}_I$:

$$\boldsymbol{Y}_{TA,I} = \frac{\boldsymbol{V}_{A,I} \times \boldsymbol{r}_I}{||\boldsymbol{V}_{A,I} \times \boldsymbol{r}_I||} \tag{2.13}$$

This equation also holds when the heading of $\boldsymbol{V}_A$ is rotated, since $\boldsymbol{Y}_{TA,I}$ rotates along with $\boldsymbol{V}_A$. Since $\boldsymbol{Z}_{TA,I}$ is perpendicular to both $\boldsymbol{X}_{TA,I}$ and $\boldsymbol{Y}_{TA,I}$, it is found with:

$$\boldsymbol{Z}_{TA,I} = \boldsymbol{X}_{TA,I} \times \boldsymbol{Y}_{TA,I} \tag{2.14}$$

The transformation matrix $\boldsymbol{C}_{I,TA}$ is then given by:

$$\boldsymbol{C}_{I,TA} = \begin{bmatrix} \boldsymbol{X}_{TA,I} & \boldsymbol{Y}_{TA,I} & \boldsymbol{Z}_{TA,I} \end{bmatrix} \tag{2.15}$$

In Chapter 6, the two methods will be compared to determine which is more efficient for TSI.

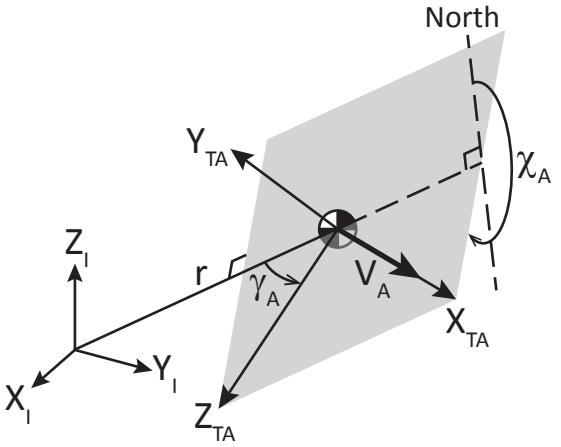


Figure 2.4: $\boldsymbol{X}_{TA}$, $\boldsymbol{Y}_{TA}$, $\boldsymbol{Z}_{TA}$ and the vertical plane spanned by $\boldsymbol{r}$, $\boldsymbol{X}_{TA}$ and $\boldsymbol{Z}_{TA}$ in grey.

Wind to Vertical Reference Frame

The transformation of the wind frame to the vertical frame is the same as that of the trajectory to the vertical frame. The only difference is that it is based on the wind velocity-vector V_W rather than the groundspeed or airspeed of the vehicle. Therefore, the geometry in Figure 2.3 is valid for $\mathcal{F}_W$ as well. The direction of the wind velocity w.r.t. the vertical frame consists of wind flight-path angle γ_W and wind heading angle χ_W. The matrix for this transformation is:

$$\boldsymbol{C}_{V,W} = \boldsymbol{C}_3(-\chi_W)\boldsymbol{C}_2(-\gamma_W) = \begin{bmatrix} \mathrm{c}\chi_W\mathrm{c}\gamma_W & -\mathrm{s}\chi_W & \mathrm{c}\chi_W\mathrm{s}\gamma_W \\ \mathrm{s}\chi_W\mathrm{c}\gamma_W & \mathrm{c}\chi_W & \mathrm{s}\chi_W\mathrm{s}\gamma_W \\ -\mathrm{s}\gamma_W & 0 & \mathrm{c}\gamma_W \end{bmatrix} \tag{2.16}$$

Aerodynamic to Trajectory Reference Frame

The rotation of $\mathcal{F}_{AG}$ to $\mathcal{F}_{TG}$ is the groundspeed-based bank angle σ_G about the X-axis. The rotation of $\mathcal{F}_{AA}$ to $\mathcal{F}_{TA}$ is the airspeed-based bank angle σ_A about the X-axis (see Figure 2.5). Both rotations are thus are given by:

$$\boldsymbol{C}_{T,A} = \boldsymbol{C}_1(\sigma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathrm{c}\sigma & \mathrm{s}\sigma \\ 0 & -\mathrm{s}\sigma & \mathrm{c}\sigma \end{bmatrix} \tag{2.17}$$

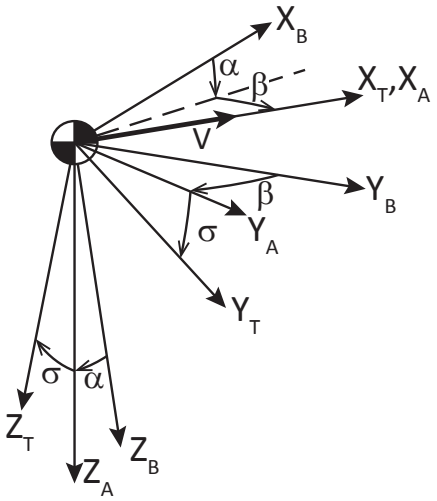


Figure 2.5: The relation between a trajectory, aerodynamic and body frame (angles are shown positive).

Body to Aerodynamic Reference Frame

This transformation starts with a rotation of the negative angle of attack α about the Y-axis, followed by a rotation of the positive side-slip angle β about the Z-axis (see Figure 2.5). For this transformation, the airspeed based angles α_A and β_A are used to transform to $\mathcal{F}_{AA}$ and the groundspeed-based angles α_G and β_G are used to transform to $\mathcal{F}_{AG}$ (in Figure 2.5, the subscripts A and G were omitted). The transformation matrix is then, without subscripts, given by:

$$\boldsymbol{C}_{A,B} = \boldsymbol{C}_3(\beta)\boldsymbol{C}_2(-\alpha) = \begin{bmatrix} \mathrm{c}\beta\mathrm{c}\alpha & \mathrm{s}\beta & \mathrm{c}\beta\mathrm{s}\alpha \\ -\mathrm{s}\beta\mathrm{c}\alpha & \mathrm{c}\beta & -\mathrm{s}\beta\mathrm{s}\alpha \\ -\mathrm{s}\alpha & 0 & \mathrm{c}\alpha \end{bmatrix} \tag{2.18}$$

As will be explained in Section 3.2, the groundspeed angles α_G, β_G and σ_G are control variables and do not originate from reentry mechanics. Their airspeed counterparts are computed as shown in Section 2.7.

2.3 State Variables

Before the translational equations of motion can be derived, one has to specify the state variables to represent the position and velocity of the reentry vehicle. As was mentioned in the introduction, the optimal state-variable set for TSI still has to be determined, so different sets are given here: Cartesian components, spherical components and spherical position with Cartesian velocity (SPCV). These are the general types of state variables; sets can contain variations

of these, such as Cartesian coordinates in different reference frames or a combination of spherical position and Cartesian velocity. The sets listed here are the most commonly used ones and were obtained from [43].

2.3.1 Cartesian Components

Cartesian components consist of x, y and z to specify the position of the center of mass of a vehicle and their derivatives w.r.t. time u, v and w to specify the velocity. The Cartesian position coordinates can be specified in either $\mathcal{F}_I$ or $\mathcal{F}_R$ and the velocity can be defined in either $\mathcal{F}_I$, $\mathcal{F}_R$ or $\mathcal{F}_V$. Velocity in $\mathcal{F}_V$ will only be used for the SPCV set in Section 2.3.3.

2.3.2 Spherical Components

The spherical components, when specifying the position w.r.t. $\mathcal{F}_R$, are distance r and geocentric longitude τ and latitude δ. The velocity is given by its magnitude V_G (groundspeed), the flight-path angle γ_G and heading angle χ_G. See Figure 2.6 for the definition of the angles. τ and χ_G take on values between -180° and 180°, where δ and γ_G have a range of -90° to 90°.

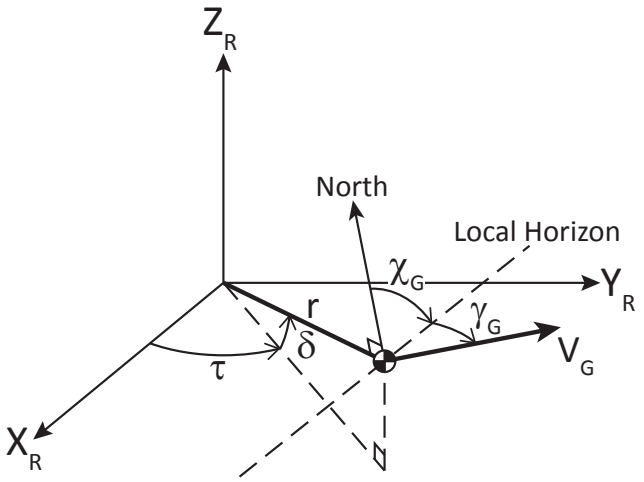


Figure 2.6: The definition of the spherical state variables (all angles are shown positive).

2.3.3 Spherical Position with Cartesian Velocity

The SPCV set also uses the spherical components for position, r, τ and δ, but the velocity is now given by the Cartesian velocity components u_V, v_V and w_V, defined in the vertical reference frame.

2.3.4 Conversions between State-Variable Sets

First of all, the conversion of Cartesian to spherical position is given. The conversion of Cartesian position from $\mathcal{F}_I$ to $\mathcal{F}_R$ or vice versa is simply a frame transformation. The spherical position

variable τ is defined in $\mathcal{F}_R$, so it is most easily obtained from the Cartesian position variables in $\mathcal{F}_R$:

$$\sin\tau = \frac{y_R}{\sqrt{x_R^2 + y_R^2}} \qquad \cos\tau = \frac{x_R}{\sqrt{x_R^2 + y_R^2}} \qquad \tan\tau = \frac{y_R}{x_R} \tag{2.19}$$

Using the position in $\mathcal{F}_I$ in these relations allows one to obtain $\tilde{\tau}$, instead of τ. r and δ can be obtained from the Cartesian position in either $\mathcal{F}_I$ or $\mathcal{F}_R$ using:

$$r = ||\boldsymbol{r}|| = \sqrt{x^2 + y^2 + z^2} \tag{2.20}$$

$$\sin\delta = \frac{z}{r} \qquad \cos\delta = \frac{\sqrt{x^2 + y^2}}{r} \qquad \tan\delta = \frac{z}{\sqrt{x^2 + y^2}} \tag{2.21}$$

where x, y and z should either all be specified in $\mathcal{F}_I$ or $\mathcal{F}_R$.

Next up is the conversion of velocity components. $\mathcal{F}_I$ is the only frame in which inertial velocity is used; in the other frames only groundspeed or airspeed is used. The groundspeed in $\mathcal{F}_I$, $\boldsymbol{V}_{G,I}$, is computed from $\boldsymbol{V}_I$ with:

$$\boldsymbol{V}_{G,I} = \boldsymbol{V}_I - \boldsymbol{\omega}_{E,I} \times \boldsymbol{r}_I \tag{2.22}$$

with $\boldsymbol{\omega}_{E,I} = (0\ 0\ \omega_E)^T$ being the rotation of the Earth about the Z_I-axis. This groundspeed vector can then be transformed to the other frames. Spherical component V_G can then be obtained from the groundspeed vector in an arbitrary frame using:

$$V_G = ||\boldsymbol{V}_G|| = \sqrt{u_G^2 + v_G^2 + w_G^2} \tag{2.23}$$

χ_G and γ_G are defined in $\mathcal{F}_V$, so they have to be obtained from the groundspeed in $\mathcal{F}_V$, using:

$$\sin\chi_G = \frac{v_{G,V}}{\sqrt{u_{G,V}^2 + v_{G,V}^2}} \qquad \cos\chi_G = \frac{u_{G,V}}{\sqrt{u_{G,V}^2 + v_{G,V}^2}} \qquad \tan\chi_G = \frac{v_{G,V}}{u_{G,V}} \tag{2.24}$$

$$\sin\gamma_G = -\frac{w_{G,V}}{V_G} \qquad \cos\gamma_G = \frac{\sqrt{u_{G,V}^2 + v_{G,V}^2}}{V_G} \qquad \tan\gamma_G = -\frac{w_{G,V}}{\sqrt{u_{G,V}^2 + v_{G,V}^2}} \tag{2.25}$$

Eqs. (2.24) and (2.25) can also be used to obtain the airspeed based angles χ_A and γ_A when using $\boldsymbol{V}_{A,V}$ instead of $\boldsymbol{V}_{G,V}$. Note that during simulation it is more accurate to compute the sines and cosines of an angle directly from the equations above than it is to first compute the angle itself and then the sine or cosine from this angle.

The conversion of spherical to Cartesian components is then as follows: the position in $\mathcal{F}_R$ is obtained from:

$$x_R = r\cos\tau\cos\delta \tag{2.26}$$

$$y_R = r\sin\tau\cos\delta \tag{2.27}$$

$$z_R = r\sin\delta \tag{2.28}$$

The groundspeed in $\mathcal{F}_V$ can then be computed with:

$$u_{G,V} = V_G\cos\chi_G\cos\gamma_G \tag{2.29}$$

$$v_{G,V} = V_G\sin\chi_G\cos\gamma_G \tag{2.30}$$

$$w_{G,V} = -V_G\sin\gamma_G \tag{2.31}$$

The position or velocity in another reference frame then requires the appropriate frame transformation(s).

2.4 Environment Models

This section lists the models for the planetary environment. As was discussed in the introduction, an Earth reentry vehicle will be used and thus only environmental models for Earth or general models will be treated. The following subsections discuss the three types of environment models used: planet shape, gravity field and atmosphere.

2.4.1 Planet Shape

The shape of the planet determines the altitude of a vehicle above the planet's surface for a given location w.r.t. the planet's center. The local atmospheric properties are a function of altitude and thus depend on the description of the Earth's shape. Furthermore, the altitude determines the current phase of the descent; whether the vehicle should prepare to land, deploy parachutes or brace for impact.

Note that the shape of the Earth is highly complex, as the surface is made up of oceans, deep valleys and high mountains. An approximation to this shape is an ellipsoid, flattened at the poles and bulging at the equator (with constant radius along the equator). The parameters, which determine this shape, are the equatorial radius R_E and the flattening f defined as:

$$f = 1 - \frac{R_P}{R_E} \tag{2.32}$$

or one can use the eccentricity e:

$$e = 1 - \frac{R_P^2}{R_E^2} = 2f - f^2 \tag{2.33}$$

where R_P is the polar radius of Earth. The values for R_E and f were obtained from the World Geodetic System 1984 (WGS84), namely $R_E = 6378.137$ km and $f = 3.35281066475 \cdot 10^{-3}$ [52]. One should note that the value for R_E is outdated, but when changing the value of R_E, one also has to determine a new value for f, so the WGS84 value should be used for altitude computation.

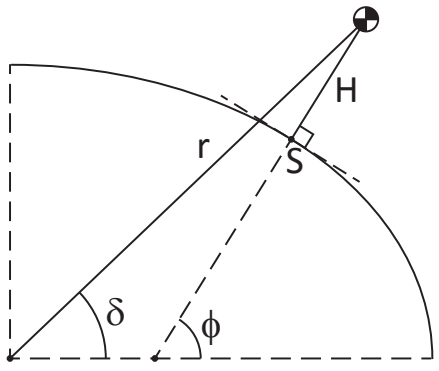


Figure 2.7: Definition of the parameters for the ellipsoid

In case of an ellipsoid, one uses the geodetic altitude H, which is is defined along a line perpendicular to the surface, as can be seen in Figure 2.7. In this figure, ϕ is the geodetic latitude and δ is the geocentric latitude of the object, the latter being one of the spherical position variables in the previous section.

To determine H exactly, an iterative process is required, which can, for instance, be found in [67]. A simplified version of a single iteration of the method described there is the approximation:

$$H \approx r - R_E \sqrt{\frac{1-e^2}{1-e^2\cos^2\delta}} \tag{2.34}$$

which will be used in the current study.

2.4.2 Gravity Field

Earth's gravity field can be modeled in different ways, ranging from simple approximations to complex, nearly exact ones. In any case, the gravity acceleration g that acts on an object within the Earth's gravity field can be found from:

$$\boldsymbol{g} = -\boldsymbol{\nabla} U \tag{2.35}$$

Where U is the gravitational potential of the planet. Because Earth has a complex shape and mass distribution, its gravity potential U has to be modeled as an infinite series of spherical harmonics [68]:

$$U = -\frac{\mu_E}{r}\left\{1 + \sum_{n=2}^{\infty}\sum_{m=0}^{n}\left(\frac{R_E}{r}\right)^n P_{n,m}\left(\sin\delta\right)\left[C_{n,m}\cos m\tau + S_{n,m}\sin m\tau\right]\right\} \tag{2.36}$$

where μ_E and R_E are the gravitational parameter and mean equatorial radius of the Earth. Using the values found by the GRACE satellite, they are equal to $3.986004415{\cdot}10^{14}$ $\mathrm{m}^3/\mathrm{s}^2$ and $6.3781363{\cdot}10^6$ m, respectively [65]. The terms $C_{n,m}$ and $S_{n,m}$ given by:

$$C_{n,m} = J_{n,m}\cos m\tau_{n,m} \tag{2.37}$$

$$S_{n,m} = J_{n,m}\sin m\tau_{n,m} \tag{2.38}$$

In these expressions, n and m mark the degree and order, respectively, of the spherical harmonics parameters. $J_{n,m}$ and $\tau_{n,m}$ are model parameters, with $J_{n,m}$ marking the magnitude of the $C_{n,m}$ and $S_{n,m}$-terms in Eq. (2.36) and $\tau_{n,m}$ being the longitude belonging to that term, with all $\tau_{n,0}$-terms being equal to 0. $P_{n,m}\left(\sin\delta\right)$ are the Legendre polynomials of degree n and order m with $\sin\delta$ as argument, given by:

$$\begin{gathered} x = \sin\delta \\ P_n(x) = \frac{1}{(-2)^n n!}\frac{d^n}{dx^n}\left(1-x^2\right)^n \\ P_{n,m}(x) = \left(1-x^2\right)^{m/2}\frac{d^m P_n(x)}{dx^m} \end{gathered} \tag{2.39}$$

Here, the spherical harmonics series will be approximated by its first term, $n = 2$ and $m = 0$, which gives the following expression for the gravitational acceleration vector [68]:

$$\boldsymbol{g} = -\frac{\mu_E}{r^3}\begin{pmatrix}x\\y\\z\end{pmatrix}\left\{1 + \frac{3}{2}J_2\left(\frac{R_E}{r}\right)^2\left[\begin{pmatrix}1\\1\\3\end{pmatrix} - 5\frac{z^2}{r^2}\right]\right\} \tag{2.40}$$

The value for Earth's J_2-coefficient (which is $J_{2,0}$) found by the GRACE satellite is $1.0826357{\cdot}10^{-3}$ [65].

2.4.3 Atmosphere

This section will address three different atmosphere models, being the exponential atmosphere, US76 and GRAM 1999. These models determine how the atmospheric quantities such as air density ρ, temperature T and pressure p depend on the position in the atmosphere. In Chapter 5, it will explained that difficulties arise for TSI when it is applied to discontinuities and discrete data. The first of the models, the exponential atmosphere, consists of a single equation for the entire atmosphere and uses no data tables. This model can be used to assess whether TSI succeeds in general to integrate a reentry trajectory with the required accuracy. US76 uses different equation sets for each of the layers of the atmosphere and uses data tables for the higher layers. This model is commonly used as it is a more accurate representation of Earth's atmosphere. In Chapter 5, techniques are described for dealing with discontinuities and discrete data and if this model is successfully implemented with TSI, then so are these techniques. The final model, GRAM 1999, is too complex to describe analytically and is only used as the basis for a wind model.

Table 2.1: Constants for the atmosphere models. Source: [53]

Name	Symbol	Value
Gravity acceleration at sea level at 45° latitude [m/s^2]	g_0	9.80665
Radius of Earth at 45° latitude [m]	R_0	$6.356766 \cdot 10^6$
Molecular mass of the atmosphere at sea level [kg/mol]	M_0	0.0289644
Absolute gas constant [J/mol/K]	$\mathcal{R}^*$	8.31432
Specific heat ratio [-]	γ	1.400
Avogadro constant [mol^{-1}]	N_A	$6.02217 \cdot 10^{23}$

Exponential Atmosphere

The exponential atmosphere gives the air density as a simple exponential function of altitude. The equations and their derivations have been obtained from [46].

The derivation of the exponential atmosphere starts with the assumption that Earth's atmosphere is an ideal gas. In that case, the ideal gas law holds, given by:

$$p = \rho R T \tag{2.41}$$

In this equation, p, ρ and T are the atmospheric pressure, density and temperature, respectively. R is the specific gas constant, the value of which depends on the type of gas. It is related to the absolute gas constant $\mathcal{R}^*$ as:

$$R = \frac{\mathcal{R}^*}{M} \tag{2.42}$$

where M is the molecular mass of the gas, which is assumed constant and equal to 28.9644 g/mol, giving R a value of 287.053 J/kg/K. Another assumption made is that the atmosphere is in hydrostatic equilibrium, in which case the the hydrostatic law holds:

$$dp = -\rho g dH \tag{2.43}$$

where g is the local gravity acceleration. The next two assumptions are that gravity is approximately constant and equal to g_0, which is the gravity acceleration at sea level, and that temperature is constant. Using these assumptions and inserting Eq. (2.41) into Eq. (2.43) yields:

$$dp = -\rho g_0 dH = RT d\rho$$

$$\frac{d\rho}{\rho} = -\frac{g_0}{RT} dH \tag{2.44}$$

Integration of Eq. (2.44) results in the equation for the exponential atmosphere:

$$\frac{\rho}{\rho_0} = e^{-\frac{g_0 H}{RT}} = e^{-\frac{H}{H_S}} \tag{2.45}$$

where ρ_0 is the atmospheric density at sea level and H_S is the scale height. The typical value for the scale height of the Earth is 7050 m [46], which corresponds to a temperature of about 240 K.

United States Standard Atmosphere 1976

The US76 model is a steady-state atmosphere model, it models the vertical distribution of a number of atmospheric properties based on mean global conditions. Unlike the exponential atmosphere, US76 does not assume constant gravity, temperature and molecular mass of the atmosphere. The equations for this model and their derivations have been obtained from [53].

Table 2.2: Parameter values for the first layers of the atmosphere. Source: [53]

i	z_i [km]	T_{M_i} [K]	L_{M_i} [K/km]	p_i [Pa]
0	0	288.15	-6.5	101325
1	11	216.65	0	22632.06
2	20	216.65	1	5474.889
3	32	228.65	2.8	868.0187
4	47	270.65	0	110.9063
5	51	270.65	-2.8	66.93887
6	71	214.65	-2	3.956420
7	84.85205	186.9459		0.3733805

First two new variables are defined, which ease the derivation of the equations by combining the variation of two parameters into one parameter. The first is the geopotential altitude z, which is defined as:

$$g_0 dz = g dH \tag{2.46}$$

Where g_0 is the gravity acceleration at sea level and g is the acceleration at an altitude H. Approximating the gravity field of Earth as an inverse square relation between gravity acceleration and radial distance, g and g_0 are related as:

$$g = g_0 \left(\frac{R_0}{R_0 + H} \right)^2 \tag{2.47}$$

Note that R_0 is the radius of Earth at the same latitude as for which g_0 is given (see Table 2.1). These values are not completely accurate, but rather they are constants defined for the US76 model. Inserting Eq. (2.47) into Eq. (2.46) and integrating yields the relation between z and H:

$$z = \frac{R_0 H}{R_0 + H} \tag{2.48}$$

The second parameter is the molecular-scale temperature, which is given by:

$$T_M = T \frac{M_0}{M_M} \tag{2.49}$$

where M_M and M_0 are the local molecular mass and the molecular mass at sea level (given in Table 2.1), respectively.

When the geometric altitude H is smaller than 86 km, T_M is a linear piecewise continuous function of altitude given by:

$$T_M = T_{M_i} + L_{M_i}(z - z_i) \tag{2.50}$$

where L_{M_i}, z_i and T_{M_i} are the lapse rate for a particular layer i of the atmosphere, the geopotential altitude at the lower boundary of that layer and the molecular-scale temperature at that boundary, respectively. These quantities are given in Table 2.2. Note that T_{M_i} and p_i (except for the first row) have been obtained through calculation, using the previous layer value and Eq. (2.50) and Eq. (2.51) or Eq. (2.52). Note also that layer 6 ends at $H = 86$ km, at which $z \approx 84.852$ km. Up to $H = 80$ km, M_M is kept constant and equal to M_0. Between 80 and 86 km, a small correction is applied to the ratio M_M/M_0, which tabulated in Appendix A.

The atmospheric pressure p can be calculated with Eq. (2.51) or Eq. (2.52), depending on whether the lapse rate is zero or not, respectively.

$$p = p_i \exp\left[\frac{-g_0 M_0 (z - z_i)}{\mathcal{R}^* T_{M_i}}\right] \tag{2.51}$$

$$p = p_i \left(\frac{T_{M_i}}{T_M}\right)^{g_0 M_0 / \mathcal{R}^* L_{M_i}} \tag{2.52}$$

Here, p_i is the pressure at the start of each layer, which can be found in Table 2.2. The air density can finally be found using the ideal gas law (Eq. (2.41)), but now corrected with the molecular-scale temperature:

$$\rho = \frac{p M_0}{\mathcal{R}^* T_M} \tag{2.53}$$

Table 2.3: Parameters for the higher layers of the atmosphere. Source: [53]

Symbol	Value	Symbol	Value
T_7	186.8672 K	T_9	240 K
		L_9	12 K/km
		H_9	110 km
T_c	263.1905 K	T_∞	1000 K
A	-76.3232 K	T_{10}	360 K
b	19.9429 km	λ	0.01875 km^{-1}
H_8	91 km	H_{10}	120 km

The temperature profiles for the next four layers of the atmosphere are a function of H, rather than z. For (86 km $\leq H \leq$ 91 km), T is constant and equal to $T_7 = 186.8672$ K. For (91 km $< H <$ 110 km), T is given by:

$$T = T_c + A\sqrt{1 - \left(\frac{H - H_8}{b}\right)^2} \tag{2.54}$$

with T_c, A, b and H_8 being constants, equal to respectively 263.1905 K, -76.3232 K, 19.9429 km and 91 km. For (110 km $\leq H \leq$ 120 km), T is given by:

$$T = T_9 + L_9 (H - H_9) \tag{2.55}$$

with T_9, L_9 and H_9 constants, equal to respectively 240 K, 12 K/km and 110 km. For altitudes larger than 120 km, an exponential function of altitude is used for the temperature:

$$T = T_\infty - (T_\infty - T_{10}) \exp\left[-\lambda \frac{(H - H_{10})(R_0 + H_{10})}{R_0 + H}\right] \tag{2.56}$$

where T_∞, T_{10}, λ and H_{10} are equal to 1000 K, 360 K, 0.01875 km^{-1} and 120 km, respectively.

Pressure for these four layers is then calculated using:

$$p = \frac{N\mathcal{R}^* T}{N_A} \tag{2.57}$$

Here, N is the sum of the number density of each gas species at a specific altitude and N_A is Avogadro's constant (given in Table 2.1). Due to its complex derivation, the sum of the number densities is best obtained from linear interpolation of tabular data [46]. The air density is again calculated using the ideal gas law, although noting that M_M is not constant, ρ is given by:

$$\rho = \frac{pM_M}{\mathcal{R}^* T} \tag{2.58}$$

M_M also has to be found by interpolation of tabular data. The tables for both N and M_M can be found in Appendix A.

Global Reference Atmospheric Model 1999

US76 is a standard atmosphere model, i.e., a model that only depends on altitude and is independent of horizontal position. The GRAM models, on the other hand, are reference atmosphere models, which means that they also include geographical and time effects. These effects include the local climate, diurnal and seasonal changes and solar heating. For GRAM 1999, the data for different layers of the atmosphere have been obtained from various sources, as is shown in Figure 2.8.

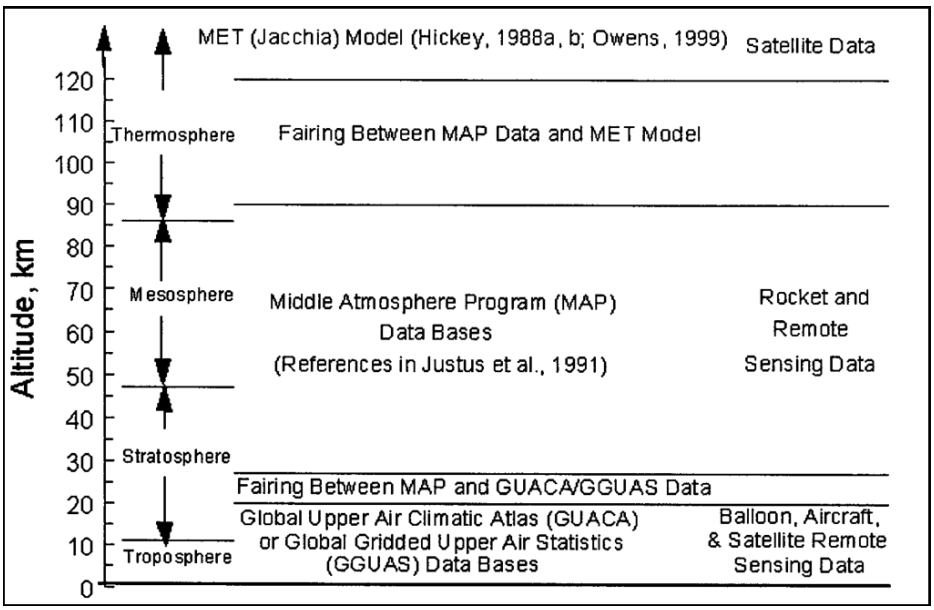


Figure 2.8: The data sources for the GRAM 1999 model of each layer of the atmosphere. Source: [31]

For the atmosphere above 90 km, the Marshall Engineering Thermosphere (MET) model of 1999

is used. Between 20 and 120 km, the Middle Atmosphere Program (MAP) data is used. Between 0 and 27 km, the Global Upper Air Climatic Atlas (GUACA) is used. In the overlapping regions of these models, a fairing method is used to provide smooth transition between the models. For more information, the reader referred to [31] and the references therein.

The inputs of GRAM 1999 are altitude H and geocentric longitude τ and latitude δ. The outputted local wind consists of the wind velocities in north-south, east-west and up-down directions. Note that H has to be computed using the reference ellipsoid of IAU76 [32], which differs slightly from the one of WGS84, which was used in Section 2.4.1. The values for the equatorial radius of Earth is in this case 6378140 m (3 m more than the value of WGS84) and the flattening is $3.35281 \cdot 10^{-3}$ (which is up to 6 digits equal to WGS84's flattening) [37].

Mach Number and Speed of Sound

The Mach number M is the ratio between the velocity of air passing an object and the local speed of sound:

$$M_\infty = \frac{V_\infty}{a_\infty} \tag{2.59}$$

where V and a are magnitude of the airspeed and the speed of sound, respectively, and the subscript ∞ was added to the terms to mark that they are qualities of the undisturbed airflow, the so-called free stream. The Mach number is one of the parameters that determine the magnitude of the aerodynamic forces acting on the reentry vehicle. The exact dependencies on the Mach number are discussed in Section 3.1.2.

The speed of sound a is defined as the speed at which sound waves can propagate through a substance, in this case air. The local speed of sound is given by [46]:

$$a = \sqrt{\gamma R T} = \sqrt{\frac{\gamma \mathcal{R}^* T}{M_M}} \tag{2.60}$$

with γ being the specific heat ratio of the local atmosphere, i.e., the ratio between the specific heat at constant pressure C_p and at constant volume C_v (note that R is also defined as $C_p - C_v$). γ typically takes on the value of 1.4 for the Earth atmosphere [53]. In case of the exponential atmosphere, where R and T are assumed constant, the speed of sound is constant and equal to 310.564 m/s.

2.5 External Forces

The external forces on a reentry vehicle consist of aerodynamic forces, gravity and thrust. Since the reference vehicle does not use thrust during reentry (and since thrust is in general uncommon for reentry), only gravity and aerodynamic forces are included here. All equations in this section were obtained from [43].

2.5.1 Aerodynamic Forces

The aerodynamic forces are typically given in the airspeed-based aerodynamic frame by:

$$\boldsymbol{F}_{A,AA} = \begin{bmatrix} -D_A \\ -S_A \\ -L_A \end{bmatrix} = \begin{bmatrix} -C_D q_{dyn} S_{ref} \\ -C_S q_{dyn} S_{ref} \\ -C_L q_{dyn} S_{ref} \end{bmatrix} \tag{2.61}$$

where D_A, S_A and L_A are the (airspeed-based) aerodynamic drag, side-force and lift, and C_D, C_L and C_S are their dimensionless coefficients, respectively. S_{ref} is the aerodynamic reference area (for HORUS-2B, this is equal to the wing surface area plus the area of the lower fuselage surface between the wings) and q_{dyn} is the dynamic pressure, given by:

$$q_{dyn} = \frac{1}{2}\rho V_A^2 \tag{2.62}$$

with V_A being the magnitude of the airspeed vector $\boldsymbol{V}_A$, defined in an arbitrary reference frame:

$$V_A = ||\boldsymbol{V}_A|| = \sqrt{u_A^2 + v_A^2 + w_A^2} \tag{2.63}$$

Note that for this equation u_A, v_A and w_A have to be in the same reference frame. When wind is absent, the airspeed vector $\boldsymbol{V}_A$ is equal to the groundspeed vector $\boldsymbol{V}_G$, which can be computed from the inertial velocity with Eq. (2.22). When wind is included in a simulation, $\boldsymbol{V}_A$ is computed as discussed in Section 2.7.

The coefficients in Eq. (2.61) depend on the aerodynamic angles α_A and β_A, on deflection angles of the vehicle's control surfaces and on the Mach number M, which is given by Eq. (2.59). In case there is no wind, α_A and β_A are equal to α_G and β_G, respectively. Otherwise, they are computed as discussed in Section 2.7. Since the rotational dynamics of the vehicle are not simulated, the angles α_G, β_G and σ_G are control variables that are specified by the user (or a guidance system) for each point along a trajectory (see Section 3.2 for the definition of the control profile). The computation of the aerodynamic coefficients for HORUS-2B is explained in Section 3.1.2.

2.5.2 Gravity

The equations for the gravity acceleration including the spherical-harmonics term J_2 were given in Section 2.4.2 and shown here once more:

$$\boldsymbol{g} = -\frac{\mu_E}{r^3}\begin{pmatrix} x \\ y \\ z \end{pmatrix}\left\{1 + \frac{3}{2}J_2\left(\frac{R_E}{r}\right)^2\left[\begin{pmatrix} 1 \\ 1 \\ 3 \end{pmatrix} - 5\frac{z^2}{r^2}\right]\right\} \tag{2.64}$$

Note that this equation is independent of longitude and is therefore valid for position variables in both $\mathcal{F}_I$ and $\mathcal{F}_R$. That is, using x, y and z defined in $\mathcal{F}_I$ will result in the gravity forces in $\mathcal{F}_I$ and using coordinates in $\mathcal{F}_R$ will result in the gravity forces in $\mathcal{F}_R$.

The gravity acceleration vector in spherical components is given in $\mathcal{F}_V$ by:

$$\boldsymbol{g} = [g_\delta \quad 0 \quad g_r]^T \tag{2.65}$$

where g_r and g_δ are the downward and northward gravity forces, which for the J_2 gravity model are given by:

$$g_r = \frac{\mu_E}{r^2}\left[1 - \frac{3}{2}J_2\left(\frac{R_E}{r}\right)^2\left(3\sin^2\delta - 1\right)\right] \tag{2.66}$$

$$g_\delta = -3J_2\frac{\mu_E}{r^2}\left(\frac{R_E}{r}\right)^2\sin\delta\cos\delta \tag{2.67}$$

2.6 Translational Equations of Motion

In this section, the translational equations of motion will be presented. The derivation of these equations and definition of all accompanying variables were obtained from [43]. This section is divided into three parts, discussing the equations of motion for Cartesian state variables, spherical state variable and spherical position with Cartesian velocity, respectively. The additional equations needed to include wind can be found in the next section.

2.6.1 Cartesian State Variables

As was mentioned in Section 2.3, the Cartesian state can be either given in $\mathcal{F}_I$ or $\mathcal{F}_R$. First, the equations for the state variables in $\mathcal{F}_I$ are given. Then the differences for the variables in $\mathcal{F}_R$ are explained.

Inertial Planetocentric Frame

The Cartesian state variables in $\mathcal{F}_I$ are in vector form given by:

$$\text{Position: } \boldsymbol{r}_I = \begin{pmatrix} x_I & y_I & z_I \end{pmatrix}^T \qquad \text{Velocity: } \boldsymbol{V}_I = \begin{pmatrix} u_I & v_I & w_I \end{pmatrix}^T$$

The kinematic and dynamic equations of motion are then respectively:

$$\frac{d\boldsymbol{r}_I}{dt} = \begin{pmatrix} \dot{x}_I & \dot{y}_I & \dot{z}_I \end{pmatrix}^T = \boldsymbol{V}_I \tag{2.68}$$

$$\frac{d\boldsymbol{V}_I}{dt} = \begin{pmatrix} \dot{u}_I & \dot{v}_I & \dot{w}_I \end{pmatrix}^T = \boldsymbol{g}_I + \frac{1}{m}\boldsymbol{F}_{A,I} \tag{2.69}$$

where $\boldsymbol{g}_I$ are the gravity accelerations and $\boldsymbol{F}_{A,I}$ are aerodynamic forces, respectively. $\boldsymbol{g}_I$ is given by Eq. (2.64) and the aerodynamic force vector is given in $\mathcal{F}_{AA}$ by Eq. (2.61). To calculate the aerodynamic forces, the airspeed vector is needed. When there is no wind, airspeed is identical to groundspeed. Otherwise, it is computed from the groundspeed in Section 2.7. The groundspeed itself can be obtained from the inertial velocity using Eq. (2.22).

The aerodynamic forces have to be transformed to the inertial frame, for which transformation matrix $\boldsymbol{C}_{I,AA}$ is needed, which was given in Section 2.2. In that section, two methods were given for obtaining $\boldsymbol{C}_{I,TA}$, from which $\boldsymbol{C}_{I,AA}$ is determined, one using transformation matrices with the angles $\tilde{\tau}$, δ, χ_A and γ_A and one using vector calculus. For the first method, $\boldsymbol{C}_{I,AA}$ is given by:

$$\begin{aligned} \boldsymbol{C}_{I,AA} &= \boldsymbol{C}_{I,T}\boldsymbol{C}_{T,AA} \\ &= \boldsymbol{C}_3(-\tilde{\tau})\boldsymbol{C}_2(\delta + \frac{\pi}{2})\boldsymbol{C}_3(-\chi_A)\boldsymbol{C}_2(-\gamma_A)\boldsymbol{C}_1(\sigma_A) \end{aligned} \tag{2.70}$$

For the vector method, $\boldsymbol{C}_{I,AA}$ is given by:

$$\boldsymbol{C}_{I,AA} = \begin{bmatrix} \boldsymbol{X}_{TA,I} & \boldsymbol{Y}_{TA,I} & \boldsymbol{Z}_{TA,I} \end{bmatrix} \boldsymbol{C}_1(\sigma_A) \tag{2.71}$$

However, both methods do not compute the correct transformation when γ_A is equal to $\pm 90°$. When that happens, both $u_{V,A}$ and $v_{V,A}$ are zero, which means that, according to Eq. (2.24), the sine and cosine of χ_A are equal to 0/0, which is usually interpreted by a computer as not-a-number. This is a computational issue for the method using transformation matrices. For

the vector method, Eq. (2.13) gives $\boldsymbol{Y}_{TA,I} = \boldsymbol{0}$ in this case, since $\boldsymbol{V}_{A,I}$ and $\boldsymbol{r}_I$ coincide. Thus for the second method, both lift and side-force will be zero after the transformation to $\mathcal{F}_I$. In practice, vertical flight occurs during rocket ascent or a parachute descent, but not during the hypersonic reentry, so this will not be a problem.

Rotating Planetocentric Frame

The Cartesian state variables in $\mathcal{F}_R$ are in vector form given by:

$$\text{Position: } \boldsymbol{r}_R = \begin{pmatrix} x_R & y_R & z_R \end{pmatrix}^T \qquad \text{Velocity: } \boldsymbol{V}_{G,R} = \begin{pmatrix} u_{G,R} & v_{G,R} & w_{G,R} \end{pmatrix}^T$$

The kinematic equations of motion are the same as for their inertial counterparts:

$$\frac{d\boldsymbol{r}_R}{dt} = \boldsymbol{V}_{G,R} \tag{2.72}$$

The dynamic equations differ though, because Newton's laws do not hold in a rotating frame. This can be solved by adding additional terms to the dynamic equations. The derivation of these terms starts with the inertial frame $\mathcal{F}_I$ and the rotating frame $\mathcal{F}_R$, both with the same origin. $\mathcal{F}_R$ rotates with constant angular velocity ω_E about the inertial frame. The relationship between the rate of change of position in an inertial and a rotating frame is [50]:

$$\frac{d\boldsymbol{r}_I}{dt} = \frac{d\boldsymbol{r}_R}{dt} + \boldsymbol{\omega}_{E,R} \times \boldsymbol{r}_R \tag{2.73}$$

Differentiation of Eq. (2.73) consists of the pure time derivative and a rotation induced effect:

$$\frac{d^2\boldsymbol{r}_I}{dt^2} = \frac{d}{dt}\left(\frac{d\boldsymbol{r}_R}{dt} + \boldsymbol{\omega}_E \times \boldsymbol{r}_R\right) + \boldsymbol{\omega}_E \times \left(\frac{d\boldsymbol{r}_R}{dt} + \boldsymbol{\omega}_E \times \boldsymbol{r}_R\right) \tag{2.74}$$

$\boldsymbol{\omega}_E$ is constant, so this equation becomes:

$$\frac{d^2\boldsymbol{r}_I}{dt^2} = \frac{d^2\boldsymbol{r}_R}{dt^2} + 2\boldsymbol{\omega}_E \times \frac{d\boldsymbol{r}_R}{dt} + \boldsymbol{\omega}_E \times (\boldsymbol{\omega}_E \times \boldsymbol{r}_R) \tag{2.75}$$

Multiplying this equation with the constant mass m of the object, it is possible to apply Newton's second law on the left hand side of the equation, since it is defined in inertial space:

$$m\frac{d^2\boldsymbol{r}_I}{dt^2} = \boldsymbol{F}_{ext}$$

Here, $\boldsymbol{F}_{ext}$ marks the external forces, which are in this case gravity and aerodynamic forces. If one were to refresh the definition of the inertial frame to coincide with the rotating frame at the current point in time, one can also say that $\boldsymbol{F}_{ext}$ is equal to the forces in $\mathcal{F}_R$. Furthermore, using Eq. (2.72) and replacing $\boldsymbol{F}_{ext}$ by the aerodynamic and gravity forces in $\mathcal{F}_R$ yields the dynamic equations of motion:

$$\frac{d\boldsymbol{V}_{G,R}}{dt} = \boldsymbol{g}_R + \frac{1}{m}\boldsymbol{F}_{A,R} - 2\boldsymbol{\omega}_E \times \boldsymbol{V}_R - \boldsymbol{\omega}_E \times (\boldsymbol{\omega}_E \times \boldsymbol{r}_R) \tag{2.76}$$

Compared to Eq. (2.69), there are two additional terms on the right-hand side. The third term on the right-hand side is the Coriolis acceleration, which is large when an object travels to the east or west fast; at the start of a trajectory, for a velocity of 7 km/s, this acceleration has a magnitude of about $2 \times 7.27 \cdot 10^{-5} \times 7000 = 1.02$ m/s^2, which is not negligible. The last term in Eq. (2.76) is the centrifugal acceleration. This term is large at high altitudes above the equator. For altitudes up to 120 km, this term has a magnitude of about $(7.27 \cdot 10^{-5})^2 \times R_E = 0.034$ m/s^2. This is much smaller than the Coriolis acceleration, but the time span of the trajectories

simulated in this report is typically more than 1000 s. After 1000 s, not including the centrifugal acceleration would cause a velocity error of about 34 m/s and a position error of about 17 km (assuming the reentry dynamics to be linear), which are not negligible errors, hence both terms will be included.

The other equations remain largely unchanged; the only difference is that the aerodynamic forces should be transformed to $\mathcal{F}_R$, instead of $\mathcal{F}_I$. This means that one should replace $\tilde{\tau}$ by τ and replace the subscript I by R in Eqs. (2.70) and (2.71), and Eqs. (2.12) to (2.15). Since these are the only changes, the wind equations in Section 2.7.1 will only be derived for Cartesian state variables in $\mathcal{F}_I$.

2.6.2 Spherical State Variables

The spherical state variables are r, τ, δ, V_G, γ_G and χ_G, of which the first three define the position in $\mathcal{F}_R$ and the latter three define the velocity in $\mathcal{F}_V$. The derivation of the equations of motion for the spherical state variables is a lengthy one and is not included here. For the full derivation, one should consult [43] and the references therein. The kinematic equations are:

$$\dot{r} = V_G \sin\gamma_G \tag{2.77}$$

$$\dot{\tau} = \frac{V_G \sin\chi_G \cos\gamma_G}{r\cos\delta} \tag{2.78}$$

$$\dot{\delta} = \frac{V_G \cos\chi_G \cos\gamma_G}{r} \tag{2.79}$$

and the dynamic equations are:

$$\dot{V}_G = -\frac{D_G}{m} - g_r \sin\gamma_G + g_\delta \cos\gamma_G \cos\chi_G + \omega_E^2 r \cos\delta \left(\sin\gamma_G \cos\delta - \cos\gamma_G \sin\delta \cos\chi_G\right) \tag{2.80}$$

$$\begin{aligned} V_G\dot{\gamma}_G = &-\frac{S_G}{m}\sin\sigma_G + \frac{L_G}{m}\cos\sigma_G - g_r\cos\gamma_G - g_\delta \sin\gamma_G\cos\chi_G + 2\omega_E V_G \cos\delta\sin\chi_G \\ &+\frac{V_G^2}{r}\cos\gamma_G + \omega_E^2 r\cos\delta\left(\cos\delta\cos\gamma_G + \sin\gamma_G\sin\delta\cos\chi_G\right) \end{aligned} \tag{2.81}$$

$$\begin{aligned} V_G\cos\gamma_G\dot{\chi}_G = &-\frac{S_G}{m}\cos\sigma_G - \frac{L_G}{m}\sin\sigma_G - g_\delta\sin\chi_G + 2\omega_E V_G\left(\sin\delta\cos\gamma_G - \cos\delta\sin\gamma_G\cos\chi_G\right) \\ &+\frac{V_G^2}{r}\cos^2\gamma_G\tan\delta\sin\chi_G + \omega_E^2 r\cos\delta\sin\delta\sin\chi_G \end{aligned} \tag{2.82}$$

where the aerodynamic forces D_G, S_G and L_G are groundspeed based, which are derived from the airspeed-based forces in Section 2.7.2. When wind is absent, these forces are identical to the airspeed-based ones and can directly be calculated with Eq. (2.61). g_r and g_δ are the spherical components of the gravity force vector and are given by Eqs. (2.66) and (2.67), respectively.

Note that in Eqs. (2.78) and (2.82), there is a division by $\cos\delta$ (in Eq. (2.82) appearing as a $\tan\delta$-term). This means that they are singular when $\delta = \pm 90°$. This means that these equations cannot be used for flight near a planet's poles, which is an unlikely scenario for Earth reentry vehicles anyway. Furthermore, in Eq. (2.82), one has to divide the right-hand side by $\cos\gamma_G$ to obtain $\dot{\chi}_G$, which means that this equation is singular when $\gamma_G = \pm 90°$. Note once more that vertical flight does not occur during hypersonic reentry.

2.6.3 Spherical Position with Cartesian Velocity

This state-variable set consists of r, τ, δ, u_V, v_V and w_V, of which the first three define the position in $\mathcal{F}_R$ and the latter three define the velocity in $\mathcal{F}_V$. The full derivation of the equations of motion is given in [43]. The kinematic equations are:

$$\dot{r} = -w_V \tag{2.83}$$

$$\dot{\tau} = \frac{v_V}{r\cos\delta} \tag{2.84}$$

$$\dot{\delta} = \frac{u_V}{r} \tag{2.85}$$

and the dynamic equations are:

$$\dot{u}_V = \frac{1}{m}F_{A,V_x} + g_\delta - 2\omega_E v_V \sin\delta - \omega_E^2 r \sin\delta\cos\delta + \frac{u_V w_V - v_V^2\tan\delta}{r} \tag{2.86}$$

$$\dot{v}_V = \frac{1}{m}F_{A,V_y} + 2\omega_E(u_V\sin\delta + w_V\cos\delta) + \frac{v_V}{r}(u_V\tan\delta + w_V) \tag{2.87}$$

$$\dot{w}_V = \frac{1}{m}F_{A,V_z} + g_r - 2\omega_E v_V\cos\delta - \omega_E^2 r\cos^2\delta + \frac{u_V^2 - v_V^2}{r} \tag{2.88}$$

where g_r and g_δ are the spherical components gravity acceleration vector as given in Eqs. (2.66) and (2.67), respectively. $\boldsymbol{F}_{A,V}$ is the aerodynamic force vector in $\mathcal{F}_V$, which requires the transformation from $\mathcal{F}_{AA}$ to $\mathcal{F}_V$:

$$\begin{aligned}\boldsymbol{C}_{V,AA} &= \boldsymbol{C}_{V,TA}\boldsymbol{C}_{TA,AA}\\ &= \boldsymbol{C}_3(-\chi_A)\boldsymbol{C}_2(-\gamma_A)\boldsymbol{C}_1(\sigma_A)\end{aligned} \tag{2.89}$$

2.7 The Influence of Wind

This section will present the influence of wind on the equations of motion. The presence of wind causes a distinction between airspeed and groundspeed-based variables, and a change of the aerodynamic forces, in both direction and magnitude, because the wind changes the airflow along the vehicle. The equations in this section were obtained from [43].

The first step of including wind, for every set of state variables is to obtain the position of vehicle in the form of the input of the wind model, which is for the GRAM 1999 model altitude H, longitude τ (w.r.t. the rotating reference frame) and latitude δ. The output of GRAM are wind velocities specified in the vertical reference frame [31].

2.7.1 Cartesian State Variables

For Cartesian state variables, the steps listed in this section are modifications to the calculation of $\boldsymbol{F}_A$, which would originally happen using groundspeed-based variables, and its transformation from $\mathcal{F}_{AA}$ to $\mathcal{F}_I$ in Section 2.6.1. Before these steps, one should have already computed the groundspeed.

First of all, of the two methods presented in Section 2.2 to obtain $\boldsymbol{C}_{I,TA}$, the method utilizing standard transformation matrices is the better option, since the matrices will be reused later. Using this method, one ends up with the transformation matrices $\boldsymbol{C}_{V,I}$ and $\boldsymbol{C}_{TG,V}$.

Should the wind be given as a magnitude V_W, flight-path angle γ_W and heading χ_W are given, then the wind velocity vector in $\mathcal{F}_V$ has to be obtained first. The relation for this can be found using the geometry of velocity, flight-path angle and heading angle as shown in Figure 2.3. This results in:

$$\boldsymbol{V}_{W,V} = V_W \begin{pmatrix} \cos\chi_W \cos\gamma_W \\ \sin\chi_W \cos\gamma_W \\ -\sin\gamma_W \end{pmatrix} \tag{2.90}$$

Figure 2.9 then shows the relationship between the wind velocity, airspeed and groundspeed. As can be seen in this figure, the airspeed is computed from the following vector subtraction:

$$\boldsymbol{V}_A = \boldsymbol{V}_G - \boldsymbol{V}_W \tag{2.91}$$

Note that this equation is valid as long as all the vectors are specified in the same frame, in this case $\mathcal{F}_V$.

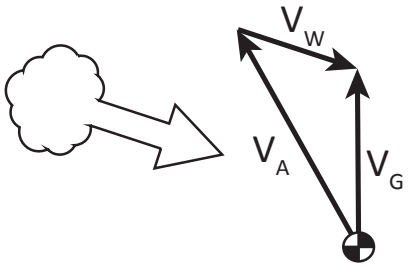


Figure 2.9: The relationship between the wind velocity, airspeed and groundspeed.

Next, the attitude of the vehicle w.r.t. the atmosphere is computed. For this, the transformation matrix $\boldsymbol{C}_{B,TA}$ is needed. $\boldsymbol{C}_{B,TA}$ is computed from:

$$\begin{aligned} \boldsymbol{C}_{B,TA} &= \boldsymbol{C}_{B,TG}\boldsymbol{C}_{TG,V}\boldsymbol{C}_{V,TA} \\ &= \boldsymbol{C}_2(\alpha_G)\boldsymbol{C}_3(-\beta_G)\boldsymbol{C}_1(-\sigma_G)\boldsymbol{C}_2(\gamma_G)\boldsymbol{C}_3(\chi_G)\boldsymbol{C}_3(-\chi_A)\boldsymbol{C}_2(-\gamma_A) \end{aligned} \tag{2.92}$$

where the sines and cosines of χ_A and γ_A can be calculated with Eqs. (2.24) and (2.25), respectively. $\boldsymbol{C}_{B,TA}$ written out in terms of α_A, β_A and σ_A gives:

$$\boldsymbol{C}_{B,TA} = \begin{bmatrix} \mathrm{c}\alpha_A\mathrm{c}\beta_A & \mathrm{s}\alpha_A\mathrm{s}\sigma_A - \mathrm{c}\alpha_A\mathrm{s}\beta_A\mathrm{c}\sigma_A & \mathrm{c}\alpha_A\mathrm{s}\beta_A\mathrm{s}\sigma_A - \mathrm{s}\alpha_A\mathrm{c}\sigma_A \\ \mathrm{s}\beta_A & \mathrm{c}\beta_A\mathrm{c}\sigma_A & -\mathrm{c}\beta_A\mathrm{s}\sigma_A \\ \mathrm{s}\alpha_A\mathrm{c}\beta_A & \mathrm{c}\alpha_A\mathrm{s}\sigma_A - \mathrm{s}\alpha_A\mathrm{s}\beta_A\mathrm{c}\sigma_A & \mathrm{c}\alpha_A\mathrm{c}\sigma_A + \mathrm{s}\alpha_A\mathrm{s}\beta_A\mathrm{s}\sigma_A \end{bmatrix} \tag{2.93}$$

The components of the matrix in this equation can then be used to obtain expressions for α_A, β_A and σ_A:

$$\sin\alpha_A = \frac{\boldsymbol{C}_{B,TA}(3,1)}{\cos\beta_A} \qquad \cos\alpha_A = \frac{\boldsymbol{C}_{B,TA}(1,1)}{\cos\beta_A} \tag{2.94}$$

$$\sin\beta_A = \boldsymbol{C}_{B,TA}(2,1) \qquad \cos\beta_A = \sqrt{\boldsymbol{C}_{B,TA}(2,2)^2 + \boldsymbol{C}_{B,TA}(2,3)^2} \tag{2.95}$$

$$\sin\sigma_A = -\frac{\boldsymbol{C}_{B,TA}(2,3)}{\cos\beta_A} \qquad \cos\sigma_A = \frac{\boldsymbol{C}_{B,TA}(2,2)}{\cos\beta_A} \tag{2.96}$$

where the notation $\boldsymbol{C}_{B,TA}(3,1)$ indicates the third row, first column entry of the right-hand side of Eq. (2.93). Knowing α_A and β_A, the aerodynamic force coefficients can be obtained from the vehicle data and in turn the aerodynamic forces in $\mathcal{F}_{AA}$ can be computed using Eq. (2.61). These then have to be transformed back to $\mathcal{F}_I$, using $\boldsymbol{C}_{I,AA}$:

$$\boldsymbol{C}_{I,AA} = \boldsymbol{C}_3(-\tilde{\tau})\boldsymbol{C}_2(\delta + \frac{\pi}{2})\boldsymbol{C}_3(-\chi_A)\boldsymbol{C}_2(-\gamma_A)\boldsymbol{C}_1(\sigma_A) \tag{2.97}$$

With $\boldsymbol{F}_{A,I}$ known, one can compute the position and velocity with the equations of motion, i.e., Eqs. (2.68) and (2.69).

2.7.2 Spherical State Variables

For spherical state variables, the addition of wind consists of the computation of the aerodynamic force vector $\boldsymbol{F}_{A,AA}$ and then transforming it to $\mathcal{F}_{AG}$.

The airspeed is calculated in the same way as for the Cartesian state variables. The groundspeed vector in $\mathcal{F}_V$ is computed from V_G, γ_G and χ_G with the groundspeed version of Eq. (2.90):

$$\boldsymbol{V}_{G,V} = V_G \begin{pmatrix} \cos\chi_G \cos\gamma_G \\ \sin\chi_G \cos\gamma_G \\ -\sin\gamma_G \end{pmatrix} \tag{2.98}$$

The aerodynamic angles α_A, β_A and σ_A are then computed in the same way as for Cartesian state variables.

The vector $\boldsymbol{F}_{A,AA}$ is then computed with Eq. (2.61). Obtaining the groundspeed based forces requires transforming this vector from $\mathcal{F}_{AA}$ to $\mathcal{F}_{AG}$ with the following transformation matrix:

$$\begin{aligned} \boldsymbol{C}_{AG,AA} &= \boldsymbol{C}_{AG,TG}\boldsymbol{C}_{TG,V}\boldsymbol{C}_{V,TA}\boldsymbol{C}_{TA,AA} \\ &= \boldsymbol{C}_1(\sigma_G)\boldsymbol{C}_2(\gamma_G)\boldsymbol{C}_3(\chi_G - \chi_A)\boldsymbol{C}_2(-\gamma_A)\boldsymbol{C}_1(\sigma_A) \end{aligned} \tag{2.99}$$

Note that the last part of this transformation is a rotation of σ_G about the X-axis. However, this transformation is not needed, because the sine and cosine of σ_G only appear in Eqs. (2.80) to (2.82) in a particular combination with the aerodynamic force terms. This combination closely resembles the aerodynamic forces expressed in $\mathcal{F}_{TG}$:

$$\boldsymbol{F}_{A,TG} = \boldsymbol{C}_1(\sigma_G)\boldsymbol{F}_{A,AG} = \begin{pmatrix} -D_G \\ -S_G\cos\sigma_G - L_G\sin\sigma_G \\ S_G\sin\sigma_G - L_G\cos\sigma_G \end{pmatrix} \tag{2.100}$$

So the aerodynamic force terms in Eqs. (2.80) to (2.82) can be obtained with:

$$-\frac{D_G}{m} = \frac{1}{m}\boldsymbol{F}_{A,TG_x} \tag{2.101}$$

$$-\frac{S_G}{m}\sin\sigma_G + \frac{L_G}{m}\cos\sigma_G = -\frac{1}{m}\boldsymbol{F}_{A,TG_y} \tag{2.102}$$

$$-\frac{S_G}{m}\cos\sigma_G - \frac{L_G}{m}\sin\sigma_G = \frac{1}{m}\boldsymbol{F}_{A,TG_z} \tag{2.103}$$

with:

$$\boldsymbol{F}_{A,TG} = \boldsymbol{C}_2(\gamma_G)\boldsymbol{C}_3(\chi_G - \chi_A)\boldsymbol{C}_2(-\gamma_A)\boldsymbol{C}_1(\sigma_A)\boldsymbol{F}_{A,AA} \tag{2.104}$$

The sines and cosines of χ_A and γ_A are computed with Eqs. (2.24) and (2.25), respectively. Now that the aerodynamic force terms are computed, one can continue with the equations of motion as described in Section 2.6.2.

2.7.3 Spherical Position with Cartesian Velocity

For this state-variable set, the addition of wind is largely the same as for the spherical state variables. The velocity state variables u_V, v_V and w_V already form the groundspeed vector

in $\mathcal{F}_V$, with which the airspeed can be computed. The aerodynamic angles are computed as described in Section 2.7.1. Knowing these, the aerodynamic force vector $\boldsymbol{F}_{A,AA}$ can be obtained. The equations of motion in Section 2.6.3 require the aerodynamic forces in $\mathcal{F}_V$, which means that the transformation $\mathcal{F}_{AA}$ to $\mathcal{F}_V$ is needed:

$$\boldsymbol{C}_{V,AA} = \boldsymbol{C}_3(-\chi_A)\boldsymbol{C}_2(-\gamma_A)\boldsymbol{C}_1(\sigma_A) \tag{2.105}$$

Now, one can continue with the equations of motion as described in Section 2.6.3.

Chapter 3

Mission Characteristics

With the equations of motion introduced in the previous chapter, this chapter addresses the reentry missions that will be simulated. The reference vehicle that will be used for all simulations is HORUS-2B. Section 3.1 will present the details of this vehicle and its reference mission. Sections 3.2 and 3.3 will explain how the controls and constraints for a trajectory are defined. Finally, Sections 3.4 and 3.5 describe alternative objectives for which HORUS' trajectory could be optimized, namely downrange, cross-range and heat load.

3.1 HORUS-2B

HORUS is a manned, Space-Shuttle-like reentry vehicle. The original purpose of HORUS was to serve as a reusable second stage to the Ariane-5 launcher. HORUS was later redesigned as the second stage of the Sänger II project, a two-stage-to-orbit concept shown in Figure 3.1, for which HORUS would be fitted with a rocket engine to bring it up to orbit. Both projects were canceled for budgetary reasons, but a complete aerodynamic database of HORUS (the version without rocket engine) was made by Messerschmidt-Boelkow-Bloehm, so HORUS can be used as a reference vehicle for reentry simulation. This section is divided into two parts, discussing its reference mission and the use of HORUS' aerodynamics database.

Figure 3.1: Artist's impression of HORUS mounted on the Sänger first-stage aircraft. Credits: D. Van Ravenswaay.

3.1.1 Reference Mission

The relevant mission details of HORUS are shown in Table 3.1. HORUS' flight starts above the Pacific Ocean and it ends at the runway at Kourou, French Guiana. The trajectory is split into two parts, the hypersonic entry phase and the Terminal Area Energy Management (TAEM) phase before landing. In this case, only the hypersonic phase is of interest and it ends when the vehicle reaches a horizontal distance of 0.75° to the target point, which is located 12 km north of the runway. The altitude-velocity profile and control profile of the reference mission are shown in Figure 3.2. HORUS flies an equilibrium-glide trajectory, i.e., its flight-path angle changes only very little over time. Its angle of attack is set to the maximal value (40°) for most of the time to keep the heat flux below the maximum value (530 kW/m^2). Since the large angle of attack implies a high lift force, the vehicle requires a large bank angle to prevent it from skipping and to keep it flying an equilibrium-glide trajectory. Note that in Figure 3.2, a number of maneuvers called bank reversals are performed, where the sign of the bank angle is flipped. The concept of bank reversals is further discussed in Section 3.2.2.

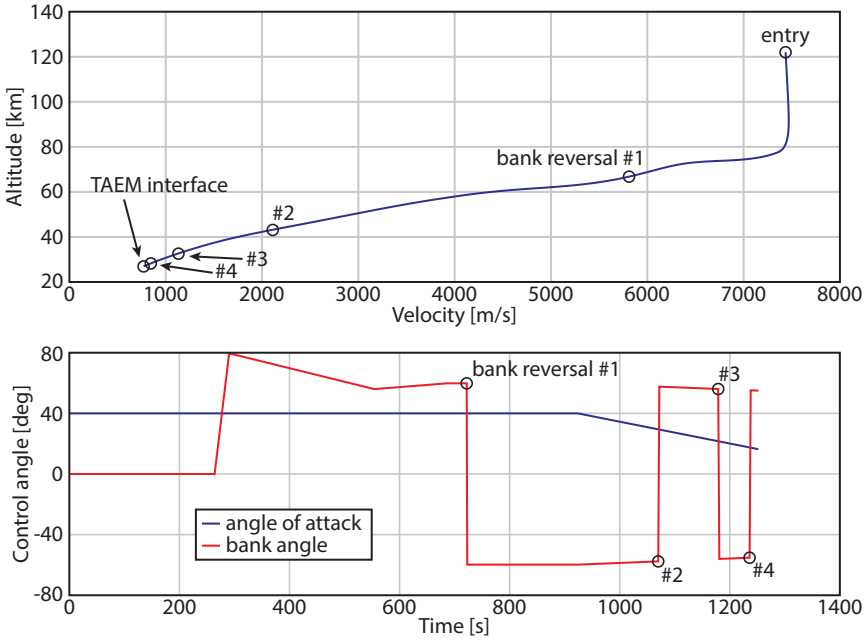


Figure 3.2: Reference trajectory and control profile for HORUS-2B. Source: [46]

Table 3.1: Main characteristics of the HORUS-2B reference mission. Source: [44]

Property	Value	Property	Value
Wing area (S_{ref}) [m^2]	110	Maximum heat flux [kW/m^2]	530
Reentry mass [kg]	26029	Maximum g-load [-]	2.5
Nose Radius [m]	0.8	Maximum dynamic pressure [N/m^2]	$1{\cdot}10^4$
Initial altitude [km]	122	Terminal altitude [km]	24.8597
Initial longitude [°]	-106.7	Terminal Mach number [-]	2.5
Initial latitude [°]	-22.3	Terminal distance [°]	0.75
Initial velocity [m/s]	7435.5	Runway longitude [°]	-53.0
Initial flight-path angle [°]	-1.43	Runway latitude [°]	5.0
Initial heading [°]	70.75		

3.1.2 Aerodynamics

The exterior design of HORUS is shown in Figure 3.3. HORUS has five control surfaces, namely the body flap located at the rear of the fuselage, two elevons at the trailing edge of the wings and two rudders located at the upstanding wingtips (note that the rudders can move only outward). The aerodynamic database of HORUS is given in tabular form in [47]. In this case, the terms of interest are the aerodynamic force coefficients C_D and C_L, and the pitch moment coefficient C_m. The pitch moment is of interest for the trimming of the vehicle; this moment should be kept at zero to prevent the vehicle from deviating from the commanded attitude. In [47], C_D, C_L and C_m are given by:

$$C_D = C_{D_0} + \Delta C_{D_b} + \Delta C_{D_{e,l}} + \Delta C_{D_{e,r}} \tag{3.1}$$

$$C_L = C_{L_0} + \Delta C_{L_b} + \Delta C_{L_{e,l}} + \Delta C_{L_{e,r}} \tag{3.2}$$

$$C_m = C_{m_0} + \Delta C_{m_b} + \Delta C_{m_{e,l}} + \Delta C_{m_{e,r}} \tag{3.3}$$

where the Δ-terms are the increments due to the deflections of the control surfaces; subscripts b and e denote body flap and elevons, respectively (l left, r right). Each of the terms in Eqs. (3.1) to (3.3) are a function of α_A and Mach number, M, and the Δ-terms are also a function of the related control surface deflection δ_b, $\delta_{e,l}$ and $\delta_{e,r}$ (all positive when the control surface is deflected downward). The terms are given in [47] for a range of α_A from 0° to 45° with increments of 5°, Mach numbers 1.2, 1.5, 2, 3, 5, 10 and 20, and a range of δ_b from -20° to 30° and δ_e from -40° to 40°, both with increments of 10°.

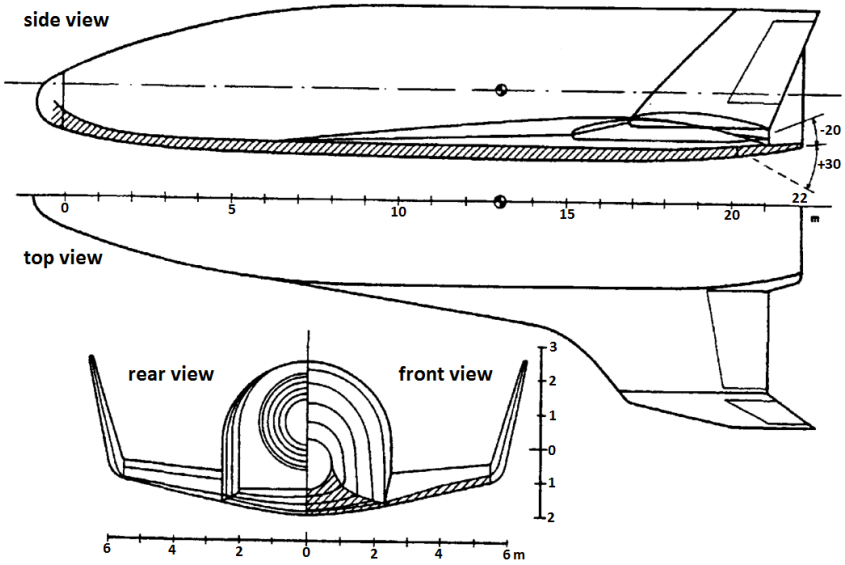


Figure 3.3: The HORUS-2B. Source: [47]

The pitching moment is zero when C_m is zero, so trimming is done by deflecting the body flap and elevons such that the sum of the Δ-terms in Eq. (3.3) equals $-C_{m_0}$. The body flap is the primary control surface for trimming C_m and only when it is at full deflection and cannot compensate for C_{m_0}, the elevons are used. $\delta_{e,l}$ and $\delta_{e,r}$ are always equal, so that the elevons do not generate lateral moments.

For TSI, the aerodynamics have to be approximated by analytical formulas. To simplify this process, the variables that determine C_D and C_L should be brought back to just α_A and M. To achieve this, the vehicle is trimmed for each combination of α_A and M. This process is done

as follows: first ΔC_{m_b} as function of δ_b is interpolated with a cubic spline (see Section 4.2.3 for a description of standard cubic splines) and a root-finding is applied to find the value of δ_b for which ΔC_{m_b} equals $-C_{m_0}$. The root-finding method in this case is a variation of Newtons method (see Section 4.1.2 for Newtons method) with $f'(x_i)$ approximated by a divided difference:

$$\delta_{b,i+1} = \delta_{b,i} - 2h \frac{\Delta C_{m_b}(\delta_{b,i})}{\Delta C_{m_b}(\delta_{b,i} + h) - \Delta C_{m_b}(\delta_{b,i} - h)} \tag{3.4}$$

with h being a small step size for the sampling of the secant method, in this case arbitrarily set to $1 \cdot 10^{-3}$ degrees. When δ_b is found, its effect on C_D and C_L is computed from spline interpolation of ΔC_{D_b} and ΔC_{L_b}. If the elevons are needed to compensate for C_{m_0}, the deflection angles $\delta_{e,l}$ and $\delta_{e,r}$ and their effect on C_D and C_L is computed in the same way as for δ_b.

Now that C_D and C_L are only a function of α_A and M, they can be fitted with analytic functions to create the desired analytic expressions for TSI. One more modification to the aerodynamics of HORUS is that only the Mach range 2.5 to 20 is used; C_D and C_L were interpolated for Mach 2.5. The reduction of the Mach range reduces the number of points that have to be fitted. The reason for choosing Mach 2.5 as lower limit is that this is a terminal condition for the hypersonic part of HORUS' flight. During simulations this condition will not directly be used as terminal constraint (see Chapter 8), but only during a small part of some trajectories, HORUS will reach Mach numbers lower than 2.5. Furthermore, this only occurs during the final part of the reentry phase, so this modification will not impact the rest of the trajectory. If the Mach number falls below 2.5, the values of C_D and C_L for Mach 2.5 are used. Similarly, if the Mach number exceeds 20, the C_D and C_L for Mach 20 will be used. The trimmed values for C_D and C_L are given in the tables of Appendix B and are plotted as circles in Figure 5.8.

3.2 Controls

Next, the way the controls are defined is discussed, followed by the an explanation of bank reversals.

3.2.1 Control Profile

As was already mentioned in Chapter 2, the reentry vehicle will be modeled as a point-mass and dynamics of its attitude will not be simulated. This means that attitude-control actuators, such as aerodynamic control surfaces and reaction-control thrusters, are not modeled. Rather, one will simply define an attitude profile for the entire trajectory and this will be applied instantaneously to the vehicle. This implies the assumption that the attitude-control systems are perfect, being able to rotate the vehicle to the desired attitude in zero time.

The attitude of the vehicle is defined by the aerodynamic groundspeed-based angles α_G, β_G and σ_G, which are, in this case, the control variables. In practice, β_G is kept zero for the entire trajectory for Shuttle-like vehicles due to the non-linear response of such vehicles to even small side-slip angles [46]. Hence, without the influence of wind, a symmetric vehicle (symmetric meaning that the left and the right side are each others exact mirror) will experience no aerodynamic side-forces. Of the two remaining angles, α_G is normally constrained in magnitude for a lifting vehicle to prevent it from stalling. For the guidance system of HORUS, α_G is limited to 43.9° to avoid stalling. For reference trajectories, which includes the trajectories simulated for this research, α_G is limited to 40°, leaving a safety margin for the guidance system. Furthermore, α_G shall not be negative, as this would expose the upper side of the vehicle to the

incoming hypersonic airflow, while the lower side has the proper shielding to withstand the heat load caused by the airflow. σ_G will always stay between $\pm 90°$, because outside this domain, the lift vector will be directed downwards. Lift, in that case, would increase the rate of descent, whereas it is normally needed in reentry to increase the stay in the upper tenuous regions of the atmosphere to lower the thermal and aerodynamic loads on the vehicle.

The controls have to be defined for the entire trajectory and are thus defined by a control profile. This profile consists of the values of α_G and σ_G defined at a number of points along the trajectory, called nodes. The location and control values of these nodes are typically found through numerical optimization. Since numerical optimization becomes harder the more variables are involved (see Section 4.5 for more details on numerical optimization), the number of nodes is kept small. The control values in between these nodes are then obtained with interpolation. In this case, cubic Hermite splines will be used to interpolate the controls, as this is a good way of generating a smooth continuous profile without overshooting the nodes (see Section 4.2.3 for a description of cubic Hermite splines).

One needs some way of defining where along the trajectory a node is located and for this the specific total energy E of the vehicle is defined:

$$E = g_0 H + \frac{1}{2} V_G^2 \tag{3.5}$$

where g_0 is the gravity acceleration at sea level, 9.81 m/s. E decreases due to dissipation of energy to the air through drag and is approximately monotonic. Only at the start of the reentry, E may increase slightly, but for the other parts of the trajectory, each value of E will mark a unique point. Note that one could have also used time to specify the node locations, but a time value contains no information on the current state of the vehicle. One could, for instance, have one trajectory where the vehicle glides for a long time in the upper regions of the atmosphere and have another where the vehicle performs a dive. At the same point in time, the vehicle could have a completely different state.

The specification of the nodes will be done using *Non-Uniform Independent Node* control. In this case, each node consists of three variables, namely the values for α_G, σ_G and E [19]. The limits for E are E_{max}, i.e., the value at the start of the trajectory, and E_{min}, the value at the end. To make sure that the profile is defined for the entire trajectory, one node should be located at E_{max} and one at E_{min}. E_{max} can be found from the initial conditions and E_{min} can be found from the terminal conditions:

$$E_{max} = g_0 H_0 + \frac{1}{2} V_{G,0}^2 \tag{3.6}$$

$$E_{min} = g_0 H_e + \frac{1}{2} \left(M_e a_e \right)^2 \tag{3.7}$$

where subscript e marks the terminal conditions and a_e is the speed of sound at the terminal altitude, which is 298.3 m/s according to US76. E_{max} and E_{min} are then equal to $2.884 \cdot 10^7$ J and $5.219 \cdot 10^5$ J, respectively.

During optimization, a control vector is given by a numerical optimizer and from this vector a trajectory is to be computed. The control vector looks as follows:

$$\mathbf{\Gamma} = \begin{bmatrix} E_1 & E_2 & \cdots & E_{n-2} & \alpha_{G,0} & \cdots & \alpha_{G,n-1} & \sigma_{G,0} & \cdots & \sigma_{G,n-1} \end{bmatrix} \tag{3.8}$$

in the case of n nodes. A randomly generated control profile is shown in Figure 3.4, where the circles indicate the position of the nodes and the profile is formed by a cubic Hermite spline.

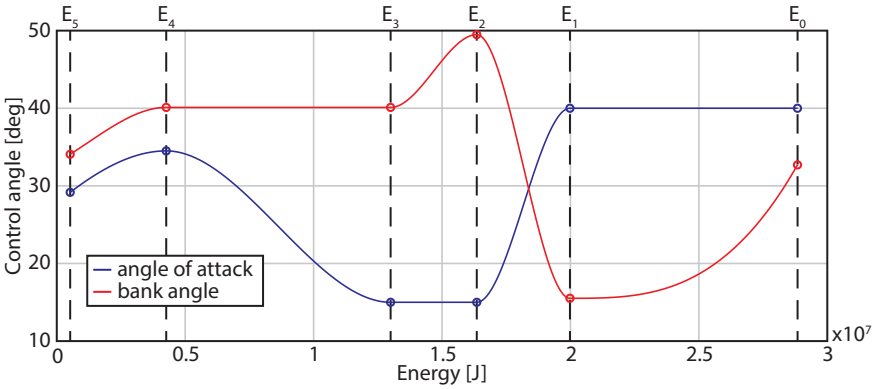


Figure 3.4: A random control profile, interpolated with Hermite splines.

For this thesis, it has been chosen to use profiles defined by 6 nodes, so the control vectors defining them will contain 16 variables (4 values for E and 6 for both α_G and σ_G).

3.2.2 Bank Reversals

The heading of the vehicle can be controlled with σ_G, as it determines the direction of the aerodynamic lift, which is the only force that can redirect the velocity vector (not including the side-force, which is zero without wind). As was mentioned in Section 3.1.1, the bank angle is typically large to keep the vehicle from skipping out of the atmosphere. If HORUS keeps banking in one direction, it will fly a spiral trajectory. To keep HORUS heading for the target, it will have to start banking in the other direction at some point in time. During simulation, σ_G is split into two variables, one for its magnitude and one for its sign. The magnitude is controlled by the trajectory's control profile, which can be optimized to meet the constraints. The sign is flipped whenever the heading error, χ_e, exceeds a predefined dead band, χ_{db}. This change of sign of σ_G is called a bank reversal and as can be seen in Figure 3.2, four of these reversals are executed during HORUS' reference trajectory. A bank reversal should only be executed if the sign of σ_G is not correct already, which can be expressed in mathematical terms as:

$$-\chi_e \text{sign}(\sigma_G) > \chi_{db} \tag{3.9}$$

χ_e is defined as:

$$\chi_e = \chi_G - \chi_T \tag{3.10}$$

where χ_T is the heading to target, given by [46]:

$$\chi_T = \text{atan2}\left(\sin(\tau_T - \tau), \tan\delta_T \cos\delta - \cos(\tau_T - \tau)\sin\delta\right) \tag{3.11}$$

where τ_T and δ_T are the longitude and latitude of the target (located 12 km north of the runway). The dead band is given in Table 3.2 as a function of the distance to target, d, given by [46]:

$$d = \arccos\left(\sin\delta\sin\delta_T + \cos\delta\cos\delta_T\cos(\tau - \tau_T)\right) \tag{3.12}$$

In between the points given in Table 3.2, χ_{db} is linearly interpolated (see Section 4.2.1 for linear interpolation). Below a distance of 0.94331°, only one final bank reversal will be executed. This is mainly because, so close to the target point, the heading error increases very quickly and one could end up having to perform many bank reversals in short succession. In reality, this would be impossible for the vehicle to perform and could lead to fatal loads on the vehicle [45].

Table 3.2: Heading-error dead band for HORUS-2B, source: [44]

d [°]	χ_{db} [°]	d [°]	χ_{db} [°]
60.0	15.0	0.90739	19.62079
30.0	15.0	0.86476	15.20725
10.0	23.0	0.85776	14.35438
0.94331	23.0	0	14.35438
0.91452	20.36779		

3.3 Trajectory Constraints

Typically, a reentry vehicle cannot fly an arbitrary trajectory through the atmosphere. Rather, a number of constraints have to be put on the trajectory to ensure the survival of the vehicle (and its crew). The constraints treated here are heating, mechanical loads and dynamic pressure. Another type of constraint is the so-called equilibrium-glide condition, which was already mentioned in Section 3.1.1. This constraint puts restrictions on the magnitude of $\dot{\gamma}$, keeping it close to zero to ensure that γ remains almost constant over time, so that the vehicle does not acutely skip during flight. These skips are usually associated with large (gradients of the) aerodynamic and thermal loads on the vehicle. By satisfying the other constraints, some of the larger skips are prevented, but smaller skips may still be possible. The equilibrium-glide condition is not used in this research so that, during optimization, the maximum range of HORUS with skips can be determined. This will mean that for such a trajectory, large gradient of the loads may occur that could cause fatigue of the structure and damage the vehicle [45].

Heating

The heating of the vehicle should be limited to prevent it from reaching temperatures at which the structural strength is degraded. Furthermore, the vehicle should not be subjected to amounts of heat, which its cooling systems can no longer cope with. Typically, a maximum temperature is given for a vehicle, which corresponds to a particular heat input, assuming the vehicle is in thermodynamic equilibrium with its surroundings [46]. The common heating constraint is a limit to the convective heating rate of the nose, which is given by [8]:

$$\dot{q}_c = \frac{C}{\sqrt{R_N}} \left(\frac{\rho}{\rho_0} \right)^n \left(\frac{V_A}{\sqrt{gr}} \right)^m \tag{3.13}$$

with the variables R_N and g representing the radius of curvature of the nose of the vehicle (0.8 m for HORUS) and the magnitude of the gravity acceleration, respectively. The value of n is equal to 0.5 for laminar flow and 0.2 for turbulent flow. The values of the constants C and m vary per source [8, 46]; C typically has a value of about $3.02 \cdot 10^6$ W/m$^{3/2}$ and the values for m range between 3 and 3.22. The quantity $\sqrt{gr}$ is also called the circular velocity. When this is assumed constant, the airflow is assumed to be laminar and all constants are grouped together in a constant C_1, one gets:

$$\dot{q}_c = \frac{C_1}{\sqrt{R_N}} \sqrt{\rho} V_A^m \leq \dot{q}_{c_{max}} \tag{3.14}$$

The used combination of values for C_1 and m is $5.28137 \cdot 10^{-5}$ Js$^{2.15}$/kg$^{0.5}$/m$^{2.15}$ and 3.15, obtained from [46]. The value for $\dot{q}_{c_{max}}$ is 530 kW/m^2 for HORUS. Note that Eq. (3.13) is a cold-wall approximation, that is, the cooling of the vehicle through radiation is not taken into account, so it will overestimate the actual heat flux, typically by about 20% [46].

Mechanical Load

The mechanical load, which when normalized is called the *g-load*, is here defined as the acceleration due to the total aerodynamic force divided by the weight at sea level [46]:

$$n_g = \frac{\sqrt{D^2 + S^2 + L^2}}{mg_0} \leq n_{g_{max}} \tag{3.15}$$

where g_0 is the gravitational acceleration at sea level. Note that, as explained in the previous section, the side-force is zero when wind is absent. The g-load acts on the entire vehicle, including all internal systems (and the crew), so the maximum allowed load is limited by the weakest link. Normally, when there is a crew present, they are the weakest link. Typical values for $n_{g_{max}}$ are 2.5 to 4 in that case. Note that when the crew does not have to perform any tasks during the reentry, the mechanical loads can be higher. The value for $n_{g_{max}}$ for HORUS is 2.5, which indicates a non-passive crew.

Dynamic Pressure

The skin of the vehicle that is exposed to the incoming airflow will be subjected to the dynamic pressure, so that determines $q_{dyn_{max}}$. If the dynamic pressure exceeds a certain (vehicle dependent) limit, the structure could be damaged. The expression for dynamic pressure, Eq. (2.62), is:

$$q_{dyn} = \frac{1}{2}\rho V_A^2 \leq q_{dyn_{max}} \tag{3.16}$$

Values of $q_{dyn_{max}}$ for winged reentry vehicles in literature vary from about $1{\cdot}10^4$ to $5{\cdot}10^4$ Pa [14, 70] and has been set to $1{\cdot}10^4$ for HORUS, as this value is not violated by its reference trajectory.

3.4 Downrange and Cross-Range

The downrange R_d is defined as the distance traveled parallel to the initial orbit and the cross-range R_c is defined as the distance from the initial orbit, perpendicular to the orbit. This is shown in Figure 3.5, where the points P_0 and P_e respectively indicate the position at the start and end of the reentry phase. Note that, to reach P_e, the vehicle can have an arbitrary trajectory; it does not need to follow the line of d.

The process of determining R_d and R_c is as follows. First the distance d is obtained from Eq. (3.12), but now using the initial position, instead of the target position:

$$d = \arccos\big(\sin\delta_e \sin\delta_0 + \cos\delta_e \cos\delta_0 \cos(\tau_e - \tau_0)\big) \tag{3.17}$$

where index e indicates the state at the end of flight. The angle ζ in Figure 3.5 can then be obtained from [46]:

$$\zeta = \arccos\left(\frac{\sin\delta_e - \cos d \sin\delta_0}{\sin d \cos\delta_0}\right) \tag{3.18}$$

This equation will always return a value for ζ between 0° and 180°. This means that if $\tau_e - \tau_0 > 180°$, ζ is defined counter-clockwise, instead of clockwise as shown in Figure 3.5. Similarly, Eq. (3.17) will in that case give a value for d pointing westward and R_d will then also be computed westward. Since HORUS flies an eastward trajectory, one is only interested in the

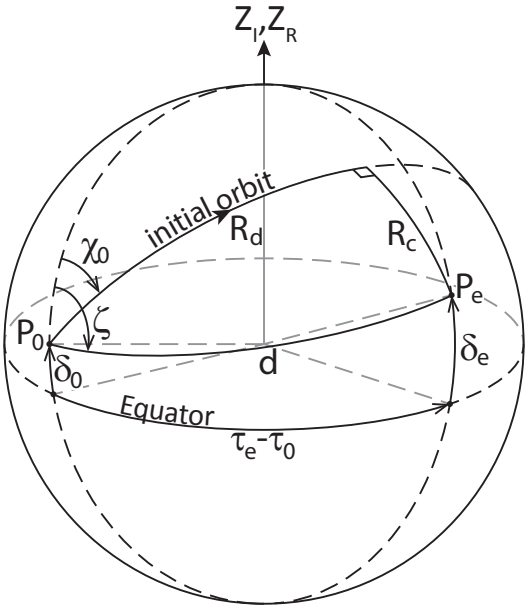


Figure 3.5: Spherical geometry for the cross-range and downrange

range eastward, so if $\tau_e - \tau_0 > 180°$, $\zeta = 180° - \zeta$ and $d = 360° - d$. Then, R_d and R_c can be found from the spherical geometry in Figure 3.5 using [69]:

$$R_d = \text{atan2}\big(\sin d \cos(\zeta - \chi_0), \cos d\big) \tag{3.19}$$

$$R_c = \arcsin\big(\sin d \sin(\zeta - \chi_0)\big) \tag{3.20}$$

Note that for R_d the atan2-function was used to ensure that if $d > 180°$, then also $R_d > 180°$. Also note that R_c is positive towards the starboard side of the initial orbit.

3.5 Heat Load

The (integrated) heat load Q is defined as the integral of the heating rate of the nose $\dot{q}_c$ (given by Eq. (3.14)) and indicates the amount of heating energy the vehicle has gathered per unit area throughout the flight. This is mainly of interest for vehicle design, as it determines how much heat the heat-protection systems of the vehicle have to absorb. The lower the Q, the less protection the vehicle needs for its mission, which means that it can be smaller, lighter and likely less expensive. Q is given by:

$$Q = \int_{t_0}^{t_e} \dot{q}_c dt \tag{3.21}$$

where t_e is the total flight time.

Chapter 4

Numerical Methods

This chapter presents the different numerical methods, which are used to solve the reentry problems and compare the performance of TSI with traditional integrators. These methods are root-finding, interpolation, least-squares regression, numerical integration and optimization and they will be treated in this order.

4.1 Root-Finding

Root-finding algorithms can be used to find the root(s) of a function $f(x)$, that is, the value(s) of x for which $f(x) = 0$. Three algorithms are presented here, namely Fixed-Point Iteration (FPI), Newton's method and the double-false position method (DFPM). FPI will be used by one of the step-size controllers of TSI in Section 5.4.1, whereas the other two methods will be used to find the location of discontinuities during integration (see Section 5.5). Of the latter two methods, Newton's method is usually faster, but uses the gradient of a function to find its roots. During integration with TSI, gradients are available, but root-finding with the other integrator, RKF5(6), will be done using DFPM, as it does not use gradients. All three algorithms are iterative and their description has been obtained from [34].

4.1.1 Fixed-Point Iteration

The FPI algorithm starts with the original function $f(x)$ and requires the user to rewrite it to the form:

$$\varphi(x) = x \tag{4.1}$$

If Eq. (4.1) is equivalent to $f(x) = 0$, then any solution of Eq. (4.1) is also a solution of $f(x) = 0$. This implies the following iteration scheme:

$$x_{i+1} = \varphi(x_i) \tag{4.2}$$

The convergence of this algorithm depends both on the choice of function $\varphi(x)$ and on the initial value of x_i. If the exact solution is $\tilde{x}$ and the error at step i is $\epsilon_i = |x_i - \tilde{x}|$, the relation between ϵ_{i+1} and ϵ_i is given by:

$$\epsilon_{i+1} = \varphi'(\xi_i)\epsilon_i \tag{4.3}$$

where ξ_i is an unknown value between x_i and $\tilde{x}$. Note from this equation that the convergence behavior of FPI is linear.

4.1.2 Newton's Method

Newton's method is based on a linear approximation to a function $f(x)$ at a point x_i and approximates the root by the point where this line crosses the axis. The formula for this method is based on the Taylor series expansion (see Section 5.1) of the function at x_i:

$$0 = f(\tilde{x}) = f(x_i) + f'(x_i)(\tilde{x} - x_i) + \cdots \tag{4.4}$$

with $\tilde{x}$ being the root. Taking only the first two elements of this series and solving for $\tilde{x}$ yields the following iteration scheme:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \tag{4.5}$$

Note that this scheme is a special case of FPI. Newton's method will converge given that the initial guess x_0 is close enough to the root $\tilde{x}$. If this is the case, Newton's method exhibits quadratic convergence behavior and will converge more rapidly than a conventional FPI function. To illustrate the convergence behavior of this method, two iterations starting from different locations are shown in Figure 4.1. Despite these two locations being relatively close to each other, the green line converges, but the red line is located too far from the root to converge.

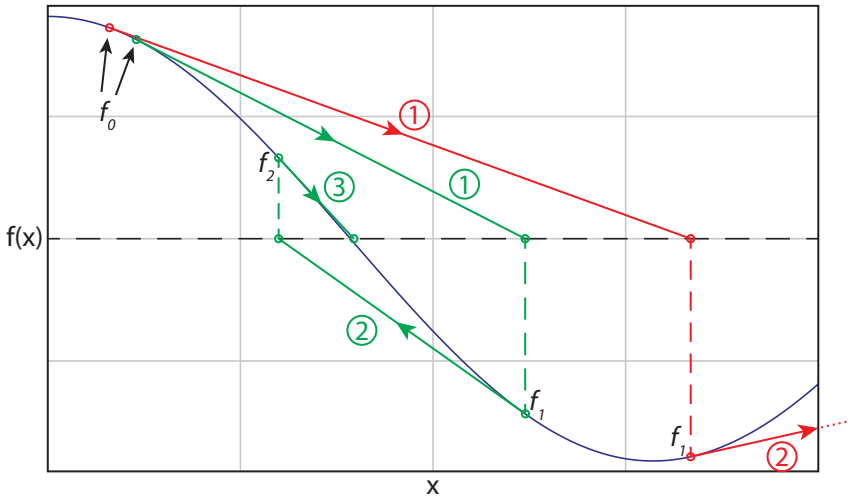


Figure 4.1: Two cases of Newton's method applied to a sine function.

4.1.3 Double-False Position Method

The double-false position method is meant for finding the root in a domain $[x_l, x_r]$, where $f(x_l)$ and $f(x_r)$ are respectively smaller and larger than zero or vice versa. This method approximates the root of $f(x)$ as the root $\hat{x}$ of a straight line from $f(x_l)$ to $f(x_r)$. The value of $\hat{x}$ for iteration i can be found from:

$$\hat{x}_i = -f(x_{l,i})\frac{x_{r,i} - x_{l,i}}{f(x_{r,i}) - f(x_{l,i})} \tag{4.6}$$

For the next iteration, x_l is replaced by $\hat{x}_i$ if $f(x_l)$ and $f(\hat{x}_i)$ have the same sign and otherwise x_r is replaced by $\hat{x}_i$. This way, the search space is reduced each step until a root is found. This method has an order of convergence, which is better than linear, but worse than quadratic, so it is mainly of interest when Newton's method cannot be used, for instance, because $f'(x)$ is not available or too hard to obtain efficiently. Another advance of this method over Newton's is that it will never diverge, although it may seize to converge prematurely.

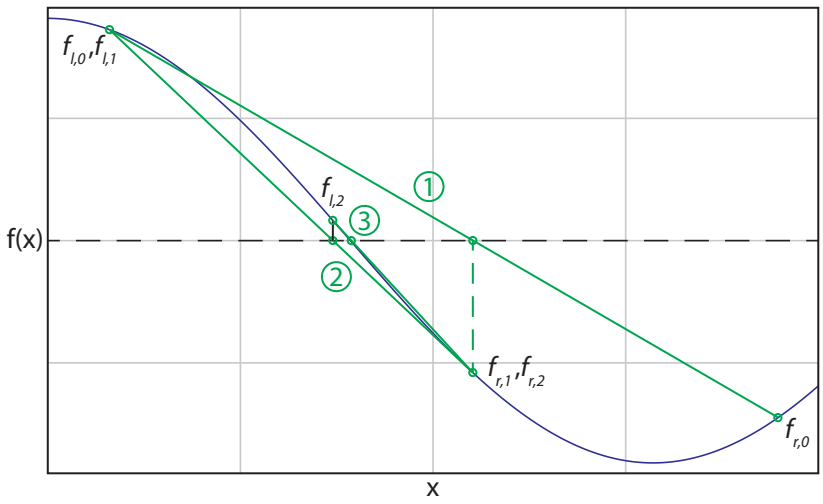


Figure 4.2: Double-false position method applied to a sine function.

4.2 Interpolation

When a function is given only at a discrete set of data points, interpolation is the process of generating values at new data points (that lie within the same range as the original data points, else it is called extrapolation). For the current subject, there are three cases in which interpolation is needed. The first is the interpolation of the results of numerical integration. The time-step size during integration is usually chosen to be as large as possible, without violating error tolerances (see Section 4.4 and Chapter 5), thus the solution may not be given at points where the user desires them, yielding the need to interpolate the results. The second case is the interpolation of discrete data for the integrators. This includes the method used to generate an analytic expression from data for TSI (see Section 5.6 for more information on this subject). The third case is the interpolation of the controls, as was mentioned in Section 3.2. The methods in this section were obtained from [34], unless specified otherwise. The types of interpolation listed in this section are linear interpolation, single-polynomial interpolation and cubic splines and they will be treated in that order.

4.2.1 Linear Interpolation

Linear interpolation here refers to the connection of adjacent nodes with straight lines. If the value of $f(x)$ is needed in between data points x_i and x_{i+1}, the linear interpolation approximation $\phi(x)$ is:

$$\phi(x) = f(x_i) + \big(f(x_{i+1}) - f(x_i)\big)\frac{x - x_i}{x_{i+1} - x_i} \tag{4.7}$$

Linear interpolation is used by traditional integrators when the to-be-integrated problem involves discrete data tables. In the current study, it is used to obtain the heading dead band (see Section 3.2.2) and to obtain air density from the tables of US76 (see Section 2.4.3).

4.2.2 Single Polynomial

One can use a single polynomial of degree p to interpolate $p+1$ data points. A reason for using polynomials as interpolants for functions is that they provide a unique solution for a specific

set of data points. This type of interpolation will not be used in the current research, but has been included here as a theoretical basis for interpolation and regression.

The standard basis for a one-dimensional interpolation polynomial is the monomial basis, given by:

$$m_i(x) = x^i \tag{4.8}$$

for $i = 0, 1, ..., p$. The interpolant then has the following form:

$$\phi(x) = \sum_{i=0}^{p} a_i x^i \tag{4.9}$$

To find the coefficients of $\phi(x)$, it is set to be equal to the function values at each of the data points x_i, which gives the linear system of $p+1$ variables and $p+1$ constraints:

$$\boldsymbol{V}\boldsymbol{a} = \boldsymbol{f} \tag{4.10}$$

with $\boldsymbol{a}$ being an array with the coefficients of the polynomial and $\boldsymbol{f}$ being an array with the values of the function. $\boldsymbol{V}$ being the so-called Vandermonde matrix, given by:

$$\boldsymbol{V} = \begin{bmatrix} 1 & x_0 & \cdots & x_0^p \\ 1 & x_1 & \cdots & x_1^p \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_p & \cdots & x_p^p \end{bmatrix} \tag{4.11}$$

One way of solving the linear system of Eq. (4.10) is by inverting $\boldsymbol{V}$, if $\boldsymbol{V}$ is invertible:

$$\boldsymbol{a} = \boldsymbol{V}^{-1}\boldsymbol{f} \tag{4.12}$$

which yields the coefficients a_i for the interpolating polynomial. For a computer, there are faster methods to solve Eq. (4.10) that do not require $\boldsymbol{V}$ to be invertible, such as QR decomposition or Singular-Value Decomposition (SVD).

Increasing the degree of the interpolating polynomial often does not mean that the polynomial converges to the actual function. In particular, it starts to oscillate increasingly wildly between the data points. The convergence of the polynomial can be improved by the choice of nodes of the data. The error is smaller for grids with more nodes near the edges of the interpolation domain and can be minimized by specifying the data points on a so-called Chebychev-Gauss grid. On the interval [0,1], a $p+1$ point grid of this type is given by:

$$\xi_i = \frac{1}{2} - \frac{1}{2}\cos\left(\frac{2i+1}{2p+2}\pi\right) \tag{4.13}$$

for $i = 0, 1, ..., p$. Contrary to common sense, the actual borders of the domain are not part of the grid and including them would harm the overall convergence. The grid can be mapped onto a domain $[a, b]$ with:

$$x_i = a + (b-a)\xi_i \tag{4.14}$$

When the choice of grid is not an option, the degree of the polynomial is best kept small, which means that the number of data points per polynomial should be limited. In [34], it is recommended not to use a higher degree than about six, unless the data is smooth. In Figure 4.3, a set of random points defined on a Chebychev-Gauss grid was interpolated using a single polynomial of degree 6.

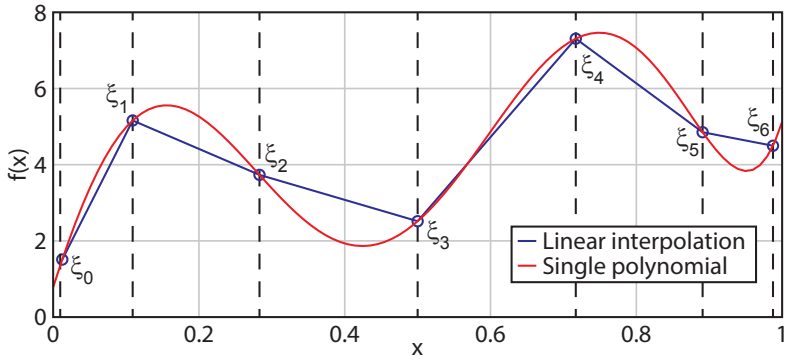


Figure 4.3: The interpolation of randomly generated points defined on a Chebychev-Gauss grid.

4.2.3 Cubic Splines

Splines are a collection of low-degree polynomials that each interpolate a small part of the discrete function. This method of interpolation is more suitable for a large number of nodes than a single polynomial. [34] gives that the interpolation error decreases as the domain of the interpolant becomes narrower, thus interpolating a function with one polynomial per adjacent two data points becomes more accurate as more nodes are given for the same domain. The most-used degree for a spline is three, since it is considered the ideal combination of flexibility and stability [34]. Two types of splines are treated, namely the standard cubic spline and the cubic Hermite spline. The first will be used to interpolate the wind model (see Section 7.2.3) and the results of traditional integrators, as it produces smoother and more accurate results when the data themselves are smooth. The second will be used to interpolate the controls, as produces a smooth interpolation without overshoots even when the data themselves are random and not smooth, which is demonstrated in Figure 4.4.

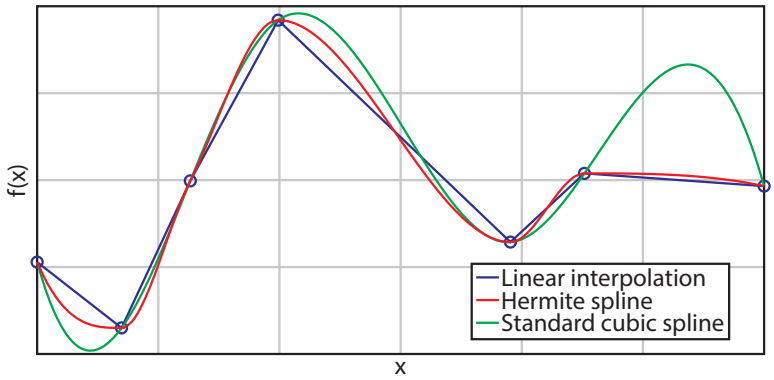


Figure 4.4: The interpolation of randomly generated points using different methods.

Standard Cubic Spline

The standard cubic spline consists of a cubic polynomial per two adjacent nodes. At each interior node, not only values of the two touching polynomials are the same, but also their first and second-order derivatives. The equation of the i^{th} cubic polynomial is:

$$P_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i \tag{4.15}$$

with $i = 0, 1, ..., p-1$ (for $p+1$ nodes). Its first and second-order derivatives are:

$$P_i'(x) = 3a_i(x - x_i)^2 + 2b_i(x - x_i) + c_i \tag{4.16}$$

$$P_i''(x) = 6a_i(x - x_i) + 2b_i \tag{4.17}$$

Setting b_i as the unknowns, the coefficients a_i can be found with the condition $P_i''(x_{i+1}) = P_{i+1}''(x_{i+1})$:

$$a_i = \frac{b_{i+1} - b_i}{3h_i} \tag{4.18}$$

with $h_i = x_{i+1} - x_i$. From Eq. (4.15) it can be found that d_i is equal to f_i. Inserting this result into Eq. (4.15) together with Eq. (4.18) results in:

$$c_i = \delta_i - \frac{1}{3}h_i\left(2b_i + b_{i+1}\right) \tag{4.19}$$

with δ_i given by:

$$\delta_i = \frac{f_{i+1} - f_i}{h_i} \tag{4.20}$$

This together with the condition $P_i'(x_i) = P_{i-1}'(x_i)$ can be used to obtain the following set of equations for the terms b_i:

$$h_{i-1}b_{i-1} + 2(h_{i-1} + h_i)b_i + h_i b_{i+1} = 3\left(\delta_i - \delta_{i-1}\right) \tag{4.21}$$

which is a system of $p-1$ equations for $p+1$ unknowns. There are different ways to obtain two additional constraints to close the system. The one used here is the *not-a-knot* condition, which sets $a_0 = a_1$ and $a_{p-2} = a_{p-1}$ and yields the following additional equations:

$$h_1 b_0 - (h_0 + h_1)b_1 + h_0 b_2 = 0 \tag{4.22}$$

$$h_{p-1}b_{p-2} - (h_{p-2} + h_{p-1})b_{p-1} + h_{p-2}b_p = 0 \tag{4.23}$$

Eqs. (4.21) to (4.23) together form the following linear system:

$$\begin{bmatrix} h_1 & -h_0 - h_1 & h_0 & & \\ h_0 & 2(h_0 + h_1) & h_1 & & \\ & \ddots & \ddots & \ddots & \\ & & h_{p-2} & 2(h_{p-2} + h_{p-1}) & h_{p-1} \\ & & h_{p-1} & -h_{p-2} - h_{p-1} & h_{p-2} \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{p-1} \\ b_p \end{bmatrix} = 3 \begin{bmatrix} 0 \\ \delta_1 - \delta_0 \\ \vdots \\ \delta_{p-1} - \delta_{p-2} \\ 0 \end{bmatrix} \tag{4.24}$$

The empty parts in the matrix of Eq. (4.24) represent zeros, which means that the matrix itself is tridiagonal, save for the third entry on the first line and the third to last entry on the last line. Solving the system of Eq. (4.24) yields the coefficients b_i (including the extra coefficient b_p, which is only needed to compute a_{p-1} and c_{p-1}), with which in turn the coefficients a_i and c_i can be determined using Eqs. (4.18) and (4.19), respectively.

Cubic Hermite Spline

Hermite interpolation is based on finding a polynomial P that satisfies the following interpolation conditions [64]:

$$P^{(k)}(x_i) = f^{(k)}(x_i) \tag{4.25}$$

$k = 0, 1, ..., k_i$. The prescription of the polynomial consists not only of the function values at each of the points x_i, but also by the first $k_i - 1$ derivatives at those points. These $p+1$ conditions specify a degree-p polynomial.

A cubic spline can be made with Hermite interpolation polynomials by giving the first derivative at each interpolation point. Each polynomial is then prescribed by four conditions and is thus of degree three. At each of the data points, the spline will be continuous up to the first-order derivative. The general procedure for setting up the expression of each polynomial is described in [64]. Applying this procedure to the i^{th} polynomial in this case yields:

$$P_i(x) = f_i \left(1 + 2\frac{x - x_i}{x_{i+1} - x_i}\right) \left(\frac{x - x_{i+1}}{x_i - x_{i+1}}\right)^2 + f_i'(x - x_i) \left(\frac{x - x_{i+1}}{x_i - x_{i+1}}\right)^2 + f_{i+1} \left(1 + 2\frac{x - x_{i+1}}{x_i - x_{i+1}}\right) \left(\frac{x - x_i}{x_{i+1} - x_i}\right)^2 + f_{i+1}'(x - x_{i+1}) \left(\frac{x - x_i}{x_{i+1} - x_i}\right)^2 \quad (4.26)$$

The polynomial can be written as:

$$P_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i \quad (4.27)$$

with the coefficients d_i given by f_i and the coefficients a_i, b_i and c_i given by:

$$a_i = 2\frac{f_i - f_{i+1}}{h_i^3} + \frac{f_i' + f_{i+1}'}{h_i^2} \quad (4.28)$$

$$b_i = 3\frac{f_{i+1} - f_i}{h_i^2} - \frac{2f_i' + f_{i+1}'}{h_i} \quad (4.29)$$

$$c_i = f_i' \quad (4.30)$$

with $h_i = x_{i+1} - x_i$.

If only a set of data points without derivative values is given, one can use the following procedure to obtain the values of f_i', which is used by the *pchip*-function of Matlab to create "shape preserving, visually pleasing interpolants without overshoots" [42]. Let δ_i again be given by Eq. (4.20). Then, for the interior points, f_i' is zero if either δ_{i-1} or δ_i is zero or if δ_{i-1} and δ_i have opposite signs. If they have the same sign, f_i' is computed using a weighted average between δ_{i-1} and δ_i:

$$f_i' = \frac{3(h_{i-1} + h_i)}{(2h_{i-1} + h_i)/\delta_i + (h_{i-1} + 2h_i)/\delta_{i-1}} \quad (4.31)$$

As for the first point, f_0' is zero if δ_0 is zero. Otherwise, it is calculated using a non-centered, three-point formula:

$$f_0' = \frac{(2h_0 + h_1)\delta_0 - h_0\delta_1}{h_0 + h_1} \quad (4.32)$$

However, if the sign of f_0' is opposite to that of δ_0, f_0' is set to zero. Furthermore, f_0' is limited in magnitude to three times that of δ_0. The approach for the last point is exactly the same, with f_{p+1}' given by:

$$f_{p+1}' = \frac{(2h_p + h_{p-1})\delta_p - h_p\delta_{p-1}}{h_p + h_{p-1}} \quad (4.33)$$

It is set to zero if either δ_p is zero or if it has a sign opposite to that of δ_p and it is limited in magnitude to $3\delta_p$.

4.3 Least-Squares Regression

Regression consists of building a function that approximates a data set, but does not necessarily reproduce it exactly. Regression is not a form of interpolation, as the interpolation condition,

i.e., the interpolant should reproduce the data exactly at the data points, is violated. Instead of using the interpolation condition to obtain the constraints of the interpolant, least-squares regression minimizes the square sum of the error at each data point. The reason for using this constraint is that it leads to a linear system. The condition that is minimized is given by:

$$R = \sum_{i=0}^{p} \left(f(x_i) - \phi(x_i)\right)^2 \tag{4.34}$$

where p is the number of data points and $\phi(x)$ is the approximating function, which is a linear combination of basis functions:

$$\phi(x) = \sum_{j=0}^{m-1} a_j \varphi_j(x) \tag{4.35}$$

When using a polynomial as approximation, the basis functions $\varphi(x)$ are monomials, as given for interpolation by Eq. (4.8), however, other basis functions can be used as well. A difference with interpolation is the fact that there can be less basis functions than data points, so $m \leq p$. The linear system to solve for least-squares is:

$$\boldsymbol{A}^T \boldsymbol{A}\boldsymbol{a} = \boldsymbol{A}^T \boldsymbol{f} \tag{4.36}$$

Note that this equation is largely the same as Eq. (4.10), with $\boldsymbol{a}$ and $\boldsymbol{f}$ containing the coefficients and function values, respectively. $\boldsymbol{A}$ is in the general case given by:

$$\boldsymbol{A} = \begin{bmatrix} \varphi_0(\boldsymbol{x}^T) & \varphi_1(\boldsymbol{x}^T) & \cdots & \varphi_m(\boldsymbol{x}^T) \end{bmatrix} \tag{4.37}$$

with $\boldsymbol{x}$ containing the data points x_i. For a polynomial approximation, this is a $p \times m$ Vandermonde matrix (see Eq. (4.11)). The system of Eq. (4.36) can then be solved by inverting $\boldsymbol{A}^T\boldsymbol{A}$, which yields:

$$\boldsymbol{a} = \left(\boldsymbol{A}^T \boldsymbol{A}\right)^{-1} \boldsymbol{A}^T \boldsymbol{f} \tag{4.38}$$

More commonly, though, one uses SVD to solve the least-squares problem. For this, one splits matrix $\boldsymbol{A}$ into a multiplication of three matrices:

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T \tag{4.39}$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthogonal matrices ($\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{I}$ and $\boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{I}$) and $\boldsymbol{S}$ is a diagonal matrix. For a derivation of $\boldsymbol{U}$, $\boldsymbol{S}$ and $\boldsymbol{V}$ from $\boldsymbol{A}$, the reader is referred to, for instance, [23]. Note that the derivation there is analytic and that computer programs use different numerical approaches. Noting that $\boldsymbol{A}^T = \boldsymbol{V}\boldsymbol{S}\boldsymbol{U}^T$ and filling Eq. (4.39) into Eq. (4.36) yields:

$$\boldsymbol{V}\boldsymbol{S}^2\boldsymbol{V}^T\boldsymbol{a} = \boldsymbol{V}\boldsymbol{S}\boldsymbol{U}^T\boldsymbol{f} \tag{4.40}$$

where use was made of the orthogonality of $\boldsymbol{U}$ and $\boldsymbol{V}$. This equation can then be written as:

$$\boldsymbol{a} = \boldsymbol{V}\boldsymbol{S}^{-1}\boldsymbol{U}^T\boldsymbol{f} \tag{4.41}$$

in which only $\boldsymbol{S}$ has to be inverted, which is simple, as it is diagonal.

4.4 Integration

In the present section, numerical integration is introduced, together with the traditional integration method that will be compared with TSI (TSI itself will be explained in the next chapter). This method is the Runge-Kutta (RK) method; in particular the RKF5(6) method, which was found to be the most efficient for reentry applications [45]. This section is split in four parts;

one on the basics of numerical integration of ordinary differential equations (ODEs), one on the errors made by numerical methods and one on the RK integration method.

4.4.1 Integrating Ordinary Differential Equations

The task of numerical-integration tools is to use a set of first order ODE to propagate the state-variables $\boldsymbol{x}$ to the next time step. A first order ODE is of the form [64]:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(t, \boldsymbol{x}) \tag{4.42}$$

where $\boldsymbol{f}(t, \boldsymbol{x})$ represents the equations of motion. For example, when using Cartesian state variables in $\mathcal{F}_I$, the equations of motion are given by Eqs. (2.68) and (2.69), with which Eq. (4.42) becomes:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{V}_I \\ \boldsymbol{g}_I + \boldsymbol{F}_{A,I}/m \end{bmatrix} \tag{4.43}$$

Integrating Eq. (4.42) yields a scheme to propagate the state in time:

$$\boldsymbol{x}(t_{n+1}) = \boldsymbol{x}(t_n) + \int_{t_n}^{t_n+1} \boldsymbol{f}(t, \boldsymbol{x}) dt \tag{4.44}$$

Numerical integration schemes use an approximation to the integral to propagate the state [54]:

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + h\boldsymbol{\Phi}(t, \boldsymbol{x}, h) \tag{4.45}$$

where subscript n denotes the current time level and h is the time step from t_n to t_{n+1}. Note that the notation $\boldsymbol{x}_{n+1}$ rather than $\boldsymbol{x}(t_{n+1})$ is used to distinguish between the numerical approximation of the value of $\boldsymbol{x}$ at t_{n+1} and the exact value. The type of numerical integrator determines function $\boldsymbol{\Phi}$. For instance, the forward Euler method, the simplest of numerical integrators, uses $\boldsymbol{f}$ evaluated at t_n [34]:

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + h\boldsymbol{f}(t_n, \boldsymbol{x}_n) \tag{4.46}$$

Numerical integrators can be categorized as single-step and multistep methods, and can be explicit or implicit. Single-step means that only the values of $\boldsymbol{f}$ of the previous time step are used, where multistep methods also use older time steps. Explicit means that $\boldsymbol{f}_{n+1}$ is not used for the computation of $\boldsymbol{x}_{n+1}$, whereas implicit methods do use $\boldsymbol{f}_{n+1}$, and since that depends on $\boldsymbol{x}_{n+1}$, an iterative process is needed to arrive at definitive values of $\boldsymbol{x}_{n+1}$. The forward Euler method is a single-step, explicit method and so are the Runge-Kutta methods.

4.4.2 Errors

There are two main types of errors for numerical integration, namely truncation errors and rounding errors [64]. Truncation errors are made when an infinite series of terms is approximated by a finite number of terms. An example of this is a Taylor series expansion of order K (Taylor series expansions are explained in more detail in Section 5.1):

$$x(t_{n+1}) = \sum_{k=0}^{K} \frac{x^{(k)}(t_n)}{k!} h^k + \mathcal{O}(h^{K+1}) \tag{4.47}$$

where $\mathcal{O}(h^{K+1})$ marks the terms the terms that are not included, which form the truncation error. For a Taylor series expansion, this error can be written as a Lagrange remainder [34]:

$$R_{K,n+1} = x(t_{n+1}) - x_{n+1} = \frac{x^{(K+1)}(t_n + \xi h)}{(K+1)!} h^{K+1} \tag{4.48}$$

with $\xi \in [0, 1]$ being an unknown constant. As can be seen, the error can usually be reduced by increasing K and/or decreasing h.

The rounding error is caused by the way computers store numbers, as only a limited number of bits is dedicated to each number. The most common way of representing numbers is the floating-point representation, using double precision. With double precision, the first 15 to 17 digits of numbers are accurate [33], the remaining digits are likely erroneous due to rounding. Where truncation errors can be controlled by using different time-step sizes or methods of different orders, rounding errors can only be decreased by raising the precision of the computer, which also increases computational time for every operation executed by the computer. Thus, in practice, the rounding error puts a limit on the achievable accuracy.

The error tolerance specified by a user consists of an absolute tolerance, ϵ_{abs}, and a relative one, ϵ_{rel}, with the latter multiplied with the current values of the state $\boldsymbol{x}$. The error tolerances for each state variable at the n^{th} time step are then:

$$\boldsymbol{\epsilon}_n = \max\{\epsilon_{rel}|\boldsymbol{x}_n|, \epsilon_{abs}\} \tag{4.49}$$

This implies that if a relative tolerance is higher than the absolute tolerance, it is used instead of the absolute tolerance. Note that, due to the fact that only the first 15 digits of the floating-point representation with double precision of a number are accurate, the minimum value for ϵ_{rel} is about $1 \cdot 10^{-15}$.

Since the Euler method in the example in the previous section is, in fact, a first-order Taylor series expansion, its local error, that is, the error made after one time step, can be assessed using the Lagrange remainder, given by:

$$\boldsymbol{R}_{1,n+1} = \frac{1}{2} h^2 \ddot{\boldsymbol{x}}(t_n + \xi h) \tag{4.50}$$

The global error, i.e., the total error at the end of the integration, is in general always one order lower than the local error [64], thus the global error of the forward Euler method is of first order. Since the order of a numerical integrator is equal to its order of global error, the forward Euler method is a first order method.

To illustrate the concept of error assessment, assume that one is simulating a falling object in a vacuum, using the constant gravity acceleration of 9.81 m/s^2. Despite that fact that this simulation is simple and the position of the object at an arbitrary point in time can be computed by hand, Eq. (4.50) shows that the forward Euler method will have an error of $4.905h^2$ each step. If one wants to limit the local error to $1 \cdot 10^{-10}$ m, he will have to use a step size of about $5 \cdot 10^{-6}$ s. This means that he would have to simulate 200,000 steps to determine the position of the object after just 1 second. Since the error made by the integrator is constant, one can determine that the global error made after 200,000 steps is about $200,000 \times 1 \cdot 10^{-10} = 2 \cdot 10^{-5}$ m, which is quite significant after only 1 second. Hence in practice, higher-order methods are used when the error tolerance is low.

4.4.3 Runge-Kutta

The Runge-Kutta methods are higher-order single-step integrators. For an RK integrator of order p, Eq. (4.45) becomes [64]:

$$\boldsymbol{x}(t_{n+1}) = \boldsymbol{x}(t_n) + h \sum_{k=0}^{K-1} c_k \boldsymbol{f}_k \tag{4.51}$$

with $\boldsymbol{f}_k$ given by:

$$\boldsymbol{f}_k = \boldsymbol{f}(t_n + \alpha_k h, \boldsymbol{x}(t_n) + h \sum_{\lambda=0}^{k-1} \beta_{k\lambda} \boldsymbol{f}_\lambda) \tag{4.52}$$

The coefficients α_k, $\beta_{k\lambda}$ and c_k depend partially on the order of the integrator, but the equations with which these are determined leave some freedom to minimize particular errors or have certain conveniences, for instance, reusing some function values $\boldsymbol{f}_k$ to decrease the total number of function evaluations. For more information on the computation of these coefficients, the reader could consult [20] or [64]. K is the number of function evaluations needed, which is equal to p for orders of 4 or lower, but increases more rapidly than p for higher-order methods.

The classical Runge-Kutta integrator is the fourth-order version, RK4, of which the coefficients are given in Table 4.1. Note that in general for explicit RK integrators, the coefficients $\beta_{k\lambda}$, for $\lambda \geq k$ are always zero, including β_{00}.

Table 4.1: The coefficients for the RK4 integrator. Source: [34]

k	α_k	β_{k0}	β_{k1}	β_{k2}	c_k
0	0	0			$\frac{1}{6}$
1	$\frac{1}{2}$	$\frac{1}{2}$			$\frac{1}{3}$
2	$\frac{1}{2}$	0	$\frac{1}{2}$		$\frac{1}{3}$
3	1	0	0	1	$\frac{1}{6}$

Writing out Table 4.1 yields the following scheme:

$$\begin{aligned}
\boldsymbol{f}_0 &= \boldsymbol{f}(t_n, \boldsymbol{x}_n) \\
\boldsymbol{f}_1 &= \boldsymbol{f}(t_n + \frac{1}{2}h, \boldsymbol{x}_n + \frac{1}{2}h\boldsymbol{f}_0) \\
\boldsymbol{f}_2 &= \boldsymbol{f}(t_n + \frac{1}{2}h, \boldsymbol{x}_n + \frac{1}{2}h\boldsymbol{f}_1) \\
\boldsymbol{f}_3 &= \boldsymbol{f}(t_n + h, \boldsymbol{x}_n + h\boldsymbol{f}_2) \\
\boldsymbol{x}_{n+1} &= \boldsymbol{x}_n + \frac{h}{6}(\boldsymbol{f}_0 + 2\boldsymbol{f}_1 + 2\boldsymbol{f}_2 + \boldsymbol{f}_3)
\end{aligned}$$

It is also possible to use RK integrators with variable step sizes. This can be done by comparing the solutions of orders p and $p+1$ and estimating the error of the p^{th} order integrator with the difference. These methods are called *Runge-Kutta-Fehlberg* methods [64]. The scheme for these methods is the same as for the normal Runge-Kutta integrators, but now the solution $\boldsymbol{x}(t_{n+1})$ of Eq. (4.51) is compared with the solution $\hat{\boldsymbol{x}}(t_{n+1})$ of:

$$\hat{\boldsymbol{x}}(t_{n+1}) = \boldsymbol{x}(t_n) + h \sum_{k=0}^{\hat{K}} \hat{c}_k \boldsymbol{f}_k \tag{4.53}$$

which is of order $p+1$, with the terms $\boldsymbol{f}_k$ still given by Eq. (4.52). The coefficients for integrators of orders five to eight with step-size control can be obtained from [20]. This type of integrators is here denoted by RKF$p(p+1)$, so RKF6(7), for instance, is a sixth-order integrator that uses a seventh-order integrator to obtain an error estimate. This error estimate should satisfy the relations:

$$|\boldsymbol{x}(t_{n+1}) - \hat{\boldsymbol{x}}(t_{n+1})| \leq \boldsymbol{\epsilon}_n \tag{4.54}$$

The largest step size h that satisfies Eq. (4.54) can be obtained with [64]:

$$h_{new} = h_n \left(\max \left\{ \frac{\boldsymbol{\epsilon}_n}{|\boldsymbol{x}(t_{n+1}) - \hat{\boldsymbol{x}}(t_{n+1})|} \right\} \right)^{\frac{1}{p+1}} \tag{4.55}$$

This equation can be used to check whether the current step size is small enough to satisfy the error tolerances; if h_{new} is smaller than h_n, the current step size is rejected and one has to recompute the current step using h_{new}. Afterwards, h_{new} can be used as a prediction for the step size of the next time step. Since it is inefficient to have to reject the step size each step, h_{new} is multiplied with a safety factor η when it is used to recompute a step or used as prediction for the next step. η is typically between 0.8 and 0.9 [64] and therefore a value of 0.85 is used during this research.

As was mentioned in the introduction of this section, the performance of RKF5(6) will be compared to that of TSI. The coefficients of this integrator are given in Table 4.2.

Table 4.2: The coefficients for the RKF5(6) integrator. Source: [20]

k	α_k	β_{k0}	β_{k1}	β_{k2}	β_{k3}	β_{k4}	β_{k5}	β_{k6}	c_k	$\hat{c}_k$
0	0	0							$\frac{31}{384}$	$\frac{7}{1408}$
1	$\frac{1}{6}$	$\frac{1}{6}$							0	0
2	$\frac{4}{15}$	$\frac{4}{75}$	$\frac{16}{75}$						$\frac{1125}{2816}$	$\frac{1125}{2816}$
3	$\frac{2}{3}$	$\frac{5}{6}$	$-\frac{8}{3}$	$\frac{5}{2}$					$\frac{9}{32}$	$\frac{9}{32}$
4	$\frac{4}{5}$	$-\frac{8}{5}$	$\frac{144}{25}$	-4	$\frac{16}{25}$				$\frac{125}{768}$	$\frac{125}{768}$
5	1	$\frac{361}{320}$	$-\frac{18}{5}$	$\frac{407}{128}$	$-\frac{11}{80}$	$\frac{55}{128}$			$\frac{5}{66}$	0
6	0	$-\frac{11}{640}$	0	$\frac{11}{256}$	$-\frac{11}{160}$	$\frac{11}{256}$	0			$\frac{5}{66}$
7	1	$\frac{93}{640}$	$-\frac{18}{5}$	$\frac{803}{256}$	$-\frac{11}{160}$	$\frac{99}{256}$	0	1		$\frac{5}{66}$

4.5 Optimization

The final type of numerical methods needed is optimization. There are many different ways to numerically optimize processes, but the methods used here are all part of the category of meta-heuristics. During the literature survey in [6], Differential Evolution (DE) was found to be the best method for reentry applications for single-objective optimization. In the same literature survey, Non-dominated Sorting Genetic Algorithm II (NSGA-II) was found to be the most commonly used method for multi-objective optimization. Furthermore, Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D) was successfully used for reentry-trajectory optimization in [57], so that method will also be tried for multi-objective optimization. This section is divided into six parts, describing respectively the basics of numerical optimization, the constraint handling, DE, multi-objective optimization, the NSGA-II and the MOEA/D.

4.5.1 Basics

The goal of optimization is to find a particular set of control variable values for which a scalar objective function $J(\boldsymbol{x}(t), \boldsymbol{u}(t), t)$ is *minimized*, with $\boldsymbol{x}$ denoting the state variables and $\boldsymbol{u}$ the control variables. Defining the set of m control variables as:

$$\boldsymbol{u}(t) \in \boldsymbol{\Omega} \tag{4.56}$$

where $\boldsymbol{\Omega}$ is the domain of the variables. The state variables have to satisfy the differential equations:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \tag{4.57}$$

which are in this case the equations of motion stated in Chapter 2.

For a trajectory optimization problem, J is typically a function of the terminal conditions or an integral of a parameter over the trajectory. The objective can, for instance, be the minimization of the heat load that is supplied to the nose of the vehicle (see Section 3.5):

$$J = Q = \int_{t_0}^{t_e} \dot{q}_c dt$$

Assuming a problem has a unique optimum, one can say that there exists no other set of control variables (within the set of constraints) for which the objective function has a better value. This optimum is then called a global optimum. A local optimum is the optimal point within a certain region around that point. Outside this region, there could be other, better optima [34]. Examples of local and global optima are shown in Figure 4.5. Note that in this figure, the axis of the objective J is upside down, so the optima are in fact minima.

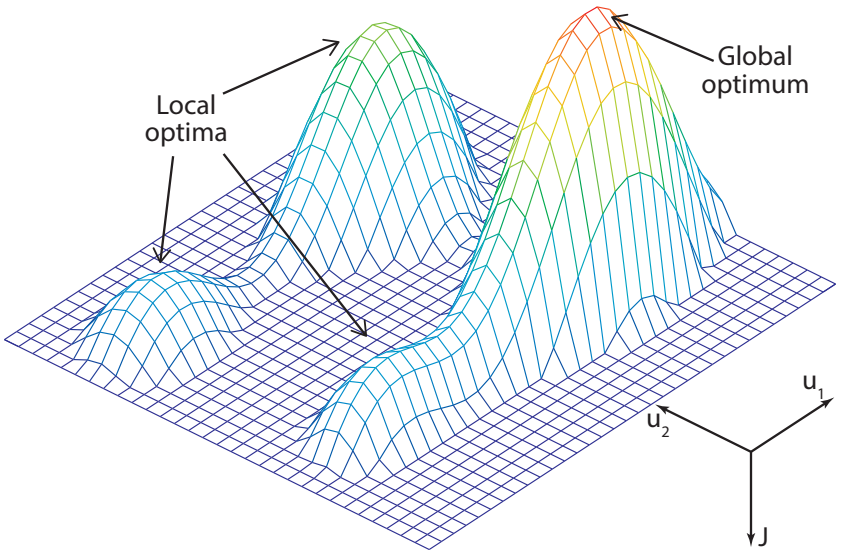


Figure 4.5: Local and global optima

4.5.2 Constraint Handling

The trajectories found through optimization must satisfy a number of constraints. These constraints for reentry were given in Section 3.3. All these constraints are inequality constraints,

which means they are of the form:

$$\boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \leq \boldsymbol{g}_{max} \tag{4.58}$$

where $\boldsymbol{g}$ indicates the constraint functions and $\boldsymbol{g}_{max}$ the constraint values. For the handling of constraints, one needs to be able to quantify how badly constraints were violated and give an individual a penalty value based on this. This value is computed as the normalized integrated constraint violation [19]:

$$J_{con} = \sum_{i=1}^{n} \int_0^{t_e} \delta_i \left(\frac{g_i}{g_{i,max}} \right) dt \tag{4.59}$$

where n is the number of constraints and δ_i is a switch function that is 0 if constraint i is not violated and 1 otherwise. Note from Eq. (4.59) how J_{con} increases the larger the constraints violations, the longer the constraints are violated and the more constraints are violated at once. This way J_{con} can be used to compare trajectories and rank them according to constraint violations. J_{con} is an additional objective function that must be equal to 0 for a trajectory to be valid. The optimization of multiple objectives at once will be discussed in Section 4.5.4.

4.5.3 Differential Evolution

DE is a single-objective optimization technique that keeps track of multiple control vectors $\boldsymbol{u}_i$, called individuals. All individuals together form a population and the goal of DE is to achieve a population where at least one individual has the best possible value for J. First, an initial population is created by assigning values to the j^{th} component of the i^{th} individual as follows [13]:

$$u_{i,j} = B_{l,j} + \text{rand}(0,1)(B_{u,j} - B_{l,j}) \tag{4.60}$$

where $B_{l,j}$ and $B_{u,j}$ are the lower and upper bound for the j^{th} control variable.

Next, DE attempts to improve the population by applying evolutionary operators to create new and better individuals. The first operator is *crossover*: for each individual $\boldsymbol{u}_i$, an offspring vector $\boldsymbol{u}_{new}$ is created, in this case through a process called binomial crossover [13]. Of $\boldsymbol{u}_{new}$, one randomly chosen component R is generated as:

$$u_{new,R} = a_R + F(b_R - c_R) \tag{4.61}$$

where $\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{c}$ are three randomly selected individuals from the population, distinct from each other and $\boldsymbol{u}_i$. Of the other components j, each one has a chance CR to be generated in the same way:

$$u_{new,j} = a_j + F(b_j - c_j) \tag{4.62}$$

otherwise it is generated as:

$$u_{new,j} = u_{i,j} \tag{4.63}$$

where CR and F are the crossover constant (between 0 and 1) and a scaling factor (between 0 and 2), respectively, both constants chosen by the user. The typical value for both parameters is 0.8 [29]. Note that each component must be within the bounds $B_{l,j}$ and $B_{u,j}$, else it is set equal to the violated bound. The fitness of $\boldsymbol{u}_{new}$ is then compared with that of $\boldsymbol{u}_i$ and only if it is better, it replaces $\boldsymbol{u}_i$ in the population.

The second evolutionary operator is *mutation*, which is applied to $\boldsymbol{u}_{new}$ after it is created. Each component of $\boldsymbol{u}_{new}$ has a small chance p (usually equal to $1/m$) to get a random value, rather than the value assigned by crossover. This value can be completely random or set by *polynomial*

mutation, which allows the user to have some control over how far the mutated value deviates from the current value [17]. For this method, one defines two parameters δ and r:

$$\delta = \frac{u_{new,j} - B_{l,j}}{B_{u,j} - B_{l,j}} \tag{4.64}$$

$$r = \text{rand}(0, 1) \tag{4.65}$$

where $\text{rand}(0, 1)$ indicates a random value between 0 and 1. If $r \leq 0.5$:

$$\delta_q = \left[2r + (1 - 2r)(1 - \delta)^{\eta}\right]^{\frac{1}{\eta}} - 1 \tag{4.66}$$

And otherwise, if $r > 0.5$:

$$\delta_q = 1 - \left[2(1 - r) + (2r - 1)\delta^{\eta}\right]^{\frac{1}{\eta}} \tag{4.67}$$

where η is the distribution index, $\eta \geq 1$. The new value of $u_{new,j}$ is then:

$$u_{new,j} = u_{new,j} + \delta_q(B_{u,j} - B_{l,j}) \tag{4.68}$$

The main purpose of mutation is to create more diversity in a population to help the optimizer fully explore the solution space and escape local optima. The parameter η determines how much the mutated value can deviate from the original value; the higher η, the closer the mutated value stays to the original [17]. Therefore, η is typically set to a small value, i.e., smaller than 10.

4.5.4 Multi-Objective Optimization

Multi-objective optimization, as the name suggests, is the simultaneous optimization of multiple objective functions. One way to do so is to make J an aggregate function of the different objectives [21]. This can, for instance, be linear:

$$J = \sum_{i=1}^{n} w_i J_i \tag{4.69}$$

where n is the number of objectives and w_i are the weights assigned to each objective. The downside of this method is that the user needs to know the ratio of the weights that gives the optimal solution. Often, determining the weights is an optimization process on its own [55].

Another way is to have a vector $\boldsymbol{J}$, in which each entry is an objective function. In that case, Pareto-front ranking can be used to rank the individuals in the population. This ranking is based on Pareto dominance: an individual is said to dominate another if for each objective it does not have a worse value and it has a better value for at least one of the objectives. So individual A dominates individual B, if $J_{i,A} \leq J_{i,B}$ for $i = 1, ..., n$ and $J_{i,A} < J_{i,B}$ for at least one i. Note that it is possible that neither of two individuals dominates the other, if A is better at some objectives and B at others. This allows one to divide the population into so-called *Pareto fronts*. A Pareto front is a group of individuals, in which none of the individuals dominates another [55]. The goal of multi-objective optimization is then to find the *Pareto set*, which contains all the solutions to a problem that are not dominated by other solutions. In Figure 4.6, the objective space of an optimization with 100 individuals is shown. In this figure, the first four Pareto fronts are displayed. In the following subsections, two multi-objective methods, NSGA-II and MOEA/D, are explained and both will be tested during simulation to see which yields the best results.

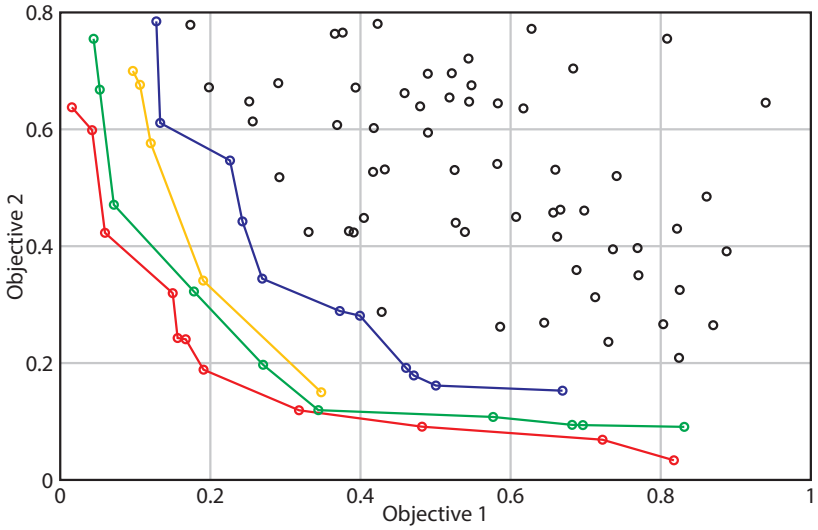


Figure 4.6: An example of Pareto fronts

4.5.5 Non-dominated Sorting Genetic Algorithm II

Of the different algorithms for multi-objective optimization, NSGA-II is the most popular. In the current study, crossover between the individuals will be done using DE, as it was found to be the best method for reentry [6]. Polynomial mutation will be used as mutation scheme (see Section 4.5.3). The crossover and mutation operators are then used on each of the individuals in the population to create an extension of the population of size N_p. The entire population of size $2N_p$ is then sorted by Pareto front and within each front, NSGA-II sorts them based on crowding. This is done by drawing a rectangle (hyper-rectangle in case of more than two objective functions) around each individual, of which the corner points are determined by the individual's neighbors in the same front (see Figure 4.7). The crowding level of an individual is equal to the sum of the dimensions of the rectangle. A high crowding level makes him less preferable than his colleagues at the less crowded parts of the front. This then allows the optimizer to select the N_p number of "best" individuals to move on to the next generation [16]. The NSGA-II algorithms for the computation of the Pareto fronts and crowding distance can, for instance, be found in [63].

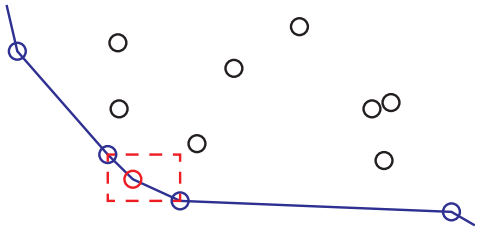


Figure 4.7: Computing the level of crowding for the red individual in NSGA-II

4.5.6 Multi-Objective Evolutionary Algorithm based on Decomposition

MOEAD is a method that decomposes a multi-objective problem into N_p scalar subproblems and then optimizes them simultaneously. Each subproblem is a linear aggregate of the objective functions J_i and each individual $\boldsymbol{u}_j$ in the population is assigned to the subproblem it minimizes. The most commonly used decomposition method is the *Tchebycheff approach* [38, 71], which

defines each subproblem as the minimization of a parameter G_j, defined as:

$$G_j(\boldsymbol{u}) = \max_{i=1...n} \{\lambda_{j,i} | J_i(\boldsymbol{u}) - z_i|\} \tag{4.70}$$

with $i = 1, ..., n$ being the index for the objective functions and $j = 1, ..., N_p$ being the index for each of the subproblems. $\boldsymbol{z}$ is the ideal objective vector, containing the minimal values for each of the objective values for the entire population:

$$z_i = \min_{j=1...n} \{J_i(\boldsymbol{u}_j)\} \tag{4.71}$$

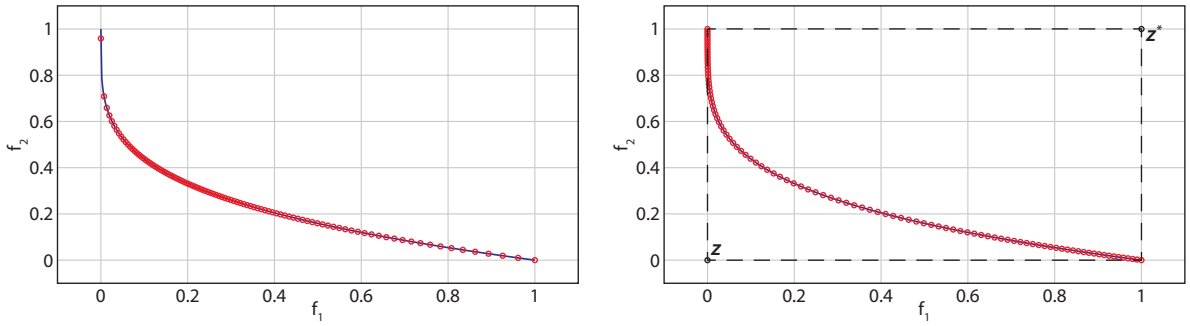
The weight vectors $\boldsymbol{\lambda}_j$ each contain n entries that sum up to 1. It is common to have the weights be evenly distributed between 0 and 1 for each i. A method to generate such weight vectors for $n = 2$ is very intuitive and is left as an exercise for the reader. For higher objective dimensions, one can use the method in [29].

The following implementation of MOEA/D was taken from [29]:

During the initialization, create the weight vectors $\boldsymbol{\lambda}_j$ and neighborhoods $\boldsymbol{B}_j$. Each $\boldsymbol{B}_j$ contains the T weight vectors with the smallest Euclidean distance to $\boldsymbol{\lambda}_j$. Create the population $\boldsymbol{P}$, containing the N_p individuals and compute $\boldsymbol{z}$. Then, each generation, for $j = 1, ..., N_p$:

1. Create a vector $\boldsymbol{u}_{new}$ with DE (see Section 4.5.3), but now using $\boldsymbol{a} = \boldsymbol{u}_j$ and $\boldsymbol{b}$ and $\boldsymbol{c}$ taken from $\boldsymbol{B}_j$ or $\boldsymbol{P}$.
2. Compute the objective function values of $\boldsymbol{u}_{new}$ and update $\boldsymbol{z}$.
3. Compare $\boldsymbol{u}_{new}$ with $\boldsymbol{u}_j$ for G_j: if $G_j(\boldsymbol{u}_{new}) < G_j(\boldsymbol{u}_j)$, $\boldsymbol{u}_{new}$ takes the place of $\boldsymbol{u}_j$ in the population.
4. Compare $\boldsymbol{u}_{new}$ with the other individuals $\boldsymbol{u}_k$ from $\boldsymbol{B}_j$ or $\boldsymbol{P}$: if $G_k(\boldsymbol{u}_{new}) < G_k(\boldsymbol{u}_k)$, $\boldsymbol{u}_{new}$ takes the place of $\boldsymbol{u}_k$ in the population. The total number of replacements (including that of step **3**) is limited to N_r.

Note that in steps 1 and 4, one randomly takes the individuals from $\boldsymbol{B}_j$ or $\boldsymbol{P}$, with a chance of 90% to take them from $\boldsymbol{B}_j$. The neighborhood size T is typically set to $N_p/10$ and maximum number of replacements N_r is typically set to 2.



(a) Tchebycheff approach

(b) Convex Tchebycheff approach

Figure 4.8: Performance of MOEAD for the "convex Pareto-optimal front"-problem from [15].

The Tchebycheff approach is less suited for problems with a convex Pareto set. Take for instance "convex Pareto-optimal front"-problem, which is given by Eq. (7.6). The results of MOEAD

with the Tchebycheff approach for this problem are shown in Figure 4.8(a), in which also the line for the Pareto set is plotted ($f_2 = 1 - f_1^{0.25}$). As can be seen, the part of the Pareto set near the end at (0,1) is poorly populated and there is no individual that has reached (0,1). To solve this problem, consider an alternative approach:

$$G_j(\boldsymbol{u}) = \min_{i=1\ldots n} \left\{ \lambda_{j,i} \left[J_i(\boldsymbol{u}) - z_i^* \right] \right\} \tag{4.72}$$

where $\boldsymbol{z}^*$ contains the maximum objective values of the population members have a minimum value for one of the objectives (and thus span the first Pareto front). $\boldsymbol{z}$ and $\boldsymbol{z}^*$ are shown in Figure 4.8(b), where also the results with this approach, that will here be called the *convex Tchebycheff approach*, are shown. The resulting population for this approach is better spread over the Pareto set and reaches all parts of it. This is because of the position of $\boldsymbol{z}$ and $\boldsymbol{z}^*$ w.r.t. to the Pareto set; for a convex Pareto set, points near the ends lie very close to one another from the point of view of $\boldsymbol{z}$. From the point of view of $\boldsymbol{z}^*$, the points are more equidistant, making subproblems that use the distances to $\boldsymbol{z}^*$ better suited for problems with convex Pareto sets. Similarly, the regular Tchebycheff approach will be better suited for problems with concave Pareto sets.

Chapter 5

Taylor Series Integration

This chapter will introduce TSI and explain the application to ODEs and the problems that arise when using it for reentry trajectories. The TSI method uses Taylor series expansions to integrate the differential equation, making use of automatic differentiation to obtain the higher-order derivatives. One of the main features of this method is that the order can be chosen by the user [49]. Using a high order allows the integrator to attain high accuracy without resorting to very small step sizes. Another advantage is that the results of integration can be interpolated with the same order of accuracy as the integration (see Section 5.1).

Automatic differentiation obtains the higher-order derivatives of a function through recurrence relations between a derivative and the lower-order derivatives (including the function itself). By using these recurrence relations, the function no longer needs to be differentiated explicitly. One drawback of the recurrence relations, though, is that they are limited to purely analytic expressions, so if a value is computed by other means (iterative methods, data table searches etc.), TSI may not be (directly) applied. Furthermore, as will be explained in Section 5.5, there should not be a discontinuity in the middle of a time step, as TSI can only process the equations on one side of the discontinuity at once.

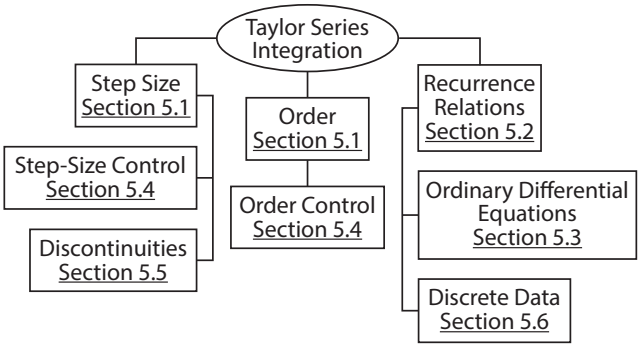


Figure 5.1: Road map for Chapter 5.

In Section 5.1, the basics of TSI are given, followed by the recurrence relations for a number of common analytic functions in Section 5.2. Then it is shown how to solve an ODE with TSI in Section 5.3, followed by the order and time-step size selection for TSI in Section 5.4. Sections 5.5 and 5.6 then describe the methods that were developed to deal with discontinuities and discrete data tables, respectively.

5.1 Basics

The derivations in this section were obtained from [49]. The basics of TSI start with the Taylor series expansion of a function x about a point in time t_n. This series expansion can be used to obtain an approximation x_{n+1} to the value of x at another point in time t_{n+1}:

$$x(t_{n+1}) \approx x_{n+1} = \sum_{k=0}^{K} \frac{x_n^{(k)}}{k!} h^k \tag{5.1}$$

In this equation, h is step size from t_n to t_{n+1}, K is the maximum order of the series and $x^{(k)}$ is the k^{th} derivative of x. The approximation x_{n+1} becomes exact as K goes to infinity, but this will never happen in any actual application. In practice, K should be large enough and h should be small enough, such that the error made by Eq. (5.1) is smaller than some user-defined error tolerance (see Section 5.4).

For TSI, Taylor coefficients (also known as normalized derivatives) are formulated as:

$$(x_n)_k = \frac{x_n^{(k)}}{k!} \tag{5.2}$$

The notation used here for the Taylor coefficients originates from of [49]. Other notations for the coefficients includes capital letters, $X_n(k)$ [62], and the notation $x_n^{[k]}$ [30]. From Eq. (5.2), it can be seen that $(x_n)_0 = x_n$. The higher-order coefficients are then obtained from recurrence relations, which are explained in the next sections. Using Taylor coefficients, Eq. (5.1) can written as:

$$x_{n+1} = \sum_{k=0}^{K} (x_n)_k h^k \tag{5.3}$$

This equation shows that, when the Taylor coefficients up to order K are known, an arbitrary value for h can be used, as long as it is smaller the limit h_{max} set by the error tolerance. Thus the Taylor coefficients at a time point t_n during an integration can be used to obtain the value of x anywhere between t_n and $t_n + h_{max}$. If one stores the Taylor coefficients computed during an integration, he can use these to interpolate the solution with the same order of accuracy as the original integration and with little computational effort. This unlike the traditional integrators discussed in the previous chapter, for which the interpolation is typically done using low order interpolation methods, such as cubic splines.

The relationship between the coefficients of a function and those of its derivative can be found using Eq. (5.2):

$$(\dot{x})_k = \frac{\dot{x}^{(k)}(t)}{k!} \tag{5.4}$$

The relationship between coefficients $(x)_k$ and $(\dot{x})_{k-1}$ is then obtained by modifying Eq. (5.4) and noting that $\dot{x}^{(k)} = x^{(k+1)}$:

$$(\dot{x})_k = \frac{\dot{x}^{(k)}(t)}{k!} \frac{(k+1)}{(k+1)} = \frac{x^{(k+1)}(t)}{(k+1)!}(k+1) = (k+1)(x)_{k+1} \tag{5.5}$$

Rearranging this equation and using $j = k + 1$ yields:

$$(x)_j = j^{-1}(\dot{x})_{j-1} \tag{5.6}$$

This is the basis for most of the recurrence relations and is also important when solving ODEs, as will be shown in Section 5.3.

5.2 Recurrence relations for Basic Mathematical Operations

The following basic recurrence relations were obtained from [30, 49]. These equations are based on Leibnitz's generalization rule for the derivatives of a multiplication $x \cdot y$:

$$(xy)^{(k)} = \sum_{j=0}^{k} \binom{k}{j} x^{(j)} y^{(k-j)} \tag{5.7}$$

where $x(t)$ and $y(t)$ are arbitrary functions, given by analytic expressions. This equation can be seen as a generalization of the chain-rule for differentiation. The basic recurrence relations are as follows:

$$(x \pm y)_k = (x)_k \pm (y)_k \tag{5.8}$$

$$(xy)_k = \sum_{j=0}^{k} (x)_j (y)_{k-j} \tag{5.9}$$

$$(x/y)_k = \frac{1}{(y)_0} \left[(x)_k - \sum_{j=1}^{k} (y)_j (x/y)_{k-j} \right] \tag{5.10}$$

$$(x^\alpha)_k = \frac{1}{(x)_0} \sum_{j=1}^{k} \left(\frac{j}{k}(\alpha+1) - 1 \right) (x)_j (x^\alpha)_{k-j} \tag{5.11}$$

$$(e^x)_k = \frac{1}{k} \sum_{j=1}^{k} j (x)_j (e^x)_{k-j} \tag{5.12}$$

$$(\sin x)_k = \frac{1}{k} \sum_{j=1}^{k} j (x)_j (\cos x)_{k-j} \tag{5.13}$$

$$(\cos x)_k = -\frac{1}{k} \sum_{j=1}^{k} j (x)_j (\sin x)_{k-j} \tag{5.14}$$

For all equations, $k \geq 1$. Note the singularity in Eq. (5.10) for $(y)_0 = 0$, although this only occurs when computing the derivatives of a division by zero, which will not occur during the simulations in this study. Eq. (5.11) has a singularity when $(x)_0 = 0$, but this will also not occur here. A full derivation of these equations is not given here, but one can see that Eq. (5.8) is directly related to one of the rules of differentiation, namely that the derivative of $(x+y)$ can be found by separately differentiating x and y and adding the results together. Also, Eq. (5.7), when written in terms of Taylor coefficients (using Eq. (5.2)), directly yields Eq. (5.9).

From the application of Eq. (5.9), two things can be noted. The first is that the arguments x and y may be swapped, i.e., $(xy)_k = (yx)_k$. The second deals with the multiplication of a variable with a constant c:

$$(cx)_k = (c)_0 (x)_k + \sum_{j=1}^{k} \cancelto{0\,\forall j}{(c)_j} (x)_{k-j} = c(x)_k \tag{5.15}$$

Similarly, $(cxy)_k = c(xy)_k$. Sometimes it is possible to combine recurrence relations without creating nested summations. If a variable is given by xy/z, i.e., a multiplication followed by a

division, its recurrence relation can be found by first setting up the recurrence relation for the division of xy by z and then inserting the relation for the multiplication of x and y:

$$(xy/z)_k = \frac{1}{(z)_0}\left[\sum_{j=0}^{k}(x)_j(y)_{k-j} - \sum_{j=1}^{k}(z)_j(xy/z)_{k-j}\right] \tag{5.16}$$

From this it also follows that a division x/y multiplied with a constant c gives the relation:

$$(cx/y)_k = \frac{1}{(y)_0}\left[c(x)_k - \sum_{j=1}^{k}(y)_j(cx/y)_{k-j}\right] \tag{5.17}$$

The relations of $\arcsin x$ and $\arccos x$ can be derived from Eqs. (5.13) and (5.14). Eq. (5.13) can be written as:

$$(x)_k = (\arcsin x)_k(y)_0 + \frac{1}{k}\sum_{j=1}^{k-1} j(\arcsin x)_j(y)_{k-j} \tag{5.18}$$

where y marks the cosine, so if $x = \sin\theta$, then $y = \cos\theta$. One could use the knowledge of trigonometry and replace y by $\sqrt{1-x^2}$, but this results in $y = |\cos\theta|$, i.e., the absolute value of the cosine. This is the wrong result in case the cosine is actually negative in the current geometry. Solving this equation for $(\arcsin x)_k$ yields:

$$(\arcsin x)_k = \frac{1}{(y)_0}\left[(x)_k - \frac{1}{k}\sum_{j=1}^{k-1} j(\arcsin x)_j(y)_{k-j}\right] \tag{5.19}$$

Since both the Taylor coefficients of the sine and cosine of the angle are needed to obtain the recurrence relation for arcsin, it is necessary to know the complete geometry of a problem. A similar expression can be found for $\arccos x$ by rewritting Eq. (5.14):

$$(\arccos x)_k = \frac{-1}{(y)_0}\left[(x)_k + \frac{1}{k}\sum_{j=1}^{k-1} j(\arccos x)_j(y)_{k-j}\right] \tag{5.20}$$

with y now marking the sine of the angle. Again both the sine and cosine of the angle are needed.

The recurrence relations of arcsin and arccos should be used with caution. This is due to the fact that their recurrence relations are both singular when $x = 1$. For arcsin, this occurs when the angle is 90° or 270° and for arccos, this occurs when the angle is 0° or 180°. Therefore, when computing an angle, use the arcsin when close to the singularities of arccos and vice versa.

5.3 Solving an Ordinary Differential Equation

The integration of first order ODEs with numerical integrators has been explained in Section 4.4.1. The application of TSI to ODEs is very similar to the RK integrators, but for TSI the following steps have to be performed prior to the integration:

1. The recurrence relations for the Taylor coefficients of each of the state variables are determined. If this step has been executed in the past for the same set of equations, there is no need to redo it.

2. Assign the initial values of the state variables to $(\boldsymbol{x}_n)_0$ in the Taylor coefficient matrices.

And the following steps during integration:

3. Compute the values of the non-state variables at this time step.
4. Compute the Taylor coefficients for each variable from $k = 1$ up to $k = K$.
5. Determine the time step size and compute the values of the state variables for the next time step. Go to step 3 for the next time step.

An important detail to notice about TSI is that the Taylor coefficients of $\boldsymbol{x}$ for all time steps should be stored. This is because i) the lower-order coefficients get reused to compute the higher-order ones (see the recurrence relations in the previous section), ii) all coefficients of a time step are needed to compute the initial value for the next time step (see example at the end of this section), and iii) they can be used after integration to interpolate the solution between time steps. For other variables, the Taylor coefficients only need to be stored for one time step, so that the user can reuse the memory for these variables at each step.

As an example, without considering its exact solution, the following one-dimensional ODE will be rewritten:

$$\dot{x} = \frac{xt + 1}{\cos t} \tag{5.21}$$

To apply automatic differentiation, this ODE is split up in two auxiliary functions, one for the numerator and one for the denominator:

$$f_1 = xt + 1 \tag{5.22}$$

$$f_2 = \cos t \tag{5.23}$$

The recurrence relation of $\cos t$ requires the coefficients of $\sin t$ (see Eq. (5.14)), so the third variable is:

$$f_3 = \sin t \tag{5.24}$$

From here on out, the time step index n is omitted to limit the number of indices in the equations. The Taylor coefficients of f_1 in turn require Taylor coefficients of t and 1, which can be found through Eq. (5.2). All derivatives of 1 are zero, so $(1)_k$ is zero. t has only a first derivative, which is equal to 1, so $(t)_1 = 1$ and $(t)_k = 0$ for $k \geq 2$. This gives:

$$(f_1)_k = (xt)_k = \sum_{j=0}^{k} (t)_j (x)_{k-j} = t(x)_k + (x)_{k-1} \tag{5.25}$$

and the coefficients of f_2 and f_3 are:

$$(f_2)_k = (\cos t)_k = -\frac{1}{k} \sum_{j=1}^{k} j(t)_j (\sin t)_{k-j} = -\frac{1}{k} (\sin t)_{k-1} = -\frac{1}{k} (f_3)_{k-1} \tag{5.26}$$

$$(f_3)_k = (\sin t)_k = \frac{1}{k} \sum_{j=1}^{k} j(t)_j (\cos t)_{k-j} = \frac{1}{k} (\cos t)_{k-1} = \frac{1}{k} (f_2)_{k-1} \tag{5.27}$$

The initial Taylor coefficients $(f_1)_0$, $(f_2)_0$ and $(f_3)_0$ are given by the right-hand sides of Eqs. (5.22) to (5.24). The recurrence relation of the original ODE is then:

$$(\dot{x})_k = (f_1/f_2)_k = \frac{1}{(f_2)_0} \left[(f_1)_k - \sum_{j=1}^{k} (f_2)_j (\dot{x})_{k-j} \right] \tag{5.28}$$

Finally, the next order Taylor coefficient of x may be determined using Eq. (5.6):

$$(x)_{k+1} = (k+1)^{-1}(\dot{x})_k \tag{5.29}$$

Overall, the sequence in which the functions have to be determined for each order k depends on the dependencies of the Taylor coefficients on one another. For this ODE, the sequence is $(x)_k$ (from initial conditions or the previous order $k-1$), followed by $(f_1)_k$, $(f_2)_k$ and $(f_3)_k$ in an arbitrary order. In Table 5.1, the recurrence relations for $k=1$ to $k=4$ are worked out. As can be seen, only the expression of $(\dot{x})_k$ grows as function of k, whereas the other relations are a simple function of previous coefficients. For reentry mechanics, simple recurrence relations are rare; most relations will grow as k increases.

Once the Taylor coefficients of x_n are determined, they can form the Taylor series expansion, using Eq. (5.3), which can then be used to obtain x_{n+1}:

$$x_{n+1} = \sum_{k=0}^{K} (x_n)_k h^k \tag{5.30}$$

The selection of the maximum order K and the time-step size h is discussed in the next section.

Table 5.1: Expressions for the Taylor coefficients of the example ODE up to order four.

$(x)_1$	$(\dot{x})_0$	$(f_2)_1$	$-(f_3)_0$
$(x)_2$	$\frac{1}{2}(\dot{x})_1$	$(f_2)_2$	$-\frac{1}{2}(f_3)_1$
$(x)_3$	$\frac{1}{3}(\dot{x})_2$	$(f_2)_3$	$-\frac{1}{3}(f_3)_2$
$(x)_4$	$\frac{1}{4}(\dot{x})_3$	$(f_2)_4$	$-\frac{1}{4}(f_3)_3$
$(f_1)_1$	$t(x)_1 + (x)_0$	$(f_3)_1$	$(f_2)_0$
$(f_1)_2$	$t(x)_2 + (x)_1$	$(f_3)_2$	$\frac{1}{2}(f_2)_1$
$(f_1)_3$	$t(x)_3 + (x)_2$	$(f_3)_3$	$\frac{1}{3}(f_2)_2$
$(f_1)_4$	$t(x)_4 + (x)_3$	$(f_3)_4$	$\frac{1}{4}(f_2)_3$
$(\dot{x})_1$	$\big[(f_1)_1 - (f_2)_1(\dot{x})_0\big]/(f_2)_0$		
$(\dot{x})_2$	$\big[(f_1)_2 - (f_2)_1(\dot{x})_1 - (f_2)_2(\dot{x})_0\big]/(f_2)_0$		
$(\dot{x})_3$	$\big[(f_1)_3 - (f_2)_1(\dot{x})_2 - (f_2)_2(\dot{x})_1 - (f_2)_2(\dot{x})_0\big]/(f_2)_0$		
$(\dot{x})_4$	$\big[(f_1)_4 - (f_2)_1(\dot{x})_3 - (f_2)_2(\dot{x})_2 - (f_2)_3(\dot{x})_1 - (f_2)_4(\dot{x})_0\big]/(f_2)_0$		

5.4 Order and Time-Step Size

The present section will list a number of methods to control the order and time-step size of TSI. As was mentioned in Section 4.4.2, increasing the order and/or decreasing the step size decreases the error made by an integration step. The goal of these controllers is then to minimize computational time, without violating the user's error tolerances. The error tolerances for each state variable at time step n are given by:

$$\boldsymbol{\epsilon}_n = \max\left\{\epsilon_{rel}|\boldsymbol{x}_n|, \epsilon_{abs}\right\} \tag{4.49}$$

5.4.1 Time-Step Size

In this subsection, three different step-size controllers are listed. During simulation, these different methods will be tested during simulation to see which best satisfies error tolerances and that method will be used during all other simulations. These methods were selected from various sources, based on whether they gave unique results (for instance, the scheme described in [62] gives similar results to step-size controller 2, but may require multiple iterations) and complexity (the methods described in [9] were considered to be too complex).

Step-Size Controller 1

In Section 4.4.2, the Lagrange remainder was given as:

$$R_{k,n+1} = x(t_{n+1}) - x_{n+1} = \frac{x^{(k+1)}(t_n + \xi h)}{(k+1)!} h^{k+1} \tag{4.48}$$

The first method determines the optimal step size for a series of order K with the Lagrange remainder of the Taylor series of order $K-2$ [3]:

$$\begin{aligned} \boldsymbol{R}_{K-2,n+1} &= \frac{\boldsymbol{x}^{(K-1)}(t_n + \xi h)}{(K-1)!} h^{K-1} \\ &= \frac{h^{K-1}}{(K-1)!} \left[\boldsymbol{x}^{(K-1)}(t_n) + \boldsymbol{x}^{(K)}(t_n)\xi h + \cdots + \frac{\boldsymbol{x}^{(K-1+i)}(t_n)}{i!} (\xi h)^i + \cdots \right] \end{aligned}$$

Taking only the first two terms of this expansion and approximating ξh by h gives:

$$\boldsymbol{R}_{K-2,n+1} \approx \frac{\boldsymbol{x}^{(K-1)}(t_n)}{(K-1)!} h^{K-1} + \frac{\boldsymbol{x}^{(K)}(t_n)}{(K-1)!} h^K \tag{5.31}$$

The Lagrange remainder should then be smaller than or equal to the error tolerance. Writing Eq. (5.31) in terms of Taylor coefficients gives the following error tolerance formulas:

$$|(\boldsymbol{x}_n)_{K-1}| h^{K-1} + K|(\boldsymbol{x}_n)_K| h^K \leq \boldsymbol{\epsilon}_n \tag{5.32}$$

In [3], a solution for Eq. (5.32) is obtained with Newton's method (see Section 4.1.2). However, as described there, the method does not converge to the proper root. Instead, an exponential form of FPI (see Section 4.1.1) is used to obtain a scheme, which gives a close approximation of the root of Eq. (5.32) in one iteration [62]:

$$h_{n,i+1} = \exp\left[\frac{1}{K-1} \ln\left(\min\left\{ \frac{\boldsymbol{\epsilon}_n}{|(\boldsymbol{x}_n)_{K-1}| + K|(\boldsymbol{x}_n)_K| h_{n,i}} \right\} \right) \right] \tag{5.33}$$

The initial value for this method is found by ignoring the term $K(\boldsymbol{x}_n)_K h^K$ in Eq. (5.32):

$$|(\boldsymbol{x}_n)_{K-1}| h_{n,0}^{K-1} \leq \boldsymbol{\epsilon}_n \tag{5.34}$$

which results in:

$$h_{n,0} = \left(\min\left\{ \frac{\boldsymbol{\epsilon}_n}{|(\boldsymbol{x}_n)_{K-1}|} \right\} \right)^{1/(K-1)} \tag{5.35}$$

Step-Size Controller 2

In [3] also a simpler controller is proposed, using Eq. (5.35), but now using both the K^{th} and the $(K-1)^{\text{th}}$ term of series:

$$h_n = \min\left\{\left(\min\left\{\frac{\boldsymbol{\epsilon}_n}{|(\boldsymbol{x}_n)_{K-1}|}\right\}\right)^{1/(K-1)}, \left(\min\left\{\frac{\boldsymbol{\epsilon}_n}{|(\boldsymbol{x}_n)_K|}\right\}\right)^{1/K}\right\} \tag{5.36}$$

Step-Size Controller 3

In [30], a step-size controller is given, which bases the step size selection on an estimate of the radius of convergence:

$$\rho_{est} = \min\left\{\rho^{(K-1)}, \rho^{(K)}\right\} \tag{5.37}$$

The radius of convergence of a Taylor series can be seen as the largest (real) value of h for which the series with $K = \infty$ converges to the exact solution. The calculation of the terms $\rho^{(j)}$ is done using:

$$\rho^{(j)} = \left(\min\left\{\frac{\boldsymbol{x}_n^*}{|(\boldsymbol{x}_n)_j|}\right\}\right)^{1/j} \tag{5.38}$$

with:

$$\boldsymbol{x}_n^* = \max\left\{|(\boldsymbol{x}_n)_0|, 1\right\} \tag{5.39}$$

In [30], it was found that dividing ρ by e^2 gives a good estimate for the optimal step size:

$$h_n = \frac{\rho_{est}}{e^2}\exp\left(\frac{0.7}{1-K}\right) \tag{5.40}$$

where $\exp\left(0.7/(1-K)\right)$ is an optional safety factor, used to reduce step sizes for low orders; it tends toward 1 as K increases.

5.4.2 Order

The step size of TSI is easier to control than the order, as the step size can be computed after the Taylor coefficients are computed, whereas the order has to be computed beforehand. Otherwise, one may end up having to compute more coefficients, which in turn influences the selection of the step size, which may require even more coefficients, etc. The first strategy is to keep the order fixed. In [3], it is stated that, for certain problems, there is an optimal fixed order that gives the best performance. However, this order will have to be determined through trial and error. On the other hand, in [30], it is argued that to increase the accuracy, it is less computationally expensive to increase the order than to decrease the step size. During simulation, the different strategies for order control will be compared to determine which yields the lowest computational times.

Order Controller 1

In [3], the following estimation for an optimal order for each time step is given, based on the previous step. It limits the possible order of the next step to three options: $K_n - p$, K_n and

$K_n + p$, with p being chosen by the user as the variation in order between two successive steps. The following scheme is used:

$$\begin{aligned}
&\text{if } K_n > K_{n-1} \text{ or } h_n < h_{n-1} \text{ then} \\
&\quad \text{if } \left(\tfrac{K_n+p+1}{K_n+1}\right)^2 < \eta_1 \tfrac{h^+_{est}}{h_n} \text{ then } K_{n+1} = K_n + p \text{ else } K_{n+1} = K_n \\
&\text{else} \\
&\quad \text{if } \left(\tfrac{K_n-p+1}{K_n+1}\right)^2 < \eta_2 \tfrac{h^-_{est}}{h_n} \text{ then } K_{n+1} = K_n - p \text{ else } K_{n+1} = K_n
\end{aligned}$$

where η_1 and η_2 are control factors that are set by the user. They determine how easily the order is increased or decreased, respectively. h^-_{est} and h^+_{est} are estimates for the current step size. This method checks whether the order should be increased if the order was also increased after the last step or if h was decreased. Otherwise, it checks whether the order can be decreased. In [3], the order of the first step, K_0, is computed using the error tolerances, but here, it is chosen to have the user pick K_0, so that it can be optimized for best performance of TSI.

Parameter h^-_{est} is given by:

$$h^-_{est} = \epsilon_n^{1/(K_n-p+1)} ||(\boldsymbol{x}_n)_{K_n-p}||_\infty^{-1/(K_n+p)} \tag{5.41}$$

Parameter h^+_{est} first requires an estimate of the radius of convergence ρ_{est} of the Taylor series:

$$\rho_{est} = \min \left\{ \left|\left| \frac{(\boldsymbol{x}_n)_{K_n-1}}{(\boldsymbol{x}_n)_{K_n}} \right|\right|_\infty, \left|\left| \frac{(\boldsymbol{x}_n)_{K_n-2}}{(\boldsymbol{x}_n)_{K_n}} \right|\right|_\infty^{1/2}, \left|\left| \frac{(\boldsymbol{x}_n)_{K_n-3}}{(\boldsymbol{x}_n)_{K_n-1}} \right|\right|_\infty^{1/2} \right\} \tag{5.42}$$

$$h^+_{est} = \epsilon_n^{1/(K_n+p+1)} \left(\frac{||(\boldsymbol{x}_n)_{K_n}||_\infty}{\rho^p_{est}} \right)^{-1/(K_n-p)} \tag{5.43}$$

Order Controller 2

The second order controller was given in [30], where it is suggested to use the following equation to compute the order for each step:

$$K_n = \text{ceil} \left\{ -\frac{1}{2} \ln \epsilon_n + 1 \right\} \tag{5.44}$$

where ceil{} indicates that the result should be rounded up. This scheme is clearly much simpler than the other scheme, but it also uses no information from Taylor coefficients, so it is mostly independent of the problem (with the exception of the relative error tolerance, which is based on the magnitude of the leading terms of a Taylor series).

5.5 Discontinuities

When a function is piecewise, such as the one shown in Figure 5.2, the shape and/or value of the function suddenly changes at certain points, caused by discontinuities of the function or its derivatives [34]. The line in Figure 5.2 has a change in its analytic equation at points t_1, t_2 and t_3. At point t_1, the discontinuity leads to a jump in value, but its derivatives remain the same. At point t_2, the shape changes, but there is no jump in value and at t_3, both shape and value change. A Taylor series expansion of a function about a point t makes a guess of the value at $t + h$, based solely on the information that the function and its derivatives offer at point t. When trying to predict the value of a function past a discontinuity from a point before it, the

Taylor series will make an error (not counting lucky guesses), as no information of the behavior of the function past the discontinuity is contained in the equation before it. The dashed lines in Figure 5.2 show the extension of each part of the function into the next part, which clearly shows the errors that could be made. This means that the time-step size should be limited to prevent jumps over discontinuities or, if possible, only fully continuous functions should be used.

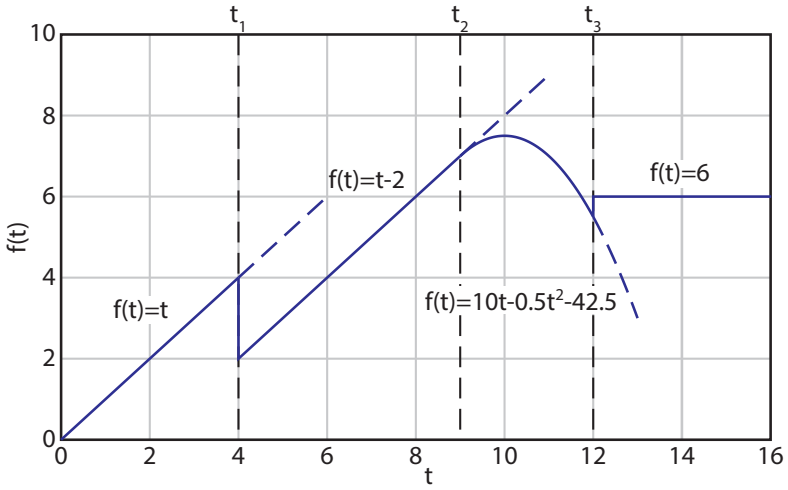


Figure 5.2: An arbitrary piecewise function

In the reentry applications discussed in this report, there are six cases of discontinuities. The first case is the equations of US76 that change for different altitude layers (see Section 2.4.3 for a description of US76). The second is the control profile that is defined as a piecewise polynomial and therefore changes when crossing each node (see Section 3.2.1 for the definition of the control profile). The third originates from the fact that the regression fits of the aerodynamics are only defined for the range of Mach numbers from 2.5 to 20 (see Section 3.1.2). When either borders of this range are passed, the aerodynamics become independent of the Mach number and thus their definition changes. The fourth case occurs whenever a bank reversal needs to be executed, which happens when the heading error χ_e exceeds the dead band value (see Section 3.2.2). The fifth case occurs when the vehicle has reached the terminal distance to its target, in which case the integration can be stopped (see Section 3.1.1). The sixth and last case is the integration of constraint violations for the constraint handling (see Section 4.5.2).

For these cases, one has to determine that during a time step, one or more of these parameters, H, E, M, χ_e, d or the trajectory-constraint variables, has exceeded a so-called event value. One then has to determine at what point in time during this step this happened. This requires root-finding. For illustrative purposes, assume that H has passed the lower border of an atmosphere layer, H_{event}, during the current step, so:

$$(H_n)_0 > H_{event} \quad \text{and} \quad (H_{n+1})_0 \leq H_{event}$$

Note that one could have steps with $(H_n)_0 > H_{event}$ and $(H_{n+1})_0 > H_{event}$, but where H briefly dropped below H_{event} during a step, but in practice, this turned out to give negligible errors. This is due to the fact that skips in the trajectory require high aerodynamic forces, which in turn cause the step size of the integrator to become small, so there is little room for such an event to occur and have a significant impact. The function of which the root has to be found is then:

$$f = H - H_{event} \tag{5.45}$$

This section discusses four different methods that are applied to the six cases. The first applies to H, M and E, as one readily has their Taylor coefficients available at the end of an integration step. The subsections after that treat bank reversals, terminal distance and the constraint handling.

5.5.1 Altitude, Mach Number and Energy

Section 4.1 gave a number of different root-finding methods with Newton's Method being the fastest. Unfortunately, Newton's method can diverge and does not have to stay within a user defined search space. The chosen solution to these problems is to combine this method with the double-false position method (see Section 4.1.3 and Figure 4.2). The function f (for instance, the one given by Eq. (5.45)) is computed at two points in time $t_{0,i}$ and $t_{1,i}$, together with the first derivative at $t_{0,i}$, a quadratic approximation to f can be made:

$$\hat{f}_i(h) = \alpha_i h^2 + \dot{f}_{0,i} h + f_{0,i} \tag{5.46}$$

where i marks the iteration, h is the step size measured from $t_{0,i}$ and α_i is given by:

$$\alpha_i = \frac{f_{1,i} - f_{0,i}}{(t_{1,i} - t_{0,i})^2} - \frac{\dot{f}_i}{t_{1,i} - t_{0,i}} \tag{5.47}$$

One of the roots of $\hat{f}_i$ can be used as approximation for the root of f. Since it is a quadratic line, the roots are given by:

$$t_{0,i+1} = t_{0,i} + \frac{-\dot{f}_{0,i} \pm \sqrt{\dot{f}_{0,i}^2 + 4\alpha_i f_{0,i}}}{2\alpha_i} \tag{5.48}$$

The first derivative $\dot{f}$ can be computed at an arbitrary point within the current step by applying Eq. (5.6) to (5.3), which yields the following Taylor series expansion:

$$\dot{f}_i(h_i) \approx \sum_{k=1}^{K} k(f)_k h_i^{k-1} \tag{5.49}$$

Eq. (5.48) actually gives two roots, depending on whether a plus or minus is used. To always get the root inside the search space, one should use the following procedure:

1. The initial value for $t_{0,i}$ is t_n. The search space is the entire step, so $t_{1,0} = t_n + h_n$.
2. Using Eq. (5.48), a new estimate for the root can be found. Use plus if the value of f on the left of the domain is smaller than 0 and minus if it is larger.
3. The method has converged when $|f(t_{0,i+1})|$ is smaller than some user-set value.
4. The root should stay within the search space, so $f_{1,i+1} = f_{0,i}$ if $f_{0,i}$ and $f_{0,i+1}$ have opposite signs, otherwise $f_{1,i+1} = f_{1,i}$. Go to step 2.

Note that if, initially, the value of f on the left is positive, then for all future steps the left value will be positive and idem in case the left value is negative. In Figure 5.3, left graph, this method is shown for a case where Newton's method diverges. The quadratic method converges slowly for cases such as the one shown in Figure 5.3, right graph. To avoid these back-and-forth movements with little convergence, the maximum jump of this method is limited to half of h_n. Note that this would give convergence in one step in case of the right example in Figure 5.3 as the root lies in the middle of the graph.

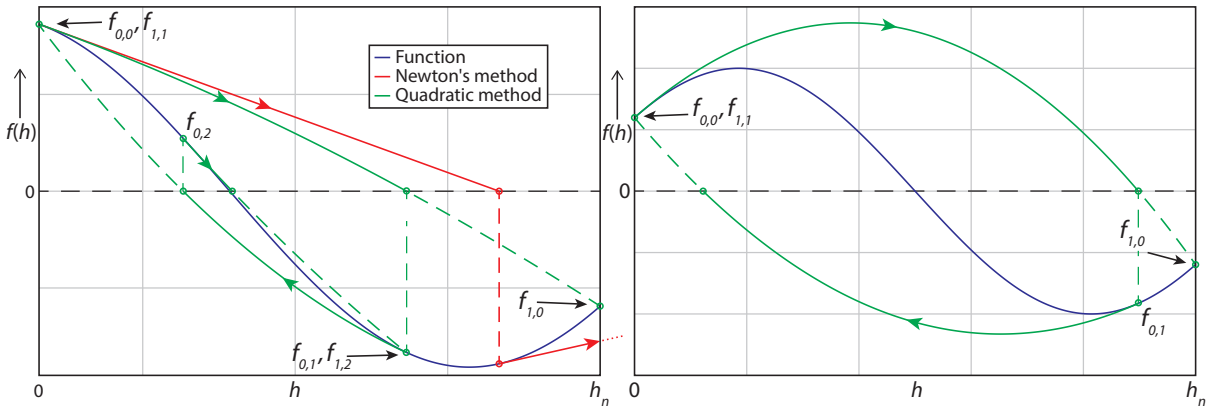


Figure 5.3: Left: Newton's method and the quadratic method applied to the same function. Right: case of slow convergence for the quadratic method.

By using this method in practice, it was found that one should not aim to hit the discontinuity exactly, but rather target a point slightly past the discontinuity. Convergence requires having a value between the discontinuity and the discontinuity plus two times this "safety distance". This way, the variable in question will have passed the discontinuity and the root-finding method is not triggered again the next step. Otherwise, this would cause wasted steps, but more importantly, this behavior could be perpetual and the integrator could get stuck. For H, M and E, safety distances of $1 \cdot 10^{-6}$ m, $M \cdot \epsilon_{rel}$ and $E \cdot \epsilon_{rel}$, respectively, were found to be about the minimal values to at least avoid infinite loops and in most cases avoid wasted steps. Note that especially H causes infinite loops, and did so in a way almost independent of the error tolerance, hence its safety distance is a fixed value.

5.5.2 Bank Reversals

For bank reversals the function f of which the root is to be found is obtained from Eq. (3.9):

$$f = -\text{sign}(\sigma_G)\chi_e - \chi_{db} \tag{5.50}$$

The Taylor coefficients of χ_e and χ_{db} are not needed for integration and would therefore need to be computed whenever a bank reversal takes place. Since the computation of these coefficients comprises of many steps and the fact that the dead band itself is piecewise defined, this would be inefficient. Instead, a root-finding method is used that creates a quadratic line by sampling f at 3 different points: $t_{0,i}$, $t_{1,i}$ and $t_{2,i}$ The line is then given by:

$$\hat{f}_i(h) = \alpha_i h^2 + \beta_i h + f_{0,i} \tag{5.51}$$

with h again being the step size measured from $t_{0,i}$. β_i and α_i are the first and second derivative of the line at $t_{0,i}$, respectively. They can be found from the equations:

$$\hat{f}_i(t_{1,i} - t_{0,i}) = \alpha_i(t_{1,i} - t_{0,i})^2 + \beta_i(t_{1,i} - t_{0,i}) + f_{0,i} = f_{1,i} \tag{5.52}$$

$$\hat{f}_i(t_{2,i} - t_{0,i}) = \alpha_i(t_{2,i} - t_{0,i})^2 + \beta_i(t_{2,i} - t_{0,i}) + f_{0,i} = f_{2,i} \tag{5.53}$$

which yield:

$$\alpha_i = \frac{1}{t_{2,i} - t_{1,i}} \left(\frac{f_{2,i} - f_{0,i}}{t_{2,i} - t_{0,i}} - \frac{f_{1,i} - f_{0,i}}{t_{1,i} - t_{0,i}} \right) \tag{5.54}$$

$$\beta_i = \frac{f_{1,i} - f_{0,i}}{t_{1,i} - t_{0,i}} - \alpha_i(t_{1,i} - t_{0,i}) \tag{5.55}$$

Of $\hat{f}$, only the positive root is needed, as a bank reversal is only needed when f is larger than zero, in which case the left values are always negative and the right are always positive. The positive root is given by:

$$t_{new} = t_{0,i} + \frac{-\beta_i + \sqrt{\beta_i^2 + 4\alpha_i f_{0,i}}}{2\alpha_i} \tag{5.56}$$

To find the value for f for a particular value of h, one first computes the state at $t_{0,i} + h$, using its Taylor coefficients and one then computes f. The root-finding procedure for bank reversals is then as follows:

1. Initially, the value at the start of a time step and at the end are used, so $t_{0,0} = t_n$ and $t_{2,0} = t_n + h_n$. The middle point is found using the double-false position method with the two outer points.
2. Using Eq. (5.56), a new estimate for the root, t_{new}, can be found.
3. The method has converged if $|f(t_{new})| \leq \epsilon_{abs}$.
4. t_{new} will become the new middle point, $t_{1,i+1}$, so if it is larger than $t_{1,i}$, $t_{0,i+1} = t_{1,i}$, else $t_{2,i+1} = t_{1,i}$. Go to step 2.

The sign of σ is flipped when the root-finding is done, which means that Eq. (3.9) is no longer satisfied and the root-finding will not be triggered for this particular bank reversal again, so the integrator cannot end up in an infinite loop. Therefore, the root-finding for bank reversals is allowed to target the root precisely, i.e., there is no safety distance needed.

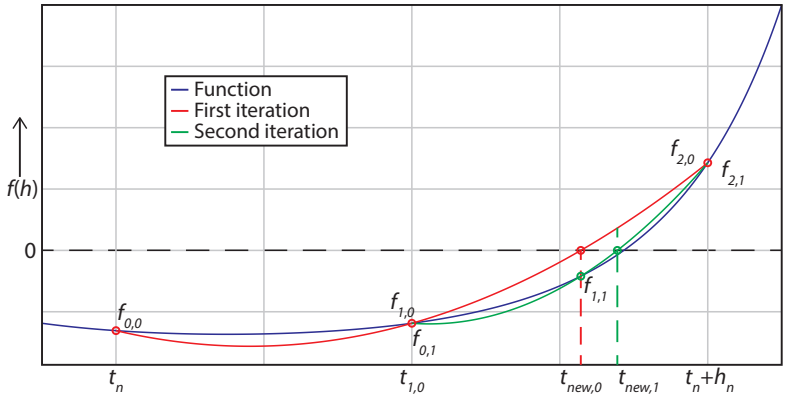


Figure 5.4: Two iterations of the root-finding method for bank reversals.

5.5.3 Terminal Distance

The integration of a trajectory can be stopped when the terminal distance to target is reached, but also when the vehicle misses the target. The used root-finding method should be able to find both cases and should clearly distinguish between them. That is, even during a time step for which the vehicle is no longer heading for the target, the vehicle may have been within terminal distance from the target. The root-finding method of choice is Newton's method. In Figure 5.5, the distance is plotted for two trajectories that pass the target at high speed. The black ring in this figure indicates the terminal distance to the target. Because of the high speed, the lines are almost straight and the chance that the terminal distance ring is overshot during a time step is large. Fortunately, the distance profile for these high speed trajectories is very typical; the distance first decreases in an asymptotic fashion to a minimum and then increases again in an

almost symmetric fashion. In Figure 5.5, one iteration of Newton's method for the trajectory that passes inside the ring stays on the left of the minimum, but for the other trajectory, the method eventually (in this case in one iteration) ends up on the right of the minimum. One can determine whether the method ends up on the left or right by evaluating the sign of the first derivative of the distance. If this is positive, one can stop the integration knowing the terminal distance ring is not reached by this trajectory. The function of which the root is to be found is the distance, given by:

$$d = \arccos\left(\sin\delta\sin\delta_T + \cos\delta\cos\delta_T\cos(\tau - \tau_T)\right) \tag{3.12}$$

Below are the derivations of the first derivative of d for each of the state-variable sets.

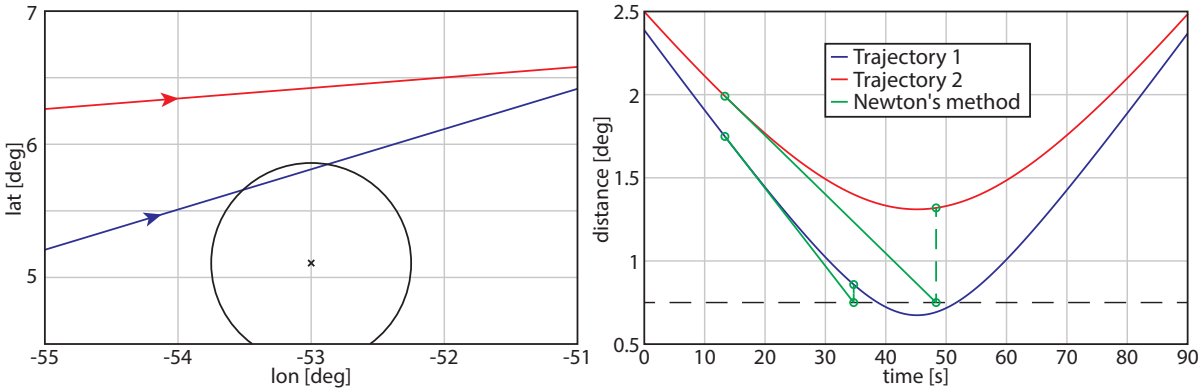


Figure 5.5: Left: longitude and latitude of two high speed trajectories. Right: distance to target of these trajectories.

Cartesian State Variables

Using Eq. (3.12), the cosine of distance for Cartesian components in $\mathcal{F}_I$ can written as:

$$f_0 = \cos d = \frac{1}{r}\left[z_I\sin\delta_T + \left(x_I\cos(\tau_T + \omega_E t) + y_I\sin(\tau_T + \omega_E t)\right)\cos\delta_T\right] \tag{5.57}$$

Note that r must be recomputed each iteration, as the position changes each iteration:

$$r = \sqrt{x_I^2 + y_I^2 + z_I^2} \tag{5.58}$$

The sine of d is computed using the relation $\sin^2 d + \cos^2 d = 1$ (this is allowed as d is always positive):

$$f_1 = \sin d = +\sqrt{1 - f_0^2} \tag{5.59}$$

d is then computed using arcsin:

$$d = \arcsin f_1 \tag{5.60}$$

The first order derivative of r can be computed using basic algebra:

$$\dot{r} = (x_I u_I + y_I v_I + z_I w_I)/r \tag{5.61}$$

Similarly, $\dot{f}_0$ can be found from:

$$\dot{f}_0 = \frac{1}{r}\left(w_I\sin\delta_T + \left(\cos(\tau_T + \omega_E t)(u_I + y_I\omega_E) + \sin(\tau_T + \omega_E t)(v_I + x_I\omega_E)\right)\cos\delta_T - \dot{r}f_0\right) \tag{5.62}$$

and $\dot{d}$ is then computed with:

$$\dot{d} = -\frac{\dot{f}_0}{f_1} \tag{5.63}$$

For the components in $\mathcal{F}_R$, the rotation of the Earth is left out of Eqs. (5.57) and (5.62):

$$f_0 = \cos d = \frac{1}{r}\left(z_R \sin\delta_T + (x_R \cos\tau_T + y_R \sin\tau_T)\cos\delta_T\right) \tag{5.64}$$

$$\dot{f}_0 = \frac{1}{r}\left(w_R \sin\delta_T + \cos\tau_T u_R + \sin\tau_T v_R)\cos\delta_T - \dot{r} f_0\right) \tag{5.65}$$

Spherical State Variables

For spherical state variables, the computation of d consists of:

$$f_0 = \cos d = \sin\delta \sin\delta_T + \cos\delta\cos\delta_T \cos(\tau - \tau_T) \tag{5.66}$$

$$f_1 = \sin d = \sqrt{1 - f_0^2} \tag{5.67}$$

$$d = \arcsin f_1 \tag{5.68}$$

For the derivatives of this variables, first the derivatives of τ and δ are needed:

$$\dot{\tau} = \frac{V_G \sin\chi_G \cos\gamma_G}{r\cos\delta} \tag{2.78}$$

$$\dot{\delta} = \frac{V_G \cos\chi_G \cos\gamma_G}{r} \tag{2.79}$$

$\dot{f}_0$ is in this case:

$$\dot{f}_0 = \dot{\delta}\cos\delta\sin\delta_T + \left(\dot{\tau}\sin(\tau_T - \tau)\cos\delta - \dot{\delta}\sin\delta\cos(\tau_T - \tau)\right)\cos\delta_T \tag{5.69}$$

$\dot{d}$ can then be found from Eq. (5.63).

Spherical Position with Cartesian Velocity

For this state-variable set, the computation is almost the same as for the spherical state variables, but now $\dot{\tau}$ and $\dot{\delta}$ are found from:

$$\dot{\tau} = \frac{v_V}{r\cos\delta} \tag{2.84}$$

$$\dot{\delta} = \frac{u_V}{r} \tag{2.85}$$

5.5.4 Constraint Violations

The constraint variables are not needed for the propagation of the state, so their Taylor coefficients are also not readily available. In this case they should be computed, so, in the next chapter, Section 6.6 is reserved for their recurrence relations. One advantage of the fact that their values do not impact the trajectory is that time steps do not have to be cut in case of a constraint violation. Instead, one has to make sure that the constraints are integrated over the correct interval(s). In the left frame of Figure 5.6, the intervals that have to be integrated are shown in red. Root finding must now be used to determine the edges of these intervals. For cases where an interval is only partially inside the time step, the quadratic method from Section 5.5.1 is used.

For small intervals, it is possible that both ends are inside a single time step. This is shown in the right frame of Figure 5.6, where also the quadratic method from Section 5.5.1 is shown. This line approximates constraint variable g with a line $\hat{f}(h)$ as given by Eq. (5.46). In this case, one is interested in the peak of $\hat{f}(h)$ and its location, given by:

$$\hat{f}_{peak} = f_{0,i} + \frac{1}{2}\dot{f}_{0,i}h_{peak} \tag{5.70}$$

$$h_{peak} = -\frac{\dot{f}_{0,i}}{2\alpha_i} \tag{5.71}$$

If this peak is inside the current time step ($0 < h_{peak} < h_n$) and $\hat{f}_{peak}$ crosses the constraint line, then the quadratic method can be used to obtain both ends of the interval.

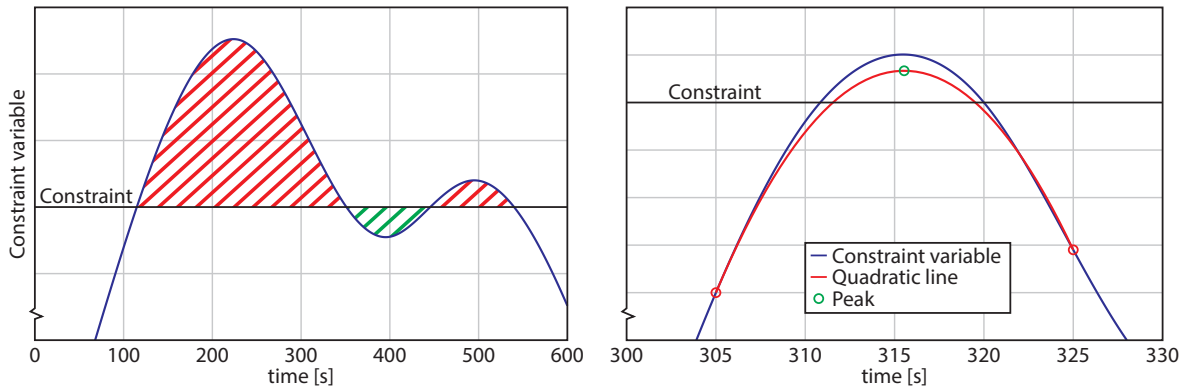


Figure 5.6: Left: arbitrary constraint profile with the violations shown in red. Right: quadratic approximation of a small constraint violation.

For small intervals below the constraint line (i.e., the green interval of the left frame of Figure 5.6), the above methods will suffice, as their impact on the integral value will be minimal if they cannot be found with those methods. However, for very small intervals above the line it may be important to try more precise methods as they can make the difference between no constraint violation and a very small constraint violation if it is the only constraint violation for a particular trajectory. The procedure of this search is then:

1. Evaluate the actual value of g at h_{peak}. If this is larger than the constraint, go to step 2, else go to step 3.
2. Use the quadratic method to find the two roots. Start at h_{peak} for both searches.
3. Use Newton's method to blindly search for a crossing of the constraint line. If this diverges or leaves the current time step, there is likely no peak.
4. If Newton's method moved towards the left and found a root (inside the current time step), this is the right root. If it moved to the right, it found the left root. Next, one can use the quadratic method to find the other root.

When the borders of the intervals inside the current time step are found, one can integrate g. For an interval between two bounds t_0 and t_e, the Taylor series expansion of the integral can be found by applying Eq. (5.6) to (5.3):

$$\int_{t_0}^{t_e} g dt \approx \sum_{k=0}^{K} \frac{(g)_k}{k+1}\left(t_e^{k+1} - t_0^{k+1}\right) \tag{5.72}$$

5.6 Discrete Data

Automatic differentiation only applies to functions that can be represented by mathematical expressions. When, on the other hand, values of variables are given in a table of discrete data, one either has to numerically obtain derivatives of that function (using, for instance, divided differences [34]) or create a mathematical expression for the relation between the data values and the variables they depend on. The second option involves fitting the data with a curve of which the mathematical expression is known. This expression then contains all the information on the derivatives. Curve fitting is here chosen as the preferred option, since it allows one to obtain multiple derivatives at once, whereas numerical differentiation with other methods can only obtain one derivative at the time and requires different schemes for different-order derivatives [34].

Since the tables of US76 and the aerodynamics of HORUS are known in advance, they can be fitted before the integration. In the following two subsections the used fitting processes for these data sets are explained and their outcome is shown. Two different techniques for obtaining a expressions for the wind velocity components from the wind model are then given in Section 5.6.3.

5.6.1 US76 Tables

For US76, Table A.2 is used to determine the values of M_M, N and ρ at altitudes above 86 km. The original tables went up to 1000 km altitude, but here, they were limited to 125 km, slightly above initial altitude of HORUS (122 km). One does not have to fit both N and ρ, since N itself is not required for the equations of motion; in fact, its only purpose is to compute ρ through Eqs. (2.57) and (2.58). In this case, the table of ρ will be fitted rather than N, as this gives a more direct computation of ρ and it was found through trial and error that N requires an equal number of terms in a regression as ρ. The goal of the fitting process was to find a function that approximates a data set without "overfitting" it and with the minimal number of terms. To define overfitting, ϵ_i is defined as the relative error at each data point:

$$\epsilon_i = \left| \frac{x_i - \bar{x}_i}{\bar{x}_i} \right| \tag{5.73}$$

where x_i is the approximated value at data point i and $\bar{x}_i$ is its actual value. The data in Table A.2 are only given up to four digits, so any approximation with a maximum relative error smaller than $5 \cdot 10^{-4}$ can be considered to overfit.

The used fitting functions are not all polynomials, which means that least-squares regression cannot be used for all functions. For instance, one cannot use least squares to determine coefficient α inside the sine function $\sin(\alpha x)$ or the exponential function $e^{\alpha x}$. Therefore, a numerical optimization tool, in this case DE (which is verified in Section 7.3.1), is used. Another reason for using a numerical optimization tool was that, instead of minimizing the least-squares condition (Eq. (4.34)), one can directly minimize the maximum value of the relative errors ϵ_i.

Molecular Mass

For M_M, polynomials up to degree 6 have been tried as fit. Furthermore, the shape of the data, when plotted, resembles a sine-like function (see Figure 5.7(a)). Thus, also two approximations using sine and cosine pairs were tested. The results are given in Table 5.2. In this Table, ϵ_{min},

ϵ_{avg} and ϵ_{max} are the minimum, average and maximum relative errors. The degree 5 and 6 polynomial and the approximation with two sine pairs have errors lower than the threshold of $5 \cdot 10^{-4}$. The degree 5 polynomial and the sine approximation would both require 4 auxiliary variables for their recurrence relations and are thus in practical applications equally as hard to evaluate. As the polynomial is not preferred over the sine approximation, the latter is chosen because of its lower maximum error. This approximation is given by:

$$M_M = a_0 + a_1 \sin(b_0 H) + a_2 \cos(b_0 H) + a_3 \sin(b_1 H) + a_4 \cos(b_1 H) \tag{5.74}$$

with the coefficients a_0 to a_4, b_0 and b_1 given up to 12 digits in Table 5.3.

Table 5.2: Results of various fits of M_M.

	# of terms	ϵ_{min}	ϵ_{avg}	ϵ_{max}
degree 3 polynomial	4	$2.700 \cdot 10^{-5}$	$7.456 \cdot 10^{-4}$	$1.234 \cdot 10^{-3}$
degree 4 polynomial	5	$1.394 \cdot 10^{-5}$	$5.943 \cdot 10^{-4}$	$9.888 \cdot 10^{-4}$
degree 5 polynomial	6	$2.245 \cdot 10^{-6}$	$2.011 \cdot 10^{-4}$	$3.854 \cdot 10^{-4}$
degree 6 polynomial	7	$3.481 \cdot 10^{-6}$	$1.185 \cdot 10^{-4}$	$2.191 \cdot 10^{-4}$
1 sine-cosine pair	3	$1.975 \cdot 10^{-5}$	$5.283 \cdot 10^{-4}$	$8.926 \cdot 10^{-4}$
2 sine-cosine pairs	5	$9.344 \cdot 10^{-6}$	$1.204 \cdot 10^{-4}$	$2.469 \cdot 10^{-4}$

Table 5.3: Coefficients of the approximation to M_M in Eq. (5.74).

Coefficient	Value
a_0	$2.72170411216 \cdot 10^{-2}$
a_1	$-7.21228033536 \cdot 10^{-4}$
a_2	$1.59853767070 \cdot 10^{-3}$
a_3	$-9.69576469474 \cdot 10^{-6}$
a_4	$-3.14919572872 \cdot 10^{-5}$
b_0	$6.71606290092 \cdot 10^{-5}$
b_1	$2.81691343060 \cdot 10^{-4}$

Air Density

As can be seen in Figure 5.7(b), the air density closely resembles an exponential function. It turns out that a large number of terms are required to fit these data and meet the requirement of $\epsilon_{max} \leq 5 \cdot 10^{-4}$. Therefore it is chosen to fit a separate function through the data of each atmosphere layer. Because of the exponential nature of the data, polynomials make for poor fitting functions, requiring a large number of terms. Instead, the exponential function of a polynomial is used to fit each layer:

$$\rho = \exp\left(\sum_{i=0}^{p} a_i H^i\right) \tag{5.75}$$

with p being the degree of the polynomial. In Table 5.4, the results are shown of the fitting process. For each atmosphere layer, the degree of the polynomial is increased until $\epsilon_{max} \leq 5 \cdot 10^{-4}$. The exception is layer 8, for which there is very little improvement from degree 5 to 6, so the degree 5 fit was selected as a limit. This mainly because higher degree fits would greatly increase the complexity of the problem without giving much improvement in quality. The coefficients of these fits are given in Table 5.5. The higher degree coefficients get very small

as they are multiplied with power of H, which is a large value itself. For instance, coefficient a_5 of layer 8 is of the order $1 \cdot 10^{-21}$ and is multiplied with $H^5 \approx \left(1 \cdot 10^5\right)^5 = 1 \cdot 10^{25}$. The outcome of the multiplication is then of the order $1 \cdot 10^4$, which is larger than coefficient a_0.

Table 5.4: Results of the fits of ρ.

Layer	Degree	ϵ_{min}	ϵ_{avg}	ϵ_{max}
7	1	$5.532 \cdot 10^{-6}$	$3.133 \cdot 10^{-4}$	$5.217 \cdot 10^{-4}$
	2	$1.037 \cdot 10^{-4}$	$1.865 \cdot 10^{-4}$	$2.617 \cdot 10^{-4}$
8	1	$1.528 \cdot 10^{-4}$	$8.565 \cdot 10^{-3}$	$1.349 \cdot 10^{-2}$
	2	$1.553 \cdot 10^{-4}$	$3.142 \cdot 10^{-3}$	$5.252 \cdot 10^{-3}$
	3	$3.490 \cdot 10^{-4}$	$2.761 \cdot 10^{-3}$	$4.381 \cdot 10^{-3}$
	4	$9.585 \cdot 10^{-5}$	$8.313 \cdot 10^{-4}$	$1.365 \cdot 10^{-3}$
	5	$1.629 \cdot 10^{-5}$	$7.892 \cdot 10^{-4}$	$1.177 \cdot 10^{-3}$
	6	$1.770 \cdot 10^{-5}$	$7.082 \cdot 10^{-4}$	$1.036 \cdot 10^{-3}$
9	1	$6.466 \cdot 10^{-3}$	$2.643 \cdot 10^{-2}$	$3.979 \cdot 10^{-2}$
	2	$3.427 \cdot 10^{-4}$	$1.943 \cdot 10^{-3}$	$2.731 \cdot 10^{-3}$
	3	$4.557 \cdot 10^{-5}$	$1.493 \cdot 10^{-4}$	$2.306 \cdot 10^{-4}$
10	1	$1.709 \cdot 10^{-3}$	$4.644 \cdot 10^{-3}$	$6.014 \cdot 10^{-3}$
	2	$7.933 \cdot 10^{-5}$	$1.288 \cdot 10^{-4}$	$1.527 \cdot 10^{-4}$

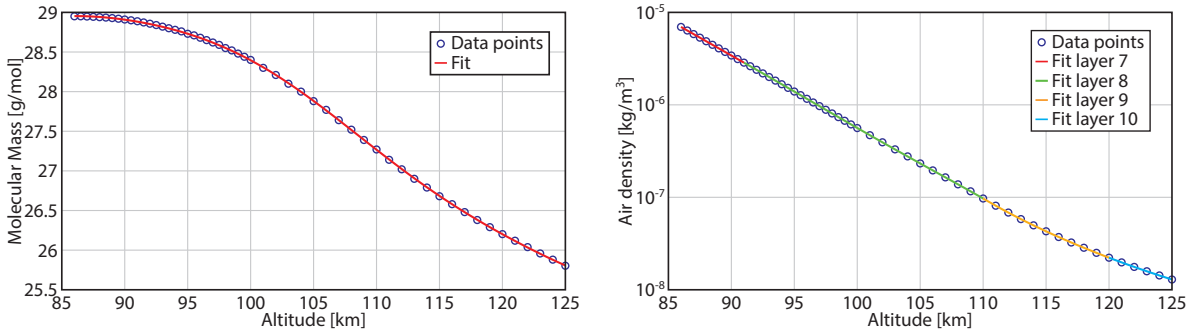
Table 5.5: Coefficients of the fits of ρ.

Layer	Coefficient	Value	Layer	Coefficient	Value
7	a_0	3.38158808697	9	a_0	174.879747121
	a_1	$-1.77053955227 \cdot 10^{-4}$		a_1	$-4.34782962139 \cdot 10^{-3}$
	a_2	$-4.15708944670 \cdot 10^{-12}$		a_2	$3.33590010087 \cdot 10^{-8}$
8	a_0	615.881175165		a_3	$-8.74608279665 \cdot 10^{-14}$
	a_1	$-3.55713113525 \cdot 10^{-2}$	10	a_0	25.3446337016
	a_2	$8.02354718879 \cdot 10^{-7}$		a_1	$-5.97599753596 \cdot 10^{-4}$
	a_3	$-8.95211959667 \cdot 10^{-12}$		a_2	$1.99617437201 \cdot 10^{-9}$
	a_4	$4.92872542676 \cdot 10^{-17}$			
	a_5	$-1.07329560520 \cdot 10^{-21}$			

5.6.2 Aerodynamics

In Section 3.1.2, the aerodynamics of HORUS were trimmed for Mach number from 2.5 to 20, yielding Tables B.1 and B.2. The fitting of these tables started with limiting the amount of data to be fitted. The first limitation is based on the assumption that, during a reentry, the angle of attack would remain large at all times, so the values for α_A up to 10° were not taken into account during the fitting process. Also, as was mentioned in Section 3.2.1, the commanded α_G is limited to 40°, so 45° was also not taken into account. For the remaining range, it was attempted to create fits with an error of about 0.005 or less, as the original data in [47] was given up to two decimal points. Furthermore, these fits should preferably have as few terms as possible.

The fitting process started with linear regression, using terms that are linear combinations of α_A and M, such as $\alpha_A^2 M$ and α_A / M^2. For C_D, this proved sufficient to obtain a fit with an



(a) Molecular mass (b) Air density

Figure 5.7: Plots of US76 atmosphere variables and their fits.

error of 0.0055 or less with a total of 10 terms. Adding more terms did not improve the fit significantly, so this fit was used. It is given by:

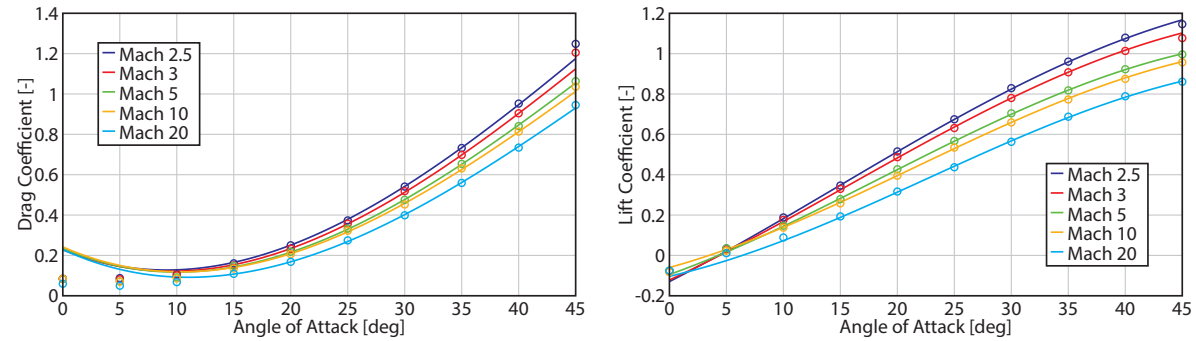
$$C_{D,trim} = a_{D,0} + a_{D,1}\alpha_A + a_{D,2}\alpha_A^2 + a_{D,3}\alpha_A^3 + a_{D,4}M + a_{D,5}M^2 + a_{D,6}\alpha_A M + a_{D,7}\frac{\alpha_A}{M} + a_{D,8}\frac{\alpha_A^2}{M} + a_{D,9}\frac{\alpha_A}{M^2} \tag{5.76}$$

with the coefficients $a_{D,i}$ given in Table 5.6.

For C_L, it proved more difficult to use linear regression, as even fits of 13 terms still gave errors of up to 0.0071. The used fit was found through a more complex process. First a sine-cosine pair with the same frequency was fit through the data of each value of M individually, yielding 5 lines as a function of α_A. Next, terms as function of M were added to the fit to combine these lines, creating the following fit:

$$C_{L,trim} = a_{L,0} + a_{L,1}M + \frac{a_{L,2}}{M + a_{L,3}} + \left(a_{L,4} + \frac{a_{L,5}}{M + a_{L,6}}\right)\sin\left(a_{L,7}\alpha_A + \frac{a_{L,8}\alpha_A}{M + a_{L,9}}\right) + \left(a_{L,10} + a_{L,11}M + \frac{a_{L,12}}{M + a_{L,13}}\right)\cos\left(a_{L,7}\alpha_A + \frac{a_{L,8}\alpha_A}{M + a_{L,9}}\right) \tag{5.77}$$

The last step was to optimize the coefficients $a_{L,i}$ to minimize the errors. The result was a maximum error of 0.0049. The coefficients $a_{D,i}$ and $a_{L,i}$ are given in Table 5.6. The fits of $C_{D,trim}$ and $C_{L,trim}$ are plotted in Figure 5.8.



(a) Drag coefficient (b) Lift coefficient

Figure 5.8: Fits of the trimmed C_D and C_L, with the original values indicated by circles.

Table 5.6: Coefficients of the fits of $C_{D,trim}$ and $C_{L,trim}$.

Coefficient	Value	Coefficient	Value
$a_{D,0}$	$2.18867646586 \cdot 10^{-1}$	$a_{L,0}$	5.41566170498
$a_{D,1}$	-1.27243888392	$a_{L,1}$	$-6.37265930291 \cdot 10^{-2}$
$a_{D,2}$	4.34450639795	$a_{L,2}$	-279.256982092
$a_{D,3}$	-1.69506701030	$a_{L,3}$	54.2977015306
$a_{D,4}$	$4.17717351727 \cdot 10^{-3}$	$a_{L,4}$	$2.29365596970 \cdot 10^{-1}$
$a_{D,5}$	$-1.83391118821 \cdot 10^{-4}$	$a_{L,5}$	1.18289308803
$a_{D,6}$	$-9.31478339288 \cdot 10^{-3}$	$a_{L,6}$	$4.99501929783 \cdot 10^{-1}$
$a_{D,7}$	$-7.54655599521 \cdot 10^{-1}$	$a_{L,7}$	2.67978011313
$a_{D,8}$	$-4.78539125537 \cdot 10^{-1}$	$a_{L,8}$	-1.39678496361
$a_{D,9}$	1.79202288046	$a_{L,9}$	$-3.84865292405 \cdot 10^{-1}$
		$a_{L,10}$	$-9.03275470332 \cdot 10^{-1}$
		$a_{L,11}$	$8.01036806574 \cdot 10^{-3}$
		$a_{L,12}$	11.8529069169
		$a_{L,13}$	26.1311915127

5.6.3 Wind Model

As will be explained in Section 7.2.3, the used wind model will give the wind velocities as a spline function of altitude. This means that the spline coefficients are readily available to compute the Taylor coefficients of the wind velocities. A downside of using the spline coefficients is that each time one moves from one spline interval to the next, one has to cut the step size, since another discontinuity is reached. Say, for instance, that the wind model uses a spline through 100 points, then also 100 steps will have to be reduced in size. This will reduce the efficiency of TSI, as one of its perks, large step sizes, is diminished. Also, 100 extra cases of root-finding will cause for extra computational time.

The opted solution for this is to sample the wind model at a number of points over a predicted altitude range, H_{pred}, for the current step and then fit a polynomial through these points using least-squares regression, as is shown in Figure 5.9. For the distribution of the sample points over the altitude range, a Chebychev-Gauss grid will be used. In Section 4.2.2, this grid was given to minimize the interpolation error for polynomials, but it was found to also work for least-squares regression.

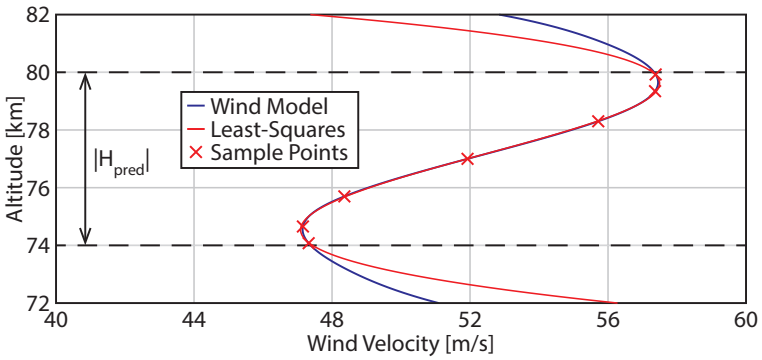


Figure 5.9: Least-squares regression of the wind model using a degree 4 polynomial with 7 sample points on a Chebychev-Gauss grid.

H_{pred} has to be determined before the Taylor coefficients of the current step are computed,

so no information of the current time step can be used. One could use the Taylor coefficients of altitude of the previous step, but it was found through trial and error that this gives poor altitude-range predictions. Instead, the trajectory without wind will be used to obtain the altitude-range prediction, as it was found that wind does not have a large influence on the resulting trajectory. This is mainly because the wind velocity is much smaller than that of the vehicle throughout the hypersonic flight.

In Figure 5.10, the timelines of the trajectories with and without wind are shown. For time step n, one needs an approximation of the step size, $\hat{h}_n$. With $\hat{h}_n$, the altitude of the trajectory without wind can be determined at $t_n + \hat{h}_n$. This altitude, $\hat{H}_{n+1}$ (which is computed with the Taylor coefficients of the original trajectory, so those will have to be stored by the user), is then used to compute the altitude-range prediction:

$$H_{pred} = \hat{H}_{n+1} - H_n \tag{5.78}$$

Three possible values for $\hat{h}_n$ are h_{n-1} of the trajectory with wind, and h_m and h_{m+1} of the trajectory without wind. These values are chosen as they are the closest available time steps to $\hat{h}_n$ on the time lines of Figure 5.10.

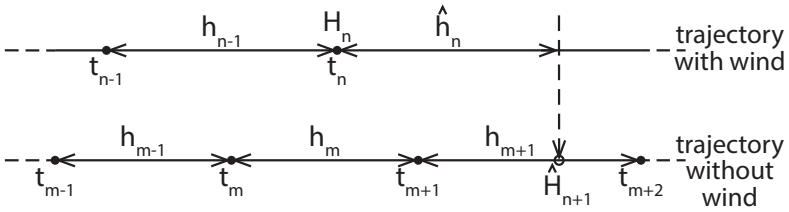


Figure 5.10: Determination of the predicted altitude range.

Note that the integration steps of the trajectory with wind will be smaller than without, so the integration without wind is done with $\epsilon = 1 \cdot 10^{-14}$ to have similar small steps sizes. H_{pred} is then computed with all three approximations and the result with the largest magnitude is used. Note that for a given degree polynomial used for the least-squares regression, there will be a limit to the altitude range that can be accurately approximated, so there is an upper limit to H_{pred}, namely H_{max}. Also, H_{pred} may in some cases be very small, and could then be smaller than the actual altitude covered by the current time step. To avoid this, there will also be a user set lower limit H_{min}. Finally, the values of H_{pred} determined by this method may be persistently too small or too large, so after computation, H_{pred} is multiplied by a user set parameter η. The optimal values for H_{max}, H_{min} and η will have to be determined through numerical optimization, together with the optimal number of sample points, s, and the degree, p, of the fitting polynomial.

Chapter 6

Reentry Recurrence Relations

Now that the mechanics of reentry are given and TSI is explained, the major preparation step of TSI can be performed. This step involves the setup of the recurrence relations for the equations of motion. The conversion to recurrence relations starts with the division of the equations into auxiliary variables such that the recurrence relation for those variables contain no nested summations. In this chapter, the indices n indicating which time step a variable belongs to are omitted for improved readability, as the recurrence relations require only variables that are defined at the same point in time as the state with which they are computed.

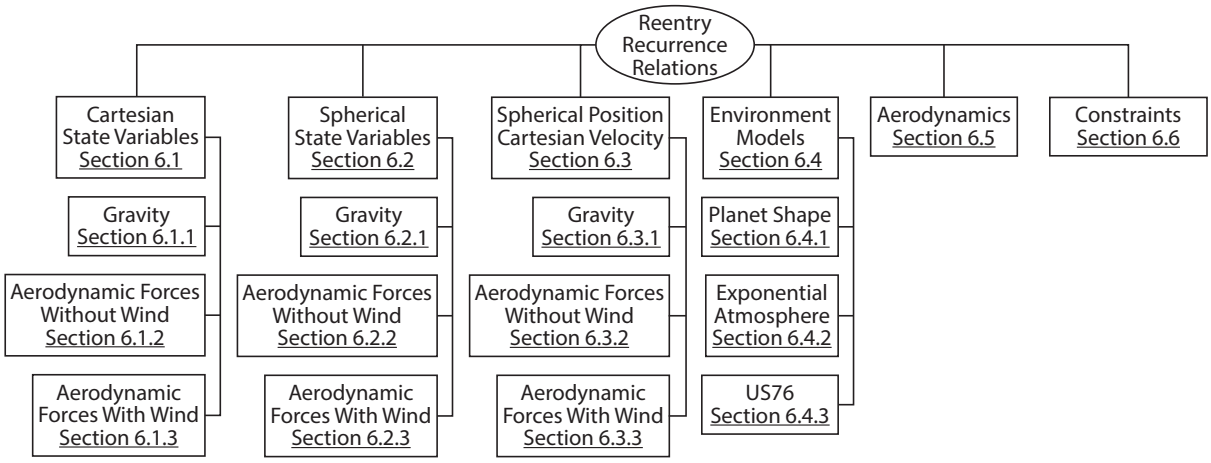


Figure 6.1: Road map for Chapter 6.

This chapter is split into six sections; the first three cover the equations of motion for each state-variable set, Section 6.4 then covers the equations for the environment models, Section 6.5 the aerodynamics and Section 6.6 the constraints.

6.1 Cartesian State Variables

This section covers the conversion of the equations of motion for the Cartesian state-variable sets and is split into the relations for the state variables themselves, gravity and aerodynamic forces, first without the influence of wind, then with wind included. The sections for the other state variables will have the same layout. This section will have an extra part discussing the changes that have to be made to the equations when Cartesian state variables in $\mathcal{F}_R$ are used.

The state variables consist of the Cartesian position and velocity vector elements in $\mathcal{F}_I$. Since these variables appear frequently in the equations, the index I will be omitted. The differential

equations of these variables are Eqs. (2.68) and (2.69):

$$\frac{d}{dt}\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} u \\ v \\ w \end{pmatrix} \tag{6.1}$$

$$\frac{d}{dt}\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \boldsymbol{g}_I + \frac{1}{m}\boldsymbol{F}_{A,I} \tag{6.2}$$

If the components of the gravity acceleration are defined as g_x, g_y and g_z and the components of the aerodynamic accelerations in $\mathcal{F}_I$ as f_x, f_y and f_z, the recurrence relations for the state variables are:

$$\begin{pmatrix} (x)_k \\ (y)_k \\ (z)_k \end{pmatrix} = \frac{1}{k}\begin{pmatrix} (u)_{k-1} \\ (v)_{k-1} \\ (w)_{k-1} \end{pmatrix} \tag{6.3}$$

$$\begin{pmatrix} (u)_k \\ (v)_k \\ (w)_k \end{pmatrix} = \frac{1}{k}\begin{pmatrix} (g_x)_{k-1} + (f_x)_{k-1} \\ (g_y)_{k-1} + (f_y)_{k-1} \\ (g_z)_{k-1} + (f_z)_{k-1} \end{pmatrix} \tag{6.4}$$

6.1.1 Gravity

The gravity acceleration vector for the J_2-model is given by Eq. (2.40), which is repeated here:

$$\boldsymbol{g}_I = -\frac{\mu_E}{r^3}\left\{1 + \frac{3}{2}J_2\left(\frac{R_E}{r}\right)^2\left[\begin{pmatrix} 1 \\ 1 \\ 3 \end{pmatrix} - 5\frac{z^2}{r^2}\right]\right\}\begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{6.5}$$

Now, a number of auxiliary variables will be defined, such that each contains no more than one multiplication and/or division or other mathematical operation. Starting with r^2:

$$f_0 = r^2 = x^2 + y^2 + z^2 \tag{6.6}$$

Although one could compute r^3 directly from f_0, it is calculated using r, since r is needed later on (for instance, for the altitude computation):

$$r = \sqrt{f_0} \tag{6.7}$$

$$f_1 = r^3 = f_0 r \tag{6.8}$$

The other variables required to compute the gravity accelerations are:

$$f_2 = \frac{z^2}{f_0} \tag{6.9}$$

$$f_3 = -\mu_E\left(1 + \frac{3}{2}J_2 R_E^2\frac{1 - 5f_2}{f_0}\right) \tag{6.10}$$

Note that the gravity acceleration in z-direction has a 3 instead of the 1 in the numerator:

$$f_3^* = -\mu_E\left(1 + \frac{3}{2}J_2 R_E^2\frac{3 - 5f_2}{f_0}\right) \tag{6.11}$$

Filling Eqs. (6.6) to (6.11) into (6.5) yields:

$$\begin{pmatrix} g_x \\ g_y \\ g_z \end{pmatrix} = \frac{1}{f_1} \begin{pmatrix} f_3 x \\ f_3 y \\ f_3^* z \end{pmatrix} \tag{6.12}$$

Next, the Taylor coefficients for $k \geq 1$ can be determined. The recurrence relation for f_0 can be found from Eq. (5.9), noting that one can combine summations of multiplications (note: this only works for multiplications, for other types of recurrence relations, one would create nested summations or very complex expressions):

$$(f_0)_k = \sum_{j=0}^{k} \left[(x)_j (x)_{k-j} + (y)_j (y)_{k-j} + (z)_j (z)_{k-j} \right] \tag{6.13}$$

The recurrence relation for a square root can be found from Eq. (5.11), with $\alpha = 1/2$:

$$(r)_k = \frac{1}{(f_0)_0} \sum_{j=1}^{k} \left[\left(\frac{3j}{2k} - 1 \right) (f_0)_j (r)_{k-j} \right] \tag{6.14}$$

f_1 consists of a single multiplication, resulting in:

$$(f_1)_k = \sum_{j=0}^{k} (f_0)_j (r)_{k-j} \tag{6.15}$$

The combination of a multiplication and division was given by Eq. (5.16), which yields:

$$(f_2)_k = \frac{1}{(f_0)_0} \left[\sum_{j=0}^{k} (z)_j (z)_{k-j} - \sum_{j=1}^{k} (f_0)_j (f_2)_{k-j} \right] \tag{6.16}$$

To obtain the recurrence relation for f_3, whose equation is packed with constants, Eq. (6.10) is first written as:

$$f_3 = -\mu_E \frac{f_0 + \frac{3}{2} J_2 R_E^2 \left(1 - 5 f_2\right)}{f_0} \tag{6.17}$$

The numerator now contains two terms with variables and one with constants, i.e., $-\mu_E \frac{3}{2} J_2 R_E^2$. The constant term has no derivatives, so it does not appear the recurrence relation, whereas the two with the variables do. The recurrence relation for f_3 is:

$$(f_3)_k = \frac{1}{(f_0)_0} \left[-\mu_E (f_0)_k + \mu_E \frac{15}{2} J_2 R_E^2 (f_2)_k - \sum_{j=1}^{k} (f_0)_j (f_3)_{k-j} \right] \tag{6.18}$$

For the Taylor coefficients of f_3^*, one should replace $(f_3)_i$ with $(f_3^*)_i$ in this equation. Note that they do not have the same derivative values, despite having the same recurrence relations. The Taylor coefficients of the gravity accelerations are then:

$$(g_x)_k = \frac{1}{(f_1)_0} \left[\sum_{j=0}^{k} (f_3)_j (x)_{k-j} - \sum_{j=1}^{k} (f_1)_j (g_x)_{k-j} \right] \tag{6.19}$$

$$(g_y)_k = \frac{1}{(f_1)_0} \left[\sum_{j=0}^{k} (f_3)_j (y)_{k-j} - \sum_{j=1}^{k} (f_1)_j (g_y)_{k-j} \right] \tag{6.20}$$

For $(g_z)_k$, one should use f_3^* instead of f_3:

$$(g_z)_k = \frac{1}{(f_1)_0}\left[\sum_{j=1}^{k}(f_3^*)_j(z)_{k-j} - \sum_{j=1}^{k}(f_1)_j(g_z)_{k-j}\right] \tag{6.21}$$

6.1.2 Aerodynamic Forces without Wind

Next, the aerodynamic force equations are rewritten. The goal of this subsection is to give the recurrence relations that are needed to obtain the aerodynamic accelerations f_x, f_y and f_z in $\mathcal{F}_I$. For the Cartesian state variables two methods were presented to obtain transformation matrix $\boldsymbol{C}_{I,TA}$ (see Section 2.2), which is used to transform the aerodynamic accelerations to $\mathcal{F}_I$. The first method consists of matrix multiplications and the second is based on vector calculus. In Section 2.7, it was explained that the first method is better when wind is included, however, it was not determined which is better for the equations without wind. The basic recurrence relations in Section 5.2 show that each of the arithmetic operations, other than addition and subtraction, require roughly the same computational time, as they all consist of a summation of multiplications of Taylor coefficients of two variables. Thus the computational time for each set of state variables can be roughly determined by the number of unique arithmetic operations excluding addition and subtraction. With this reasoning, it can be found that the method involving vector calculus is better when wind is not included. This means that this method is worked out in this section, while the method with matrices is worked out in Section 6.1.3. The counting of the number of relevant operations (not including addition and subtraction) for both methods is explained in the following two paragraphs.

Computing $\boldsymbol{C}_{I,TA}$ with transformation matrices starts with the computation of the sines and cosines of $\tilde{\tau}$ and δ, which requires 7 relevant operations, as can be seen in Eqs. (6.66) to (6.68). Next up is the transformation of $\boldsymbol{V}_A$ to $\mathcal{F}_V$, which consists of the multiplication with $\boldsymbol{C}_3(-\tilde{\tau})$ and $\boldsymbol{C}_2(\delta + \frac{\pi}{2})$. A multiplication with a transformation matrix, for instance $\boldsymbol{C}_3(-\tilde{\tau})$:

$$\boldsymbol{C}_3(-\tilde{\tau}) = \begin{bmatrix} \cos\tilde{\tau} & -\sin\tilde{\tau} & 0 \\ \sin\tilde{\tau} & \cos\tilde{\tau} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

requires 4 relevant operations, namely the multiplications with the sines and cosines (multiplying with 1 is not counted). Thus, the multiplication with $\boldsymbol{C}_3(-\tilde{\tau})$ and $\boldsymbol{C}_2(\delta + \frac{\pi}{2})$ requires in 8 relevant operations. Next up is the calculation of the sines and cosines of γ_A and χ_A, consisting of 11 relevant operations, as can be seen in Eqs. (6.73) to (6.77). Finally the transformation of $\boldsymbol{F}_A$ to $\mathcal{F}_I$ consists of five matrix multiplications, thus needing another 20 operations. The total number of operations then becomes 46.

The vector method starts with 4 relevant operations to obtain V_A (see Eq. 6.24), followed by 3 for the components of $\boldsymbol{X}_{TA,I}$, through Eq. (6.25). $\boldsymbol{Y}_{TA,I}$, using Eqs. (6.26) to (6.30) requires a total of 13 relevant operations. $\boldsymbol{Z}_{TA,I}$, using (6.31), needs 6 operations. The transformation of $\boldsymbol{F}_A$ to $\mathcal{F}_I$ then consists of Eqs. (6.32) to (6.34), which add up to 13 operations. The total is then 39 operations, making the vector method the better of the two when wind is excluded.

By combining Eqs. (2.61) and (2.15), the aerodynamic accelerations in $\mathcal{F}_I$ are given by:

$$\frac{1}{m}\boldsymbol{F}_{A,I} = \begin{pmatrix} f_x \\ f_y \\ f_z \end{pmatrix} = \begin{bmatrix} \boldsymbol{X}_T & \boldsymbol{Y}_T & \boldsymbol{Z}_T \end{bmatrix} \begin{pmatrix} -C_D \\ -C_S\cos\sigma_A - C_L\sin\sigma_A \\ C_S\sin\sigma_A - C_L\cos\sigma_A \end{pmatrix} \frac{\rho V_A^2 S_{ref}}{2m} \tag{6.22}$$

Note that the relationship between the aerodynamic coefficients, and the aerodynamic angles and Mach number are given in Section 6.5. The computation of the aerodynamic accelerations starts with computing the groundspeed components in $\mathcal{F}_I$ (Eq. (2.22)):

$$\begin{pmatrix} u_G \\ v_G \\ w_G \end{pmatrix} = \begin{pmatrix} u + \omega_E y \\ v - \omega_E x \\ w \end{pmatrix} \tag{6.23}$$

In the absence of wind, the components of the airspeed and groundspeed are the same and (6.23) can directly be used to compute the airspeed components. To rewrite Eq. (6.22), the following auxiliary variables are defined. For $\boldsymbol{X}_T$ (using Eq. (2.12)), they are:

$$f_4 = u_G^2 + v_G^2 + w_G^2 \qquad V_G = \sqrt{f_4} \tag{6.24}$$

$$\begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} = \begin{pmatrix} u_G/V_G \\ v_G/V_G \\ w_G/V_G \end{pmatrix} = \boldsymbol{X}_T \tag{6.25}$$

For $\boldsymbol{Y}_T$ (using Eq. (2.13)), they are:

$$f_5 = v_G z - w_G y \tag{6.26}$$

$$f_6 = w_G x - u_G z \tag{6.27}$$

$$f_7 = u_G y - v_G x \tag{6.28}$$

$$f_8 = f_5^2 + f_6^2 + f_7^2 \qquad f_9 = \sqrt{f_8} \tag{6.29}$$

$$\begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \end{pmatrix} = \begin{pmatrix} f_5/f_9 \\ f_6/f_9 \\ f_7/f_9 \end{pmatrix} = \boldsymbol{Y}_T \tag{6.30}$$

And for $\boldsymbol{Z}_T$ (using Eq. (2.14)), they are:

$$\begin{pmatrix} Z_1 \\ Z_2 \\ Z_3 \end{pmatrix} = \begin{pmatrix} X_2 Y_3 - X_3 Y_2 \\ X_3 Y_1 - X_1 Y_3 \\ X_1 Y_2 - X_2 Y_1 \end{pmatrix} = \boldsymbol{Z}_T \tag{6.31}$$

The fourth group of variables are:

$$C_S^* = C_S \mathrm{c}\sigma_A + C_L \mathrm{s}\sigma_A \tag{6.32}$$

$$C_L^* = -C_S \mathrm{s}\sigma_A + C_L \mathrm{c}\sigma_A \tag{6.33}$$

For the remainder of this chapter, s and c will be the shorthand notation for the sine and cosine of an angle, so $\mathrm{s}\sigma_A$ and $\mathrm{c}\sigma_A$ are the sine and cosine of σ_A, respectively. Note that sines and cosines themselves are auxiliary variables with their own recursion relations, as can be seen in Eqs. (6.54) and (6.56). The aerodynamic force coefficients in $\mathcal{F}_I$, here defined as C_x, C_y and C_z, can be determined with:

$$\begin{pmatrix} C_x \\ C_y \\ C_z \end{pmatrix} = \begin{pmatrix} C_D X_1 + C_S^* Y_1 + C_L^* Z_1 \\ C_D X_2 + C_S^* Y_2 + C_L^* Z_2 \\ C_D X_3 + C_S^* Y_3 + C_L^* Z_3 \end{pmatrix} \tag{6.34}$$

and the final variable f_{10} is given by:

$$f_{10} = -\frac{\rho f_4 S_{ref}}{2m} \tag{6.35}$$

Combining the auxiliary variables with Eq. (6.21) yields the aerodynamic accelerations:

$$\begin{pmatrix} f_x \\ f_y \\ f_z \end{pmatrix} = \begin{pmatrix} C_x f_{10} \\ C_y f_{10} \\ C_z f_{10} \end{pmatrix} \tag{6.36}$$

The Taylor coefficients, in order of dependency, are then as follows ($k \geq 1$):

$$\begin{pmatrix} (u_G)_k \\ (v_G)_k \\ (w_G)_k \end{pmatrix} = \frac{1}{k} \begin{pmatrix} (u)_k + \omega_E(y)_k \\ (v)_k - \omega_E(x)_k \\ (w)_k \end{pmatrix} \tag{6.37}$$

$$(f_4)_k = \sum_{j=0}^{k} \left[(u_G)_j (u_G)_{k-j} + (v_G)_j (v_G)_{k-j} + (w_G)_j (w_G)_{k-j} \right] \tag{6.38}$$

$$(V_G)_k = \frac{1}{(f_4)_0} \sum_{j=1}^{k} \left[\left(\frac{3j}{2k} - 1 \right) (f_4)_j (V_G)_{k-j} \right] \tag{6.39}$$

$$(X_1)_k = \frac{1}{(V_G)_0} \left[(u_G)_k - \sum_{j=1}^{k} (V_G)_j (X_1)_{k-j} \right] \tag{6.40}$$

$$(X_2)_k = \frac{1}{(V_G)_0} \left[(v_G)_k - \sum_{j=1}^{k} (V_G)_j (X_2)_{k-j} \right] \tag{6.41}$$

$$(X_3)_k = \frac{1}{(V_G)_0} \left[(w_G)_k - \sum_{j=1}^{k} (V_G)_j (X_3)_{k-j} \right] \tag{6.42}$$

$$(f_5)_k = \sum_{j=0}^{k} \left[(v_G)_j (z)_{k-j} - (w_G)_j (y)_{k-j} \right] \tag{6.43}$$

$$(f_6)_k = \sum_{j=0}^{k} \left[(w_G)_j (x)_{k-j} - (u_G)_j (z)_{k-j} \right] \tag{6.44}$$

$$(f_7)_k = \sum_{j=0}^{k} \left[(u_G)_j (y)_{k-j} - (v_G)_j (x)_{k-j} \right] \tag{6.45}$$

$$(f_8)_k = \sum_{j=0}^{k} \left[(f_5)_j (f_5)_{k-j} + (f_6)_j (f_6)_{k-j} + (f_7)_j (f_7)_{k-j} \right] \tag{6.46}$$

$$(f_9)_k = \frac{1}{(f_8)_0} \sum_{j=1}^{k} \left[\left(\frac{3j}{2k} - 1 \right) (f_8)_j (f_9)_{k-j} \right] \tag{6.47}$$

$$(Y_1)_k = \frac{1}{(f_9)_0} \left[(f_5)_k - \sum_{j=1}^{k} (f_9)_j (Y_1)_{k-j} \right] \tag{6.48}$$

$$(Y_2)_k = \frac{1}{(f_9)_0} \left[(f_6)_k - \sum_{j=1}^{k} (f_9)_j (Y_2)_{k-j} \right] \tag{6.49}$$

$$(Y_3)_k = \frac{1}{(f_9)_0}\left[(f_7)_k - \sum_{j=1}^{k}(f_9)_j(Y_3)_{k-j}\right] \tag{6.50}$$

$$(Z_1)_k = \sum_{j=0}^{k}\left[(X_2)_j(Y_3)_{k-j} - (X_3)_j(Y_2)_{k-j}\right] \tag{6.51}$$

$$(Z_2)_k = \sum_{j=0}^{k}\left[(X_3)_j(Y_1)_{k-j} - (X_1)_j(Y_3)_{k-j}\right] \tag{6.52}$$

$$(Z_3)_k = \sum_{j=0}^{k}\left[(X_1)_j(Y_2)_{k-j} - (X_2)_j(Y_1)_{k-j}\right] \tag{6.53}$$

$$(\mathrm{s}\sigma_A) = \frac{1}{k}\sum_{j=1}^{k} j(\sigma_A)_j(\mathrm{c}\sigma_A)_{k-j} \tag{6.54}$$

$$(\mathrm{c}\sigma_A) = -\frac{1}{k}\sum_{j=1}^{k} j(\sigma_A)_j(\mathrm{s}\sigma_A)_{k-j} \tag{6.55}$$

$$(C_S^*)_k = \sum_{j=0}^{k}\left[(C_S)_j(\mathrm{c}\sigma_A)_{k-j} + (C_L)_j(\mathrm{s}\sigma_A)_{k-j}\right] \tag{6.56}$$

$$(C_L^*)_k = \sum_{j=0}^{k}\left[-(C_S)_j(\mathrm{s}\sigma_A)_{k-j} + (C_L)_j(\mathrm{c}\sigma_A)_{k-j}\right] \tag{6.57}$$

$$(C_x)_k = \sum_{j=0}^{k}\left[(C_D)_j(X_1)_{k-j} + (C_S^*)_j(Y_1)_{k-j} + (C_L^*)_j(Z_1)_{k-j}\right] \tag{6.58}$$

$$(C_y)_k = \sum_{j=0}^{k}\left[(C_D)_j(X_2)_{k-j} + (C_S^*)_j(Y_2)_{k-j} + (C_L^*)_j(Z_2)_{k-j}\right] \tag{6.59}$$

$$(C_z)_k = \sum_{j=0}^{k}\left[(C_D)_j(X_3)_{k-j} + (C_S^*)_j(Y_3)_{k-j} + (C_L^*)_j(Z_3)_{k-j}\right] \tag{6.60}$$

$$(f_{10})_k = -\frac{S_{ref}}{2m}\sum_{j=0}^{k}(\rho)_j(f_4)_{k-j} \tag{6.61}$$

$$(f_x)_k = \sum_{j=0}^{k}(C_x)_j(f_{10})_{k-j} \tag{6.62}$$

$$(f_y)_k = \sum_{j=0}^{k}(C_y)_j(f_{10})_{k-j} \tag{6.63}$$

$$(f_z)_k = \sum_{j=0}^{k}(C_z)_j(f_{10})_{k-j} \tag{6.64}$$

6.1.3 Aerodynamic Forces with Wind

The computation of aerodynamic forces with wind also starts with determining the groundspeed:

$$\boldsymbol{V}_G = \begin{pmatrix} u_G \\ v_G \\ w_G \end{pmatrix} = \begin{pmatrix} u + \omega_E y \\ v - \omega_E x \\ w \end{pmatrix} \tag{6.65}$$

Next, $\boldsymbol{V}_G$ is transformed from $\mathcal{F}_I$ to $\mathcal{F}_V$, which requires the sines and cosines of $\tilde{\tau}$ and δ:

$$f_{11} = x^2 + y^2 \tag{6.66}$$

$$f_{12} = \sqrt{f_{11}} \tag{6.67}$$

$$\mathrm{s}\tilde{\tau} = \frac{y}{f_{12}} \qquad \mathrm{c}\tilde{\tau} = \frac{x}{f_{12}} \qquad \mathrm{s}\delta = \frac{z}{r} \qquad \mathrm{c}\delta = \frac{f_{12}}{r} \tag{6.68}$$

$$f_{13} = -\mathrm{s}\tilde{\tau}\mathrm{s}\delta \tag{6.69}$$

$$f_{14} = -\mathrm{s}\tilde{\tau}\mathrm{c}\delta \tag{6.70}$$

$$f_{15} = -\mathrm{c}\tilde{\tau}\mathrm{s}\delta \tag{6.71}$$

$$f_{16} = -\mathrm{c}\tilde{\tau}\mathrm{c}\delta \tag{6.72}$$

$$u_{G,V} = u_G f_{15} + v_G f_{13} + w_G \mathrm{c}\delta \tag{6.73}$$

$$v_{G,V} = -u_G \mathrm{s}\tilde{\tau} + v_G \mathrm{c}\tilde{\tau} \tag{6.74}$$

$$w_{G,V} = u_G f_{16} + v_G f_{14} - w_G \mathrm{s}\delta \tag{6.75}$$

Now the sines and cosines of χ_G and γ_G can be obtained:

$$f_{17} = u_{G,V}^2 + v_{G,V}^2 \tag{6.76}$$

$$f_{18} = \sqrt{f_{17}} \tag{6.77}$$

$$f_{19} = f_{17} + w_{G,V}^2 \tag{6.78}$$

$$V_G = \sqrt{f_{19}} \tag{6.79}$$

$$\mathrm{s}\chi_G = \frac{v_{G,V}}{f_{18}} \qquad \mathrm{c}\chi_G = \frac{u_{G,V}}{f_{18}} \qquad \mathrm{s}\gamma_G = -\frac{w_{G,V}}{V_G} \qquad \mathrm{c}\gamma_G = \frac{f_{18}}{V_G} \tag{6.80}$$

If $\boldsymbol{V}_W$ is provided in its components in $\mathcal{F}_V$, the components of $\boldsymbol{V}_{A,V}$ are computed from:

$$u_{A,V} = u_{G,V} - u_{W,V} \qquad v_{A,V} = v_{G,V} - v_{W,V} \qquad w_{A,V} = w_{G,V} - w_{W,V} \tag{6.81}$$

Now V_A and the sines and cosines of γ_A and χ_A can be obtained in similar fashion as for their groundspeed counterparts, using Eqs. (6.76) to (6.80). The variables f_{20}, f_{21} and f_{22} will then be the airspeed versions of f_{17}, f_{18} and f_{19}, respectively.

Next one has to choose the most efficient way to compute the sines and cosines of α_A, β_A and σ_A, while making sure that the signs of these variables are correct. For α_A and β_A, it is assumed that they stay within $\pm 90°$, thus only their sines have to be computed from elements of matrix $\boldsymbol{C}_{B,TA}$. Their cosines are then calculated using the trigonometric identity:

$$\cos^2 x = 1 - \sin^2 x \tag{6.82}$$

σ_A does not have to remain within $\pm 90°$; even when the commanded angle σ_G stays within this range, the wind can cause the magnitude of σ_A to exceed 90°. The necessary components of matrix $\boldsymbol{C}_{B,TA}$ can be computed using Eq. (2.92). From Eqs. (2.94) to (2.96) it follows that

elements (2,1), (2,2), (2,3) and (3,1) are needed to compute the sines of α_A, β_A and the sine and cosine of σ_A. These elements in turn require the following auxiliary variables:

$$f_{23} = \sin(\chi_G - \chi_A) = \mathrm{s}\chi_G\mathrm{c}\chi_A - \mathrm{c}\chi_G\mathrm{s}\chi_A \tag{6.83}$$

$$f_{24} = \cos(\chi_G - \chi_A) = \mathrm{s}\chi_G\mathrm{s}\chi_A + \mathrm{c}\chi_G\mathrm{c}\chi_A \tag{6.84}$$

$$f_{25} = \mathrm{s}\gamma_G\mathrm{s}\gamma_A \tag{6.85}$$

$$f_{26} = \mathrm{s}\gamma_G\mathrm{c}\gamma_A \tag{6.86}$$

$$f_{27} = \mathrm{c}\gamma_G\mathrm{c}\gamma_A \tag{6.87}$$

$$f_{28} = f_{24}f_{26} - \mathrm{c}\gamma_G\mathrm{s}\gamma_A \tag{6.88}$$

$$f_{29} = -f_{23}\mathrm{c}\gamma_A \tag{6.89}$$

$$f_{30} = f_{23}\mathrm{s}\gamma_G \tag{6.90}$$

$$f_{31} = f_{25} + f_{24}f_{27} \tag{6.91}$$

$$f_{32} = f_{27} + f_{24}f_{25} \tag{6.92}$$

$$f_{33} = -f_{23}\mathrm{s}\gamma_A \tag{6.93}$$

Entries (2,1), (2,2), (2,3) and (3,1) of $\boldsymbol{C}_{B,TA}$ can then be computed as follows, assuming that the commanded angle β_G is always equal to zero:

$$f_{34} = f_{28}\mathrm{c}\sigma_G + f_{29}\mathrm{s}\sigma_G \tag{6.94}$$

$$\boldsymbol{C}_{B,TA}(2,1) = f_{29}\mathrm{c}\sigma_G - f_{28}\mathrm{s}\sigma_G \tag{6.95}$$

$$\boldsymbol{C}_{B,TA}(2,2) = f_{24}\mathrm{c}\sigma_G + f_{30}\mathrm{s}\sigma_G \tag{6.96}$$

$$\boldsymbol{C}_{B,TA}(2,3) = f_{33}\mathrm{c}\sigma_G - f_{32}\mathrm{s}\sigma_G \tag{6.97}$$

$$\boldsymbol{C}_{B,TA}(3,1) = f_{31}\mathrm{s}\alpha_G + f_{34}\mathrm{c}\alpha_G \tag{6.98}$$

The cosines and sines of α_A, β_A and σ_A can then be determined:

$$\mathrm{s}\beta_A = \boldsymbol{C}_{B,TA}(2,1) \tag{6.99}$$

$$f_{35} = 1 - (\mathrm{s}\beta_A)^2 \tag{6.100}$$

$$\mathrm{c}\beta_A = \sqrt{f_{35}} \tag{6.101}$$

$$\mathrm{s}\alpha_A = \frac{\boldsymbol{C}_{B,TA}(3,1)}{\mathrm{c}\beta_A} \tag{6.102}$$

$$f_{36} = 1 - (\mathrm{s}\alpha_A)^2 \tag{6.103}$$

$$\mathrm{c}\alpha_A = \sqrt{f_{36}} \tag{6.104}$$

$$\mathrm{s}\sigma_A = \frac{-\boldsymbol{C}_{B,TA}(2,3)}{\mathrm{c}\beta_A} \tag{6.105}$$

$$\mathrm{c}\sigma_A = \frac{\boldsymbol{C}_{B,TA}(2,2)}{\mathrm{c}\beta_A} \tag{6.106}$$

The values of α_A and β_A are then obtained from the arcsin of $\mathrm{s}\alpha_A$ and $\mathrm{s}\beta_A$ (one cannot use arccos when the angles have values near $0°$, see Section 5.2). With those, the aerodynamic force coefficients C_D, C_S and C_L can be determined.

The second to last step of including wind is the transformation of the coefficients to $\mathcal{F}_I$, using the transformation in Eq. (2.97):

$$f_{37} = C_S \mathrm{c}\sigma_A + C_L \mathrm{s}\sigma_A \tag{6.107}$$

$$f_{38} = -C_S \mathrm{s}\sigma_A + C_L \mathrm{c}\sigma_A \tag{6.108}$$

$$f_{39} = C_D \mathrm{c}\gamma_A + f_{38} \mathrm{s}\gamma_A \tag{6.109}$$

$$f_{40} = -C_D \mathrm{s}\gamma_A + f_{38} \mathrm{c}\gamma_A \tag{6.110}$$

$$f_{41} = f_{39} \mathrm{c}\chi_A - f_{37} \mathrm{s}\chi_A \tag{6.111}$$

$$f_{42} = f_{39} \mathrm{s}\chi_A + f_{37} \mathrm{c}\chi_A \tag{6.112}$$

The aerodynamic force coefficients in $\mathcal{F}_I$ are then:

$$C_x = f_{41} f_{15} - f_{42} \mathrm{s}\tilde{\tau} + f_{40} f_{16} \tag{6.113}$$

$$C_y = f_{41} f_{13} + f_{42} \mathrm{c}\tilde{\tau} + f_{40} f_{14} \tag{6.114}$$

$$C_z = f_{41} \mathrm{c}\delta - f_{40} \mathrm{s}\delta \tag{6.115}$$

The aerodynamic accelerations are then calculated using:

$$f_x = C_x f_{43} \tag{6.116}$$

$$f_y = C_y f_{43} \tag{6.117}$$

$$f_z = C_z f_{43} \tag{6.118}$$

with f_{43} given by:

$$f_{43} = -\frac{\rho f_{22} S_{ref}}{2m} \tag{6.119}$$

The list of equations of this subsection is long and consists of only a few different types of mathematical operations. Therefore it has been decided to not list all of the recurrence relations, as it would be a very repetitive list. Rather, only one of each type of operation is rewritten to recurrence relation here. The first is the sum of two multiplications for f_{11}:

$$(f_{11})_k = \sum_{j=0}^{k} \left[(x)_j (x)_{k-j} + (y)_j (y)_{k-j} \right] \tag{6.120}$$

The square root for f_{12}, Eq. (6.67):

$$(f_{12})_k = \frac{1}{(f_{11})_0} \sum_{j=1}^{k} \left[\left(\frac{3j}{2k} - 1 \right) (f_{11})_j (f_{12})_{k-j} \right] \tag{6.121}$$

The division for $\mathrm{s}\tilde{\tau}$, Eq. (6.68):

$$(\mathrm{s}\tilde{\tau})_k = \frac{1}{(f_{12})_0} \left[(y)_k - \sum_{j=1}^{k} (f_{12})_j (\mathrm{s}\tilde{\tau})_{k-j} \right] \tag{6.122}$$

The sum of a variable and a square for f_{19}, Eq. (6.78):

$$(f_{19})_k = (f_{17})_k + \sum_{j=0}^{k} (w_{G,V})_j (w_{G,V})_{k-j} \tag{6.123}$$

The subtraction of one variable from another for $u_{A,V}$, Eq. (6.81):

$$(u_{A,V})_k = (u_{G,V})_k - (u_{W,V})_k \tag{6.124}$$

Finally, 1 subtracted by the square of a variable for f_{35}, Eq. (6.100):

$$(f_{35})_k = -\sum_{j=0}^{k} (\mathrm{s}\beta_A)_j (\mathrm{s}\beta_A)_{k-j} \tag{6.125}$$

and the multiplication of two variables and some constants for f_{43}, Eq. (6.119):

$$(f_{43})_k = -\frac{S_{ref}}{2m} \sum_{j=0}^{k} (f_{22})_j (\rho)_{k-j} \tag{6.126}$$

6.1.4 Rotating Planetocentric Frame

The equations of motion for Cartesian components in $\mathcal{F}_R$ are (Eqs. (2.72) and (2.76)):

$$\frac{d}{dt}\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} u \\ v \\ w \end{pmatrix} \tag{6.127}$$

$$\frac{d}{dt}\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \boldsymbol{g}_R + \frac{1}{m}\boldsymbol{F}_{A,R} - 2\begin{pmatrix} 0 \\ 0 \\ \omega_E \end{pmatrix} \times \begin{pmatrix} u \\ v \\ w \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \\ \omega_E \end{pmatrix} \times \left[\begin{pmatrix} 0 \\ 0 \\ \omega_E \end{pmatrix} \times \begin{pmatrix} x \\ y \\ z \end{pmatrix} \right] \tag{6.128}$$

When working out the cross products and replacing the vector $\boldsymbol{F}_{A,R}/m$ by the aerodynamic accelerations, one gets:

$$\frac{d}{dt}\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} g_x \\ g_y \\ g_z \end{pmatrix} + \begin{pmatrix} f_x \\ f_y \\ f_z \end{pmatrix} + 2\omega_E \begin{pmatrix} v \\ -u \\ 0 \end{pmatrix} + \omega_E^2 \begin{pmatrix} x \\ y \\ 0 \end{pmatrix} \tag{6.129}$$

In these equations, the index R has been omitted, as these state variables can directly replace those in the earlier equations of this section. As was mentioned in Section 2.6.1, the only practical change to the equations of motion is that groundspeed no longer has to be computed, this means that Eqs. (6.23), (6.37) and (6.65) are not needed. Furthermore, u, v and w replace u_G, v_G and w_G, respectively, in Eqs. (6.24) to (6.28), (6.38) to (6.45) and (6.73) to (6.75). Also note that formally $\tilde{\tau}$ should now be τ, but this is just a change of symbol and does not impact the equations or their outcome.

6.2 Spherical State Variables

Next up are the spherical state variables. This section is also divided into a part on the equations of motion, gravity and aerodynamic forces with and without wind.

The equations of motion are given by Eqs. (2.77) to (2.82):

$$\dot{r} = V_G \sin\gamma_G \tag{6.130}$$

$$\dot{\tau} = \frac{V_G \sin\chi_G \cos\gamma_G}{r\cos\delta} \tag{6.131}$$

$$\dot{\delta} = \frac{V_G \cos\chi_G \cos\gamma_G}{r} \tag{6.132}$$

$$\dot{V}_G = f_D - g_r \sin\gamma_G + g_\delta \cos\gamma_G \cos\chi_G + \omega_E^2 r \cos\delta \left(\sin\gamma_G \cos\delta - \cos\gamma_G \sin\delta \cos\chi_G\right) \tag{6.133}$$

$$\begin{aligned} V_G\dot{\gamma}_G = {} & f_S \sin\sigma_G - f_L \cos\sigma_G - g_r \cos\gamma_G - g_\delta \sin\gamma_G \cos\chi_G + 2\omega_E V_G \cos\delta \sin\chi_G \\ & + \frac{{V_G}^2}{r} \cos\gamma_G + \omega_E^2 r \cos\delta \left(\cos\delta \cos\gamma_G + \sin\gamma_G \sin\delta \cos\chi_G\right) \end{aligned} \tag{6.134}$$

$$\begin{aligned} V_G \cos\gamma_G \dot{\chi}_G = {} & f_S \cos\sigma_G + f_L \sin\sigma_G - g_\delta \sin\chi_G + 2\omega_E V_G \left(\sin\delta \cos\gamma_G - \cos\delta \sin\gamma_G \cos\chi_G\right) \\ & + \frac{{V_G}^2}{r} \cos^2\gamma_G \tan\delta \sin\chi_G + \omega_E^2 r \cos\delta \sin\delta \sin\chi_G \end{aligned} \tag{6.135}$$

where terms $-D_G/m$, $-S_G/m$ and $-L_G/m$ have been replaced by the aerodynamic accelerations f_D, f_S and f_L, respectively. Furthermore, a number of reappearing terms has been marked with colors and these terms will be some of the auxiliary variables.

However, the first auxiliary variables will have to be the cosine and sine of the five angles (τ, δ, γ_G, χ_G and σ_G) in these formulas. These are stored as auxiliary variables, written in the same format as for the Cartesian state variables, i.e., the sine of γ_G is written $s\gamma_G$. The next four variables are the color-coded terms in Eqs. (6.130) to (6.135):

$$f_0 = \frac{V_G \mathrm{c}\gamma_G}{r} \tag{6.136}$$

$$f_1 = \mathrm{c}\gamma_G \mathrm{c}\chi_G \tag{6.137}$$

$$f_2 = \omega_E^2 r \mathrm{c}\delta \tag{6.138}$$

$$f_3 = \mathrm{s}\gamma_G \mathrm{c}\chi_G \tag{6.139}$$

The other auxiliary variables are:

$$f_4 = \mathrm{s}\gamma_G \mathrm{c}\delta - f_1 \mathrm{s}\delta \tag{6.140}$$

$$f_5 = \mathrm{c}\delta \mathrm{c}\gamma_G + f_3 \mathrm{s}\delta \tag{6.141}$$

$$f_6 = f_2 \mathrm{s}\delta - g_\delta \tag{6.142}$$

The terms in Eq. (6.134) that have to be divided by V_G are grouped into one variable:

$$f_7 = f_S \mathrm{s}\sigma_G - f_L \mathrm{c}\sigma_G - g_r \mathrm{c}\gamma_G - g_\delta f_3 + f_2 f_5 \tag{6.143}$$

$$f_8 = \frac{f_7}{V_G} \tag{6.144}$$

The terms in Eq. (6.135) that are divided by V_G are summed:

$$f_9 = f_S \mathrm{c}\sigma_G + f_L \mathrm{s}\sigma_G + f_6 \mathrm{s}\chi_G \tag{6.145}$$

$$f_{10} = \frac{f_9}{V_G} \tag{6.146}$$

Similar to f_7 and f_9, the terms that will be divided by $\mathrm{c}\gamma_G$ are grouped into a variable:

$$f_{11} = -2\omega_E \mathrm{c}\delta f_3 + f_{10} \tag{6.147}$$

$$f_{12} = \frac{f_{11}}{\mathrm{c}\gamma_G} \tag{6.148}$$

The equations of motion are then written as:

$$\dot{r} = V_G \mathrm{s}\gamma_G \tag{6.149}$$

$$\dot{\tau} = \frac{f_0 \mathrm{s}\chi_G}{\mathrm{c}\delta} \tag{6.150}$$

$$\dot{\delta} = f_0 \mathrm{c}\chi_G \tag{6.151}$$

$$\dot{V}_G = f_D - g_r \mathrm{s}\gamma_G + g_\delta f_1 + f_2 f_4 \tag{6.152}$$

$$\dot{\gamma}_G = f_8 + 2\omega_E \mathrm{c}\delta \mathrm{s}\chi_G + f_0 \tag{6.153}$$

Remarking that the second to last term in Eq. (6.135) resembles $\dot{\tau}\sin\delta$, the equation for $\dot{\chi}_G$ can be written as:

$$\dot{\chi}_G = f_{12} + (2\omega_E + \dot{\tau})\, s\delta \tag{6.154}$$

The list of recurrence relations is then as follows:

$$(f_0)_k = \frac{1}{(r)_0}\left[\sum_{j=0}^{k}(V_G)_j(\mathrm{c}\gamma_G)_{k-j} - \sum_{j=1}^{k}(r)_j(f_0)_{k-j}\right] \tag{6.155}$$

$$(f_1)_k = \sum_{j=0}^{k}(\mathrm{c}\gamma_G)_j(\mathrm{c}\chi_G)_{k-j} \tag{6.156}$$

$$(f_2)_k = \omega_E^2 \sum_{j=0}^{k}(r)_j(\mathrm{c}\delta)_{k-j} \tag{6.157}$$

$$(f_3)_k = \sum_{j=0}^{k}(\mathrm{s}\gamma_G)_j(\mathrm{c}\chi_G)_{k-j} \tag{6.158}$$

$$(f_4)_k = \sum_{j=0}^{k}\left[(\mathrm{s}\gamma_G)_j(\mathrm{c}\delta)_{k-j} - (f_1)_j(\mathrm{s}\delta)_{k-j}\right] \tag{6.159}$$

$$(f_5)_k = \sum_{j=0}^{k}\left[(\mathrm{c}\delta)_j(\mathrm{c}\gamma_G)_{k-j} + (f_3)_j(\mathrm{s}\delta)_{k-j}\right] \tag{6.160}$$

$$(f_6)_k = -(g_\delta)_k + \sum_{j=0}^{k}(f_2)_j(\mathrm{s}\delta)_{k-j} \tag{6.161}$$

$$(f_7)_k = \sum_{j=0}^{k}\left[(f_S)_j(\mathrm{s}\sigma_G)_{k-j} - (f_L)_j(\mathrm{c}\sigma_G)_{k-j} - (g_r)_j(\mathrm{c}\gamma_G)_{k-j} - (g_\delta)_j(f_3)_{k-j} + (f_2)_j(f_5)_{k-j}\right] \tag{6.162}$$

$$(f_8)_k = \frac{1}{(V_G)_0}\left[(f_7)_k - \sum_{j=1}^{k}(V_G)_j(f_8)_{k-j}\right] \tag{6.163}$$

$$(f_9)_k = \sum_{j=0}^{k}\left[(f_S)_j(\mathrm{c}\sigma_G)_{k-j} + (f_L)_j(\mathrm{s}\sigma_G)_{k-j} + (f_6)_j(\mathrm{s}\chi_G)_{k-j}\right] \tag{6.164}$$

$$(f_{10})_k = \frac{1}{(V_G)_0}\left[(f_9)_k - \sum_{j=1}^{k}(V_G)_j(f_{10})_{k-j}\right] \tag{6.165}$$

$$(f_{11})_k = (f_{10})_k - 2\omega_E \sum_{j=0}^{k}(\mathrm{c}\delta)_j(f_3)_{k-j} \tag{6.166}$$

$$(f_{12})_k = \frac{1}{(\mathrm{c}\gamma_G)_0}\left[(f_{11})_k - \sum_{j=1}^{k}(\mathrm{c}\gamma_G)_j(f_{12})_{k-j}\right] \tag{6.167}$$

$$(\dot{r})_k = \sum_{j=0}^{k}(V_G)_j(\mathrm{s}\gamma_G)_{k-j} \tag{6.168}$$

$$(\dot{\tau})_k = \frac{1}{(\mathrm{c}\delta)_0}\left[\sum_{j=0}^{k}(f_0)_j(\mathrm{s}\chi_G)_{k-j} - \sum_{j=1}^{k}(\mathrm{c}\delta)_j(\dot{\tau})_{k-j}\right] \tag{6.169}$$

$$(\dot{\delta})_k = \sum_{j=0}^{k}(f_0)_j(\mathrm{c}\chi_G)_{k-j} \tag{6.170}$$

$$(\dot{V}_G)_k = (f_D)_k + \sum_{j=0}^{k}\left[-(g_r)_j(\mathrm{s}\gamma_G)_{k-j} + (g_\delta)_j(f_1)_{k-j} + (f_2)_j(f_4)_{k-j}\right] \tag{6.171}$$

$$(\dot{\gamma}_G)_k = (f_0)_k + (f_8)_k + 2\omega_E\sum_{j=0}^{k}(\mathrm{c}\delta)_j(\mathrm{s}\chi_G)_{k-j} \tag{6.172}$$

$$(\dot{\chi}_G)_k = (f_{12})_k + 2\omega_E(\mathrm{s}\delta)_k + \sum_{j=0}^{k}(\dot{\tau})_j(\mathrm{s}\delta)_{k-j} \tag{6.173}$$

And finally the Taylor coefficients of the state variables themselves are:

$$(r)_k = \frac{1}{k}(\dot{r})_{k-1} \qquad (\tau)_k = \frac{1}{k}(\dot{\tau})_{k-1} \qquad (\delta)_k = \frac{1}{k}(\dot{\delta})_{k-1} \tag{6.174}$$

$$(V_G)_k = \frac{1}{k}(\dot{V}_G)_{k-1} \qquad (\gamma_G)_k = \frac{1}{k}(\dot{\gamma}_G)_{k-1} \qquad (\chi_G)_k = \frac{1}{k}(\dot{\chi}_G)_{k-1} \tag{6.175}$$

6.2.1 Gravity

The equations of gravity for the spherical coordinates consist of Eqs. (2.66) and (2.67):

$$g_r = \frac{\mu_E}{r^2}\left[1 - \frac{3}{2}J_2\left(\frac{R_E}{r}\right)^2\left(3\sin^2\delta - 1\right)\right] \tag{6.176}$$

$$g_\delta = -3J_2\frac{\mu_E}{r^2}\left(\frac{R_E}{r}\right)^2\sin\delta\cos\delta \tag{6.177}$$

The auxiliary variables used are:

$$f_{13} = r^2 \tag{6.178}$$

$$f_{14} = 1 - \frac{3}{2}J_2\frac{R_E^2}{f_{13}}\left(3(\mathrm{s}\delta)^2 - 1\right) \tag{6.179}$$

$$f_{15} = -3J_2\frac{R_E^2}{f_{13}}\mathrm{s}\delta\mathrm{c}\delta \tag{6.180}$$

with which Eqs. (6.176) and (6.177) turn into:

$$g_r = \frac{\mu_E}{f_{13}}f_{14} \tag{6.181}$$

$$g_\delta = \frac{\mu_E}{f_{13}} f_{15} \tag{6.182}$$

The recurrence relation of f_{13} is simply:

$$(f_{13})_k = \sum_{j=0}^{k} (r)_j (r)_{k-j} \tag{6.183}$$

For f_{14}, one make use of the fact that the Taylor coefficients of $\left(3(\mathrm{s}\delta)^2 - 1\right)$ are equal those of $3(\mathrm{s}\delta)^2$, with the exception of the first ($k = 0$) term. Since recurrence relations only hold for $k \geq 1$, the -1 does not appear in the recurrence relation:

$$(f_{14})_k = \frac{1}{(f_{13})_0} \left[(f_{13})_k - \frac{9}{2} J_2 R_E^2 \sum_{j=0}^{k} (\mathrm{s}\delta)_j (\mathrm{s}\delta)_{k-j} - \sum_{j=1}^{k} (f_{13})_j (f_{14})_{k-j} \right] \tag{6.184}$$

The other relations are:

$$(f_{15})_k = \frac{-1}{(f_{13})_0} \left[3 J_2 R_E^2 \sum_{j=0}^{k} (\mathrm{s}\delta)_j (\mathrm{c}\delta)_{k-j} + \sum_{j=1}^{k} (f_{13})_j (f_{15})_{k-j} \right] \tag{6.185}$$

$$(g_r)_k = \frac{1}{(f_{13})_0} \left[\mu_E (f_{14})_k - \sum_{j=1}^{k} (f_{13})_j (g_r)_{k-j} \right] \tag{6.186}$$

$$(g_\delta)_k = \frac{1}{(f_{13})_0} \left[\mu_E (f_{15})_k - \sum_{j=1}^{k} (f_{13})_j (g_\delta)_{k-j} \right] \tag{6.187}$$

6.2.2 Aerodynamic Forces without Wind

Next, the aerodynamic force equations are rewritten. Note that the relationship between the aerodynamic coefficients and the aerodynamic angles and the Mach number are vehicle specific and are therefore not given in this report, thus their recurrence relations are not derived here.

Aerodynamic forces are given by Eq. (2.61), which gives the following relation for the aerodynamic accelerations:

$$\begin{pmatrix} f_D \\ f_S \\ f_L \end{pmatrix} = - \begin{pmatrix} C_D \\ C_S \\ C_L \end{pmatrix} \frac{\rho V_G^2 S_{ref}}{2m} \tag{6.188}$$

Without wind, the groundspeed and airspeed-based forces are equal, and the terms f_D, f_S and f_L can be directly filled into the equations of motion. The auxiliary variables are:

$$f_{16} = V_G^2 \tag{6.189}$$

$$f_{17} = -\frac{\rho f_{16} S_{ref}}{2m} \tag{6.190}$$

The recurrence relations are then:

$$(f_{16})_k = \sum_{j=0}^{k} (V_G)_j (V_G)_{k-j} \tag{6.191}$$

$$(f_{17})_k = -\frac{S_{ref}}{2m}\sum_{j=0}^{k}(\rho)_j(f_{16})_{k-j} \tag{6.192}$$

$$(f_D)_k = \sum_{j=0}^{k}(C_D)_j(f_{17})_{k-j} \tag{6.193}$$

$$(f_S)_k = \sum_{j=0}^{k}(C_S)_j(f_{17})_{k-j} \tag{6.194}$$

$$(f_L)_k = \sum_{j=0}^{k}(C_L)_j(f_{17})_{k-j} \tag{6.195}$$

6.2.3 Aerodynamic Forces with Wind

Firstly, the components of $V_{G,V}$ are determined with Eq. (2.98):

$$f_{18} = \mathrm{s}\chi_G\mathrm{c}\gamma_G \tag{6.196}$$

$$\begin{pmatrix} u_{G,V} \\ v_{G,V} \\ w_{G,V} \end{pmatrix} = V_G \begin{pmatrix} \cos\chi_G\cos\gamma_G \\ \sin\chi_G\cos\gamma_G \\ -\sin\gamma_G \end{pmatrix} = V_G \begin{pmatrix} f_1 \\ f_{18} \\ -\mathrm{s}\gamma_G \end{pmatrix} \tag{6.197}$$

The components of $V_{A,V}$ and the airspeed-based angles α_A, β_A and σ_A are then determined in the same way as for the Cartesian state variables (see Section 6.1.3, Eqs. (6.81) to (6.119)). Since the auxiliary variables f_{20} to f_{36} are still vacant, they can be copied directly from Section 6.1.3. Auxiliary-variable numbering in this section will start at 37.

With α_A and β_A, the aerodynamic coefficients can be determined and those can in turn be multiplied with:

$$f_{37} = -\frac{\rho f_{22} S_{ref}}{2m} \tag{6.198}$$

to obtain the aerodynamic accelerations $f_{D,AA}$, $f_{S,AA}$ and $f_{L,AA}$. These have to be transformed to $\mathcal{F}_{TG}$, using Eq. (2.104). For this, one can use the transpose of matrix $\boldsymbol{C}_{TA,TG}$ [43]:

$$\begin{bmatrix} \mathrm{c}\gamma_G\mathrm{c}(\chi_G-\chi_A)\mathrm{c}\gamma_A + \mathrm{s}\gamma_G\mathrm{s}\gamma_A & \mathrm{c}\gamma_G\mathrm{s}(\chi_G-\chi_A) & \mathrm{c}\gamma_G\mathrm{c}(\chi_G-\chi_A)\mathrm{s}\gamma_A - \mathrm{s}\gamma_G\mathrm{c}\gamma_A \\ -\mathrm{s}(\chi_G-\chi_A)\mathrm{c}\gamma_A & \mathrm{c}(\chi_G-\chi_A) & \mathrm{s}(\chi_G-\chi_A)\mathrm{s}\gamma_A \\ \mathrm{s}\gamma_G\mathrm{c}(\chi_G-\chi_A)\mathrm{c}\gamma_A - \mathrm{c}\gamma_G\mathrm{s}\gamma_A & \mathrm{s}\gamma_G\mathrm{s}(\chi_G-\chi_A) & \mathrm{s}\gamma_G\mathrm{c}(\chi_G-\chi_A)\mathrm{s}\gamma_A - \mathrm{c}\gamma_G\mathrm{c}\gamma_A \end{bmatrix} \tag{6.199}$$

of which seven elements were already determined in the process of determining α_A, β_A and σ_A. The remaining two are determined as follows:

$$f_{38} = \mathrm{c}\gamma_G f_{23} \tag{6.200}$$

$$f_{39} = \mathrm{c}\gamma_G\mathrm{s}\gamma_A \tag{6.201}$$

$$f_{40} = f_{39}f_{24} - f_{26} \tag{6.202}$$

Now $\boldsymbol{C}_{TA,TG}$ is given by:

$$\begin{bmatrix} f_{31} & f_{38} & f_{40} \\ f_{29} & f_{24} & -f_{33} \\ f_{28} & f_{30} & f_{32} \end{bmatrix} \tag{6.203}$$

The aerodynamic accelerations are then transformed from $\mathcal{F}_{AA}$ to $\mathcal{F}_{TG}$ using $\boldsymbol{C}_{TA,AA}$ and the transpose of $\boldsymbol{C}_{TA,TG}$:

$$f_{41} = f_{S,AA}\mathrm{c}\sigma_A + f_{L,AA}\mathrm{s}\sigma_A \tag{6.204}$$

$$f_{42} = -f_{S,AA}\mathrm{s}\sigma_A + f_{L,AA}c\sigma_A \tag{6.205}$$

$$f_{D,TG} = f_{D,AA}f_{31} + f_{41}f_{29} + f_{42}f_{28} \tag{6.206}$$

$$f_{S,TG} = f_{D,AA}f_{38} + f_{41}f_{24} + f_{42}f_{30} \tag{6.207}$$

$$f_{L,TG} = f_{D,AA}f_{40} - f_{41}f_{33} + f_{42}f_{32} \tag{6.208}$$

One can then fill these terms into the equations of motion, using Eqs. (2.101) to (2.103). This requires an adaption of the equations for f_7 and f_9 (Eqs. (6.143) and (6.145), respectively):

$$f_7 = -f_{L,TG} - g_r\mathrm{c}\gamma_G + g_\delta f_3 + f_2 f_5 \tag{6.209}$$

$$f_9 = f_{S,TG} + f_6\mathrm{s}\chi_G \tag{6.210}$$

The recurrence relations are not included in this subsection, as the equations here are solely multiplications, summations and subtractions, which have been treated exhaustively.

6.3 Spherical Position with Cartesian Velocity

In this section, the recurrence relations themselves will no longer be given; only the process of splitting the equations into auxiliary variables is shown.

The kinematic equations for this state-variable set are (Eqs. (2.83) to (2.85)):

$$\dot{r} = -w \qquad \dot{\tau} = \frac{v}{r\cos\delta} \qquad \dot{\delta} = \frac{u}{r} \tag{6.211}$$

The dynamic equations (Eqs. (2.86) to (2.88)) are here given in a reordered form:

$$\dot{u} = f_x + g_\delta - (2\omega_E v + \omega_E^2 r\cos\delta + \dot{\tau}v)\sin\delta + \dot{\delta}w \tag{6.212}$$

$$\dot{v} = f_y + (2\omega_E + \dot{\tau})(u\sin\delta + w\cos\delta) \tag{6.213}$$

$$\dot{w} = f_z + g_r - (2\omega_E v + \omega_E^2 r\cos\delta + \dot{\tau}v)\cos\delta - \dot{\delta}u \tag{6.214}$$

where f_x, f_y and f_z mark the aerodynamic accelerations in $\mathcal{F}_V$. The auxiliary variables are in this case:

$$f_0 = r\mathrm{c}\delta \tag{6.215}$$

$$f_1 = (2\omega_E + \dot{\tau})\, v + \omega_E^2 f_0 \tag{6.216}$$

$$f_2 = u\mathrm{s}\delta + w\mathrm{c}\delta \tag{6.217}$$

with which the equation for $\dot{\tau}$ and the dynamic equations become:

$$\dot{\tau} = \frac{v}{f_0} \tag{6.218}$$

$$\dot{u} = f_x + g_\delta - f_1\mathrm{s}\delta + \dot{\delta}w \tag{6.219}$$

$$\dot{v} = f_y + (2\omega_E + \dot{\tau})f_2 \tag{6.220}$$

$$\dot{w} = f_z + g_r - f_1\mathrm{c}\delta - \dot{\delta}u \tag{6.221}$$

6.3.1 Gravity

The equations for the gravity accelerations are exactly the same as for the spherical state variables, consisting of g_r and g_δ (Eqs. (2.66) and (2.67)):

$$g_r = \frac{\mu_E}{r^2}\left[1 - \frac{3}{2}J_2\left(\frac{R_E}{r}\right)^2\left(3\sin^2\delta - 1\right)\right] \tag{6.222}$$

$$g_\delta = -3J_2\frac{\mu_E}{r^2}\left(\frac{R_E}{r}\right)^2 \sin\delta\cos\delta \tag{6.223}$$

Thus the auxiliary variables are the same, the only difference is the numbering:

$$f_3 = r^2 \tag{6.224}$$

$$f_4 = 1 - \frac{3}{2}J_2\frac{R_E^2}{f_3}\left(3(\mathrm{s}\delta)^2 - 1\right) \tag{6.225}$$

$$f_5 = -3J_2\frac{R_E^2}{f_3}\mathrm{s}\delta\mathrm{c}\delta \tag{6.226}$$

with which Eqs. (6.222) and (6.223) turn into:

$$g_r = \frac{\mu_E}{f_3}f_4 \tag{6.227}$$

$$g_\delta = \frac{\mu_E}{f_3}f_5 \tag{6.228}$$

6.3.2 Aerodynamic Forces without Wind

Similar to the spherical state variables, the aerodynamic accelerations in $\mathcal{F}_{AG}$ are computed first:

$$\begin{pmatrix} f_D \\ f_S \\ f_L \end{pmatrix} = \begin{pmatrix} C_D \\ C_S \\ C_L \end{pmatrix} f_8 \tag{6.229}$$

with f_8 computed as follows:

$$f_6 = u^2 + v^2 \tag{6.230}$$

$$f_7 = f_6 + w^2 \tag{6.231}$$

$$f_8 = \frac{\rho f_7 S_{ref}}{2m} \tag{6.232}$$

where f_6 and f_7 are also computed for later use. The accelerations are then transformed to $\mathcal{F}_{TG}$. f_D remains unchanged, but the other two are transformed to respectively:

$$f_S^* = f_S\mathrm{c}\sigma_G + f_L\mathrm{s}\sigma_G \tag{6.233}$$

$$f_L^* = -f_S\mathrm{s}\sigma_G + f_L\mathrm{c}\sigma_G \tag{6.234}$$

The accelerations f_x, f_y and f_z are then determined with transformation matrix $\boldsymbol{C}_{V,TG}$ (Eq. (2.10)). The sines and cosines of this matrix can be calculated using the relations in Eqs. (2.24) and (2.25). The transformation has been optimized to use the smallest number of variables, yielding:

$$f_9 = \sqrt{f_6} \tag{6.235}$$

$$V_G = \sqrt{f_7} \tag{6.236}$$

$$f_{10} = \frac{f_D}{V_G} \tag{6.237}$$

$$f_{11} = \frac{f_S^*}{f_9} \tag{6.238}$$

$$f_{12} = \frac{f_L^*}{V_G} \tag{6.239}$$

$$f_{13} = \frac{f_{12} w}{f_9} \tag{6.240}$$

with which the accelerations become:

$$f_x = f_{10} u - f_{11} v - f_{13} u \tag{6.241}$$

$$f_y = f_{10} v + f_{11} u - f_{13} v \tag{6.242}$$

$$f_z = f_{10} w - f_{12} f_9 \tag{6.243}$$

6.3.3 Aerodynamic Forces with Wind

The process of obtaining the aerodynamic accelerations with wind is largely the same as for the spherical state variables given Section 6.2.3. The auxiliary variables f_{20} to f_{37} are taken from that section. The difference with spherical state variables is that now the aerodynamic accelerations f_D, f_S and f_L will be transformed from $\mathcal{F}_{AA}$ to $\mathcal{F}_V$, instead of $\mathcal{F}_{TG}$. Transformation matrix $\boldsymbol{C}_{V,TA}$ is given by Eq. (2.10). This transformation requires the following variables:

$$f_{38} = \mathrm{c}\chi_A \mathrm{c}\gamma_A \tag{6.244}$$

$$f_{39} = \mathrm{c}\chi_A \mathrm{s}\gamma_A \tag{6.245}$$

$$f_{40} = \mathrm{s}\chi_A \mathrm{c}\gamma_A \tag{6.246}$$

$$f_{41} = \mathrm{s}\chi_A \mathrm{s}\gamma_A \tag{6.247}$$

with which the transformation to $\mathcal{F}_V$ is given by:

$$f_{42} = f_{S,AA} \mathrm{c}\sigma_A + f_{L,AA} \mathrm{s}\sigma_A \tag{6.248}$$

$$f_{43} = -f_{S,AA} \mathrm{s}\sigma_A + f_{L,AA} c\sigma_A \tag{6.249}$$

$$f_x = f_{D,AA} f_{38} - f_{42} \mathrm{s}\chi_A + f_{43} f_{39} \tag{6.250}$$

$$f_y = f_{D,AA} f_{40} + f_{42} \mathrm{c}\chi_A - f_{43} f_{41} \tag{6.251}$$

$$f_z = -f_{D,AA} \mathrm{c}\gamma_A + f_{43} \mathrm{c}\gamma_A \tag{6.252}$$

6.4 Environment Models

This section is devoted to the recurrence relations of the environment models. This excludes the gravity field model, as it was already included in the previous sections. Note that the indices of the auxiliary variables will start at 0 again, but note that these variables have to coexist with those of the used state-variable set's equations of motion.

6.4.1 Planet Shape

The planet shape approximation consisted of only a single equation, namely Eq. (2.34):

$$H = r - R_E\sqrt{\frac{1-e^2}{1-e^2\mathrm{c}^2\delta}} \tag{6.253}$$

This creates the need for two auxiliary variables:

$$f_0 = 1 - e^2(c\delta)^2 \tag{6.254}$$

$$f_1 = R_E\sqrt{1-e^2}f_0^{-1/2} \tag{6.255}$$

and this yields the following recurrence relations:

$$(f_0)_k = -e^2\sum_{j=0}^{k}(\mathrm{c}\delta)_j(\mathrm{c}\delta)_{k-j} \tag{6.256}$$

$$(f_1)_k = \frac{1}{(f_0)_0}\sum_{j=1}^{k}\left[\left(\frac{j}{2k}-1\right)(f_0)_j(f_1)_{k-j}\right] \tag{6.257}$$

$$(H)_k = (r)_k - (f_1)_k \tag{6.258}$$

Notice that in Eq. (6.257), the multiplication with a constant does not appear, as the first term of the series $(f_1)_0$ already contains this multiplication and the higher-order terms are all a function of $(f_1)_0$. Thus all the higher-order terms will automatically be multiplied by the constant. This applies to, inter alia, functions of the form $c \cdot x^\alpha$ and $c \cdot \exp(x)$, with c being a constant. Note that functions of the form $\exp(x+c)$ can be written as $\exp(c) \cdot \exp(x)$ and thus the same applies. However, for functions of the form $\exp(c \cdot x)$, c will appear in the expressions of the Taylor coefficients.

6.4.2 Exponential Atmosphere

Two equations are of interest for the exponential atmosphere, namely that of ρ and of M, which are given by Eq. (2.45) and (2.59), respectively:

$$\rho = \rho_0 \exp\left(-\frac{H}{H_S}\right) \tag{6.259}$$

$$M = \frac{V_A}{a} \tag{6.260}$$

where speed of sound a (given by Eq. (2.60)) is constant, since the exponential atmosphere assumes constant temperature and molecular mass. No auxiliary variables are needed, so the recurrence relations are:

$$(\rho)_k = -\frac{1}{kH_S}\sum_{j=1}^{k} j(H)_j(\rho)_{k-j} \tag{6.261}$$

$$(M)_k = \frac{1}{a}(V_A)_k \tag{6.262}$$

6.4.3 United States Standard Atmosphere 1976

The original equations of US76 were given Section 2.4.3 and are converted here. Since the US76 model uses different equations for different layers of the atmosphere, the recurrence relations are derived per layer. This starts with the layers up to 86 km altitude:

$$z = \frac{R_0 H}{R_0 + H} \tag{6.263}$$

$$T_M = T_{M_i} + L_{M_i}(z - z_i) \tag{6.264}$$

$$p = p_i \cdot T_{M_i}^{C/L_{M_i}} \cdot T_M^{-C/L_{M_i}} \tag{6.265}$$

$$\rho = \frac{M_0 p}{\mathcal{R}^* T_M} \tag{6.266}$$

where index i indicates the layer and C is a constant (independent of the layer), given by:

$$C = \frac{g_0 M_0}{\mathcal{R}^*} \tag{6.267}$$

For layers 1 and 4, the lapse rate L_{M_i} is zero and instead of Eq. (6.265), one should use:

$$p = p_i \exp\left[\frac{-C\,(z - z_i)}{T_{M_i}}\right] \tag{6.268}$$

a can be computed with T_M, instead of T, so that the correction factor M_M/M_0 given in Table A.1 is not needed:

$$a = \sqrt{\frac{\gamma \mathcal{R}^* T_M}{M_0}} \tag{6.269}$$

The recurrence relations of z and T_M are given by:

$$(z)_k = \frac{1}{R_0 + (H)_0}\left[R_0 (H)_k - \sum_{j=1}^{k} (H)_j (z)_{k-j}\right] \tag{6.270}$$

$$(T_M)_k = L_{M_i}(z)_k \tag{6.271}$$

The first equation of p gives:

$$(p)_k = \frac{1}{(T_M)_0}\sum_{j=1}^{k}\left\{\left[\frac{j}{k}\left(1 - \frac{C}{L_{M_i}}\right) - 1\right](T_M)_j (p)_{k-j}\right\} \tag{6.272}$$

The second equation yields:

$$(p)_k = \frac{-C}{k T_{M_i}}\sum_{j=1}^{k} j (z)_j (p)_{k-j} \tag{6.273}$$

The equations for ρ and a give:

$$(\rho)_k = \frac{1}{(T_M)_0}\left[\frac{M_0}{\mathcal{R}^*}(p)_k - \sum_{j=1}^{k}(T_M)_j (\rho)_{k-j}\right] \tag{6.274}$$

$$(a)_k = \frac{1}{(T_M)_0}\sum_{j=1}^{k}\left[\left(\frac{3j}{2k} - 1\right)(T_M)_j (a)_{k-j}\right] \tag{6.275}$$

Next up is layer 7, for which the temperature T is constant and equal to T_7. For this layer, a is given by:

$$a = \sqrt{\frac{\gamma \mathcal{R}^* T_7}{M_M}} \tag{6.276}$$

The recurrence relation for a is now:

$$(a)_k = \frac{1}{(M_M)_0} \sum_{j=1}^{k} \left[\left(\frac{j}{2k} - 1 \right) (M_M)_j (a)_{k-j} \right] \tag{6.277}$$

For layer 8, T is given by:

$$T = T_c + A\sqrt{f_2} \tag{6.278}$$

$$f_2 = 1 - \left(\frac{H - H_8}{b} \right)^2 \tag{6.279}$$

yielding the following recurrence relations:

$$(f_2)_k = 2\frac{H_8 (H)_k}{b^2} - \frac{1}{b^2} \sum_{j=0}^{k} (H)_j (H)_{k-j} \tag{6.280}$$

$$(T)_k = \frac{1}{(f_2)_0} \left\{ \frac{1}{2} (f_2)_k \big[(T)_0 - T_c \big] + \sum_{j=1}^{k-1} \left[\left(\frac{3j}{2k} - 1 \right) (f_2)_j (T)_{k-j} \right] \right\} \tag{6.281}$$

For layer 9, the equation and recurrence relation of T are:

$$T = T_9 + L_9 (H - H_9) \tag{6.282}$$

$$(T)_k = L_9 (H)_k \tag{6.283}$$

For layer 10, T and the necessary auxiliary variables are given by:

$$f_3 = -\lambda \frac{(H - H_{10})(R_0 + H_{10})}{R_0 + H} \tag{6.284}$$

$$f_4 = (T_\infty - T_{10}) \exp(f_3) \tag{6.285}$$

$$T = T_\infty - f_4 \tag{6.286}$$

which yield the following recurrence relations:

$$(f_3)_k = \frac{1}{R_0 + (H)_0} \left[-\lambda (H - H_{10}) (H)_k - \sum_{j=1}^{k} (H)_j (f_3)_{k-j} \right] \tag{6.287}$$

$$(f_4)_k = \frac{1}{k} \sum_{j=1}^{k} j (f_3)_j (f_4)_{k-j} \tag{6.288}$$

$$(T)_k = -(f_4)_k \tag{6.289}$$

For layers 8 to 10, a is given by:

$$a = \sqrt{f_5} \tag{6.290}$$

with f_5 given by:

$$f_5 = \frac{\gamma \mathcal{R}^* T}{M_M} \tag{6.291}$$

yielding the recurrence relations:

$$(f_5)_k = \frac{1}{(M_M)_0}\left[\gamma\mathcal{R}^*(T)_k - \sum_{j=1}^{k}(M_M)_j(f_5)_{k-j}\right] \tag{6.292}$$

$$(a)_k = \frac{1}{(f_5)_0}\sum_{j=1}^{k}\left[\left(\frac{3j}{2k}-1\right)(f_5)_j(a)_{k-j}\right] \tag{6.293}$$

Finally, the Mach number is determined using Eq. (6.260), which in this case yields the following recurrence relation:

$$(M)_k = \frac{1}{(a)_0}\left[(V_A)_k - \sum_{j=1}^{k}(a)_j(M)_{k-j}\right] \tag{6.294}$$

Next up are the fits of the tables of US76, starting with M_M, given by:

$$M_M = a_0 + a_1\sin(b_0H) + a_2\cos(b_0H) + a_3\sin(b_1H) + a_4\cos(b_1H) \tag{6.295}$$

This requires the following auxiliary variables:

$$f_6 = a_1\sin(b_0H) \tag{6.296}$$

$$f_7 = a_2\cos(b_0H) \tag{6.297}$$

$$f_8 = a_3\sin(b_1H) \tag{6.298}$$

$$f_9 = a_4\cos(b_1H) \tag{6.299}$$

The recurrence relations of these variables are:

$$(f_6)_k = \frac{b_0}{k}\sum_{j=1}^{k}j(H)_j(f_7)_{k-j} \tag{6.300}$$

$$(f_7)_k = \frac{-b_0}{k}\sum_{j=1}^{k}j(H)_j(f_6)_{k-j} \tag{6.301}$$

$$(f_8)_k = \frac{b_1}{k}\sum_{j=1}^{k}j(H)_j(f_9)_{k-j} \tag{6.302}$$

$$(f_9)_k = \frac{-b_1}{k}\sum_{j=1}^{k}j(H)_j(f_8)_{k-j} \tag{6.303}$$

$$(M_M)_k = (f_6)_k + (f_7)_k + (f_8)_k + (f_9)_k \tag{6.304}$$

The fits of ρ are all exponential functions of polynomials. Rather than giving the relations for all layers, assume that ρ is given by the exponent of a fourth order polynomial of H:

$$\rho = \exp\left(a_0 + a_1H + a_2H^2 + a_3H^3 + a_4H^4\right) \tag{6.305}$$

The auxiliary variables for the polynomial are then set to:

$$f_{10} = a_2 + a_3H + a_4H^2 \tag{6.306}$$

$$f_{11} = a_1 + Hf_{10} \tag{6.307}$$

$$f_{12} = a_0 + H f_{11} \tag{6.308}$$

with which the recurrence relations are:

$$(f_{10})_k = a_3 (H)_k + a_4 \sum_{j=0}^{k} (H)_j (H)_{k-j} \tag{6.309}$$

$$(f_{11})_k = \sum_{j=0}^{k} (H)_j (f_{10})_{k-j} \tag{6.310}$$

$$(f_{12})_k = \sum_{j=0}^{k} (H)_j (f_{11})_{k-j} \tag{6.311}$$

$$(\rho)_k = \frac{1}{k} \sum_{j=1}^{k} j (f_{12})_j (\rho)_{k-j} \tag{6.312}$$

Note that the expressions for f_{10}, f_{11} and f_{12} were chosen such that the overall number of multiplications to compute ρ is minimized. One can expand this process to increase the order of the polynomial or decrease the order and remove variables.

6.5 Aerodynamics

Here, the recurrence relations of $C_{D,trim}$ and $C_{L,trim}$ are obtained.

$$\begin{aligned} C_{D,trim} = {} & a_{D,0} + a_{D,1}\alpha_A + a_{D,2}\alpha_A^2 + a_{D,3}\alpha_A^3 + a_{D,4}M + a_{D,5}M^2 \\ & + a_{D,6}\alpha_A M + a_{D,7}\frac{\alpha_A}{M} + a_{D,8}\frac{\alpha_A^2}{M} + a_{D,9}\frac{\alpha_A}{M^2} \end{aligned} \tag{6.313}$$

$$\begin{aligned} C_{L,trim} = {} & a_{L,0} + a_{L,1}M + \frac{a_{L,2}}{M + a_{L,3}} + \left(a_{L,4} + \frac{a_{L,5}}{M + a_{L,6}}\right) \sin\left(a_{L,7}\alpha_A + \frac{a_{L,8}\alpha_A}{M + a_{L,9}}\right) \\ & + \left(a_{L,10} + a_{L,11}M + \frac{a_{L,12}}{M + a_{L,13}}\right) \cos\left(a_{L,7}\alpha_A + \frac{a_{L,8}\alpha_A}{M + a_{L,9}}\right) \end{aligned} \tag{6.314}$$

For this, the following auxiliary variables are defined:

$$f_{13} = \alpha_A^2 \tag{6.315}$$

$$f_{14} = \frac{\alpha_A}{M} \tag{6.316}$$

$$f_{15} = \frac{f_{14}}{M} \tag{6.317}$$

$$f_{16} = \frac{a_{L,2}}{M + a_{L,3}} \tag{6.318}$$

$$f_{17} = \frac{a_{L,5}}{M + a_{L,6}} \tag{6.319}$$

$$f_{18} = \frac{a_{L,8}\alpha_A}{a_{L,7}(M + a_{L,9})} \tag{6.320}$$

$$f_{19} = \frac{a_{L,12}}{a_{L,11}(M + a_{L,13})} \tag{6.321}$$

$$f_{20} = \sin\left(a_{L,7}(\alpha_A + f_{18})\right) \tag{6.322}$$

$$f_{21} = \cos\left(a_{L,7}(\alpha_A + f_{18})\right) \tag{6.323}$$

These allow $C_{D,trim}$ and $C_{L,trim}$ to be written as:

$$\begin{aligned} C_{D,trim} = {} & a_{D,0} + a_{D,1}\alpha_A + \left(a_{D,2} + a_{D,3}\alpha_A\right) f_{13} + \left(a_{D,4} + a_{D,5}M + a_{D,6}\alpha_A\right) M \\ & + \left(a_{D,7} + a_{D,8}\alpha_A\right) f_{14} + a_{D,9} f_{15} \end{aligned} \tag{6.324}$$

$$C_{L,trim} = a_{L,0} + a_{L,1}M + f_{16} + \left(a_{L,4} + f_{17}\right) f_{20} + \left(a_{L,10} + a_{L,11}(M + f_{19})\right) f_{21} \tag{6.325}$$

The list of recurrence relations is then:

$$(f_{13})_k = \sum_{j=0}^{k} (\alpha_A)_j (\alpha_A)_{k-j} \tag{6.326}$$

$$(f_{14})_k = \frac{1}{(M)_0}\left[(\alpha_A)_k - \sum_{j=1}^{k} (M)_j (f_{14})_{k-j}\right] \tag{6.327}$$

$$(f_{15})_k = \frac{1}{(M)_0}\left[(f_{14})_k - \sum_{j=1}^{k} (M)_j (f_{15})_{k-j}\right] \tag{6.328}$$

$$(f_{16})_k = \frac{-1}{(M)_0 + a_{L,3}} \sum_{j=1}^{k} (M)_j (f_{16})_{k-j} \tag{6.329}$$

$$(f_{17})_k = \frac{-1}{(M)_0 + a_{L,6}} \sum_{j=1}^{k} (M)_j (f_{17})_{k-j} \tag{6.330}$$

$$(f_{18})_k = \frac{1}{(M)_0 + a_{L,9}}\left[\frac{a_{L,8}}{a_{L,7}}(\alpha_A)_k - \sum_{j=1}^{k} (M)_j (f_{18})_{k-j}\right] \tag{6.331}$$

$$(f_{19})_k = \frac{-1}{(M)_0 + a_{L,13}} \sum_{j=1}^{k} (M)_j (f_{19})_{k-j} \tag{6.332}$$

$$(f_{20})_k = \frac{a_{L,7}}{k} \sum_{j=1}^{k} j\left[(\alpha_A)_j + (f_{18})_j\right](f_{21})_{k-j} \tag{6.333}$$

$$(f_{21})_k = \frac{-a_{L,7}}{k} \sum_{j=1}^{k} j\left[(\alpha_A)_j + (f_{18})_j\right](f_{20})_{k-j} \tag{6.334}$$

$$\begin{aligned} (C_{D,trim})_k = {} & a_{D,1}(\alpha_A)_k + a_{D,2}(f_{13})_k + a_{D,4}(M)_k + a_{D,7}(f_{14})_k + a_{D,9}(f_{15})_k \\ & + \sum_{j=0}^{k} \Big\{ a_{D,3}(\alpha_A)_j (f_{13})_{k-j} + \left[a_{D,5}(M)_j + a_{D,6}(\alpha_A)_j\right](M)_{k-j} \\ & + a_{D,8}(\alpha_A)_j (f_{14})_{k-j} \Big\} \end{aligned} \tag{6.335}$$

$$\begin{aligned} (C_{L,trim})_k = {} & a_{L,1}(M)_k + (f_{16})_k + a_{L,4}(f_{20})_k + a_{L,10}(f_{21})_k \\ & + \sum_{j=0}^{k} \Big\{ (f_{17})_j (f_{20})_{k-j} + a_{L,11}\left[(M)_j + (f_{19})_j\right](f_{21})_{k-j} \Big\} \end{aligned} \tag{6.336}$$

6.6 Constraints

The final set of equations that have to be reshaped into recurrence relations are the constraint variables $\dot{q}_c$, n_g and q_{dyn}, given in Section 3.3:

$$\dot{q}_c = \frac{C_1}{\sqrt{R_N}} \sqrt{\rho} V_A^{3.15} \tag{6.337}$$

$$n_g = \frac{\sqrt{f_D^2 + f_S^2 + f_L^2}}{g_0} \tag{6.338}$$

$$q_{dyn} = \frac{1}{2} \rho V_A^2 \tag{6.339}$$

The needed auxiliary variables for $\dot{q}_c$ are:

$$f_{22} = \sqrt{\rho} \tag{6.340}$$

$$f_{23} = V_A^{3.15} \tag{6.341}$$

The auxiliary variable for n_g is:

$$f_{24} = f_D^2 + f_S^2 + f_L^2 \tag{6.342}$$

Note that this variable may also be computed with another set of 3 orthogonal aerodynamic accelerations (for instance, the f_x, f_y and f_z given in Eq. (6.22)). The final auxiliary variable is:

$$f_{25} = V_A^2 \tag{6.343}$$

Note that, at this point, this variable will already have been computed for each state-variable sets. The recurrence relations are:

$$(f_{22})_k = \frac{1}{(\rho)_0} \sum_{j=1}^{k} \left[\left(\frac{3j}{2k} - 1 \right) (\rho)_j (f_{22})_{k-j} \right] \tag{6.344}$$

$$(f_{23})_k = \frac{1}{(V_A)_0} \sum_{j=1}^{k} \left[\left(\frac{4.15j}{k} - 1 \right) (V_A)_j (f_{23})_{k-j} \right] \tag{6.345}$$

$$(f_{24})_k = \sum_{j=0}^{k} \left[(f_D)_j (f_D)_{k-j} + (f_S)_j (f_S)_{k-j} + (f_L)_j (f_L)_{k-j} \right] \tag{6.346}$$

$$(f_{25})_k = \sum_{j=0}^{k} (V_A)_j (V_A)_{k-j} \tag{6.347}$$

$$(\dot{q}_c)_k = \frac{C_1}{\sqrt{R_N}} \sum_{j=0}^{k} (f_{22})_k (f_{23})_{k-j} \tag{6.348}$$

$$(n_g)_k = \frac{1}{(f_{24})_0} \sum_{j=1}^{k} \left[\left(\frac{3j}{2k} - 1 \right) (f_{24})_j (n_g)_{k-j} \right] \tag{6.349}$$

$$(q_{dyn})_k = \frac{1}{2} \sum_{j=0}^{k} (\rho)_j (f_{25})_{k-j} \tag{6.350}$$

Chapter 7

Software

This chapter will provide an overview of the software that was used. In Section 7.1, the used external software packages are discussed and in Section 7.2 the custom software is discussed. Finally, Section 7.3 will discuss the verification and validation of various routines. All custom code is written in C++, unless otherwise mentioned.

7.1 External Software

Not all software has to be programmed, as some tools are already available from external sources. This section is dedicated to the description of the used external tools, being TU Delft Astrodynamics Toolbox (Tudat), Eigen, Boost, PaGMO, and the software of GRAM 1999.

7.1.1 Tudat, Eigen and Boost

Tudat is a toolbox containing a number of C++ libraries for astrodynamics simulations. It is developed and maintained by students and staff of the Astrodynamics & Space Missions chair at the Faculty of Aerospace Engineering of the Delft University of Technology. The libraries in this toolbox can be used for a wide range of subjects, including reentry and are still being improved upon and expanded with new codes. For Tudat, there are Working Group Meetings that take place every two weeks, during which users can ask questions to developers and other users. Furthermore, the forum and wiki of Tudat and other users of Tudat can also be a source of information.[1]

Tudat contains the libraries Tudat and Tudat Core, and is packaged together with the external libraries Boost and Eigen. CMake is used to compile the codes of Tudat. The Tudat core library contains the RK4 integrator, whereas the Tudat library contains the blueprint for variable-step RK integrators. Although it is not mentioned on the website, inspection of the C++ files of Tudat revealed that it also contains a build-in version of the RKF5(6) integrator, in the form of the class *RungeKuttaVariableStepSizeIntegrator* and the list of coefficients for RKF5(6). Tudat further comes with a number of unit tests with which the integrators have been tested.

The integrators use the matrix classes from the Eigen library for the state vectors. Furthermore, Eigen is used for matrix multiplications in the case of reference-frame transformations and for SVD when performing least-squares regression (see Section 4.3 for an explanation of SVD).[2]

Of Boost, the libraries Boost Chrono and Boost Random were used. Boost Chrono is a library that provided different clocks that can be used to time C++ programs. The used clock is the

[1] The wiki of Tudat can be found at `tudat.tudelft.nl/projects/tudat/wiki`.
[2] The documentation of Eigen can be found at `eigen.tuxfamily.org/dox`.

process_cpu_clock, which is designed to measure only the CPU time of a process, without the overhead time of other processes. It was found through trial and error that this clock has a resolution of about 1 ms. Boost Random contains several random number generators, of which the generator *mt19937* was used. This generator has a average speed compared to the others and has a cycle length of 2^{19937}-1. The main reason for using this generator is that this one is advised by the documentation of Boost and it is also used by PaGMO [29].[3]

7.1.2 PaGMO

PaGMO, Parallel Global Multi-objective Optimizer, is a C++ library containing the codes for a number of state-of-the-art numerical optimization methods. It was developed by the European Space Agency and was initially meant for the optimization of interplanetary and other spacecraft trajectories. One of the advantages of PaGMO is that it can perform parallel computing, i.e., it can compute the objective function for multiple individuals at once by utilizing multiple processor cores (assuming the user is in the possession of a computer with multiple cores, but for most modern day computers, this is the case). The PaGMO package includes codes for DE, NSGA-II and MOEA/D [29], so these methods do not have to be developed.

Due to compiler incompatibilities, the author was unable to use the PaGMO package itself and instead, the codes for the different optimization techniques were adapted from PaGMO and made into a custom C++ class. This is further discussed in Section 7.2.4.

7.1.3 GRAM 1999

GRAM 1999 is an atmosphere model written in FORTRAN 77, whereas the other software is written in C++. Fortunately, it is possible to have C++ call FORTRAN routines and send variables back and forth between the two languages. The GRAM 1999 package consists a number of source code files and some data files for the models used by GRAM. One of the models that has its own data files is the GUACA model (see Section 2.4.3 for a description of the different models used by GRAM), of which only the data files are included for the month January; the complete set of GUACA data has to be ordered separately. Furthermore, it was chosen not to include the RRA model, as it only provides data near the RRA sites, which is a feature that is only useful in case one wants to simulate a landing at a site with RRA data. GRAM 1999 is provided with a sample code "gramtraj.f", which shows how to incorporate GRAM 1999 into a trajectory calculation program. The interfaces with GRAM 1999 were based on this sample code. GRAM 1999 was only used as wind model and was altered for this purpose, which is explained in Section 7.2.3.

7.2 Custom Software

As was mentioned at the beginning of this chapter, C++ is the programming language of choice, but Matlab will be used to plot and check the results, mainly because of the user-friendly tools for plotting included in Matlab. The results from the C++ codes will be stored in data files and these will be read by custom Matlab routines, so that they can be plotted and analyzed.

[3] The documentation of Boost can be found at `www.boost.org/doc`.

The custom programs discussed in this section are the optimization framework for the Runge-Kutta integrators, TSI and the wind model (which is based on GRAM 1999).

7.2.1 Runge-Kutta

As mentioned in Section 7.1.1, the Runge-Kutta integrators are present as a class in Tudat, so they do not have to be developed from scratch. That just leaves the task of creating a framework for using the integrator during a numerical optimization or sensitivity analysis. For this, a custom class called "controller" is created, which contains the methods that compute the equations of motion, and those in turn can be integrated by the RK integrator class. This class also contains methods that take a control vector and integrate the trajectory it defines and output one or more objective function values. The flowchart of this method is shown in Figure 7.1.

Before the optimization, the user gives a number of initial settings to the controller, such as the state-variable set, initial conditions and initial step size to which the integrator should be reset at the start of each integration. The computation of objective function values then starts with the numerical optimizer calling the controller's objective function method with a control vector. The controller then creates the control profile using Hermite splines. Next, the integrator is called, which integrates the equations of motion. These, in turn, use the earlier created control profile. When the terminal constraints are reached, the integration is ended and the objective-function values can be computed and given back to the numerical optimizer.

In case of a sensitivity analysis, different inputs can be varied, such as the controls or initial conditions and it is possible to include a wind model to compute the effect of wind disturbances (see Section 8.4 for a complete description of the sensitivity analyses). In the latter case, the communication with the wind model will be as explained in Section 7.2.3. In Figure 7.1, the wind model is connected by dashed lines, since it will not be used during optimization, because wind is random and one cannot optimize random events.

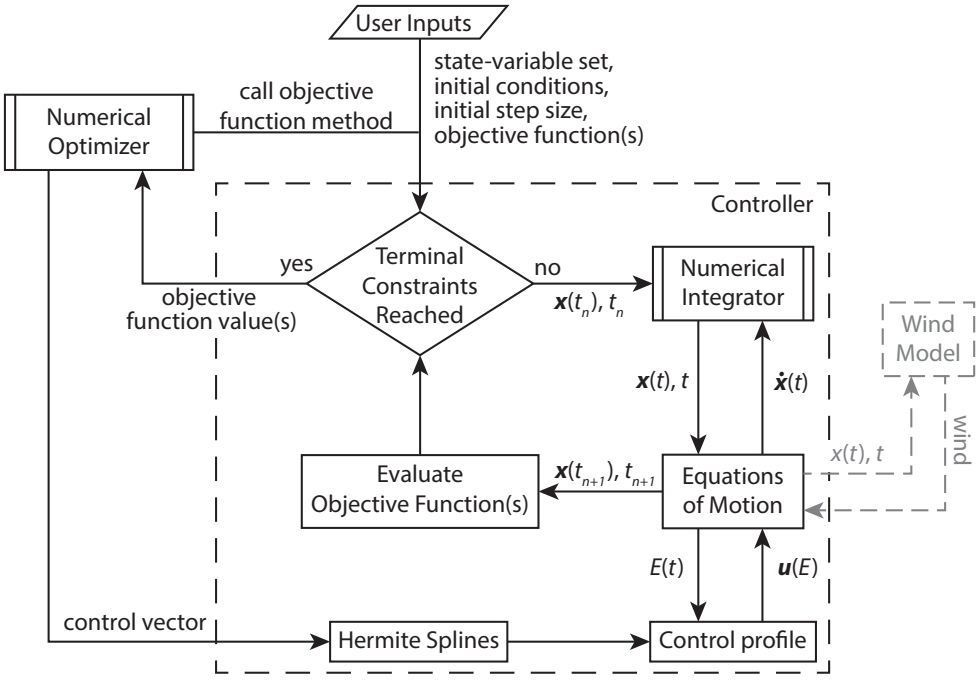


Figure 7.1: Flowchart of the Runge-Kutta integrators within an optimization routine

7.2.2 Taylor Series Integration

The general flowchart for TSI during an optimization is identical to that of the RK integrators shown in Figure 7.1, although now the blocks "Numerical Integrator" and "Equations of Motion" are combined into a single TSI class. The flowchart for this case is shown in Figure 7.2.

For each state-variable set, two classes will be made; one for the equations of motion with wind and one without them. These classes will be derivatives of a main class that contains the general methods, such as the order control and updating of the state variables. The order control is called at the start of each step size to set the order for this step. Then the auxiliary variables and Taylor coefficients are generated. Note that TSI requests the spline coefficients for the current interval, rather than just the control values. The step-size control will compute a step size based on an error estimation, after which this step size is reduced in case of a discontinuity during the current step (as was discussed in Section 5.5). The summation of the Taylor series of the state is implemented using the Horner scheme, which is the "classic" numerical way of evaluating a high-order polynomial [24]:

$$
\begin{aligned}
&x_{n+1} = (x_n)_K \\
&\text{for } i = (K-1) : -1 : 0 \\
&\quad x_{n+1} = (x_n)_i + x_{n+1} h \\
&\text{endfor}
\end{aligned}
$$

TSI also has the option to output the entire list of Taylor coefficients of the state, which can for instance be used to interpolate the resulting trajectory. When the sensitivity analysis to wind is performed, the interface between the "Generate Auxiliary Variables" block and the wind model will be as explained in Section 7.2.3.

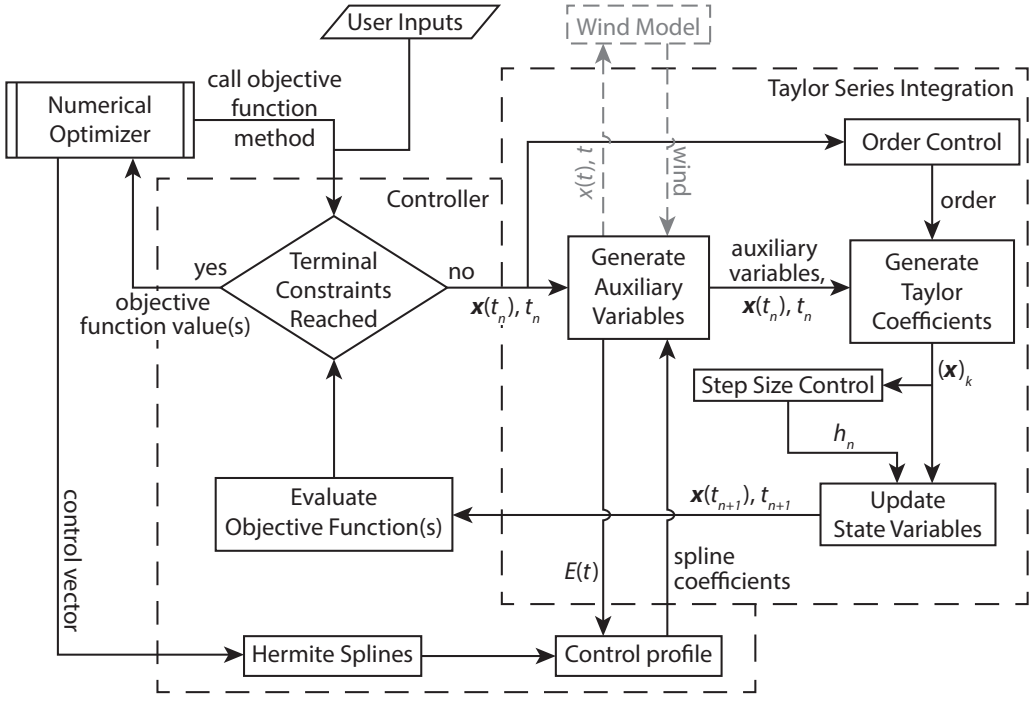


Figure 7.2: Flowchart of the Taylor Series Integration programs within an optimization routine

7.2.3 Wind Model

Since GRAM was written in FORTRAN 77, so will be the wind-model program that distills the wind data from GRAM and sends it to the C++ programs. As was mentioned in Section 7.1.3, only the GUACA data of January are available, so for simulation, only dates in January 1999 will be used.

Using an unaltered version of GRAM 1999 as a wind model for trajectory integration will lead to a number of problems. The first cause for problems originates from the way GRAM computes random wind. The random wind is not fixed for a specific location at a specific point in time, hence if one were to move from a location A to location B and then back to A again, one would get two different wind vectors at A, even when the time at which the wind is sampled is the same. This is a problem for RK integrators, as they sample certain points in time multiple times. RKF5(6), for instance, samples the point $t = t_n$ twice (see the second column of Table 4.2, where $\alpha_k = 0$ appears twice). Also, RKF5(6) may reject a time step and recompute it, leading to different winds and thus a different trajectory. The chosen solution to this problem is to sample the wind prior to integration and create a wind profile along the trajectory as a function of altitude only. The wind model then takes an altitude as input and returns the wind for that altitude.

The randomness of GRAM also means that the altitudes at which the wind is sample to create the profile will input the shape of this profile. An example of this is shown in Figure 7.3.

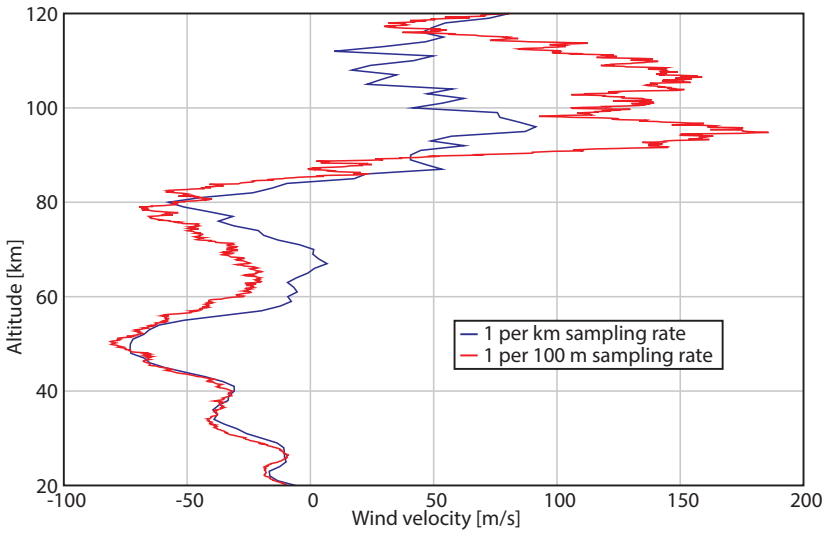


Figure 7.3: Sampling the wind of GRAM 1999 at two different rates.

Here, the wind is sampled once per kilometer and once per 100 meter for the same random seed, yielding two different profiles. Another issue of the random wind is the spikiness of the profile. Real wind is assumed to be more smooth, as is also suggested by the data shown in [35]. Furthermore, it is not possible to approximate spiky data well with least-squares regression, which would make the method suggested in Section 5.6.3 ineffective. It has therefore been decided to smooth the data. Note from Figure 7.3 that the spikiness is not decreased by increasing the sampling rate, thus the wind profile was first smoothed using a weighted moving average. The weights w_i for this are obtained from the build-in tri-cubic weight function of

Matlab, given by:

$$w_i = \frac{\left[1 - \left(\frac{d_i}{\max(d)}\right)^3\right]^3}{\sum_{j=1}^{n}\left[1 - \left(\frac{d_j}{\max(d)}\right)^3\right]^3} \tag{7.1}$$

where d_i is is the distance (in altitude) between the i^{th} point and the current point that is smoothed. $\max(d)$ is then the maximum of these distances. $n = 7$ was used for this smoothing, so point $i = 4$ is the point that is smoothed (so $d_4 = 0$) and since GRAM was sampled with an equal spacing, the weights for the points 1 and 7 are zero. Note that near the borders of the domain, some of the points are outside the domain and these are then not taken into account. To also have smooth results in between the data points, a spline was fitted through the data and this spline forms the wind profile that is used for the trajectory integrations. Some raw GRAM wind data and the smoothed spline through this data are shown in the left graph of Figure 7.4.

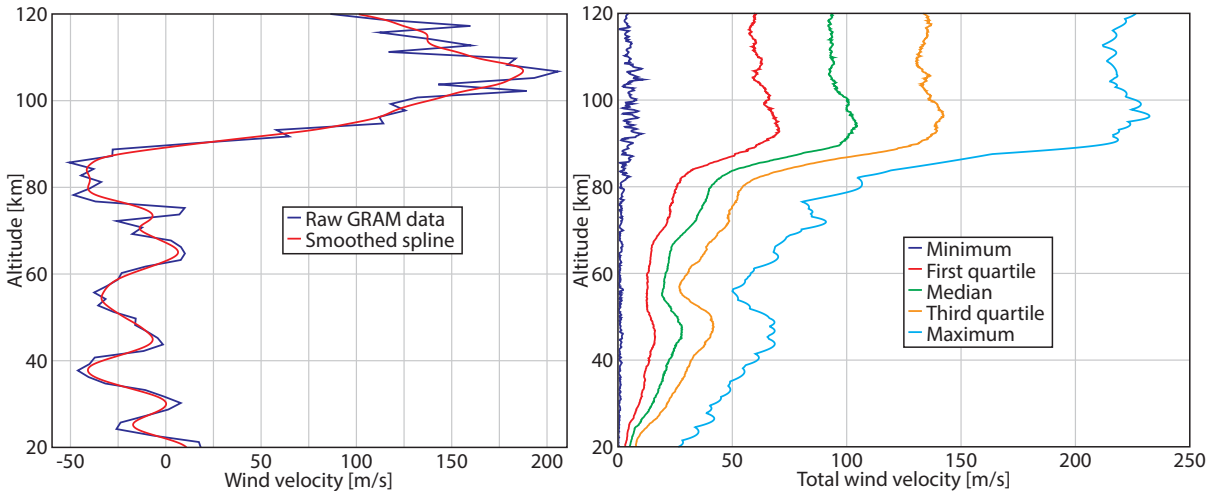


Figure 7.4: Left: smoothing and spline fitting of GRAM generated wind. Right: distribution of the total wind velocity of the wind model for 1000 random samples.

The net effect of the smoothing of the wind model on the resulting trajectory is small, as the large differences in wind velocity with the raw data in the upper regions of the atmosphere cause negligible effects, because the air density there is low. At lower altitudes, the differences are less than 15 m/s, which is small compared to the velocity of the vehicle (which is always larger than 750 m/s). The smoothing does allow the integrators to make larger time steps.

Finally, the data in [35] was used as a sanity check for the wind model. There, it is stated that over 60% of the observations have wind velocities of 100 m/s and few of the observations exceed 200 m/s. The latter was violated badly in GRAM 1999, as total wind velocities could often exceed 200 m/s and could reach 280 m/s. To fix this, the wind velocities were reduced by a factor η when the total wind V_w exceeds 190 m/s:

$$\eta = \frac{V_w + 190}{2V_w} \tag{7.2}$$

The distribution of 1000 random wind profiles is shown in the right graph of Figure 7.4 (first quartile means 25% of the data is smaller than or equal to this line and third quartile means 75%). As can be seen, slightly over 50% of the profiles exceeds 100 m/s and about 3% exceed 200 m/s, which are percentages that correspond quite well with the data of [35]. Furthermore,

in [35], the maximum wind velocities occur between 100 and 110 km altitude, whereas here, they occur between 90 and 100 km. This difference was accepted as fixing this requires modifying the internal code of GRAM. Finally, the graphs in [35] show oscillations in the wind velocities with a minimum half-period of slightly below 2 km, so the sampling rate of GRAM was set to 2 km to mimic this behavior.

For the interface with RKF5(6), the wind model simply returns the wind velocities for a given altitude, but an extra option was added to return the spline coefficients, in case they are required by TSI. The ways TSI obtains the Taylor coefficients of the wind velocities was discussed in Section 5.6.3.

7.2.4 Optimization

In Section 7.1.2, it was mentioned that the optimization routines were adapted from the PaGMO source code into a custom class. This class contains a population of individuals for the optimization and the methods to optimize this population for a given objective function. The methods are DE for single-objective optimization, and NSGA-II with DE and MOEA/D for multi-objective optimization. These methods will optimize for a user specified number of generations. The optimization techniques themselves were explained in Section 4.5. The inputs for the class are the bounds for each of the control parameters (e.g., E should be between E_{min} and E_{max}), the population size and the parameters for each method. These parameters are, unless otherwise specified, set to the default values of PaGMO, which are given in Table 7.1. Note that, in this table, m marks the number of control variables and N_p is the population size. Furthermore, this class can output the current population to a file, together with the objective function values for each individual.

Table 7.1: Default values for optimization parameters

Parameter	Symbol	Value
Scaling factor DE	F	0.8
Crossover constant DE	CR	0.9
Mutation chance	p	$1/m$
Mutation distribution index	η	11
MOEA/D neighborhood size	T	$N_p/10$
MOEA/D maximum number of replacements	N_r	2

7.3 Verification and Validation

Before numerical tools are used, they have to be verified, to assess whether they have been correctly implemented, which is done in Section 7.3.1. Then, in Section 7.3.2, the fits of the US76 atmosphere model are verified. Also, the equations of motion should be validated to check whether they represent reality, which is done in Section 7.3.3. The validation of the wind model was already done in Section 7.2.3.

7.3.1 Verification of the Numerical Methods

The numerical methods that are verified here are those for the (implementation of the) recurrence relations, root-finding, spline interpolation, least-squares regression, numerical inte-

gration, and optimization. For the recurrence relations, root-finding, spline interpolation and least-squares regression, no graphs are shown of the verification process, as they are required to reproduce the verification data exactly (within the limits of the double-precision floating-point representation). It was deemed not informative to show graphs of two line overlapping almost exactly. Furthermore, the number of verification cases for the recurrence relations and root-finding methods is too large to show any representative graphs of.

Recurrence Relations

The implementation of the recurrence relations were first verified individually, since their large quantity makes it hard to determine the cause of errors in the integration. The way this was done was by removing the impact of this particular variable on the equations of motion (i.e., if the variable was part of the computation of the aerodynamic forces, remove the aerodynamic forces from the equations of motion). Then one can look whether the Taylor series of this variable accurately describes its value at the end of an integration step (which is computed from the state at the end of the step).

Root-Finding

The root-finding methods for TSI were manually checked for large batches of integrations to see whether they would trigger when needed, whether they actually converge to the root and how many iterations were needed to achieve convergence. The root-finding methods work better the smaller the step-size, hence they were checked with a low error tolerance of $1 \cdot 10^{-8}$ for orders up to 18. In that case they still reached convergence in less than 6 iterations, which is the chosen maximum number of iterations.

Spline Interpolation and Least-Squares Regression

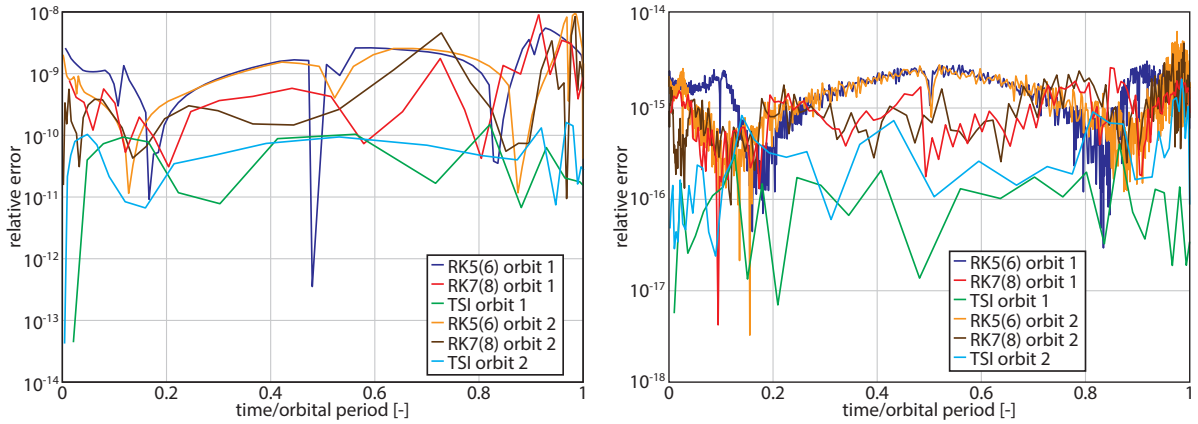
The implementation of splines and least-squares regression in C++ and Fortran was verified using the build-in functions of Matlab. The relative errors made were in the order of $1 \cdot 10^{-15}$ or less, which is about equal to the rounding error of double precision floating point representation and thus negligible.

Numerical Integrators

Next, the integrators were verified. Note that for TSI, time-step controller 1 with a safety factor of 0.95 was used, as this was selected in 8.1.1.

Firstly, TSI, RKF5(6) and RKF7(8) were verified using a 2D Kepler orbit problem. The reason for verifying RKF7(8) is that it will be used in Section 8.1.1 to compute the errors made by the step-size controllers of TSI. The equations and variables for a 2D Kepler orbit are discussed in Appendix C. This kind of orbits is defined by a semi-major axis a and an eccentricity e. The errors made by the integrators were checked for two different orbits, one with $a = 7 \cdot 10^6$ m (Earth radius plus about 522 km) and $e = 0.5$ and one with $a = 3 \cdot 10^7$ m and $e = 0.8$. The integrators were checked with an error tolerance ϵ (both relative and absolute) of $1 \cdot 10^{-8}$ and $1 \cdot 10^{-14}$. The results are given in Figure 7.5, in which per time step the maximum of relative errors in radius and velocity is plotted (note that some lines have missing parts when the error is 0, because of the logarithmic scale of the y-axis). After each time step, the state vectors

were reset back to the correct values, so that the error accumulation over multiple time steps is not taken into account. As can be seen, the error tolerances were not violated by any of the integrators, which means they are verified. Note that the results for $\epsilon = 1 \cdot 10^{-14}$ are shown, as this was the lowest tolerance all integrators could satisfy. TSI turned out to also be successful at integrating with $\epsilon = 1 \cdot 10^{-15}$, but RKF5(6) and RKF7(8) then had errors of up to $5 \cdot 10^{-15}$ and $4 \cdot 10^{-15}$, respectively. The errors made in that case are still at all time steps lower than for $\epsilon = 1 \cdot 10^{-14}$. $\epsilon = 1 \cdot 10^{-15}$ can thus be used to obtain more accurate results. Lowering the tolerance further had no positive effect on the size of the errors.



(a) $\epsilon = 1 \cdot 10^{-8}$ (b) $\epsilon = 1 \cdot 10^{-14}$

Figure 7.5: Maximum of the relative errors made in radius and velocity by the integrators.

Optimization

Here, the optimization methods Differential Evolution, NSGA-II and MOEAD, are verified.

DE was verified with the partial Himmelblau function and the modified Deb-Tan Function. The Himmelblau function is given by [63]:

$$f(u_1, u_2) = (u_1^2 + u_2 - 11)^2 + (u_1 + u_2^2 - 7)^2 \tag{7.3}$$

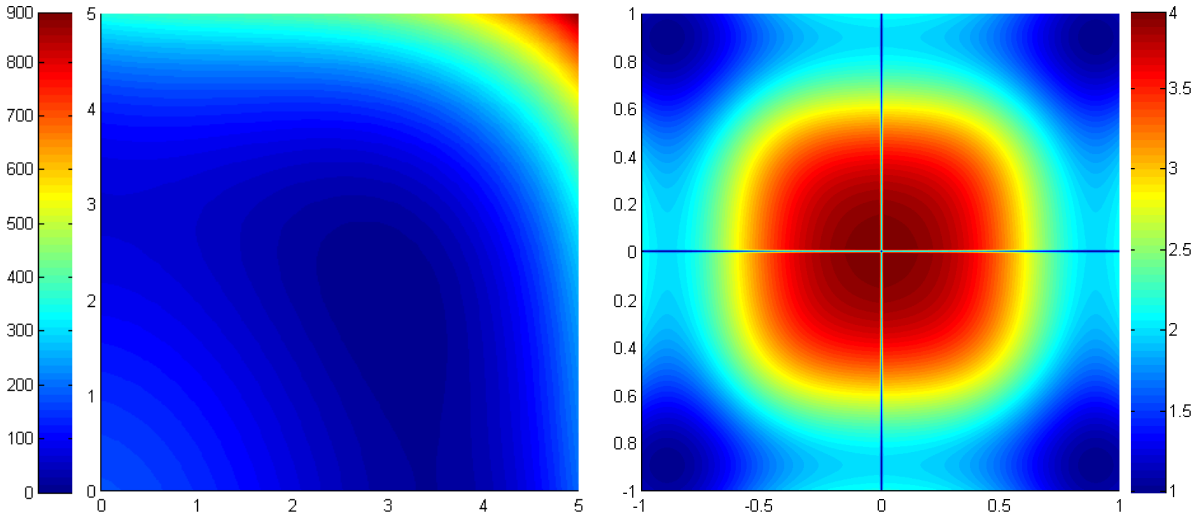
with $0 \leq u_1, u_2 \leq 5$. The Himmelblau function for this range is shown in Figure 7.6(a). Himmelblau function normally has four global optima, but within the range of u_1 and u_2, it only has one, located at (3,2) equal to 0. This function not very difficult for DE, as even with a small population of 10 individuals, the optimum is found in all of the runs, as long as F is set larger than 0.5.

The modified 2-dimensional Deb-Tan function is given by [63]:

$$\begin{aligned} g_1(u_1, u_2) &= 2 - \exp\left(-\frac{u_1^2}{1.6 \cdot 10^{-5}}\right) - \exp\left(-\frac{(|u_1| - 0.9)^2}{0.16}\right) \\ g_2(u_1, u_2) &= 2 - \exp\left(-\frac{u_2^2}{1.6 \cdot 10^{-5}}\right) - \exp\left(-\frac{(|u_2| - 0.9)^2}{0.16}\right) \\ f(u_1, u_2) &= g_1 g_2 \end{aligned} \tag{7.4}$$

with $-1 \leq u_1, u_2 \leq 1$. This function is shown in Figure 7.6(b) and has one global optimum at (0,0) and eight local optima at (-0.9,-0.9), (-0.9,0), (-0.9,0.9), (0,-0.9), (0,0.9), (0.9,-0.9), (0.9,0)

and (0.9,0.9). It is far easier for DE to reach the local minima in the corners than reaching the other optima; the "trenches" in the graph have a width of less than 2.5% of the complete graph width. This means that a random sampling would have about 0.0625% chance to find the global optimum. In Figure 7.7, the cross-section of the trench for $u_2 = 0$ is plotted. When using the standard F of 0.8, a population size of at least 80 was needed to find the global optimum. Note that the optima are in a square grid and the corner optima are the easiest to find, so using $F = 0.5$ can guide DE from these optima to the others, as long as there are individuals at each of the optima. Then, population sizes of about 40 can find the global optimum in most of the simulations. Overall, DE managed to find the optima of both problems, so its functionality is verified.



(a) Himmelblau function (b) Modified 2D Deb-Tan function

Figure 7.6: Verification problems for single-objective optimization (colors indicate the objective values).

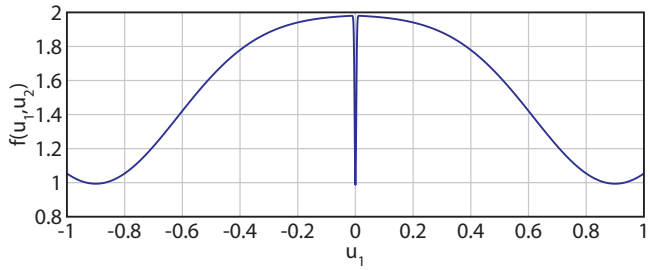


Figure 7.7: Trench of the modified Deb-Tan function for $u_2 = 0$.

NSGA-II and MOEAD were verified with the "discrete Pareto-optimal front" and "convex Pareto-optimal front"-problems from [15]. The "discrete Pareto-optimal front"-problem has a Pareto set that consists of four separate parts, as can be seen in Figure 7.8(a). The objectives f_1 and f_2 of this problem are given by:

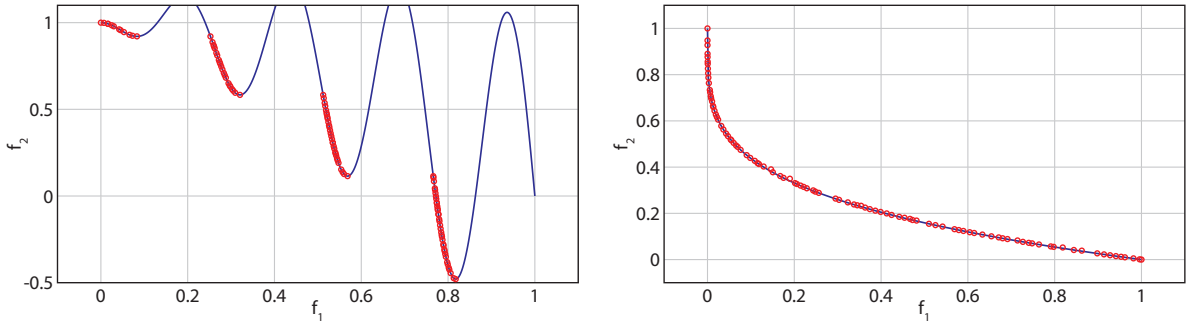
$$\begin{aligned} f_1(u_1, u_2) &= u_1 \\ g &= 1 + 10u_2 \\ f_2(u_1, u_2) &= g - \frac{u_1^2}{g} - u_1 \sin(8\pi u_1) \end{aligned} \tag{7.5}$$

with $0 \leq u_1, u_2 \leq 1$. Note that the goal of the optimization methods is to locate all parts of the Pareto set and have solutions that are approximately evenly distributed over the entire set. The results of NSGA-II have been shown in Figure 7.8(a), together with the line for $g = 1$, which is a condition for the Pareto set. MOEAD also managed to find the entire Pareto set with an evenly distributed population (with both approaches) and hence its results are not shown here. For both methods, $N_p = 100$ and 1000 generations were used.

The "convex Pareto-optimal front"-problem is given by [15]:

$$
\begin{aligned}
f_1(u_1, u_2) &= 4u_1 \\
\text{if } u_2 \leq 0.4 \quad \text{then} \quad & g = 4 - 3\exp\left(-\tfrac{(u_2-0.2)^2}{4\cdot 10^4}\right) \\
\text{else} \quad & g = 4 - 2\exp\left(-\tfrac{(u_2-0.7)^2}{0.04}\right) \\
\alpha &= 0.25 + 3.75(g-1) \\
\text{if } f_1 \leq g \quad \text{then} \quad & f_2(u_1, u_2) = g - g\left(\tfrac{f_1}{g}\right)^{\alpha} \\
\text{else} \quad & f_2(u_1, u_2) = 0
\end{aligned}
\tag{7.6}
$$

The results for NSGA-II for this problem are shown in Figure 7.8(b) and the results for MOEAD were shown in Figure 4.8. For both methods, $N_p = 100$ and 1000 generations were used. The resulting population for NSGA-II is less evenly distributed than for MOEAD (with convex Tchebycheff approach), but since it found the entire Pareto set for both verification problems, just like MOEAD, both methods are considered verified.



(a) Discrete Pareto-optimal front

(b) Convex Pareto-optimal front

Figure 7.8: Solution space of the verification problems for multi-objective optimization.

7.3.2 Verification of the US76 Fits

In Section 5.6.1, the tables of the US76 model were fitted with different equations. Here, the impact of using these fits, rather than the table values, is assessed by integrating HORUS' reference trajectory with RKF5(6), set to an error tolerance of $1 \cdot 10^{-14}$. The resulting differences in altitude and velocity are shown in Figure 7.9. Note that the tables are only used for altitudes above 86 km and this altitude is reached after about 203 seconds. Furthermore, the goal of the fits was to have relative errors of no more $1 \cdot 10^{-4}$ in terms of air density and molecular mass and, as can be seen in Figure 7.9, the resulting errors in the trajectory are no more than $3 \cdot 10^{-5}$. Thus, it is concluded that using the fits of the US76 tables does not cause significant errors.

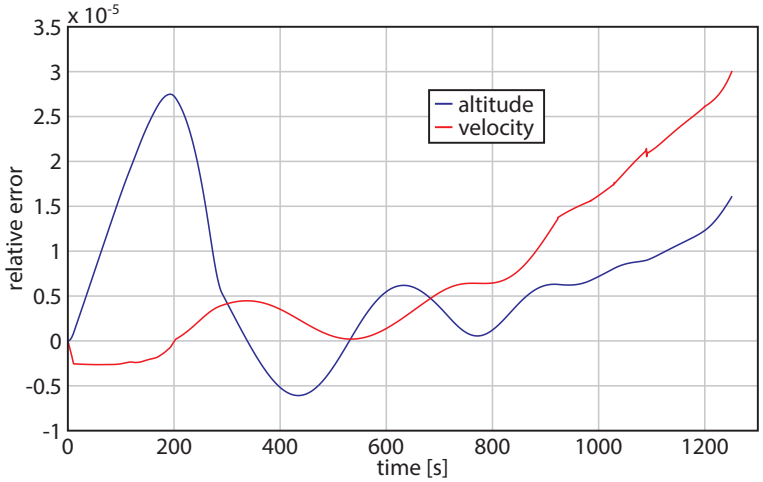


Figure 7.9: Relative differences in H and V when using the fits of US76, instead of the table values.

7.3.3 Validation of the Equations of Motion

The equations of motion are validated in two parts; first the reentry equations without wind are validated and after that the equations of wind.

Reentry Equations

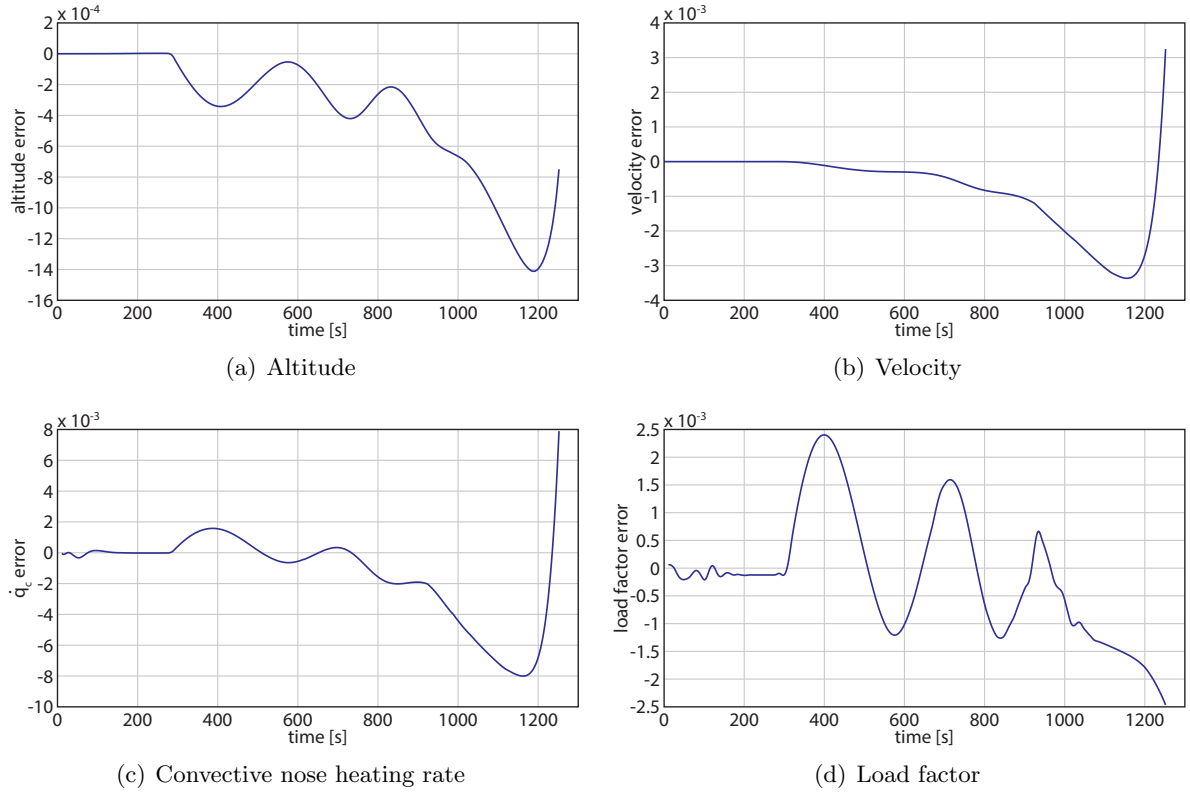
Validation of the reentry equations of motion was not possible with real-life data, because there is non available (HORUS never flew). Instead, a reference trajectory provided by E. Mooij that was made with the reentry simulator START is used [44]. This trajectory is the standard reference trajectory for HORUS, for which the data and control profile are given in Table 3.1 and Figure 3.2. The reference trajectory was simulated with the aerodynamic fits developed in Section 5.6.2 and uses the US76 atmosphere model, but there is also a number of differences for this trajectory compared to the way reentry simulation has been discussed in this report [45]:

- A spherically shaped Earth was used, rather than an ellipsoid shape.
- The controls are defined as function of time, rather than energy
- The air density for layer 10 of US76 is zero.

Furthermore, for RKF5(6), the fits of US76 were used that were developed in Section 5.6.1, the order of TSI was fixed to 18 and $\epsilon = 1 \cdot 10^{-12}$ for both integrators. The relative errors w.r.t. the reference trajectory are shown in Figure 7.10. In each of these graphs, only one line is shown, because the lines for RKF5(6) and TSI nearly overlap. As can be seen, these errors have an order magnitude of $1 \cdot 10^{-3}$ for the entire trajectory, hence the reentry equations of motion are considered to be validated.

Wind Equations

There was no reference trajectory available with wind, so instead the implementation of the wind equations (the equations for obtaining the airspeed-based angles from the groundspeed angles and a local wind vector, see Section 2.7) were verified and validated with manually computed solutions. These cases are shown in Table 7.2. The first case is the seemingly trivial



(a) Altitude

(b) Velocity

(c) Convective nose heating rate

(d) Load factor

Figure 7.10: Relative errors made by RKF5(6) and TSI w.r.t. the reference trajectory of HORUS.

case of no wind, where the airspeed angles should match the groundspeed angles. After that, the implementation was checked for cases of horizontal flight and pure side-wind, where only β should change, and wind from below, where only α should change. Then, the equations were tested for a few complex cases, where the "manual" solutions were computed using matrix operations in Matlab. In the end, all results matched, so the equations for the airspeed-based angles are verified.

Table 7.2: Verification cases for the equations used to obtain the airspeed-based angles.

groundspeed angles [deg]			groundspeed components [m/s]			wind velocity components [m/s]			airspeed angles [deg]		
α_G	β_G	σ_G	$u_{G,V}$	$v_{G,V}$	$w_{G,V}$	$u_{W,V}$	$v_{W,V}$	$w_{W,V}$	α_A	β_A	σ_A
5	0	25	0	7500	0	0	0	0	5	0	25
0	0	0	0	7500	0	1000	0	0	0	7.5946	0
0	0	0	0	7500	0	0	0	-1000	7.5946	0	0
0	0	90	0	7500	0	-500	0	0	-3.8141	0	90
0	0	90	-500	7000	1000	-500	0	0	-3.9550	0.0398	89.7206
25	0	45	-500	7000	1000	-500	400	200	20.6384	-2.2429	44.8169

The final verification was a comparison of the integrations of TSI and RKF7(8), similar to the procedure in Section 8.1.1 (see that section for the reasoning behind the use of RKF7(8)). TSI made use of the spline coefficients of the wind model and RKF7(8) was set to $\epsilon = 1 \cdot 10^{-14}$. Furthermore, all discontinuities in the trajectories were disabled and the differences between the results of the integrators were examined per step. The resulting graphs of the errors are

not shown here, as they are considered to give little useful information; when TSI is set to error tolerances of $1 \cdot 10^{-8}$ to $1 \cdot 10^{-12}$, the errors were always smaller than the respective tolerance.

Chapter 8

Simulation Results

Now that the software has been selected/build and verified, it can be used to obtain the simulation results for this thesis. The simulations were performed on a HP® EliteBook 8540w, with an Intel® Core™ i7 clocked at 1.60 GHz. The optimal settings for TSI are determined in Section 8.1. Next the optimal version of TSI is compared with RKF5(6) for integration speed in Section 8.2. The results of the integrators are compared for optimization in Section 8.3 and for sensitivity analysis in Section 8.4.

Unless otherwise mentioned in the text, in each section, the equations of motion without wind will be used. Each integration will start at the initial conditions of HORUS' reference mission. Unless otherwise mentioned, integration is stopped when i) the terminal distance is reached, ii) the vehicle is no lower heading for the target (i.e., when the first derivative of distance to target is positive) or iii) the altitude is larger than 122 km, at which point it is assumed the vehicle has skipped out of the atmosphere. Furthermore, in Sections 8.1 and 8.2, the integrations are stopped when the energy is less than the terminal energy ($5.219 \cdot 10^5$ J, computed in Section 3.2.1), whereas in Sections 8.3 and 8.4, a range of possible terminal energy values may be used.

8.1 Optimal Settings for Taylor Series Integration

In this section, the optimal settings for TSI are determined. Firstly, in Section 8.1.1, a step-size controller is selected for TSI. Secondly, in Section 8.1.2, TSI is timed for different state-variable sets and order-control strategies to determine which combination yields the lowest computational load. Then, in Section 8.1.3, the optimal state-variable set is selected and finally, in Section 8.1.4, the optimal settings are determined for TSI in case of wind.

8.1.1 Step-size Control

In this section, the errors made by TSI when using the different step-size controllers of Section 5.4.1 are presented and analyzed. For this error analysis, the result of TSI was compared with that of RKF7(8). For each time step of TSI, the equations of motion were also integrated with RK7(8), starting at the same point in time and with the same state. RKF7(8) was set to an error tolerance of $1 \cdot 10^{-15}$, which in Section 7.3.1 was found to give the lowest errors. The reason for using RKF7(8) is that it is the highest-order RKF integrator in the Tudat package. The higher order of this integrator allows it to make larger steps than RKF5(6). This means that for each step of TSI, the number of time steps of RKF7(8) can be lower than that of RKF5(6) and this means that RKF7(8) has less error accumulation over multiple steps, which allows it to better serve as "real" solution.

Furthermore, the simulations were done with equations of motion without discontinuities. Using the reasoning of Section 5.5, TSI will make errors when integrating over a discontinuity and since these errors are not directly due to too large step sizes, discontinuities would cloud the results. To eliminate all discontinuities, the exponential-atmosphere model was used and the Mach number has been artificially fixed to 10 when computing the aerodynamic coefficients. The controls must be defined by a single polynomial, which was done by specifying the control angles and their first order derivatives at E_{min} and E_{max} and directly feeding them to the Hermite spline interpolator through Eq. (4.26) to (4.30).

To generate different random trajectories, these control inputs were randomized (in such a way that the control limits of HORUS were never violated), together with the initial conditions, which were varied from the nominal HORUS mission values given in Table 3.1. H_0 was randomly set between 110 and 122 km, V_0 between 7 and 7.5 km/s, γ_0 between -3° and -0.5°, δ_0 between -45° and 0° (note: there is no point in including the Northern Hemisphere, since the equations of motion are symmetric around the equator) and χ_0 between 45° and 135°. Each integration was ended when either the terminal energy level was reached or the trajectory would skip out of the atmosphere ($H > 122$ km).

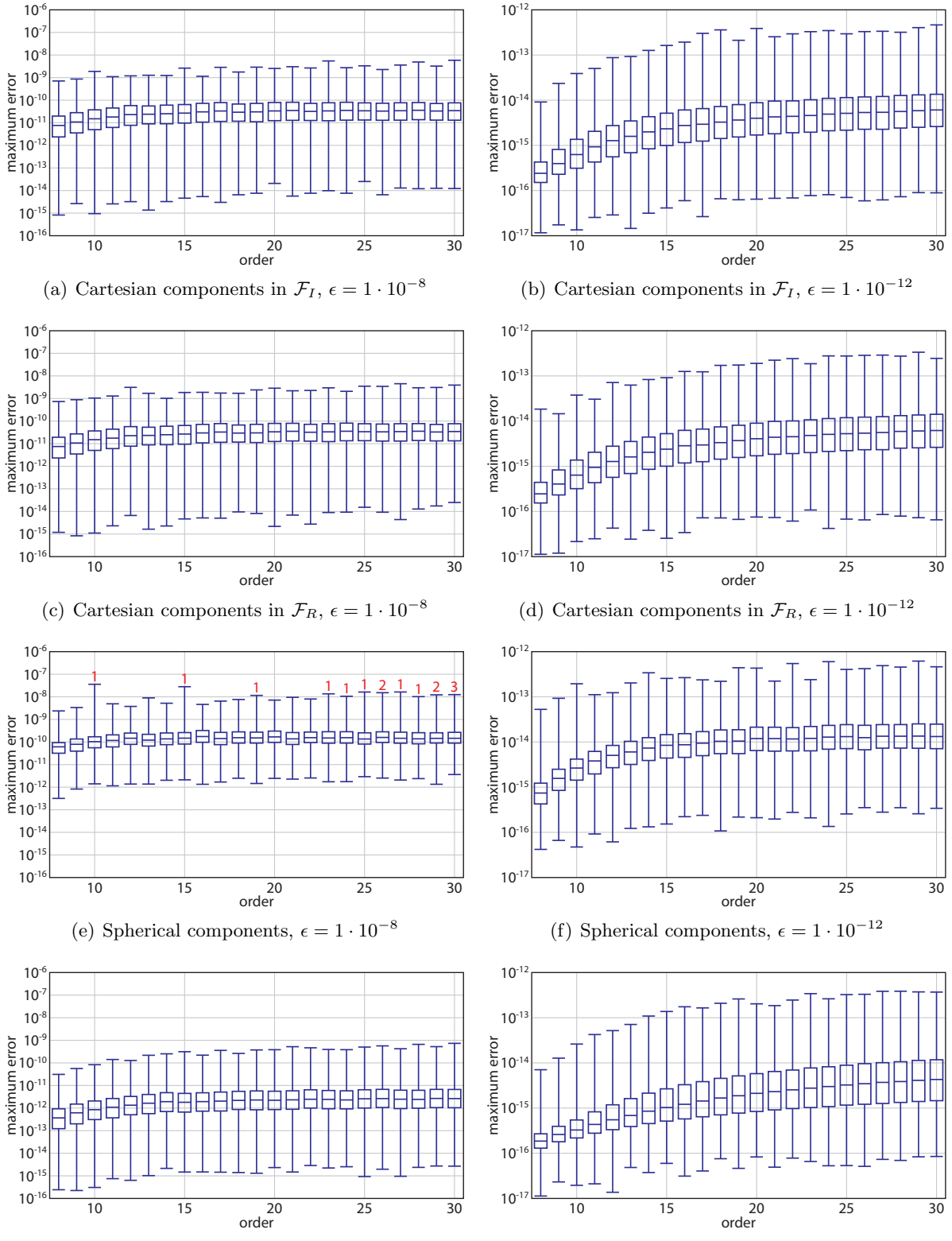
The results are shown for each combination of step-size controller and state-variable set and for error tolerances of $1 \cdot 10^{-8}$ and $1 \cdot 10^{-12}$ in Figures 8.1 to 8.3. These graphs give the distribution of the maximum errors for 30,000 time steps in the form of boxplots. The top and bottom of a boxplot indicate the maximum and minimum of the data, respectively. 25% of the data lies below the center box and 25% lies above it. The center line indicates the median, so 50% of the data lies below and above it. Note that the median does not have to split the center box in two equal parts. The vertical axis name "Maximum errors" here means the maximum of absolute and relative errors of each state variable. The numbers above some of the boxplots indicate the number of error tolerance violations. For Cartesian velocity and position, the errors of the norms of the position and velocity were checked rather than the state variables themselves, because, otherwise, the relative errors depend on the rotation of the reference frame (for instance, if you have an absolute error of 1 m, then the same position can have an X-coordinate of $1 \cdot 10^6$ in one reference frame, which gives a low relative error, and 100 in another, which gives a much higher relative error). Similarly, the relative error w.r.t. angles also differs per reference frame, so for angles, only the absolute error was taken into account. During simulations, the errors were also larger for velocity components than position components. Furthermore, the errors for angles are larger than for Cartesian components, so the spherical state variables have the largest errors.

In Figures 8.1 and 8.2, there is an increase in errors as function of the order. One explanation for this is that these two controllers use the last two coefficients of a Taylor series to estimate the error and try to make these coefficients smaller than a certain value. The higher the order, the more terms there are in the series and the less influence the two last terms have on the total sum of the series and thus making the step size controller less effective.

For controller 3, it should be noted that is was designed to be used together with order controller 2 (see Section 5.4.2), which gives a relation between the order and the error (Eq. (5.44)). This relation has also been plotted in Figure 8.3. As can be seen, for error tolerances of $1 \cdot 10^{-12}$ or higher, this controller works well. However, a practical limitation is that one always has to use the order advised by order controller 2 (see Table 8.2), and these orders are not always optimal (see the next section for the comparison of order controllers). Therefore, it will not be used.

From Figures 8.1 and 8.2, it can be concluded that step-size controller 1 is better than controller 2, as it almost does not violate the error tolerances. Multiplying the suggested step size of

controller 1 by a factor of 0.98 turns out to be enough to fully eliminate the error-tolerance violations in Figure 8.1(e). To be safe, step-size controller 1 with a safety factor of 0.95 will be used from here on.



(a) Cartesian components in $\mathcal{F}_I$, $\epsilon = 1 \cdot 10^{-8}$

(b) Cartesian components in $\mathcal{F}_I$, $\epsilon = 1 \cdot 10^{-12}$

(c) Cartesian components in $\mathcal{F}_R$, $\epsilon = 1 \cdot 10^{-8}$

(d) Cartesian components in $\mathcal{F}_R$, $\epsilon = 1 \cdot 10^{-12}$

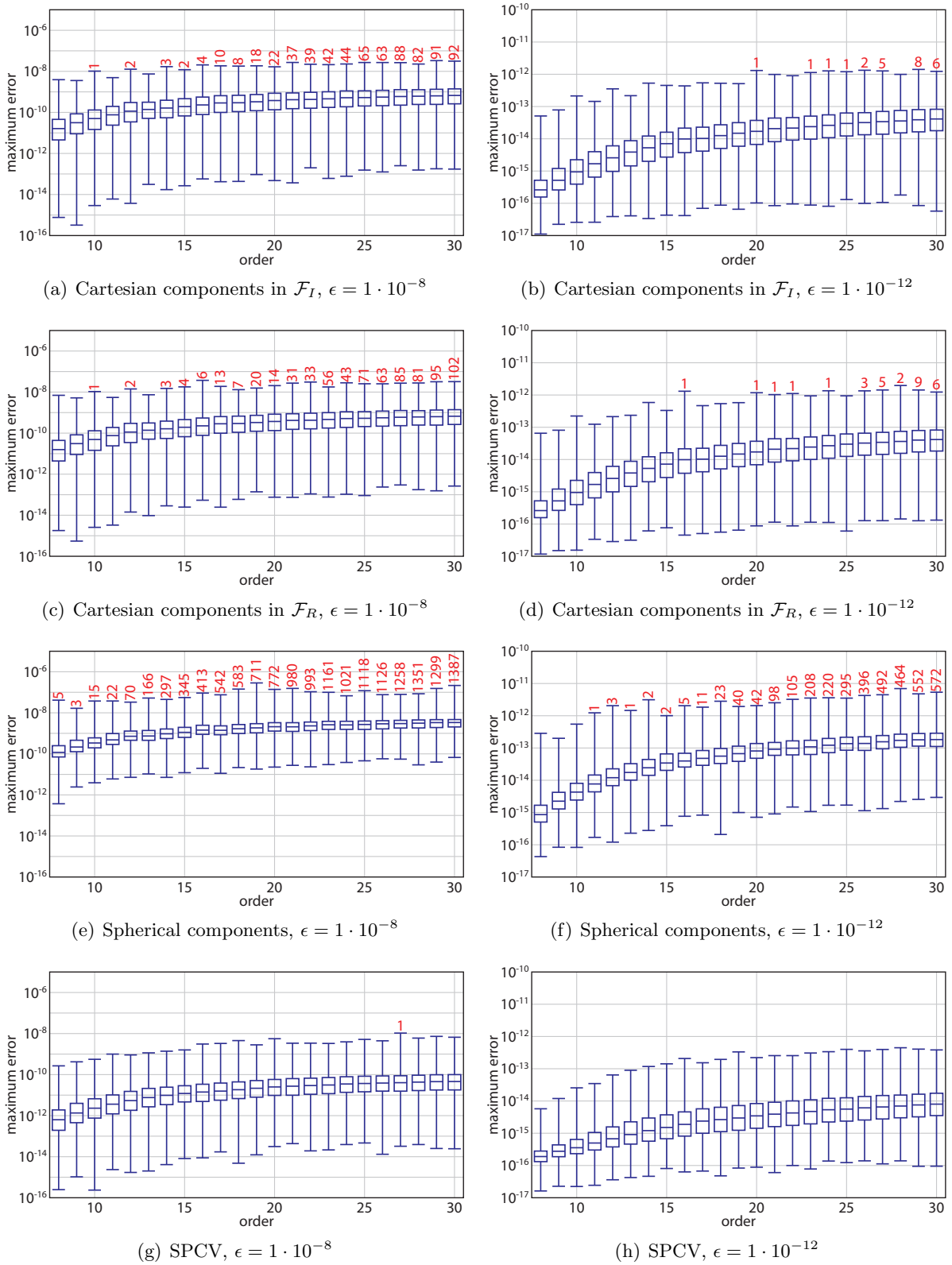
(e) Spherical components, $\epsilon = 1 \cdot 10^{-8}$

(f) Spherical components, $\epsilon = 1 \cdot 10^{-12}$

(g) SPCV, $\epsilon = 1 \cdot 10^{-8}$

(h) SPCV, $\epsilon = 1 \cdot 10^{-12}$

Figure 8.1: Distribution of the maximum errors of step-size controller 1 for 30,000 steps



(a) Cartesian components in $\mathcal{F}_I$, $\epsilon = 1 \cdot 10^{-8}$

(b) Cartesian components in $\mathcal{F}_I$, $\epsilon = 1 \cdot 10^{-12}$

(c) Cartesian components in $\mathcal{F}_R$, $\epsilon = 1 \cdot 10^{-8}$

(d) Cartesian components in $\mathcal{F}_R$, $\epsilon = 1 \cdot 10^{-12}$

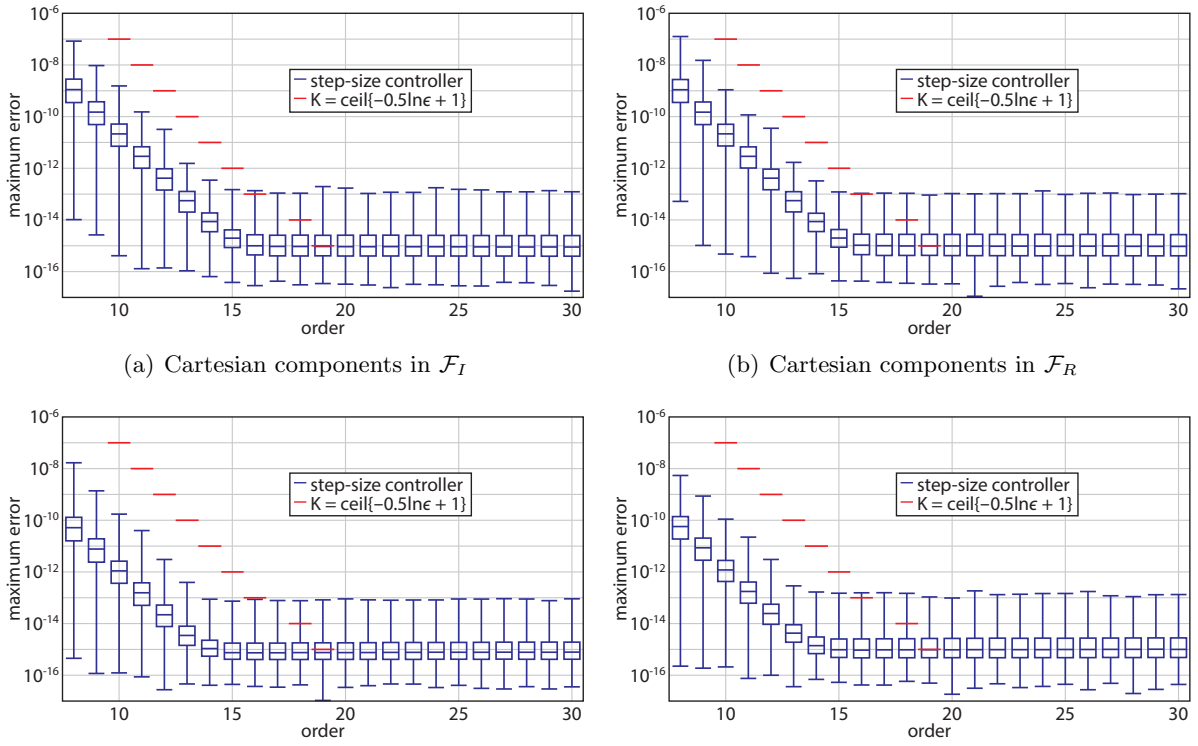
(e) Spherical components, $\epsilon = 1 \cdot 10^{-8}$

(f) Spherical components, $\epsilon = 1 \cdot 10^{-12}$

(g) SPCV, $\epsilon = 1 \cdot 10^{-8}$

(h) SPCV, $\epsilon = 1 \cdot 10^{-12}$

Figure 8.2: Distribution of the maximum errors of step-size controller 2 for 30,000 steps



(a) Cartesian components in $\mathcal{F}_I$

(b) Cartesian components in $\mathcal{F}_R$

(c) Spherical components

(d) Spherical position, Cartesian velocity

Figure 8.3: Distribution of the maximum errors of step-size controller 3 for 30,000 steps

8.1.2 Order Control

Next, the influence of the order-control strategy for TSI on the computational time is studied.

Fixed Order

First, 5,000 integrations with TSI were timed using orders varying from 8 to 25 and error tolerances from $1 \cdot 10^{-8}$ to $1 \cdot 10^{-15}$. The complete set of results is given in Appendix D. In Table 8.1, the best results for each error tolerance are given. Note that the results for $1 \cdot 10^{-15}$ are not included, as this error tolerance could not always be satisfied by the spherical state variables and the SPCV set without resorting to extremely small step sizes. Furthermore, the reader may expect a more smooth trend in the hierarchy of orders for each tolerance, but this is not always the case. Also the best orders do not always increase when the tolerance is lowered. The reason for this is that some time steps are cut at discontinuities or the end of the integration, causing integrations to have different efficiency. To explain this, the inefficiency of an integration is here defined as the percentage of integration time the integration that overshoots a limit t_e. In Figure 8.4, this is shown for three different integrations. As can be seen, the second has the highest efficiency for this particular t_e, which could mean that, even when it is in general slower than its competitors, it could be the fastest here due to its efficiency. One could therefore say that some orders have more "luck" than others. The best few orders for each tolerance have been thoroughly checked to make sure that the hierarchy is correct, but the timings may differ in future runs.

In Section 5.4.2, it can be found that order controller 2 gives the order directly as function of

the error tolerance. The resulting orders for each error tolerance are shown in Table 8.2. As can be seen, they are close to the results in Table 8.1, which shows that they are useful as estimate of the optimal orders, but for the reentry problems in this report, the orders in Table 8.1 can be used for better performance.

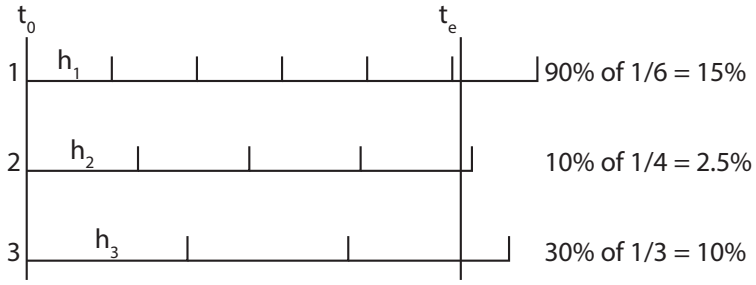


Figure 8.4: Inefficiency of three different integrations to the same end point t_e.

Table 8.1: Best orders and their times to integrate 5,000 trajectories for each error tolerance and state-variable set.

Cartesian state variables in $\mathcal{F}_I$

ϵ	$1 \cdot 10^{-8}$	$1 \cdot 10^{-9}$	$1 \cdot 10^{-10}$	$1 \cdot 10^{-11}$	$1 \cdot 10^{-12}$	$1 \cdot 10^{-13}$	$1 \cdot 10^{-14}$
order	11	12	14	16	17	19	18
time [s]	2.829	3.437	4.082	4.779	5.455	6.244	7.016

Cartesian state variables in $\mathcal{F}_R$

ϵ	$1 \cdot 10^{-8}$	$1 \cdot 10^{-9}$	$1 \cdot 10^{-10}$	$1 \cdot 10^{-11}$	$1 \cdot 10^{-12}$	$1 \cdot 10^{-13}$	$1 \cdot 10^{-14}$
order	11	13	14	16	17	18	20
time [s]	3.136	3.768	4.436	5.221	5.957	6.743	7.571

Spherical state variables

ϵ	$1 \cdot 10^{-8}$	$1 \cdot 10^{-9}$	$1 \cdot 10^{-10}$	$1 \cdot 10^{-11}$	$1 \cdot 10^{-12}$	$1 \cdot 10^{-13}$	$1 \cdot 10^{-14}$
order	10	12	13	15	17	18	17
time [s]	2.288	2.803	3.315	3.905	4.441	5.151	5.725

Spherical position with Cartesian velocity

ϵ	$1 \cdot 10^{-8}$	$1 \cdot 10^{-9}$	$1 \cdot 10^{-10}$	$1 \cdot 10^{-11}$	$1 \cdot 10^{-12}$	$1 \cdot 10^{-13}$	$1 \cdot 10^{-14}$
order	13	14	16	17	17	20	19
time [s]	2.938	3.463	4.056	4.623	5.355	5.985	6.556

Table 8.2: Orders computed with order controller 2.

ϵ	$1 \cdot 10^{-8}$	$1 \cdot 10^{-9}$	$1 \cdot 10^{-10}$	$1 \cdot 10^{-11}$	$1 \cdot 10^{-12}$	$1 \cdot 10^{-13}$	$1 \cdot 10^{-14}$
order	11	12	13	14	15	16	18

Variable Order

Order controller 1 is the only strategy for variable order. It has three parameters which are user set and can thus be optimized, being p, η_1 and η_2. In this case, it has been chosen to also manually set the order of the first step, K_0, and have a minimum and maximum order, K_{min} and K_{max}, which are also optimized. The times to beat are given in Table 8.1. Variable order control will only be used if it is clearly faster than fixed orders; in case of doubt, fixed

orders will be used. Furthermore, different trajectories may have different optimal settings, but the goal here was to find specific settings that are the best overall for a given error tolerance. The results of manual tweaking of these settings for spherical state variables (as they had the lowest times for the fixed orders) are shown in Table 8.3. Overall, it was found that the lowest computational times were achieved when $K_0 = K_{min}$ and the integrator slowly increases the order up to or slightly over the optimal fixed order. The resulting orders for the reference trajectory of HORUS are shown in Figure 8.5 as function of time. The order initially increases up to the point where the aerodynamic forces become large and curve the trajectory to almost horizontal. When this point is passed the order is decreased for a while before stabilizing at the maximum value. Note how, in Figure 8.5, the order controller and step-size controller together cause very similar step sizes for the different error tolerances. The number of time steps for the error tolerances of $1 \cdot 10^{-8}$, $1 \cdot 10^{-10}$, $1 \cdot 10^{-12}$ and $1 \cdot 10^{-14}$ is respectively 53, 54, 53 and 58. This backs up the statement in [30] that to increase the accuracy, it is less computationally expensive to increase the order than to decrease the step size.

Table 8.3: Optimal settings for order controller 1 for spherical state variables.

ϵ	$1 \cdot 10^{-8}$	$1 \cdot 10^{-10}$	$1 \cdot 10^{-12}$	$1 \cdot 10^{-14}$
K_0	6	8	8	11
K_{min}	6	8	8	11
K_{max}	11	14	17	18
p	1	1	1	1
η_1	0.60	0.65	0.75	0.75
η_2	0.40	0.40	0.40	0.50

For the first two tolerances in Table 8.3, variable order is slower than the best fixed order. For $1 \cdot 10^{-12}$, variable order performs about similar to fixed order and only for the lowest tolerance, variable order is slightly faster. The differences in computational time with the fixed order strategy are all within about 1% of the total computational time. Overall, it is concluded that order controller 1 does not yield a significant performance boost for spherical state variables w.r.t. the increase in number of parameters the user has to optimize before integration. It was therefore not tested for the other state-variable sets.

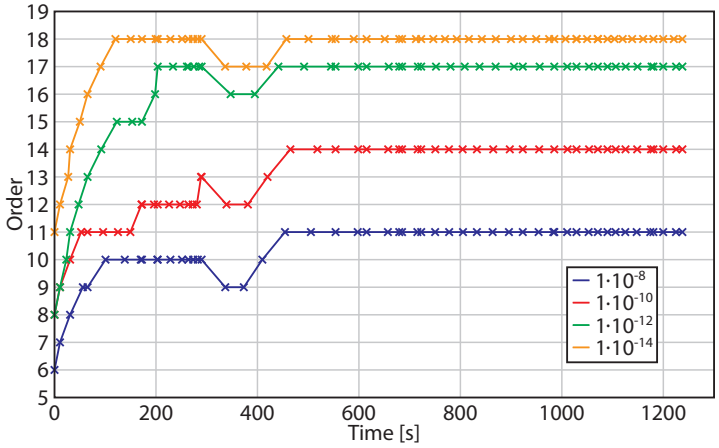


Figure 8.5: Order versus time for order controller 1 and HORUS' reference trajectory.

8.1.3 State-Variable Set

The state-variable set selection was basically performed along with the fixed-order selection. From Table 8.1, it can be concluded that spherical state variables have the most favorable computational time for all cases. Here, a number of analyses is done to determine why this is the case.

One reason why the spherical state variables turned out to be the fastest is the fact that the average step-size for this state-variable set is larger. Take, for example, $K = 11$ and $\epsilon = 1 \cdot 10^{-8}$, for which spherical state variables are the fastest, followed by $\mathcal{F}_I$ Cartesian, SPCV and $\mathcal{F}_R$ Cartesian. The sets have an average step size of 26.0 s, 22.8 s, 18.2 s and 22.6 s, respectively (measured over 5,000 integations). Spherical state variables thus have the largest step size, and even with $K = 10$, the optimal order for them, they still have an average step size of 23.0 s.

The step-size controller bases its step-size selection on the ratio between $\boldsymbol{\epsilon}_n$ and the highest-order coefficients of the state. In Figure 8.6, the average magnitude of the velocity coefficients relative to the velocity state variables themselves are plotted for the first 15 orders. The state variables for the position are not taken into account, as their coefficients are smaller (relatively) and thus do not determine the step sizes. From left to right, each group of blue dots represents u_I, v_I and w_I, each group of red dots V_G, γ_G and χ_G and the green dots are u_V, v_V and w_V. As can be seen, the red dots have the lowest relative magnitude, safe for the ones of γ_G, which are higher than the blue dots and this implies that spherical components should have smaller step sizes than Cartesian components. However, the definition of the error tolerance, Eq. (4.49), is very favorable to γ_G. This definition states that the absolute tolerance is used if the relative tolerance is lower than the absolute. Since the value of γ_G is always in the order of a few degrees (<0.1 rad) for HORUS' gliding flight, the absolute error tolerance is used. The absolute magnitude of the γ_G dots in Figure 8.6 is about as high as the relative magnitude of the V_G dots, which explains the small step sizes. This could be seen as unfair or wrong, but say that one were to replace γ_G with its complement $\gamma'_G = 90^\circ - \gamma_G$. This angle is just like γ_G able to describe the steepness of the flight, but it has a value that remains close to 1.57 rad. Because of its definition, the Taylor coefficients of γ'_G will be equal to those of γ_G multiplied by -1. This means that the step sizes could be even larger for γ'_G. On the other hand, the velocity component w_V of SPCV also describes the steepness of the flight and is therefore small. This makes its Taylor coefficients relatively large, as can be seen in Figure 8.6. Because w_V is not dimensionless, it is in general larger than 1 m/s, which means it does not benefit from the absolute error tolerance, making it the main reason for the small step sizes for SPCV.

To see whether it is justifiable that the spherical state variables have larger steps, the errors made by them, $\mathcal{F}_I$ Cartesian state variables and SPCV have been checked for the step size of the spherical set in Figure 8.7 (again, only the velocity components, as their errors are larger than those of the position). In this Figure, the errors are also shown when transforming the state to the other sets. Note here that the errors for the dimensional velocity components have been normalized with V_G. The "real" solutions were in this case created using TSI with order 35 and the same step sizes. The relative differences between the real solutions, computed with different state-variable sets, were about $1 \cdot 10^{-15}$ or less. As can be seen in Figure 8.7, the spherical state variables indeed have lower errors for all variables, except for V_G, where the Cartesian set has slightly smaller errors. This figure also shows that the steps for SPCV actually do not have to be smaller than for Cartesian state variables.

The other reason why spherical state variables have low computational times is the low average computational time per step. Also, SPCV, despite having the smaller steps, outperforms the Cartesian state variables in $\mathcal{F}_R$. This means that it too must have a lower average computational

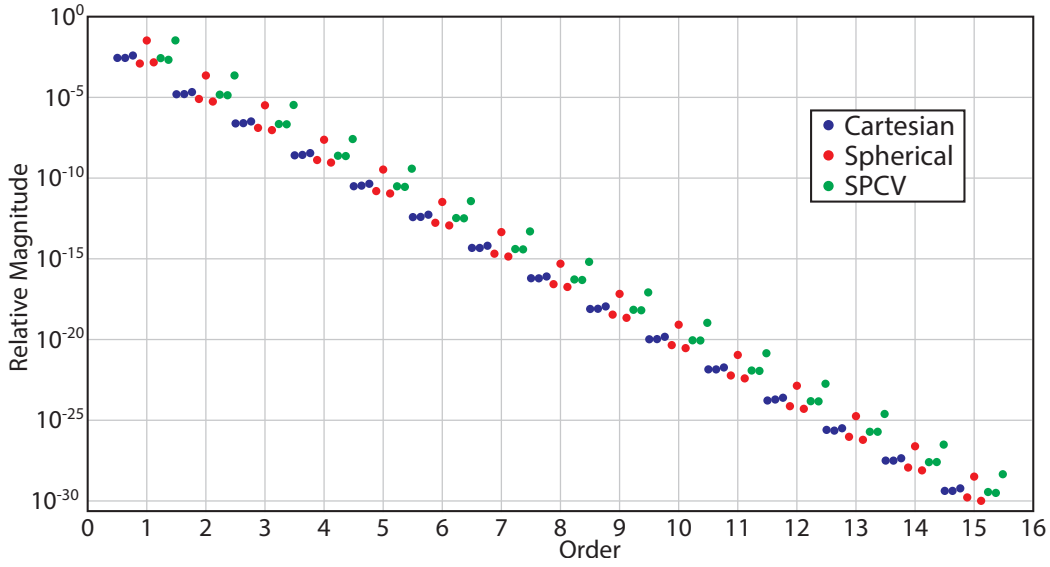


Figure 8.6: Average magnitude of the velocity Taylor coefficients relative to the leading terms.

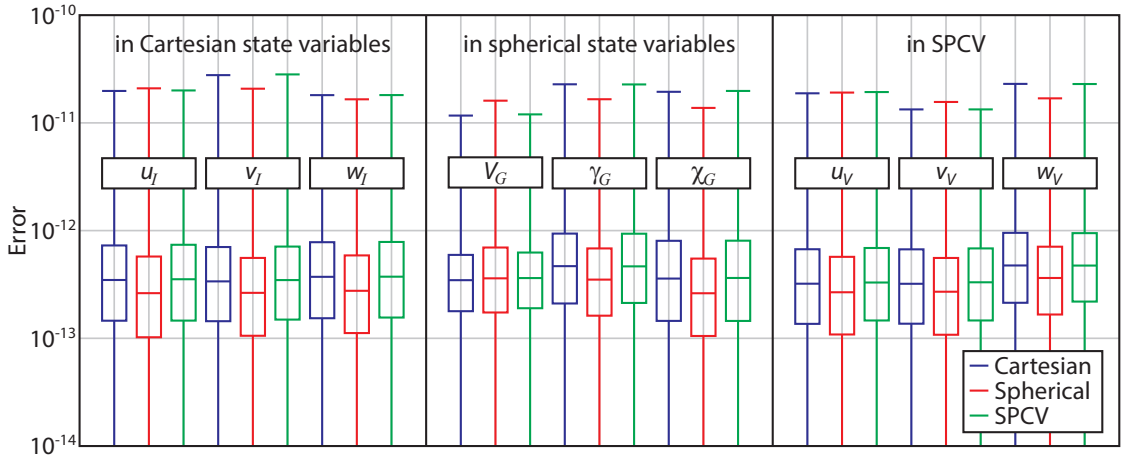


Figure 8.7: Errors made by three different state-variable sets, transformed to each of the sets, using the step size for spherical state variables with $K = 15$ and $\epsilon = 1 \cdot 10^{-10}$.

time per step. In Table 8.4, the profiling of the integration process is shown for each state-variable set for $K = 11$ and $\epsilon = 1 \cdot 10^{-8}$. Per step, resizing and contraints integration should take roughly the same amount of time for all state-variable sets. The state variable specific part, the generation of auxiliary variables and Taylor coefficients are indeed lowest for SPCV, followed by the spherical set and then the two Cartesian sets. These results can at least partially be explained by counting the number mathematical operations that cause summed multiplications in the recurrence relations, which is all safe for addition and subtraction (see Section 5.2). From the recurrence relations in Chapter 6, it can be found that Cartesian state variables have 53 of such operations for the equations of motion without wind, where spherical state variables have 43, and SPCV has 41.

8.1.4 Wind

In the previous subsections, the optimal settings for TSI have been determined in case of absence of wind. Here, the optimal settings are determined in case wind is included. In Section 5.6.3, two different methods of sampling the wind model were suggested. The first, which directly

Table 8.4: Average percentage of computational time spend on each sub-process of TSI over five runs. The standard deviation is about 0.5% of the total time.

	auxiliary variables	Taylor coefficients	resize steps	integrate constraints	other processes
$\mathcal{F}_I$ Cartesian	6.3%	79.1%	6.5%	1.0%	7.0%
$\mathcal{F}_R$ Cartesian	6.4%	84.4%	4.7%	1.0%	3.4%
spherical	7.8%	73.7%	7.5%	1.2%	9.8%
SPCV	7.3%	72.1%	7.8%	1.4%	11.3%

uses the spline coefficients with which the wind velocities are defined, is exact and the second, which fits least-squares regressions through a number of samples, is an approximation, but is not bound in step size to the size of the spline intervals.

Spline Method

The spline method has been timed for 2,000 trajectories, each with different control profiles and different wind profiles to determine the optimal order and state-variable combination. The SPCV set was not programmed due the fact that it was among the slowest of state-variable sets for both integrators. Programs for Cartesian state variables in $\mathcal{F}_I$ and $\mathcal{F}_R$ differ little from each other, thus it took little time and effort to create one when the other was created. Similar to Table 8.1, the best orders for each state-variable set and error tolerance are given in Table 8.5. The complete tables are again given in Appendix D. As can be seen, the spherical state variables again have the lowest computational time. Also, the optimal orders are significantly lower for these simulations than for the ones without wind. This is at least partially due to the fact that the wind profile spline has a discontinuity every 2 km altitude, which makes the larger step sizes associated with higher orders less efficient.

Table 8.5: Best orders and their times to integrate 2,000 trajectories with wind for each error tolerance and state-variable set.

Cartesian state variables in $\mathcal{F}_I$

ϵ	$1 \cdot 10^{-8}$	$1 \cdot 10^{-9}$	$1 \cdot 10^{-10}$	$1 \cdot 10^{-11}$	$1 \cdot 10^{-12}$	$1 \cdot 10^{-13}$	$1 \cdot 10^{-14}$
order	9	9	10	11	11	13	13
time [s]	3.142	3.716	4.409	5.117	5.822	6.633	7.438

Cartesian state variables in $\mathcal{F}_R$

ϵ	$1 \cdot 10^{-8}$	$1 \cdot 10^{-9}$	$1 \cdot 10^{-10}$	$1 \cdot 10^{-11}$	$1 \cdot 10^{-12}$	$1 \cdot 10^{-13}$	$1 \cdot 10^{-14}$
order	9	9	10	10	11	13	13
time [s]	3.111	3.707	4.393	5.145	5.962	6.696	7.507

Spherical state variables

ϵ	$1 \cdot 10^{-8}$	$1 \cdot 10^{-9}$	$1 \cdot 10^{-10}$	$1 \cdot 10^{-11}$	$1 \cdot 10^{-12}$	$1 \cdot 10^{-13}$	$1 \cdot 10^{-14}$
order	8	9	10	10	11	12	13
time [s]	2.652	3.130	3.671	4.311	5.049	5.736	6.516

Least-Squares Method

In Section 5.6.3, a least-squares method was described that computes a prediction, H_{pred} (multiplied by η and set within limits H_{min} and H_{max}), for the altitude range covered during a time

step. The wind velocities are then sampled at s different altitudes defined by a Chebychev-Gauss grid (see Section 4.2.2) within this altitude range. Then a degree p polynomial is fitted through the sample points to form an analytical expression for the wind velocities. Here, the user settings s, p, η, H_{min} and H_{max} are optimized.

First though, the optimal integration order K has to be determined for a number of error tolerances. This was done by integrating the HORUS' reference trajectory with 2,000 different random wind profiles using different combinations of the parameters s, p, η, H_{min} and H_{max}. Since these parameters impact the optimal integration order and testing all possible combinations with a large number of wind profiles would be too time consuming, only an estimate of this order can be determined. For tolerances of $1 \cdot 10^{-8}$, $1 \cdot 10^{-10}$ and $1 \cdot 10^{-12}$, they are roughly 10, 12 and 13.

Next, for each of the tolerances, an optimization of the parameters was performed to determine which combination yields the least amount of integration steps and has no errors larger than the respective tolerance. The errors were determined per step by also integrating with the spline method, set to $\epsilon = 1 \cdot 10^{-14}$ and comparing the results. Unfortunately, no combination of parameters was found for the least-squares method that could make it satisfy any of the error tolerances without resorting to step sizes much smaller than 1 s or constant altitude steps of about 100 m.

The method was then tried once more for $\epsilon = 1 \cdot 10^{-8}$, but now using an integration order of 9. This yielded a number of combinations of parameters, which satisfy the error tolerances for all 2,000 wind profiles. The combination of parameters with the lowest computational time was $p = 4$, $s = 5$, $\eta = 5.077$, $H_{min} = 889.5$ m, and H_{max} set to an arbitrary large value (6,000 m). The computational time in that case is 4.056 s, whereas the spline method for this particular trajectory has an average time of 3.276 s. The spline method is thus available for a larger range of error tolerances (it was verified at the end of Section 7.3.3 up to a tolerance of $1 \cdot 10^{-12}$) and has a lower computational time for the only case where the least-squares method works. Therefore, the spline method will from here on be used to integrate trajectories with wind.

8.2 Integration

Now that the optimal settings for TSI are determined, RKF5(6) is timed to determine which of both integrators is the fastest for reentry integration.

8.2.1 Equations of Motion without Wind

Similar to TSI, RKF5(6) was timed for batches of 5,000 random trajectories without wind, using each of the state-variable sets. The results are shown in Table 8.6. In this table, also the ratio between the times of RKF5(6) and TSI, for which the results were given in Table 8.1, for each state-variable set are shown. For RKF5(6), Cartesian state-variables are faster than the spherical variables and SPCV. Using $\epsilon = 1 \cdot 10^{-8}$, the average step size for $\mathcal{F}_I$ Cartesian, $\mathcal{F}_R$ Cartesian and spherical state variables and SPCV is 10.52 s, 10.54 s, 15.46 s and 10.63 s, respectively. This explains why $\mathcal{F}_R$ Cartesian is better than $\mathcal{F}_I$ Cartesian and shows that the average computational time per step has to be larger for spherical state variables and SPCV than for Cartesian state variables. The ratios between the best performing set of RKF5(6), the Cartesian state variables in $\mathcal{F}_R$, and the best set of TSI, spherical state variables, are then given

in Table 8.7. As can be seen, RKF5(6) is per state-variable set at least a factor 2.39 slower and overall a factor 3.28 slower. Furthermore, the ratio increases as the tolerance decreases, showing that TSI is especially preferred for high accuracy.

Table 8.6: Timing of RKF5(6) and ratio with TSI's times.

ϵ	$1 \cdot 10^{-8}$	$1 \cdot 10^{-9}$	$1 \cdot 10^{-10}$	$1 \cdot 10^{-11}$	$1 \cdot 10^{-12}$	$1 \cdot 10^{-13}$	$1 \cdot 10^{-14}$
$\mathcal{F}_I$ Cartesian [s]	7.555	10.57	15.10	21.57	31.25	45.56	66.94
RKF5(6)/TSI	2.67	3.08	3.70	4.51	5.73	7.30	9.54
$\mathcal{F}_R$ Cartesian [s]	7.504	10.51	14.88	21.36	31.02	45.29	66.44
RKF5(6)/TSI	2.39	2.79	3.35	4.09	5.21	6.72	8.78
spherical [s]	10.86	15.24	21.46	31.23	45.24	63.84	93.29
RKF5(6)/TSI	4.75	5.44	6.47	8.00	10.19	12.39	16.29
SPCV [s]	10.54	15.26	21.55	31.28	45.22	65.73	96.52
RKF5(6)/TSI	3.59	4.41	5.31	6.77	8.44	10.98	14.72

Table 8.7: Ratios between the best performing state-variable sets of the two integrators.

ϵ	$1 \cdot 10^{-8}$	$1 \cdot 10^{-9}$	$1 \cdot 10^{-10}$	$1 \cdot 10^{-11}$	$1 \cdot 10^{-12}$	$1 \cdot 10^{-13}$	$1 \cdot 10^{-14}$
RKF5(6)/TSI	3.28	3.75	4.49	5.47	6.99	8.79	11.61

8.2.2 Equations of Motion with Wind

Next, RKF5(6) is timed using batches of 2,000 random trajectories with random wind profiles, in the same way as TSI was timed. The results are shown in Table 8.8. The average step sizes for $\epsilon = 1 \cdot 10^{-8}$ are 10.53 s, 10.54 s and 15.33 s, respectively, which are almost identical to those for no wind. This means that the addition of wind especially has a large impact on the average computational time per step for Cartesian state variables, because they are now slower than spherical state variables. Since the spherical set is now the fastest, the best performing state-variable sets are the same for both integrators and the last row of Table 8.8 gives the ratios between the best sets of the integrators. These ratios show that TSI is still faster than RKF5(6), but this time by a smaller margin; the smallest ratio is 1.24. One reason for this is the fact that TSI has to resize its step sizes for each spline interval. Another cause could be the fact that the optimal orders of TSI for wind are lower than for no wind, which means that the optimal order lies closer to the order 5 of RKF5(6). Note, however, that the lower orders for TSI could be completely due to the fact that TSI has to reduce its step sizes more often due to the additional discontinuities.

Table 8.8: Timing of RKF5(6) with wind and ratio with TSI's timing.

ϵ	$1 \cdot 10^{-8}$	$1 \cdot 10^{-9}$	$1 \cdot 10^{-10}$	$1 \cdot 10^{-11}$	$1 \cdot 10^{-12}$	$1 \cdot 10^{-13}$	$1 \cdot 10^{-14}$
$\mathcal{F}_I$ Cartesian [s]	4.217	5.918	8.356	11.94	17.20	24.93	33.87
RKF5(6)/TSI	1.34	1.59	1.90	2.33	2.95	3.76	4.55
$\mathcal{F}_R$ Cartesian [s]	3.929	5.468	7.725	11.03	15.91	23.04	33.63
RKF5(6)/TSI	1.26	1.48	1.76	2.14	2.67	3.44	4.48
spherical [s]	3.289	4.590	6.555	9.410	13.53	19.53	28.33
RKF5(6)/TSI	1.24	1.47	1.79	2.18	2.69	3.41	4.35

8.3 Optimization

Two different optimizations were performed using TSI and RKF5(6). The first is HORUS' flight to the TAEM interface. The second is the combined optimization of downrange and integrated heat load. Both problems were optimized with the default settings of PaGMO (see Section 7.2.4) and a population size of 100. The integrators were set to their optimal state-variable set and an error tolerance of $1 \cdot 10^{-8}$, the latter to reduce the time taken by the optimization and to test if this tolerance is low enough for the integrators to produce valid trajectories.

8.3.1 Flight to TAEM

The objective functions $\boldsymbol{J}$ of this problem are:

$$\boldsymbol{J} = \begin{bmatrix} w_0|d - d_{term}| + J_{con} \\ w_1|E - E_{term}| + J_{con} \end{bmatrix} \tag{8.1}$$

where d_{term} and E_{term} are the terminal distance and energy, 0.75° and $5.219 \cdot 10^5$ J (computed in Section 3.2.1), respectively. J_{con} is the sum of the integrated constraint violations, as given by Eq. (4.59). The weights w_0 and w_1 are set equal to 1 and $1 \cdot 10^{-8}$. w_1 is set such that $w_1|E - E_{term}|$ is always smaller than 1 (maximum energy level is $2.884 \cdot 10^7$ J), which is the minimum non-zero value of J_{con}, so that setting J_{con} to zero should be the main priority of the optimizer. Different levels of terminal energy are acceptable when reaching the TAEM interface, as long as HORUS can reach the runway during the TAEM phase. This energy range is shown in Figure 12.2 of [14] and is approximately between 40 km and 60 km energy height, which is the equivalent of $3.924{\cdot}10^5$ J to $5.886{\cdot}10^5$ J. For this optimization, the minimal energy of the controls is not set to $5.219 \cdot 10^5$ J, but to $3.924{\cdot}10^5$ J, so that the controls are defined for the entire energy range. An advantage of this is that the controls are still defined in case a trajectory is perturbed, for instance, during a sensitivity analysis, and the vehicle ends up reaching the interface with less energy than E_{term}.

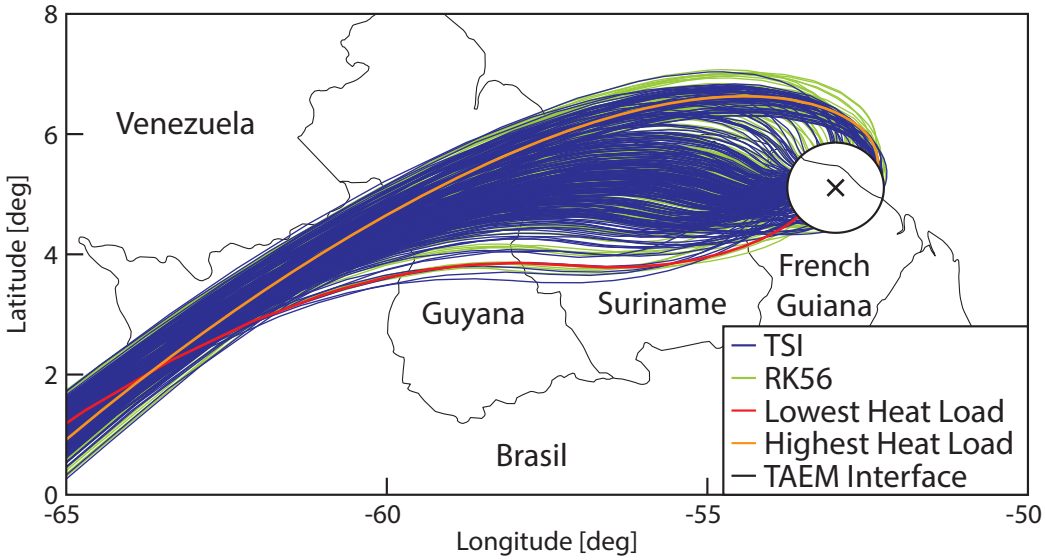
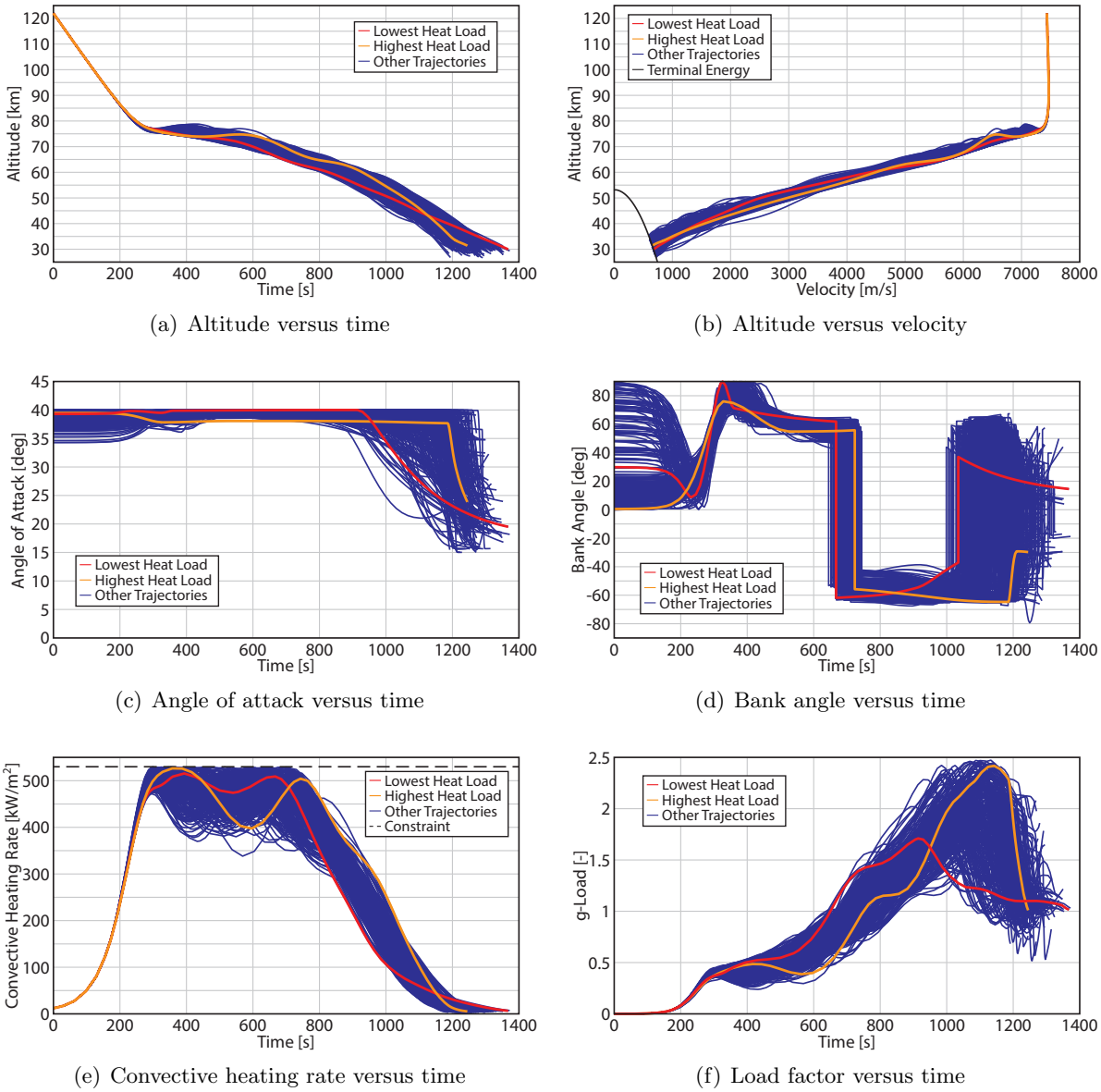


Figure 8.8: Trajectories to the TAEM interface found through optimization with RKF5(6) and TSI.

The optimal values for this problem are 0 for both objectives. MOEAD with the original Tchebycheff approach yielded the best results, reaching values of practically zero (0 difference

in distance and $6.4 \cdot 10^{-10}$ J energy difference) for both objectives simultaneously. This, however, meant it could only produce one result per optimization, as the entire population ends up having the same control values. NSGA-II had worse results (best individual had 0 difference in distance and 1.9 J energy difference), but it actually kept the semi-optimal trajectories as well. MOEAD with the convex Tchebycheff approach persistently converged to values significantly higher than 0. NSGA-II managed to find per optimization about 90 trajectories that precisely hit the TAEM interface with both integrators. In Figure 8.8, the trajectories found in three optimization runs with NSGA-II and the one with MOEAD using both integrators are shown. In Figure 8.9, some of the other properties of the trajectories found with TSI are plotted; the lines of RKF5(6) are not in these graphs as they would be visible behind the the lines of TSI.



(a) Altitude versus time

(b) Altitude versus velocity

(c) Angle of attack versus time

(d) Bank angle versus time

(e) Convective heating rate versus time

(f) Load factor versus time

Figure 8.9: Results of the optimization of the flight to TAEM with TSI set to $\epsilon = 1 \cdot 10^{-8}$.

Despite the fact that the amount of energy left is one of the objectives of this optimization, there is little point in ranking the trajectories using this as criterion, as all trajectories are well within the acceptable energy range; the maximum energy difference was about 3490 J. It is

more interesting to rank the trajectories by integrated heat load (see Section 3.5), although this was not used as an objective function to keep the optimization simple. The trajectories with the lowest and highest heat load are indicated in Figures 8.8 and 8.9 by red and yellow lines, respectively. The trajectories have heat loads of $3.5362 \cdot 10^8$ J/m^2 and $3.8372 \cdot 10^8$ J/m^2, respectively; the first is actually below the heat load of HORUS' reference trajectory ($3.5931 \cdot 10^8$ J/m^2). As can be seen in Figure 8.8, the lowest heat load trajectory reaches the front of the interface and the highest reaches the back. For the latter trajectory, HORUS has to fly further and stay up high in the atmosphere for longer, meaning it has to make a small skip. This causes the high heat load, together with the tight turn (of almost 2.5 g) made near the end of the trajectory, during which the vehicle drops about 30 km in altitude.

For all trajectories found, the distribution of the errors w.r.t. integrators set to $\epsilon = 1 \cdot 10^{-14}$ is displayed in Figure 8.10. In the figure, the errors in the final altitude, velocity and integrated heat load are shown. Note that the trajectories found with TSI were compared with integrators also set to spherical state variables and trajectories found with RKF5(6) were compared with integrators set to $\mathcal{F}_R$ Cartesian state variables. As can be seen in Figure 8.10, TSI has lower errors than RKF5(6), even when compared to the low tolerance RKF5(6) integrator. Compared to the low tolerance RKF5(6) integrator, TSI has medians that are about two orders of magnitude lower than RKF5(6). Compared to the low tolerance TSI integration, the medians are four to five orders of magnitude lower. Other than the lower errors, TSI produces very similar results to RKF5(6) and both integrators agree on which trajectories violate constraints and which do not.

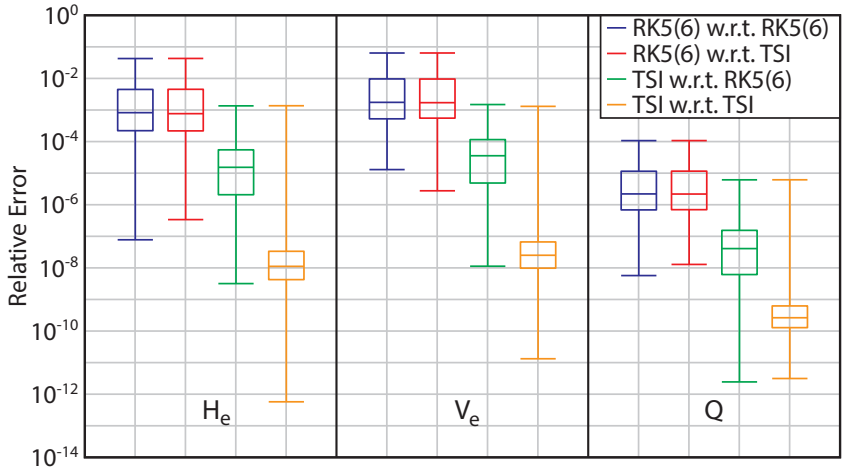


Figure 8.10: Relative errors of RKF5(6) and TSI w.r.t. integrators set to a tolerance of $1 \cdot 10^{-14}$.

8.3.2 Downrange and Integrated Heat Load

The objective functions of this problem are:

$$\boldsymbol{J} = \begin{bmatrix} -w_0 R_d + J_{con} \\ w_1 Q + J_{con} \end{bmatrix} \tag{8.2}$$

with R_d being the downrange, which can be found from Eq. (3.19). Note that for this optimization, bank reversals are disabled, so $\sigma_G \geq 0°$ holds for the entire trajectory. w_0 and w_1 were set to 1 and $1 \cdot 10^{-9}$. During optimization, it was found that NSGA-II is better at finding the extreme values for each objective individually, but MOEAD (both versions) was better at

improving the leading Pareto front overall. After a few different optimizations with NSGA-II, a population was found with about 200° downrange, whereas the other optimizations got stuck at about 190°. This population was then used as starting point for optimizations with MOEAD, one using TSI and one using RKF5(6). These optimizations were continued until no further improvement was found within 1000 generations. The resulting Pareto fronts (both of 100 individuals) are shown in Figure 8.11.

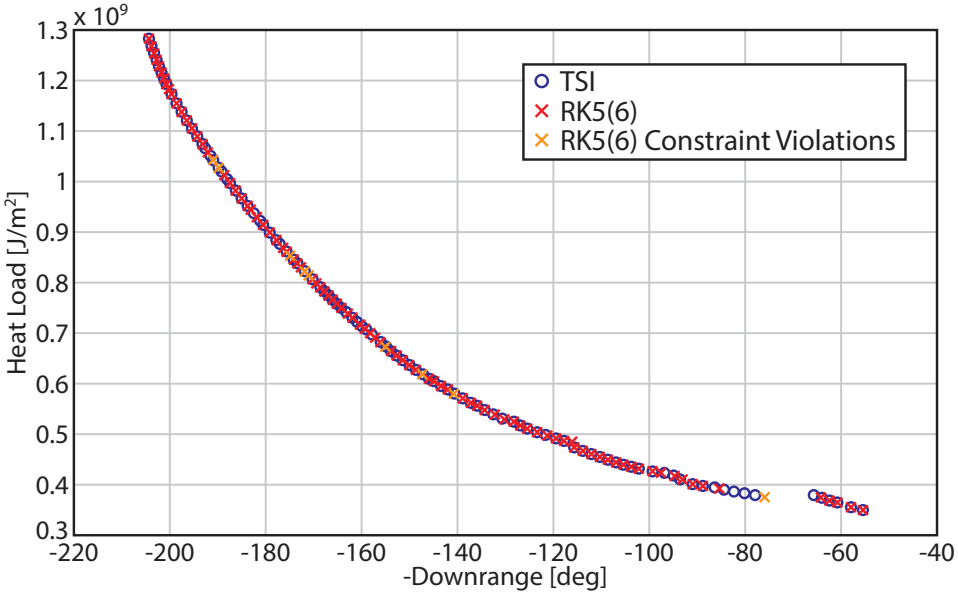


Figure 8.11: Pareto sets of the second optimization problem found with TSI and RKF5(6).

As can be seen, the Pareto fronts almost perfectly overlap, safe for the part near -80° downrange. Since most of the trajectories of TSI and RKF5(6) are very similar, in Figures 8.12 and 8.13, only the properties those found with TSI have been plotted. The trajectory with the minimal heat load ($3.498 \cdot 10^8$ J/m^2) has a range of 55.45° and the trajectory with the longest range (204.3°) has a heat load of $1.283 \cdot 10^9$ J/m^2, which is more than 3 times higher than any of the trajectories to the TAEM interface. The minimum heat-load trajectory ends above the Grenadines and the maximum range trajectory ends up over the Nicobar Islands.

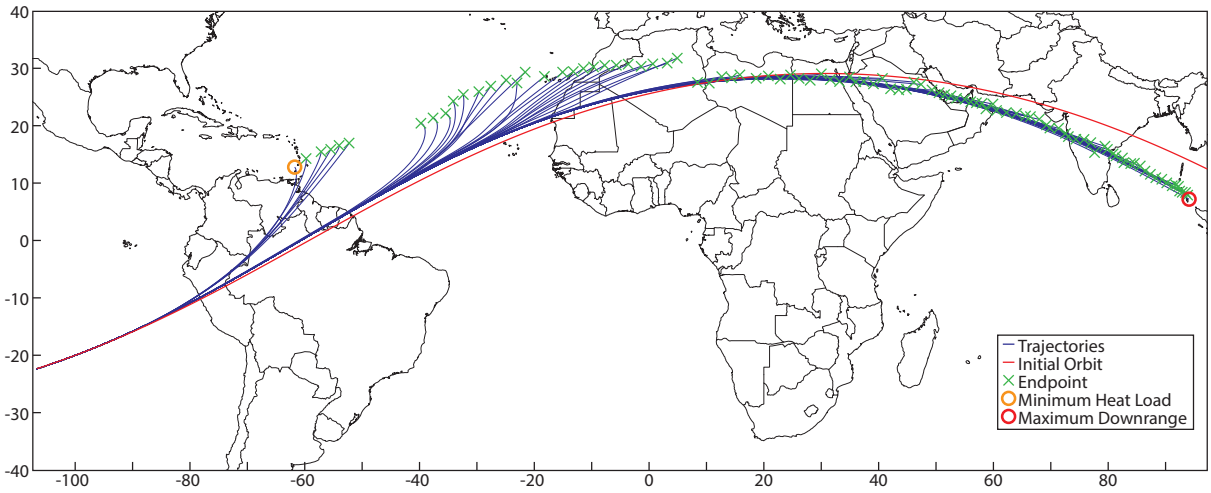
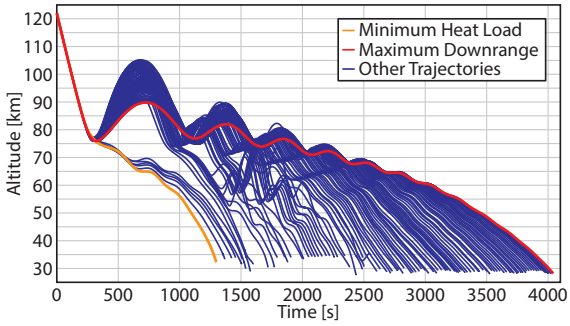
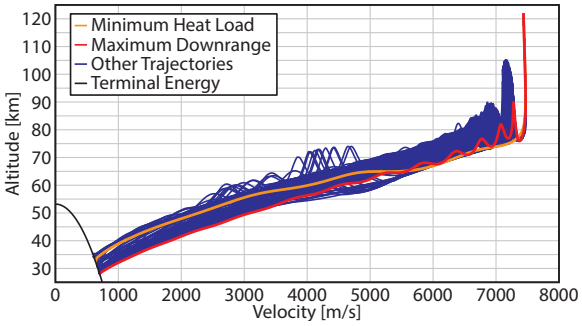


Figure 8.12: Trajectories found optimizing for maximum downrange and minimum heat load.

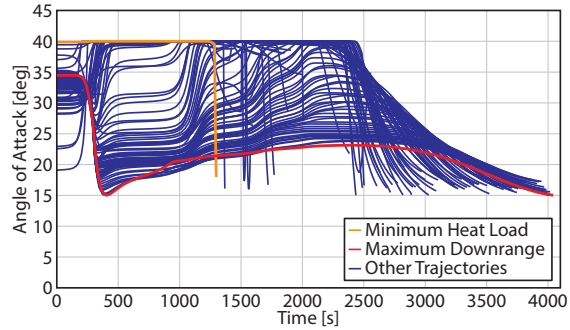
For the minimal heat load, the vehicle flies with maximal angle of attack almost the entire



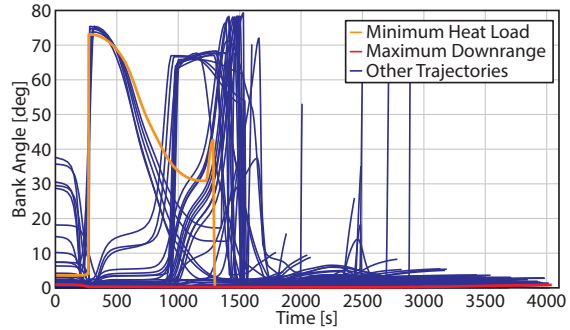
(a) Altitude versus time



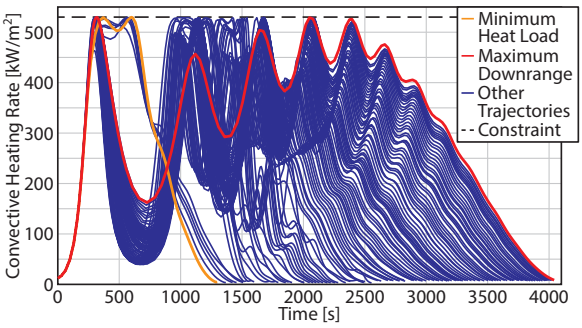
(b) Altitude versus velocity



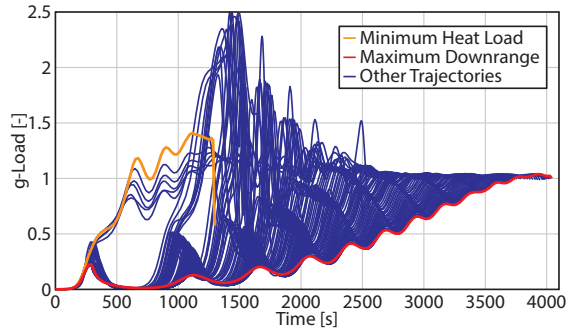
(c) Angle of attack versus time



(d) Bank angle versus time



(e) Convective heating rate versus time



(f) Load factor versus time

Figure 8.13: Results of the downrange and heat-load optimization with TSI set to $\epsilon = 1 \cdot 10^{-8}$.

way, before performing a small dive at the end of the flight. This trajectory actually touches the heating rate constraint twice, but because it then subtly glides down, manages to keep the heating rate low. Since Q is an integral over time, this trajectory also has the shortest flight time of all trajectories and the dive at the end is meant to end the flight slightly faster. For the longest range, the vehicle has to perform the largest possible skip early on without violating the heating rate constraint. To maximize the downrange, HORUS has to fly with practically zero bank angle and the maximum lift over drag ratio [46]. The latter, for the regression fits of HORUS' aerodynamics, is achieved with an angle of attack of approximately 16.6° to 18.1°, depending on the Mach number. The reason the angle of attack is higher than this range for most of the trajectory is to keep the heating rate below the constraint value (notice the two peaks in heating rate in Figure 8.13(e) between 2000 s and 2500 s just touching the constraint line). The angle of attack drops to about 15° to maximize the time HORUS flies at maximum L/D between the second to last control node (located at about 2400 s) and the end of the

flight, without overshooting the heating constraint. Note that if an extra node could be added between these points, it could be used to increase the time of flight at maximum L/D.

In Figure 8.11, 9 individuals (of the 100 in total) found with RKF5(6) were marked in yellow, as integration with a tolerance of $1 \cdot 10^{-14}$ reveals that these trajectories violate the heating constraint. Integration with the high-tolerance TSI also gives this result. On the other hand, none of the trajectories found with TSI violated a constraint when integrated with the low-tolerance integrators. Similar to Figure 8.10, the relative errors of RKF5(6) and TSI when integrating all of the found trajectories, are plotted in Figure 8.14. As can be seen, TSI again has lower errors than RKF5(6), especially when compared to the low-tolerance TSI integrator. Compared to the low-tolerance RKF5(6) integrator, TSI has error of about one-to-two orders of magnitude lower than RKF5(6) and compared to the low-tolerance TSI integrator, the errors are about two-to-six orders lower. From this and the contraint violations of the trajectories found with RKF5(6), it can be concluded that the quality of the results found with TSI is better.

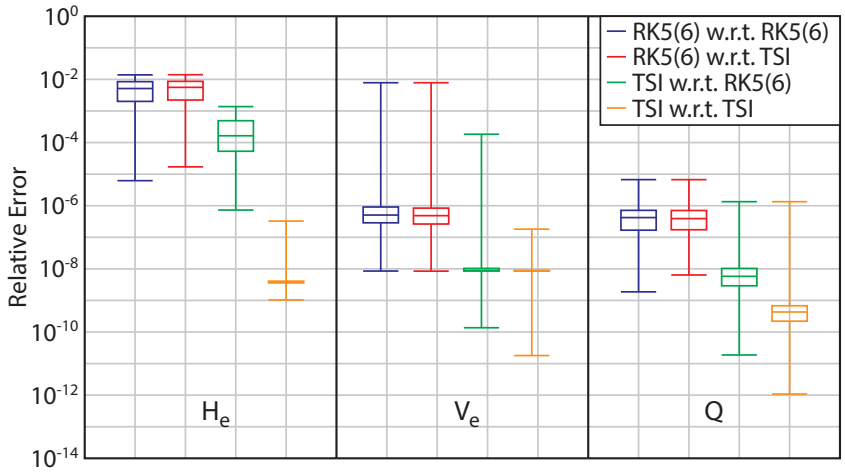


Figure 8.14: Relative errors of RKF5(6) and TSI w.r.t. integrators set to a tolerance of $1 \cdot 10^{-14}$.

8.4 Sensitivity Analysis

Sensitivity analysis consists of applying perturbations to a trajectory and assessing the impact. Normally, one would have a guidance/control system that will try to correct disturbances. In that case, the sensitivity analysis can be used to assess the robustness of the guidance/control system. For this research, no guidance/control system was programmed, so this analysis is merely to assess the necessity of such systems and compare the outcomes of TSI and RKF5(6). In Section 8.4.1, the effects of errors in the controls are analyzed and in Section 8.4.2, the effects of wind perturbations are studied. The sensitivity analyses will be performed on the lowest heat-load trajectory of the first optimization and the minimum heat-load and maximum range trajectories of the second optimization.

8.4.1 Control Perturbation

Here, the value of α_G and σ_G at all the nodes of the control profiles will be perturbed by -5°, -2.5°, +2.5° and +5°. The impact these perturbations have on the final conditions is listed

in Table 8.9. These results have been obtained with a TSI integrator set to a tolerance of $1 \cdot 10^{-14}$, to obtain about as accurate results as possible. In this table, H, M, E, d, R_d, Q and J_{con} are the altitude, Mach number, energy, distance to target, downrange, heat load and normalized integral of the control violations (see Eq. (4.59) for the latter), respectively. Note that for the first trajectory, the terminal conditions were $d = 0.75°$ or $E = 392.4$ kJ, and the other trajectories are stopped at $E = 521.94$ kJ. The full set of graphs of the trajectories with perturbations can be found in Appendix E.

Table 8.9: Final conditions of the trajectories with different control perturbations.

Lowest heat-load trajectory of the first optimization

	H [km]	M [-]	E [kJ]	d [deg]	Q [MJ/m^2]	J_{con} [-]
unperturbed	29.924	2.2429	522.45	0.75	353.62	0
$\Delta\alpha_G = -5°$	52.953	9.6815	5537.0	1.9317	411.08	174.65
$\Delta\alpha_G = -2.5°$	46.255	6.6724	2847.9	0.75	381.62	71.762
$\Delta\alpha_G = 2.5°$	26.896	1.6921	392.4	3.9139	320.76	0
$\Delta\alpha_G = 5°$	27.530	1.6483	392.4	7.0022	292.76	0
$\Delta\sigma_G = -5°$	51.752	8.1022	4065.4	1.9998	362.17	0
$\Delta\sigma_G = -2.5°$	44.909	5.9762	2334.5	0.75	358.81	0
$\Delta\sigma_G = 2.5°$	26.133	1.7437	392.4	3.5733	340.51	124.23
$\Delta\sigma_G = 5°$	26.086	1.7468	392.4	6.6500	326.57	241.80

Maximum downrange trajectory of the second optimization

	H [km]	M [-]	E [kJ]	R_d [deg]	Q [MJ/m^2]	J_{con} [-]
unperturbed	28.168	2.3324	521.94	204.34	1282.9	0
$\Delta\alpha_G = -5°$	24.841	2.5009	521.94	208.48	1638.9	1587.2
$\Delta\alpha_G = -2.5°$	26.725	2.4063	521.94	212.18	1476.7	404.76
$\Delta\alpha_G = 2.5°$	29.301	2.2733	521.94	190.55	1095.8	0
$\Delta\alpha_G = 5°$	30.254	2.2229	521.94	174.86	934.67	0
$\Delta\sigma_G = -5°$	28.147	2.3335	521.94	203.66	1281.6	26.956
$\Delta\sigma_G = -2.5°$	28.162	2.3327	521.94	204.28	1283.0	4.3324
$\Delta\sigma_G = 2.5°$	28.166	2.3325	521.94	203.86	1281.3	23.257
$\Delta\sigma_G = 5°$	28.154	2.3331	521.94	202.84	1278.3	42.605

Minimum heat-load trajectory of the second optimization

	H [km]	M [-]	E [kJ]	R_d [deg]	Q [MJ/m^2]	J_{con} [-]
unperturbed	32.219	2.1161	521.94	55.445	349.83	0
$\Delta\alpha_G = -5°$	31.098	2.1776	521.94	62.316	432.77	189.87
$\Delta\alpha_G = -2.5°$	31.706	2.1446	521.94	58.660	387.56	130.04
$\Delta\alpha_G = 2.5°$	32.697	2.0858	521.94	52.589	318.05	30.405
$\Delta\alpha_G = 5°$	33.158	2.0565	521.94	50.025	291.02	47.488
$\Delta\sigma_G = -5°$	32.449	2.1015	521.94	61.933	372.81	0
$\Delta\sigma_G = -2.5°$	32.346	2.1081	521.94	58.668	361.67	0
$\Delta\sigma_G = 2.5°$	32.095	2.1234	521.94	52.299	337.31	273.71
$\Delta\sigma_G = 5°$	31.970	2.1302	521.94	49.259	324.17	281.50

As can be seen in Table 8.9, for the trajectory to TAEM, HORUS only reaches the TAEM interface ($d = 0.75°$) with $\Delta\alpha_G = -2.5°$ and $\Delta\sigma_G = -2.5°$. Of these two perturbations, only $\Delta\sigma_G = -2.5°$ does not cause any constraint violations. The cases where either control angle is increased causes HORUS to run out of energy before reaching the TAEM interface and $\Delta\alpha_G = -5°$ and $\Delta\sigma_G = -5°$ cause the vehicle to miss the TAEM interface (see Figures E.5(g) and E.6(g)), meaning that the trajectories were terminated when it was no longer heading for

the target. Lowering α_G brings it closer to the value for maximum L/D, which means that HORUS has more velocity left at a particular altitude and that causes the heating rate to increase beyond the constraint value at certain points.

Next, the effects of the perturbations on the maximum range trajectory of the second optimization are analyzed. Lowering α_G again increases the range, but also the heating rate, causing it to exceed the constraint value. Increasing α_G decreases the range and heating rate. Modifying σ_G has almost no impact on the trajectory; the range decreases slightly, because HORUS drifts away from the optimal trajectory. Also, because the original σ_G was practically zero, all perturbations to σ_G cause its absolute value to increase and this causes each skip to happen at a slightly lower altitude, which in turn increases the heating rate during each skip, causing small constraint violations.

For the minimum heat-load trajectory, lowering α_G causes HORUS to make more of a skip when it first touches the denser regions of the atmosphere. This at first lowers the heating rate, but leaves HORUS with more velocity after the skip. This then causes the heating rate to rise above the constraint value at the second skip. Increasing α_G decreases the first skip, causing HORUS to drop deeper into the atmosphere with the same amount of velocity and causes the peak in heating rate to slightly exceed the constraint. Lowering the bank angle increases the skips, which actually lowers the heating rate during the first skip. This also keeps HORUS higher in the atmosphere, delaying the second skip and increasing the flight time, which increases the overall heat load. Increasing the bank angle eliminates the first skip, allowing HORUS to lose altitude more quickly and causing the heating rate to exceed the limit. It also decreases the flight time, which is beneficial to the heat load.

Overall, the control perturbations cause large changes to the trajectories, justifying the need for a guidance (and control) system.

In Table 8.10, the relative differences between the results of TSI and those of RKF5(6) set to the same error tolerance are shown. Note that negative values in this table indicate that the value of RKF5(6) is smaller than that of TSI. To make sure that the results of TSI and RKF5(6) were compared at the same point in time, the results of TSI are interpolated to the final time of RKF5(6). Note that the integrators agree on which trajectories reach the TAEM and which run out of energy and agree on which trajectories violate constraints. However, for some of the parameters in the table, the integrators only agree on the first three digits.

To show the main cause for the relatively large differences in Table 8.10, the lowest heat-load trajectory of the first optimization is integrated with integrators set to $\epsilon = 1 \cdot 10^{-15}$. The relative differences in altitude and velocity are shown by the red and blue lines in Figure 8.15. Note that this figure uses logarithmic scale, so the sign of the differences is not shown and each downward peak actually indicates a change of sign. The most noticeable features of figure 8.15 are jumps in the red line and certain points at which the two lines suddenly increase. As has been marked in the graph, these events coincide with discontinuities such as US76 layer changes and bank reversals. In Section 5.5.1, it was described that some of the root-finding methods do not target the root itself, but aim for a point slightly beyond the root. It was tested to see whether the safety distances of the root-finding methods were the cause of the jumps in Figure 8.15, but decreasing these distances only increases the differences between TSI and RKF5(6), albeit by a negligible amount (compared to the size of the jumps). So the step size of RKF5(6) was then fixed to 0.001 s (it was originally between 0.5 and 1.5 s) and the result is the green line in Figure 8.15. This line is lower than the original line (safe for the spike in the red line where it changes sign), which suggests that TSI is more accurate. The altitude difference for

Table 8.10: Differences between the final conditions found with TSI and RKF5(6), relative to the values of TSI.

Lowest heat-load trajectory of the first optimization

	H [-]	M [-]	E [-]	d [-]	Q [-]	J_{con} [-]
unperturbed	-2.929e-7	-9.884e-7	-1.047e-6	1.291e-4	-1.813e-4	0
$\Delta\alpha_G = -5°$	-4.355e-7	-1.210e-6	-2.021e-6	1.075e-4	-2.131e-3	3.544e-4
$\Delta\alpha_G = -2.5°$	-1.782e-7	-4.207e-7	-8.072e-7	8.451e-5	-9.211e-4	2.869e-4
$\Delta\alpha_G = 2.5°$	4.307e-4	-2.461e-5	1.104e-6	5.071e-5	-1.343e-4	0
$\Delta\alpha_G = 5°$	1.628e-3	-9.943e-5	-2.570e-7	3.184e-6	-5.599e-8	0
$\Delta\sigma_G = -5°$	-3.352e-7	-7.804e-7	-1.254e-6	6.855e-5	-3.003e-3	0
$\Delta\sigma_G = -2.5°$	-1.826e-6	-4.874e-6	-8.949e-6	1.290e-3	-2.064e-4	0
$\Delta\sigma_G = 2.5°$	2.516e-4	-1.517e-5	-5.645e-7	-3.060e-6	6.106e-9	-6.597e-4
$\Delta\sigma_G = 5°$	4.299e-4	-2.498e-5	7.715e-8	4.214e-6	-1.561e-4	-6.850e-4

Maximum downrange trajectory of the second optimization

	H [-]	M [-]	E [-]	R_d [-]	Q [-]	J_{con} [-]
unperturbed	1.817e-4	-1.128e-5	2.360e-9	1.518e-10	-1.050e-4	0
$\Delta\alpha_G = -5°$	3.910e-4	-2.176e-5	1.095e-10	-7.705e-12	-4.358e-7	-1.317e-4
$\Delta\alpha_G = -2.5°$	2.848e-4	-1.689e-5	6.383e-9	-1.603e-10	-1.035e-4	9.327e-4
$\Delta\alpha_G = 2.5°$	1.318e-3	-8.471e-5	-1.954e-9	-1.349e-10	-1.305e-8	0
$\Delta\alpha_G = 5°$	3.558e-4	-2.349e-5	2.522e-8	1.461e-9	-1.064e-4	0
$\Delta\sigma_G = -5°$	3.457e-4	-2.147e-5	-6.679e-9	-3.054e-10	-2.587e-9	-8.751e-4
$\Delta\sigma_G = -2.5°$	1.316e-3	-8.170e-5	6.667e-9	4.043e-10	-1.049e-4	1.774e-2
$\Delta\sigma_G = 2.5°$	6.376e-5	-3.949e-6	1.421e-8	7.554e-10	-1.051e-4	-2.028e-4
$\Delta\sigma_G = 5°$	2.112e-4	-1.312e-5	-2.181e-9	-7.009e-11	-4.935e-9	-1.894e-3

Minimum heat-load trajectory of the second optimization

	H [-]	M [-]	E [-]	R_d [-]	Q [-]	J_{con} [-]
unperturbed	3.215e-4	-6.273e-5	-8.337e-9	-3.133e-9	-1.569e-9	0
$\Delta\alpha_G = -5°$	1.803e-4	-1.220e-5	2.431e-9	3.200e-10	-1.158e-4	7.723e-4
$\Delta\alpha_G = -2.5°$	2.573e-4	-1.769e-5	-1.926e-9	-8.429e-10	7.510e-12	-1.715e-4
$\Delta\alpha_G = 2.5°$	2.815e-6	-5.512e-7	-8.899e-9	-3.183e-9	-5.502e-9	-1.601e-3
$\Delta\alpha_G = 5°$	8.084e-6	-1.606e-6	-5.885e-9	-3.297e-9	-4.018e-9	7.757e-4
$\Delta\sigma_G = -5°$	2.674e-5	-5.240e-6	-2.067e-8	-7.383e-9	-1.253e-8	0
$\Delta\sigma_G = -2.5°$	1.509e-4	-2.952e-5	-1.118e-8	-4.749e-9	-5.557e-9	0
$\Delta\sigma_G = 2.5°$	3.746e-5	-2.603e-6	1.647e-8	6.898e-9	-1.304e-4	4.077e-4
$\Delta\sigma_G = 5°$	4.319e-5	-2.992e-6	-1.809e-8	-7.808e-9	-2.334e-9	-4.701e-4

this case was not plotted to avoid crowding in the figure, but it is of similar order of magnitude as the velocity.

Next, it is of interest to see how large the differences are when root-finding for RKF5(6) performed for all discontinuities and this yields the yellow line. This was done using the double-false position method (see Section 4.1.3). It was found that the roots had to be determined up to 13 significant digits accuracy to make the jumps in the difference with TSI disappear. Now that these discontinuities are absent, the two integrators agree almost up to machine precision for the first 164 seconds and afterwards the difference increases very subtly. Also, now the discontinuities after the bank reversal are noticeable. Overall, it can be concluded that TSI is more accurate RKF5(6) without discontinuity step reduction for trajectories without wind.

In Figure 8.15, the first bank reversal appears to be largest cause of errors, as the red, blue and green lines start to rapidly rise afterwards. Thus it is of interest to see what the red line looks

like when only root-finding for the bank reversals is included, which is shown by the cyan line. As can be seen, now the difference remain of approximately the same order, so bank reversals appear to cause the largest differences between the integrators.

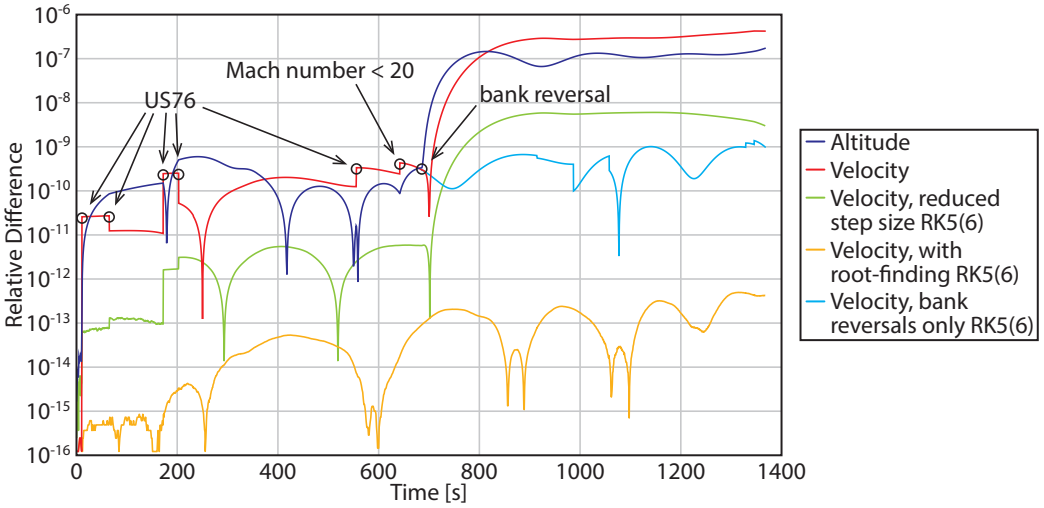


Figure 8.15: Differences in H and V between integrations with RKF5(6) and TSI set to $\epsilon = 1 \cdot 10^{-15}$.

Finally, the differences between the integrators are recomputed, this time using discontinuity step reduction for RKF5(6). The results are shown in Table 8.11. The largest magnitude of the differences, not including the last column, is now about $1 \cdot 10^{-11}$. This suggests that the results obtained with TSI are correct. Discontinuity step reduction was not included for the constraint violations, which would be necessary to lower the differences in J_{con}, as the low differences in Q suggest that the values of $\dot{q}_c$ are very similar for both integrators, so discontinuities must be the cause for the large differences. However, for J_{con}, it is mainly of importance whether it is zero or not and that it is of the correct order of magnitude, so that it can steer the optimization towards trajectories without constraint violations.

Table 8.11: Differences between the final conditions found with TSI and RKF5(6) with discontinuity step reduction, relative to the values of TSI.

Lowest heat-load trajectory of the first optimization

	H [-]	M [-]	E [-]	d [-]	Q [-]	J_{con} [-]
unperturbed	-1.216e-12	-1.152e-12	-1.760e-12	1.900e-11	-1.091e-13	0
$\Delta\alpha_G = -5°$	-1.926e-12	-1.959e-12	-2.771e-12	-8.456e-12	-1.366e-14	-4.737e-4
$\Delta\alpha_G = -2.5°$	9.477e-13	-1.289e-12	-1.635e-12	8.902e-12	-3.248e-14	-1.001e-4
$\Delta\alpha_G = 2.5°$	-1.109e-12	-2.100e-12	-2.163e-12	4.550e-12	-1.501e-13	0
$\Delta\alpha_G = 5°$	-5.078e-13	-7.244e-13	-8.199e-13	1.006e-12	-3.909e-14	0
$\Delta\sigma_G = -5°$	-9.306e-13	-2.913e-12	-4.791e-12	-9.413e-12	2.802e-15	0
$\Delta\sigma_G = -2.5°$	7.269e-13	-7.973e-13	-8.780e-13	6.828e-12	-9.434e-14	0
$\Delta\sigma_G = 2.5°$	-4.635e-13	-1.228e-12	-1.172e-12	2.915e-12	-1.008e-13	-9.468e-4
$\Delta\sigma_G = 5°$	-2.501e-13	-9.166e-13	-8.103e-13	1.140e-12	-1.223e-13	-2.579e-4

Maximum downrange trajectory of the second optimization

	H [-]	M [-]	E [-]	R_d [-]	Q [-]	J_{con} [-]
unperturbed	3.642e-13	-6.914e-13	-4.342e-13	1.140e-13	1.079e-12	0
$\Delta\alpha_G = -5°$	-1.087e-12	9.314e-10	9.922e-10	6.798e-14	6.120e-13	-3.481e-5
$\Delta\alpha_G = -2.5°$	2.233e-12	9.743e-13	2.223e-12	1.364e-13	8.312e-13	4.211e-4
$\Delta\alpha_G = 2.5°$	1.844e-12	3.338e-12	4.120e-12	2.345e-13	1.470e-12	0
$\Delta\alpha_G = 5°$	3.083e-14	4.177e-12	3.611e-12	3.127e-13	1.610e-12	0
$\Delta\sigma_G = -5°$	1.027e-12	2.149e-12	2.625e-12	3.063e-13	1.218e-12	8.114e-4
$\Delta\sigma_G = -2.5°$	2.283e-12	3.285e-12	4.432e-12	4.279e-13	1.178e-12	3.942e-3
$\Delta\sigma_G = 2.5°$	1.291e-12	2.204e-12	2.829e-12	3.070e-13	1.124e-12	-3.694e-3
$\Delta\sigma_G = 5°$	6.289e-13	1.909e-12	2.165e-12	2.622e-13	1.076e-12	2.780e-3

Minimum heat-load trajectory of the second optimization

	H [-]	M [-]	E [-]	R_d [-]	Q [-]	J_{con} [-]
unperturbed	-1.038e-11	2.014e-12	-6.294e-12	4.274e-12	-1.388e-12	0
$\Delta\alpha_G = -5°$	-8.369e-12	1.172e-12	-4.388e-12	-1.253e-12	-6.892e-13	7.381e-4
$\Delta\alpha_G = -2.5°$	-8.552e-12	1.503e-12	-4.356e-12	6.656e-13	-8.153e-13	1.095e-3
$\Delta\alpha_G = 2.5°$	-6.153e-12	3.424e-13	-4.451e-12	4.131e-12	-9.904e-13	-3.683e-3
$\Delta\alpha_G = 5°$	-8.100e-12	4.988e-14	-6.222e-12	6.014e-12	-1.316e-12	-3.555e-3
$\Delta\sigma_G = -5°$	-4.976e-12	1.229e-12	-2.837e-12	-9.426e-13	-3.863e-13	0
$\Delta\sigma_G = -2.5°$	-5.648e-12	1.380e-12	-3.216e-12	5.735e-13	-5.799e-13	0
$\Delta\sigma_G = 2.5°$	-8.038e-12	2.181e-13	-5.119e-12	4.821e-12	-1.245e-12	-2.085e-4
$\Delta\sigma_G = 5°$	-7.609e-12	8.184e-13	-4.338e-12	4.436e-12	-1.134e-12	2.774e-4

8.4.2 Wind Perturbation

Here, the sensitivity to different wind conditions is checked. For this, 8 out of a batch of 2000 random wind profiles were selected for each trajectory, based on the effects they had on the final conditions of the trajectory. They were selected for causing the highest or lowest value of altitude, Mach number, longitude, latitude, distance from the unperturbed final position, heat load, range or constraint violations. Since some of the trajectories caused the multiple extreme values, the total number of unique wind profiles is less than or equal to 8 for all trajectories (if it was less, random profiles were added). The effects of the wind disturbances on the final conditions, computed with TSI (set to $\epsilon = 1 \cdot 10^{-14}$), are shown in Table 8.12.

Table 8.12: Final conditions of the trajectories with different wind disturbances.

Lowest heat-load trajectory of the first optimization

profile	H [km]	M [-]	E [kJ]	d [deg]	Q [MJ/m^2]	J_{con} [-]
-	29.924	2.2429	522.45	0.75	353.62	0
0	31.008	2.3828	563.75	0.75	354.06	0
1	28.769	2.0416	470.90	0.75	356.50	0
2	32.715	2.7443	669.26	0.75	351.04	0
3	27.685	1.8796	430.75	0.75	348.10	0
4	29.097	2.0917	483.79	0.75	354.62	0
5	31.883	2.5443	609.82	0.75	350.15	0
6	30.875	2.3933	564.58	0.75	354.34	0
7	32.121	2.6001	625.65	0.75	350.86	0

Maximum downrange trajectory of the second optimization

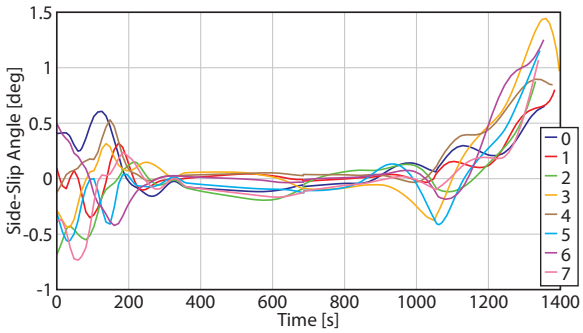
profile	H [km]	M [-]	E [kJ]	R_d [deg]	Q [MJ/m^2]	J_{con} [-]
-	28.168	2.3323	521.94	204.34	1282.9	0
0	28.180	2.3318	521.94	203.96	1278.8	0
1	28.181	2.3317	521.94	204.34	1276.1	20.987
2	28.069	2.3375	521.94	204.36	1281.9	0
3	28.172	2.3322	521.94	204.49	1286.7	57.391
4	28.272	2.3270	521.94	203.99	1275.5	0
5	28.221	2.3296	521.94	204.88	1263.2	0
6	28.205	2.3304	521.94	204.11	1288.7	8.6722
7	28.411	2.3198	521.94	204.23	1257.6	0

Minimum heat-load trajectory of the second optimization

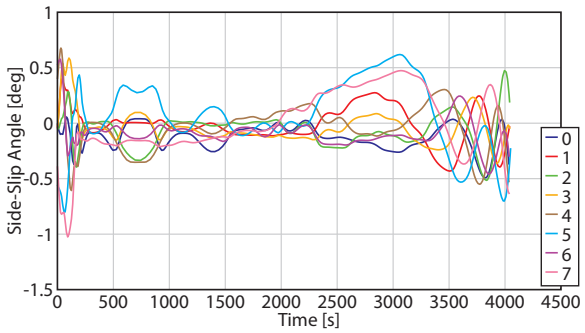
profile	H [km]	M [-]	E [kJ]	R_d [deg]	Q [MJ/m^2]	J_{con} [-]
-	32.219	2.1161	521.94	55.445	349.83	0
0	32.406	2.1043	521.94	55.771	347.25	0
1	32.240	2.1147	521.94	55.776	346.92	0
2	32.260	2.1135	521.94	55.267	352.21	126.52
3	32.239	2.1148	521.94	55.642	343.86	0
4	32.365	2.1068	521.94	55.446	351.43	93.448
5	31.921	2.1329	521.94	55.568	349.67	19.300
6	32.278	2.1123	521.94	55.765	347.36	0
7	32.560	2.0945	521.94	55.493	348.07	34.066

The deviations from the original trajectory due to wind are quite small, compared to the deviations caused by the control perturbations, hence no graphs are shown of the trajectories with wind, as it would be hard to distinguish them from one another. It turns out that the first trajectory is quite safe in case of different wind conditions; non of the wind profiles of

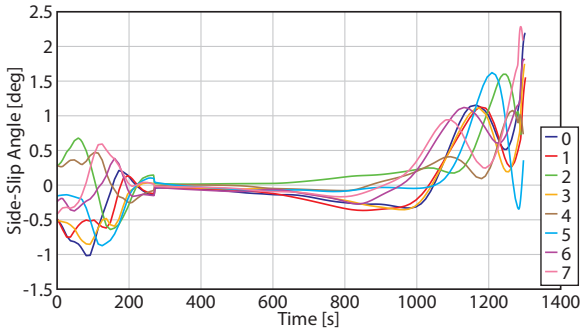
the entire batch of 2000 managed to prevent HORUS from reaching the TAEM or caused any constraint violations. The latter is due to the fact that the original trajectory had some margin between its maximum heating rate and the heating constraint (see Figure 8.9(e)). For the second trajectory, about a quarter of the batch causes heating-constraint violations. For the last trajectory, around 50% of the wind profiles in its batch caused constraint violations. In Figure 8.16, the side-slip angle induced by the wind is shown for each of the trajectories. As can be seen, the last wind profile for the last trajectory causes the highest side-slip angle, which is about 2.29°. This value can be seen as an approximation of the side-slip a control system would have to counter in case of wind.



(a) Lowest heat-load to TAEM trajectory



(b) Maximum range trajectory



(c) Minimum heat-load trajectory

Figure 8.16: Airspeed-based side-slip angle versus time caused by the different wind profiles.

Just like with the control perturbations, the results obtained with TSI were compared with those obtained with RKF5(6) and these are shown in Table 8.13. This time, discontinuity step reduction was right away applied to RKF5(6). The highest order of magnitude of the differences is about $1 \cdot 10^{-10}$, from which it is concluded that TSI obtains accurate results when integrating reentry trajectories with wind.

Table 8.13: Relative differences between the final conditions found with TSI and RKF5(6) with discontinuity step reduction.

Lowest heat-load trajectory of the first optimization

	H [-]	M [-]	E [-]	d [-]	Q [-]	J_{con} [-]
0	-1.504e-12	-9.616e-13	-1.790e-12	2.558e-11	-1.845e-13	0
1	-1.329e-12	-2.082e-12	-2.529e-12	2.484e-11	-2.235e-13	0
2	5.090e-11	6.066e-12	4.118e-11	3.948e-12	-2.922e-13	0
3	-3.027e-13	-1.312e-12	-1.174e-12	6.479e-13	-1.257e-13	0
4	6.412e-12	1.319e-11	1.491e-11	-1.864e-10	1.778e-13	0
5	-1.258e-12	-8.281e-13	-1.535e-12	2.233e-11	-1.508e-13	0
6	-1.329e-12	-1.088e-12	-1.803e-12	2.293e-11	-1.719e-13	0
7	-1.335e-12	1.869e-14	-7.481e-13	1.248e-11	-5.606e-14	0

Maximum downrange trajectory of the second optimization

	H [-]	M [-]	E [-]	R_d [-]	Q [-]	J_{con} [-]
0	1.432e-11	3.409e-11	4.044e-11	3.048e-12	1.447e-12	0
1	7.581e-11	4.202e-11	8.411e-11	2.563e-12	1.372e-12	-3.950e-3
2	7.736e-12	9.243e-12	1.326e-11	4.166e-13	1.399e-12	0
3	9.261e-13	-2.045e-13	3.526e-13	1.359e-13	1.080e-12	2.001e-4
4	1.420e-10	1.006e-10	1.779e-10	5.300e-13	8.027e-13	0
5	4.016e-11	3.049e-11	5.225e-11	2.567e-13	1.248e-12	0
6	6.188e-11	1.327e-10	1.608e-10	1.297e-12	4.664e-13	4.968e-3
7	1.930e-11	2.465e-11	3.436e-11	8.361e-13	7.894e-13	0

Minimum heat-load trajectory of the second optimization

	H [-]	M [-]	E [-]	R_d [-]	Q [-]	J_{con} [-]
0	1.063e-12	-1.027e-12	7.695e-15	6.515e-13	-4.017e-13	0
1	-3.900e-12	8.755e-13	-2.273e-12	1.267e-12	-4.744e-13	0
2	4.049e-11	-1.261e-11	2.085e-11	-1.262e-11	1.105e-12	4.963e-5
3	5.777e-10	-2.468e-10	2.444e-10	-4.257e-12	-2.799e-12	0
4	-2.993e-12	2.154e-12	-5.922e-13	3.390e-12	-1.043e-12	-1.336e-3
5	2.546e-10	-5.060e-11	1.263e-10	-1.126e-11	-1.107e-12	1.278e-3
6	1.447e-10	-5.925e-11	6.341e-11	2.569e-13	-7.584e-13	0
7	3.291e-12	-1.274e-12	1.527e-12	-9.095e-13	-3.060e-13	-1.714e-3

Chapter 9

Conclusion and Recommendations

Now that the results of the research have been obtained, the conclusions of the research and the recommendations for future research are given in this chapter.

9.1 Conclusions

The main research question of this thesis was formulated as:

Does Taylor Series Integration with automatic differentiation have a better performance for reentry trajectory propagation than traditional numerical integrators?

To answer this question, a TSI reentry integrator was programmed and verified, and its performance was compared to that of RKF5(6), which, in previous studies, was found to be the fastest of the traditional method for integrating reentry trajectories. To cope with discontinuities in the equations of motion, time steps were shortened whenever TSI integrates over a discontinuity, so that TSI does not accumulate errors by integrating a function that is no longer valid after the discontinuity. The end of the current time step is set at the discontinuity (in case of bank reversals or terminal distance) or a small safety distance past it (all other cases). The latter is to prevent the integrator from getting stuck at the discontinuity. Furthermore, regression lines were fitted through the discrete data of environment and aerodynamic models, to obtain an approximation to these models of which the mathematical expression was known.

Optimal Settings for Taylor Series Integration

The optimal step-size controller for TSI was found to be controller 1, which bases the step size on the magnitude of the last two terms of the Taylor series of each state variable, relative to the error tolerance. This step size controller was chosen, because of the fact that it can reliably satisfy error tolerances as low as $1 \cdot 10^{-12}$ for an arbitrary order. It was then found that a variable order strategy does not yield better computational times than a fixed order, except when using an error tolerance of $1 \cdot 10^{-14}$. In that case, the time reduction was about 1%, which was considered not worth the effort, as multiple parameters of the order controller had to be optimized to yield this result. Therefore, a fixed order control strategy was used, with the optimal orders obtained for each state-variable set and error tolerance obtained from timing integrations. From the results, it can be concluded that the optimal order in most cases increases as the error tolerance decreases. The optimal state-variable set was found to be the spherical set for both the equations of motion with and without wind.

For the equations with wind, two different approaches were developed to read the wind model; one directly reading the spline coefficients with which the model is defined and the other fitting

a least-squares regression curve through a number of samples of the model. With the latter, it turned out to be difficult to obtain results that are accurate enough to satisfy the error tolerance, without taking very small time steps and having high computational time. For the error tolerance of $1 \cdot 10^{-8}$, some solutions were found that yielded a decent performance, but this was still slower than using the spline coefficients, hence the spline method was selected.

Integration

For integration, TSI yielded lower computational times than RKF5(6) for all state-variable sets and for integration with wind and without. Furthermore, it turned out to have a relative better performance for low error tolerances; the ratio between computational times of RKF5(6) to TSI was 3.28 for $\epsilon = 1 \cdot 10^{-8}$ to 11.61 for $\epsilon = 1 \cdot 10^{-14}$ without wind, and 1.24 for $\epsilon = 1 \cdot 10^{-8}$ to 4.48 for $\epsilon = 1 \cdot 10^{-14}$ in case of wind. The better performance for lower tolerances is mainly attributed to TSI's ability to increase its order when the tolerance is decreased. The fact that the ratios are lower in case of wind is likely due to fact that TSI has to reduce additional steps whenever it reaches a new segment of the spline with which the wind model is defined. Also the optimal orders of TSI are lower for wind than for no wind, so they are closer to the order 5 of RKF5(6), so TSI gains less speed relative to RKF5(6) from having an adaptive order.

Optimization

The results of optimization were verified with TSI and RKF5(6) set to an error tolerance of $\epsilon = 1 \cdot 10^{-14}$. Compared to both low-tolerance integrators, TSI has smaller errors than RKF5(6), set to the same error tolerance. Also, during the optimization for minimum heat load and maximum downrange, RKF5(6) had 9 individuals in its final population of 100 individuals that violated one or more of the constraints according to the low-tolerance RKF5(6) integrator and TSI set to both $\epsilon = 1 \cdot 10^{-8}$ and $\epsilon = 1 \cdot 10^{-14}$. For TSI, a special mechanism was set up to detect constraint violations, even when making large time steps. This mechanism uses multiple root-finding methods to detect even the smallest constraint violations and from the results of optimization, it can be concluded that this mechanism works correctly. Furthermore, the constraint violations of RKF5(6) show that it needs a similar mechanism when optimizing with high tolerances to avoid accepting trajectories that turn out to violate constraints when integrated with higher precision.

Sensitivity Analysis

For the sensitivity analyses, both integrators were set to an error tolerance of $1 \cdot 10^{-14}$. This allowed for an analysis of the differences between the integrators when set to maximum precision (when using double precision floating point representation). When compared, the results of TSI and RKF5(6) have relative differences of up to $1 \cdot 10^{-3}$ at the end of the integration, which translates to absolute differences of up to $1 \cdot 10^{2}$ m in altitude and 1 m/s in velocity. The cause for these large differences (when compared to the tolerance the integrators were set to) turns out to be the fact that RKF5(6) also accumulates errors when integrating over a discontinuity. Therefore, it was decided to add discontinuity step reduction to RKF5(6), which causes the relative differences between the integrators to be less than $1 \cdot 10^{-10}$ and in most cases less than $1 \cdot 10^{-12}$ at the end of integration, for trajectories both with and without wind. From this, it can be concluded that TSI generates more accurate results than RKF5(6) without discontinuity step reduction. This also means that discontinuity step reduction is necessary for RKF methods when integrating with high accuracy, albeit that is not commonly done. Furthermore, this means

that the comparisons of integration speed in this report were done with an unfair advantage for RKF5(6), as discontinuity step reduction would slow down the integrator. However, as TSI was already faster, this does not change the general conclusion.

Overall, it can be concluded that TSI is faster than traditional integrators for reentry trajectory propagation. The use of TSI is advised for reentry applications such as integration, optimization and sensitivity analysis, as long as the application does not involve a guidance/navigation/control system that limits its step sizes to very small values.

9.2 Recommendations

The recommendations for future work are:

- Apply TSI to other cases of atmospheric flight, such as the ascending flight of space planes or parachute descent. TSI may especially yield large reductions in computational time when it is applied to long duration cruise flight, as these cases have long mission times and and very slow dynamics.
- Include the equations for rotational dynamics, so that attitude can be determined, for instance, during a free fall (of space debris or meteors).
- When including the rotational dynamics, one can also add a guidance or control system. The guidance system can be of a predictor-corrector type that makes a guess of the controls at the start of the step and then corrects this guess until the desired vehicle state is achieved during the step. TSI can provide the derivatives of any variable up to an arbitrary order, which may be of use for the guidance. Note that there will little point in using TSI with a controller that requires the integrator to make very small steps, as large time steps are the main reason TSI has low computational times.
- TSI could be used in an onboard system that supports the guidance system by re-optimizing the nominal trajectory when the vehicle deviates from the original nominal trajectory, as the optimization needs to happen in a short amount of time.
- TSI can be used in combination with interval analysis to obtain bounds on the truncation error of each integration step, allowing one to set up an interval within which the exact solution of the equations of motion lies, which is useful for a sensitivity analysis.
- When performing regression on the aerodynamics, split up the regime for Mach and the aerodynamic angles in different parts, rather than to try to fit mathematical expression through all points at once. This will make the equations for the aerodynamic coefficients simpler, which may actually speed up the integration time. Also, here, regression was only performed on Mach numbers of 2.5 and higher, but it turned out that the vehicle reaches lower Mach numbers, so the regression model of HORUS' aerodynamics should be extended to lower Mach numbers.
- The least-squares method for the wind implementation of TSI was slower than the spline method, but such a method does allow for the use of an environment model of which not all equations are known in advance. It may be of interest to expand upon the concept of fitting a model while integrating, to avoid having to fit all environment and aerodynamic models before integration.

- Compare the performance of possibly other state-variable sets than the ones used in this report, although it is at this point not known what sets could yield lower computational times for reentry integration.
- When one wants to integrate the motion of different vehicles or use other environment models (for other planets than Earth), the relevant equations and/or fits of data tables will have to transformed into recurrence relations and added to the integrator.

Bibliography

[1] Airey, J.R. "Emden Functions". *British Association for the Advancement of Science Mathematical Tables*, vol. II, 1932.

[2] Allen, H.J. and Eggers, A.J., Jr. A Study of the Motion and Aerodynamic Heating of Missiles Entering the Earth's Atmosphere. Technical report, NACA Report 1381, Washington D.C., 1958.

[3] Barrio, R. "Performance of the Taylor Series Method for ODEs/DAEs". *Applied Mathematics and Computation*, vol. 163, pp. 525-545, 2005.

[4] Barton, D., Willers, I.M. and Zahar, R.V.M. "Taylor Series Methods for Ordinary Differential Equations - an Evaluation". *Mathematical Software*, 1971.

[5] Barton, D., Willers, I.M. and Zahar, R.V.M. "The Automatic Solution of Ordinary Differential Equations by the Method of Taylor Series". *The Computer Journal*, vol. 14, 1971.

[6] Bergsma, M.C.W. Application of Taylor Series Integration to Reentry Problems, Literature study, Delft University of Technology, 2014.

[7] Chang, Y.F. "Automatic Solution of Differential Equations". *Lecture Notes in Mathematics*, vol. 430, 1974.

[8] Chapman, D., Vinh, N., Chern, J. and Lin, C. An Approximate Analytical Method for Studying Entry Into Planetary Atmospheres. Technical report, NACA, Washington, D.C., 1958.

[9] Corliss, G.F. and Chang, Y.F. "Ratio-Like and Recurrence Relation Tests for Convergence of Series". *IMA Journal of Applied Mathematics*, vol. 25:4, pp. 349-359, 1980.

[10] Corliss, G.F. and Chang, Y.F. "ATOMFT: Solving ODEs and DAEs Using Taylor Series". *Computers & Mathematics with Applications*, vol. 28:10, pp. 209-233, 1994.

[11] Corliss, G.F. and Chang, Y.F. "Solving Ordinary Differential Equations Using Taylor Series". *ACM Transactions on Mathematical Software*, vol. 8:2, pp. 114-144, 1982.

[12] Corliss, G.F. et al. "High-order Stiff ODE Solvers via Automatic Differentiation and Rational Prediction". *Lecture Notes in Computational Science*, vol. 1196, pp. 114-125, 1997.

[13] Das, S., Abraham, A. and Konar, A. "Particle Swarm Optimization and Differential Evolution Algorithms: Technical Analysis, Applications and Hybridization Perspectives". *Studies in Computational Intelligence*, 2008.

[14] De Ridder, S. Study on Optimal Trajectories and Energy Management Capabilities of a Winged Re-Entry Vehicle during the Terminal Area. Master's thesis, Delft University of Technology, 2009.

[15] Deb, K. Multi-objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems. Technical report, University of Dortmund, 1998.

[16] Deb, K., Agrawal, S., Pratap, A. and Meyarivan, T. "A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II". In *Lecture Notes in Computer Science 1917*. pp. 849-858, Springer, 2000.

[17] Deb, K. and Goyal, M. "A Combined Genetic Adaptive Search". *Computer Science and Informatics*, vol. 28, pp. 30-45, 1996.

[18] Deprit, A. and Zahar, R.V.M. "Numerical Integration of an Orbit and Its Concomitant Variations by Recurrent Power Series". *Zeitschrift für Angewandte Mathematik und Physik ZAMP*, vol. 17, pp. 425-430, 1966.

[19] Dijkstra, M. Trajectory Optimization of Hyperion-II for the Study of Hypersonic Aerothermodynamic Phenomena. Master's thesis, Delft University of Technology, 2012.

[20] Fehlberg, E. Classical Fifth-, Sixth-, Seventh-, and Eight-Order Runge-Kutta Formulas with Stepsize Control. Technical report, NASA, Washington, D. C., 1968.

[21] Feoktistov, V. *Differential Evolution - In Search of Solutions*. Springer, New York, 2006.

[22] Gibbons, A. "A program for the automatic integration of differential equations using the method of Taylor series". *The Computer Journal*, vol. 3, pp. 108-111, 1960.

[23] Golub, G. H., Reinsch, C. "Singular value decomposition and least squares solutions". *Numerische Mathematik*, vol. 14:5, pp. 403-420, 1970.

[24] Graillat, S., Langlois, P. and Louvet, N. Compensated Horner Scheme. Technical report, Université de Perpignan Via Domitia, 2005.

[25] Guckenheimer, J. and Choe, W.G. "Computing Periodic Orbits With High Accuracy". *Computer Methods in Applied Mechanics and Engineering*, vol. 170, pp. 331-341, 1999.

[26] Hoefkens, F., Berz, M. and Makino, K. "Computing Validated Solutions of Implicit Differential Equations". *Advances in Computational Mathematics*, vol. 19, pp. 231-253, 2003.

[27] Hull, T.E., Enright, W.H., Fellen, B.M. and Sedgwick, A.E. "Comparing Numerical Methods for Ordinary Differential Equations". *SIAM Journal on Numerical Analysis*, vol. 9:4, pp. 603-637, 1972.

[28] Irvine, D.H. and Savageau, M.A. "Efficient Solution of Nonlinear Ordinary Differential Equations Expressed in S-System Canonical Form". *SIAM Journal on Numerical Analysis*, vol. 27:3, pp. 704-735, 1990.

[29] Izzo, D. et al. PaGMO Documentation and Source Code, Last accessed on December 15, 2014. URL `github.com/esa/pagmo`.

[30] Jorba, A. and Zou, M. "A Software Package for the Numerical Integration of ODEs by Means of High-Order Taylor Methods". *Experimental Mathematics*, vol. 14:1, pp. 99-117, 2005.

[31] Justus, C.G. and Johnson, D.L. The NASA/MSFC Global Reference Atmospheric Model, 1999. URL `trs.nasa.gov/archive/00000503/`.

[32] Justus, C.G. and Leslie, F.W. The NASA MSFC Global Reference Atmospheric Model - 2007 Version, 2008.

[33] Kahan, W. Lecture Notes on the Status of: IEEE Standard 754 for Binary Floating-Point Arithmetic, Last accessed on November 19, 2013. URL `www.cs.berkeley.edu/~wkahan/ieee754status/IEEE754.PDF`.

[34] Klees, R. and Dwight, R.P. *Lecture Notes AE2212 Applied Numerical Analysis.* Delft University of Technology, 2012.

[35] Larsen, M.F. "Winds and Shears in the Mesosphere and Lower Thermosphere: Results from Four Decades of Chemical Release Wind Measurements". *Journal of Geophysical Research*, vol. 107:A8, 2002.

[36] Leavitt, J. "Methods and Applications of Power Series". *Mathematics of Computation*, vol. 20, pp. 46-52, 1966.

[37] Lederle, T. "The IAU (1976) System of Astronomical Constants". *NASA Astrophysics Data System*, 1980.

[38] Liu, B. et al. "An Enhanced MOEA/D-DE and Its Application to Multiobjective Analog Cell Sizing". *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pp. 1-7, 2010.

[39] Loh, W.H.T. *Re-Entry and Planetary Entry Physics and Technology I.* Sphringer Verlag, Berlin, 1968.

[40] Loh, W.H.T. *Re-Entry and Planetary Entry Physics and Technology II.* Sphringer Verlag, Berlin, 1969.

[41] Miller, J.C.P. "The Airy Integral". *British Association for the Advancement of Science Mathematical Tables*, vol. B, 1946.

[42] Moler, C. *Numerical Computing with MATLAB.* SIAM, Philadelphia, 2004.

[43] Mooij, E. *The Motion of a Vehicle in a Planetary Atmosphere.* Delft University Press, Delft, 1997.

[44] Mooij, E. *Aerospace-Plane Flight Dynamics: Analysis of Guidance and Control Concepts.* PhD thesis, Delft University of Technology, 1998.

[45] Mooij, E. Private Communication, 2013.

[46] Mooij, E. *Reentry Systems: Lecture Notes AE4870B.* Delft University of Technology, 2013.

[47] Mooij, E. "The HORUS-2B Reference Vehicle", Delft University of Technology, Memorandum M-682, 1995.

[48] Moore, R.E. *Interval Analysis.* Prentice-Hall, Englewod Cliffs, N.J., 1966.

[49] Moore, R.E. *Methods and Applications of Interval Analysis.* SIAM, Philadelphia, 1979.

[50] Mulder, J.A. et al. *Flight Dynamics: Lecture Notes AE3202.* Delft University of Technology, 2013.

[51] Nedialkov, N.S., Jackson, K.R. and Corliss, G.F. "Validated Solutions of Initial Value Problems for Ordinary Differential Equations". *Applied Mathematics and Computation*, vol. 105, pp. 21-68, 1999.

[52] NIMA. World Geodetic System 1984: Its Definition and Relationships with Local Geodetic Systems. Technical report, NIMA, 2000.

[53] NOAA, NASA, and USAF. *U.S. Standard Atmosphere, 1976.* Washington, D.C., 1976.

[54] Noomen, R. AE4878 Space Mission Design Slides: Integrators, Delft University of Technology, v. 4-3, 2013.

[55] Noomen, R. AE4878 Space Mission Design Slides: Optimization, Delft University of Technology, v. 4-7, 2013.

[56] Norman, A.C. "Expanding the Solutions of Implicit Sets of Ordinary Differential Equations in Power Series". *The Computer Journal*, vol. 19, 1976.

[57] Papp, Z. Mission Planner for Heating-Optimal Re-Entry Trajectories with Extended Range Capability. Master's thesis, Delft University of Technology, 2014.

[58] Phipps, E., Casey, R. and Guckenheimer, J. "Period Orbits of Hybrid Systems and Parameter Estimation via AD". In *Automatic Differentiation: Applications, Theory, and Implementations*. vol. 50, pp. 211-223, Springer Berlin Heidelberg, 2006.

[59] Pryce, J.D. "Solving High-index DAEs by Taylor Series". *Numerical Algorithms*, vol. 19, pp. 195-211, 1998.

[60] Römgens, B., Mooij, E. and Naeije, M.C. "Satellite Collision Prediction Using Verified Interval Orbit Propagation". *Journal of Guidance, Control and Dynamics*, vol. 36, pp. 821-832, 2013.

[61] Sänger, E. and Bredt, J. *A Rocket Drive for Long Range Bombers*. Technical Information Branch, U.S. Navy Bureau of Aeronautics, Translation CGD-32 (1955), 1944.

[62] Scott, J.R. and Martini, M.C. "High Speed Solution of Spacecraft Trajectory Problems Using Taylor Series Integration". In *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, 2008.

[63] Spaans, J. Improving Global Optimization Methods for Low Thrust Trajectories. Master's thesis, Delft University of Technology, 2009.

[64] Stoer, J. and Bulirsch, R. *Introduction to Numerical Analysis*. Springer-Verlag, 2nd edition, 1993.

[65] Tapley, B. et al. "GGM02 - An improved Earth gravity field model from GRACE". *Journal of Geodesy*, vol. 79:8, pp. 467-478, 2005.

[66] Vinh, N.X., Busemann, A. and Culp, R.D. *Hypersonic and Planetary Entry Flight Mechanics*. University of Michigan Press, 1980.

[67] Wakker, K.F. *Astrodynamics I*. Delft University of Technology, 2010.

[68] Wakker, K.F. *Astrodynamics II*. Delft University of Technology, 2010.

[69] Wertz, J.R. *Orbit & Constellation Design & Management*. Microcosm Press, 2009.

[70] Yokoyama, N. and Suzuki, S. "Modified Genetic Algorithm for Constrained Trajectory Optimization". *Journal of Guidance, Control and Dynamics*, vol. 26:1, pp. 139-144, 2005.

[71] Zhang, Q. and Li, H. "MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition". *IEEE Transactions on Evolutionary Computation*, vol. 11:6, 2007.

Appendix A

Tables of US76

In this Appendix, the relevant tables from [53] are given. The Table A.1 gives the correction to the molecular mass ratio M/M_0 for the geometric altitude range 80-86 km. This ratio is used to determine the temperature T from molecular temperature T_M. Note that in [53], the geometric and geopotential altitudes are denoted by Z and H, respectively, whereas here they are denoted by H and z, respectively. Furthermore, it was determined in Subsection 6.4.3 that this table is not needed for the current problems, as temperature is not explicitly needed. However, this table is still given here for the sake of completeness. Table A.2 then gives the total number density N, mean molecular mass M and air density ρ as function of geometric altitude for the altitude range 86-125 km.

Table A.1: Molecular mass ratio vs. geopotential and geometric altitude, Table 8 of [53]

z [m]	h [m]	M/M_0	h [m]	z [m]	M/M_0
79000	79994.1	1.000000	80000	79005.7	1.000000
79500	80506.8	0.999996	80500	79493.3	0.999996
80000	81019.6	0.999988	81000	79980.9	0.999989
80500	81532.5	0.999969	81500	80468.3	0.999971
81000	82045.5	0.999938	82000	80955.7	0.999941
81500	82558.5	0.999904	82500	81443.0	0.999909
82000	83071.6	0.999864	83000	81930.2	0.999870
82500	83584.8	0.999822	83500	82417.4	0.999829
83000	84098.1	0.999778	84000	82904.5	0.999786
83500	84611.4	0.999731	84500	83391.5	0.999741
84000	85124.9	0.999681	85000	83878.4	0.999694
84500	85638.4	0.999679	85500	84365.3	0.999641
			86000	84852.1	0.999579

Table A.2: Total number density, molecular mass and air density as function of geometric altitude, distilled from Table I and II of [53]

H [m]	N [m^{-3}]	M [kg/mol]	ρ [kg/m^3]	H [m]	N [m^{-3}]	M [kg/mol]	ρ [kg/m^3]
86000	$1.447 \cdot 10^{20}$	0.02895	$6.958 \cdot 10^{8}$	100000	$1.189 \cdot 10^{19}$	0.02840	$5.609 \cdot 10^{7}$
86500	$1.324 \cdot 10^{20}$	0.02895	$6.366 \cdot 10^{8}$	101000	$9.990 \cdot 10^{18}$	0.02830	$4.695 \cdot 10^{7}$
87000	$1.212 \cdot 10^{20}$	0.02895	$5.824 \cdot 10^{8}$	102000	$8.402 \cdot 10^{18}$	0.02821	$3.935 \cdot 10^{7}$
87500	$1.109 \cdot 10^{20}$	0.02894	$5.328 \cdot 10^{8}$	103000	$7.071 \cdot 10^{18}$	0.02810	$3.300 \cdot 10^{7}$
88000	$1.014 \cdot 10^{20}$	0.02894	$4.875 \cdot 10^{8}$	104000	$5.956 \cdot 10^{18}$	0.02800	$2.769 \cdot 10^{7}$
88500	$9.284 \cdot 10^{19}$	0.02893	$4.460 \cdot 10^{8}$	105000	$5.021 \cdot 10^{18}$	0.02788	$2.325 \cdot 10^{7}$
89000	$8.496 \cdot 10^{19}$	0.02893	$4.081 \cdot 10^{8}$	106000	$4.237 \cdot 10^{18}$	0.02777	$1.954 \cdot 10^{7}$
89500	$7.775 \cdot 10^{19}$	0.02892	$3.734 \cdot 10^{8}$	107000	$3.578 \cdot 10^{18}$	0.02764	$1.643 \cdot 10^{7}$
90000	$7.116 \cdot 10^{19}$	0.02891	$3.416 \cdot 10^{8}$	108000	$3.023 \cdot 10^{18}$	0.02752	$1.381 \cdot 10^{7}$
90500	$6.513 \cdot 10^{19}$	0.02890	$3.126 \cdot 10^{8}$	109000	$2.552 \cdot 10^{18}$	0.02739	$1.161 \cdot 10^{7}$
91000	$5.962 \cdot 10^{19}$	0.02889	$2.860 \cdot 10^{8}$	110000	$2.144 \cdot 10^{18}$	0.02727	$9.708 \cdot 10^{6}$
91500	$5.456 \cdot 10^{19}$	0.02887	$2.616 \cdot 10^{8}$	111000	$1.800 \cdot 10^{18}$	0.02714	$8.111 \cdot 10^{6}$
92000	$4.993 \cdot 10^{19}$	0.02886	$2.393 \cdot 10^{8}$	112000	$1.524 \cdot 10^{18}$	0.02702	$6.838 \cdot 10^{6}$
92500	$4.568 \cdot 10^{19}$	0.02884	$2.188 \cdot 10^{8}$	113000	$1.301 \cdot 10^{18}$	0.02690	$5.811 \cdot 10^{6}$
93000	$4.178 \cdot 10^{19}$	0.02882	$2.000 \cdot 10^{8}$	114000	$1.118 \cdot 10^{18}$	0.02679	$4.975 \cdot 10^{6}$
93500	$3.821 \cdot 10^{19}$	0.02880	$1.828 \cdot 10^{8}$	115000	$9.681 \cdot 10^{17}$	0.02668	$4.289 \cdot 10^{6}$
94000	$3.494 \cdot 10^{19}$	0.02878	$1.670 \cdot 10^{8}$	116000	$8.430 \cdot 10^{17}$	0.02658	$3.720 \cdot 10^{6}$
94500	$3.194 \cdot 10^{19}$	0.02876	$1.526 \cdot 10^{8}$	117000	$7.382 \cdot 10^{17}$	0.02648	$3.246 \cdot 10^{6}$
95000	$2.920 \cdot 10^{19}$	0.02873	$1.393 \cdot 10^{8}$	118000	$6.498 \cdot 10^{17}$	0.02638	$2.847 \cdot 10^{6}$
95500	$2.669 \cdot 10^{19}$	0.02871	$1.273 \cdot 10^{8}$	119000	$5.748 \cdot 10^{17}$	0.02629	$2.509 \cdot 10^{6}$
96000	$2.440 \cdot 10^{19}$	0.02868	$1.162 \cdot 10^{8}$	120000	$5.107 \cdot 10^{17}$	0.02620	$2.222 \cdot 10^{6}$
96500	$2.230 \cdot 10^{19}$	0.02865	$1.061 \cdot 10^{8}$	121000	$4.558 \cdot 10^{17}$	0.02612	$1.977 \cdot 10^{6}$
97000	$2.038 \cdot 10^{19}$	0.02862	$9.685 \cdot 10^{7}$	122000	$4.086 \cdot 10^{17}$	0.02604	$1.767 \cdot 10^{6}$
97500	$1.862 \cdot 10^{19}$	0.02859	$8.842 \cdot 10^{7}$	123000	$3.677 \cdot 10^{17}$	0.02596	$1.585 \cdot 10^{6}$
98000	$1.702 \cdot 10^{19}$	0.02855	$8.071 \cdot 10^{7}$	124000	$3.323 \cdot 10^{17}$	0.02588	$1.428 \cdot 10^{6}$
98500	$1.556 \cdot 10^{19}$	0.02852	$7.367 \cdot 10^{7}$	125000	$3.013 \cdot 10^{17}$	0.02580	$1.291 \cdot 10^{6}$
99000	$1.422 \cdot 10^{19}$	0.02848	$6.725 \cdot 10^{7}$				
99500	$1.300 \cdot 10^{19}$	0.02844	$6.141 \cdot 10^{7}$				

Appendix B

Aerodynamic Tables of HORUS-2B

Subsection 3.1.2 described how the trimming of HORUS was included in the tables for C_D and C_L. The outcome of this process is shown in Tables B.1 and B.2, where α_A is the airspeed-based angle of attack and M is the Mach number. For the combination of Mach 10 and zero angle of attack, the C_{m_0} of HORUS was too negative to trim even with the combined full deflection of the body flap and both elevons. In the tables, these entries are replaced with "N.T." (Not Trimmable).

Table B.1: Trimmed values of C_D

α_A \ M	2.5	3	5	10	20
0°	0.08470	0.08333	0.08412	N.T.	0.05953
5°	0.08735	0.08173	0.07123	0.06977	0.04984
10°	0.10604	0.10217	0.09134	0.08563	0.06775
15°	0.16072	0.15000	0.14852	0.13485	0.10878
20°	0.25026	0.23345	0.21797	0.20614	0.16809
25°	0.37345	0.35738	0.33292	0.32228	0.27443
30°	0.54106	0.51840	0.47563	0.45251	0.39896
35°	0.73229	0.69805	0.65370	0.63111	0.55915
40°	0.95237	0.90449	0.84083	0.81343	0.73430
45°	1.24844	1.20502	1.06323	1.03595	0.94539

Table B.2: Trimmed values of C_L

α_A \ M	2.5	3	5	10	20
0°	-0.07738	-0.07891	-0.07592	N.T.	-0.07538
5°	0.03420	0.03033	0.03522	0.02344	0.01108
10°	0.18807	0.17618	0.14709	0.13767	0.08889
15°	0.34640	0.33000	0.27916	0.25816	0.19336
20°	0.51533	0.48721	0.42705	0.39527	0.31661
25°	0.67473	0.63141	0.56732	0.53448	0.43708
30°	0.82894	0.78018	0.70423	0.65904	0.56297
35°	0.96049	0.90768	0.81801	0.77289	0.68756
40°	1.07919	1.01352	0.92273	0.87495	0.78885
45°	1.14607	1.07703	0.99608	0.95698	0.86098

Appendix C

2D Kepler Orbit

Here, the equations of motion for an elliptic Kepler orbit are given for both Cartesian components, as well as for Kepler elements, together with the conversions between the two. Equations in this chapter were obtained from [67].

Kepler Elements

The shape of a 2D Kepler orbit is defined by semi-major axis a and eccentricity e. These remain constant for the entire orbit. For an elliptic orbit, the eccentricity must be smaller than 1. The position in this orbit is then fixed through an anomaly angle, either true anomaly θ, eccentric anomaly E or mean anomaly M. The meaning of some of these variables is shown in Figure C.1. In this figure, the central body is indicated by F, the spacecraft by S and b is the semi-minor axis. The relations between the different anomalies are given by:

$$\tan\frac{\theta}{2} = \sqrt{\frac{1+e}{1-e}}\tan\frac{E}{2} \tag{C.1}$$

$$M = E - e\sin E \tag{C.2}$$

Eq. (C.1) can easily be modified to obtain θ from E or vice versa. Eq. (C.2) on the other hand cannot be inverted to obtain E from M. This means that a root-finding method must be deployed. Using Newton's method (see Section 4.1.2) on Eq. (C.2) yields:

$$E_{i+1} = E_i + \frac{M - E_i + e\sin E_i}{1 - e\cos E_i} \tag{C.3}$$

The initial guess for E can be $E_0 = M$. The benefit of using M is that it changes linearly with time, which allows one to determine the anomaly at an arbitrary point in time:

$$M = M_0 + n(t - t_0) \tag{C.4}$$

where M_0 is the mean anomaly at some time t_0. n is the mean angular motion, which is given by:

$$n = \sqrt{\frac{\mu}{a^3}} \tag{C.5}$$

where μ indicates the gravitational parameter of the central body.

Equations of Motion

The position is given by the inertial components x and y, which are here defined in an axis system with the origin at the location of the central body and the X-axis pointing along the semi-major axis, as shown in Figure C.1. The velocities in X and Y-direction are u and v,

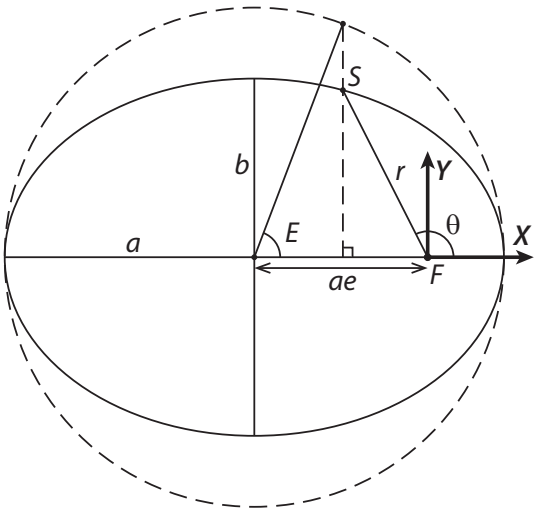


Figure C.1: An elliptic Kepler orbit

respectively. The equations of motion of a Kepler orbit with Cartesian position and velocity in inertial space are given by the inverse square gravity law:

$$\frac{d}{dt}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} u \\ v \end{bmatrix} \tag{C.6}$$

$$\frac{d}{dt}\begin{bmatrix} u \\ v \end{bmatrix} = -\frac{\mu_E}{r^3}\begin{bmatrix} x \\ y \end{bmatrix} \tag{C.7}$$

Conversions

Finally, the conversions between Cartesian components and Kepler elements are given., starting with the conversion of Cartesian components to Kepler elements. Semi-major axis a and eccentricity e are found from:

$$a = \frac{1}{2/r - V^2/\mu} \tag{C.8}$$

$$e = \sqrt{1 - \frac{H_{ang}^2}{a\mu}} \tag{C.9}$$

Where V indicates the magnitude of the velocity vector, r the radial distance (magnitude of the position vector) and H_{ang} is the angular momentum, given by:

$$H_{ang} = \boldsymbol{r} \times \boldsymbol{V} = xv - yu \tag{C.10}$$

θ is then given by:

$$\theta = \operatorname{atan2}(y, x) \tag{C.11}$$

The conversion from Kepler elements to Cartesian position starts with computing the radial distance r using:

$$r = \frac{a(1-e^2)}{1 + e\cos\theta} = a(1 - e\cos E) \tag{C.12}$$

The position is then given by:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} r\cos\theta \\ r\sin\theta \end{bmatrix} \tag{C.13}$$

For the Cartesian velocity components, first the angular momentum must be determined:

$$H_{ang} = \sqrt{a\mu(1-e^2)} \tag{C.14}$$

Note from this equation that the sign of H_{ang} is always positive, since the Kepler elements a and e do not determine whether the orbit is clockwise or counterclockwise. The velocity components are then given by:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{\mu}{H_{ang}} \begin{bmatrix} -\sin\theta \\ -e - \cos\theta \end{bmatrix} \tag{C.15}$$

Appendix D

Fixed Order Timing

In the Tables D.1 to D.4, the results of the timing of the integration of 5,000 trajectories for each state-variable set are given. The numbers on the left of each table represent the (fixed) order and the numbers on top represent the error tolerances. Note that the error tolerance of $1 \cdot 10^{-15}$ is missing for the last two state-variable sets, as their times were much larger for this tolerance (in the order of minutes) and without a clear hierarchy. This indicates that the integrator could not satisfy this tolerance. In Tables D.5 to D.7, the times of TSI to integrate 2,000 trajectories with wind using the spline coefficients of the wind model are given. The values in each of the tables are the average of five or more separate runs, each with its own random seed. The best few orders for each tolerance have been checked with extra runs to verify that each table indeed shows the proper hierarchy of orders for each tolerance.

Table D.1: Time (in seconds) taken to integrate 5,000 trajectories using Cartesian state variables in $\mathcal{F}_I$.

	$1{\cdot}10^{-8}$	$1{\cdot}10^{-9}$	$1{\cdot}10^{-10}$	$1{\cdot}10^{-11}$	$1{\cdot}10^{-12}$	$1{\cdot}10^{-13}$	$1{\cdot}10^{-14}$	$1{\cdot}10^{-15}$
8	3.396	4.503	6.079	8.226	11.222	15.309	20.946	28.512
9	3.026	3.838	4.956	6.422	8.382	10.878	14.321	18.554
10	2.881	3.567	4.394	5.548	7.030	8.830	11.107	13.972
11	2.829	3.505	4.212	5.091	6.266	7.764	9.495	11.466
12	2.912	3.437	4.160	4.898	5.829	7.062	8.502	10.067
13	2.985	3.448	4.139	4.805	5.621	6.552	7.904	9.131
14	3.125	3.562	4.082	4.831	5.528	6.344	7.374	8.658
15	3.203	3.702	4.170	4.836	5.548	6.297	7.093	8.050
16	3.390	3.817	4.300	4.779	5.564	6.252	7.098	7.920
17	3.526	3.910	4.472	4.904	5.455	6.279	7.028	7.748
18	3.734	4.056	4.649	5.112	5.595	6.261	7.036	7.712
19	3.947	4.306	4.758	5.288	5.756	6.244	7.110	7.727
20	4.170	4.519	4.930	5.444	5.970	6.432	7.016	7.816
21	4.404	4.763	5.138	5.658	6.136	6.588	7.176	7.800
22	4.659	4.956	5.382	5.751	6.313	6.833	7.389	7.883
23	4.904	5.273	5.647	5.949	6.620	7.046	7.592	8.003
24	5.184	5.538	5.897	6.245	6.692	7.327	7.800	8.211
25	5.444	5.814	6.230	6.505	6.895	7.519	8.096	8.429

Table D.2: Time (in seconds) taken to integrate 5,000 trajectories using Cartesian state variables in $\mathcal{F}_R$.

	$1{\cdot}10^{-8}$	$1{\cdot}10^{-9}$	$1{\cdot}10^{-10}$	$1{\cdot}10^{-11}$	$1{\cdot}10^{-12}$	$1{\cdot}10^{-13}$	$1{\cdot}10^{-14}$	$1{\cdot}10^{-15}$
8	3.723	4.961	6.677	9.121	12.431	17.015	23.093	31.725
9	3.354	4.274	5.533	7.238	9.386	12.277	15.964	20.850
10	3.214	4.035	4.976	6.276	7.972	10.033	12.532	15.824
11	3.136	3.926	4.711	5.756	7.062	8.666	10.629	13.096
12	3.193	3.791	4.623	5.517	6.503	7.904	9.412	12.145
13	3.245	3.768	4.550	5.398	6.146	7.280	8.679	10.462
14	3.390	3.890	4.436	5.413	6.092	6.963	8.091	9.487
15	3.468	4.051	4.612	5.273	6.029	6.937	7.909	8.944
16	3.619	4.134	4.685	5.221	6.001	6.749	7.675	8.653
17	3.801	4.238	4.846	5.398	5.957	6.807	7.608	8.525
18	3.988	4.399	5.013	5.559	6.045	6.743	7.582	8.403
19	4.212	4.612	5.054	5.767	6.175	6.755	7.701	8.377
20	4.430	4.846	5.273	5.902	6.354	6.937	7.571	8.414
21	4.690	5.054	5.507	6.032	6.557	7.106	7.665	8.397
22	4.940	5.320	5.736	6.172	6.776	7.340	7.862	8.523
23	5.190	5.554	5.970	6.365	7.010	7.574	8.086	8.583
24	5.444	5.855	6.214	6.729	7.080	7.792	8.268	8.801
25	5.751	6.152	6.542	6.932	7.342	8.029	8.564	9.438

Table D.3: Time (in seconds) taken to integrate 5,000 trajectories using spherical state variables.

	$1{\cdot}10^{-8}$	$1{\cdot}10^{-9}$	$1{\cdot}10^{-10}$	$1{\cdot}10^{-11}$	$1{\cdot}10^{-12}$	$1{\cdot}10^{-13}$	$1{\cdot}10^{-14}$
8	2.642	3.474	4.706	6.295	8.486	11.690	15.961
9	2.439	3.052	3.942	5.052	6.484	8.559	11.287
10	2.288	2.896	3.583	4.456	5.548	6.994	8.915
11	2.335	2.834	3.420	4.142	4.961	6.167	7.595
12	2.408	2.803	3.416	4.004	4.716	5.746	6.877
13	2.465	2.886	3.315	3.978	4.566	5.335	6.344
14	2.527	2.943	3.319	3.926	4.467	5.255	6.001
15	2.673	3.068	3.440	3.905	4.514	5.273	5.886
16	2.777	3.130	3.588	3.960	4.508	5.262	5.803
17	2.933	3.271	3.718	4.074	4.441	5.205	5.725
18	3.125	3.401	3.895	4.220	4.602	5.151	5.792
19	3.286	3.578	3.890	4.381	4.722	5.153	5.766
20	3.489	3.744	4.087	4.514	4.914	5.369	5.727
21	3.692	3.968	4.248	4.631	5.065	5.525	5.985
22	3.890	4.155	4.477	4.781	5.210	5.725	6.178
23	4.077	4.436	4.670	4.992	5.356	5.938	6.263
24	4.332	4.633	4.904	5.182	5.481	6.042	6.531
25	4.545	4.867	5.169	5.416	5.710	6.201	6.716

Table D.4: Time (in seconds) taken to integrate 5,000 trajectories using SPCV.

	$1{\cdot}10^{-8}$	$1{\cdot}10^{-9}$	$1{\cdot}10^{-10}$	$1{\cdot}10^{-11}$	$1{\cdot}10^{-12}$	$1{\cdot}10^{-13}$	$1{\cdot}10^{-14}$
8	4.139	5.543	8.351	10.301	14.201	19.563	26.770
9	3.453	4.472	6.438	7.535	9.922	13.109	17.197
10	3.141	3.921	5.403	6.250	7.966	10.041	12.834
11	3.047	3.718	4.508	5.574	6.874	8.383	10.431
12	2.995	3.593	4.298	5.101	6.214	7.545	9.048
13	2.938	3.536	4.154	4.831	5.764	6.890	8.102
14	3.006	3.463	4.103	4.768	5.456	6.406	7.576
15	3.104	3.536	4.083	4.727	5.468	6.152	7.063
16	3.234	3.604	4.056	4.680	5.382	6.074	6.778
17	3.266	3.708	4.126	4.623	5.355	6.032	6.646
18	3.401	3.910	4.274	4.696	5.390	6.006	6.622
19	3.552	3.952	4.472	4.815	5.495	6.063	6.556
20	3.692	4.056	4.581	4.992	5.577	5.985	6.657
21	3.864	4.212	4.649	5.169	5.736	6.081	6.587
22	4.077	4.368	4.732	5.210	5.876	6.162	6.657
23	4.280	5.018	4.883	5.351	6.058	6.266	6.770
24	4.456	5.366	5.091	5.528	6.167	6.495	6.926
25	4.696	5.512	5.335	5.684	6.209	6.666	7.030

Table D.5: Time (in seconds) taken to integrate 2,000 trajectories with wind using Cartesian state variables in $\mathcal{F}_I$.

	$1{\cdot}10^{-8}$	$1{\cdot}10^{-9}$	$1{\cdot}10^{-10}$	$1{\cdot}10^{-11}$	$1{\cdot}10^{-12}$	$1{\cdot}10^{-13}$	$1{\cdot}10^{-14}$
6	4.390	6.443	9.775	14.923	23.101	36.049	56.525
7	3.445	4.599	6.324	8.777	12.324	17.619	25.191
8	3.170	3.916	5.029	6.561	8.599	11.450	15.372
9	3.142	3.716	4.487	5.513	6.989	8.699	11.167
10	3.295	3.744	4.409	5.142	6.178	7.600	9.226
11	3.557	3.937	4.483	5.117	5.822	6.920	8.249
12	3.913	4.209	4.571	5.161	5.856	6.646	7.641
13	4.359	4.580	4.927	5.363	5.903	6.633	7.438
14	4.774	5.011	5.323	5.638	6.112	6.730	7.541
15	5.307	5.532	5.756	6.056	6.424	6.914	7.560
16	5.853	6.037	6.246	6.487	6.823	7.257	7.794
17	6.412	6.596	6.799	6.967	7.276	7.632	8.031
18	7.070	7.223	7.385	7.607	7.822	8.165	8.543
19	7.688	7.850	7.965	8.171	8.408	8.617	9.036
20	8.408	8.571	8.714	8.842	9.107	9.282	9.569

Table D.6: Time (in seconds) taken to integrate 2,000 trajectories with wind using Cartesian state variables in $\mathcal{F}_R$.

	$1{\cdot}10^{-8}$	$1{\cdot}10^{-9}$	$1{\cdot}10^{-10}$	$1{\cdot}10^{-11}$	$1{\cdot}10^{-12}$	$1{\cdot}10^{-13}$	$1{\cdot}10^{-14}$
6	4.315	6.380	9.628	14.780	23.020	35.531	55.833
7	3.379	4.574	6.284	8.692	12.305	17.410	25.051
8	3.142	3.866	5.020	6.552	8.667	11.326	15.329
9	3.111	3.707	4.455	5.507	7.076	8.671	11.101
10	3.264	3.766	4.393	5.145	6.262	7.579	9.232
11	3.579	3.944	4.446	5.167	5.962	6.914	8.352
12	3.909	4.249	4.668	5.226	6.034	6.702	7.759
13	4.331	4.615	4.964	5.463	6.075	6.696	7.507
14	4.811	5.070	5.360	5.763	6.221	6.811	7.600
15	5.292	5.519	5.834	6.125	6.518	6.992	7.716
16	5.872	6.128	6.331	6.611	6.936	7.366	7.906
17	6.427	6.633	6.870	7.145	7.410	7.756	8.243
18	7.126	7.345	7.516	7.756	7.909	8.259	8.661
19	7.834	7.928	8.159	8.334	8.518	8.789	9.182
20	8.455	8.661	8.861	9.051	9.213	9.407	9.722

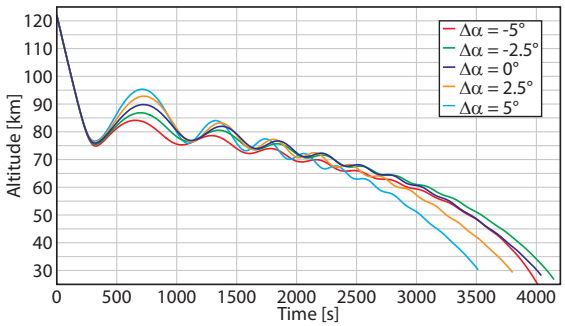
Table D.7: Time (in seconds) taken to integrate 2,000 trajectories with wind using spherical state variables.

	$1{\cdot}10^{-8}$	$1{\cdot}10^{-9}$	$1{\cdot}10^{-10}$	$1{\cdot}10^{-11}$	$1{\cdot}10^{-12}$	$1{\cdot}10^{-13}$	$1{\cdot}10^{-14}$
6	3.312	4.781	7.067	10.668	16.445	25.834	39.895
7	2.813	3.635	4.917	6.711	9.355	13.387	18.972
8	2.652	3.292	4.033	5.242	6.794	8.980	11.981
9	2.688	3.130	3.728	4.482	5.671	7.038	8.954
10	2.904	3.208	3.671	4.311	5.096	6.237	7.571
11	3.237	3.474	3.838	4.342	5.049	5.811	6.960
12	3.598	3.796	4.092	4.485	5.067	5.736	6.560
13	4.059	4.181	4.425	4.724	5.187	5.790	6.516
14	4.472	4.620	4.789	5.044	5.463	5.931	6.588
15	4.919	5.073	5.268	5.463	5.811	6.185	6.716
16	5.437	5.572	5.717	5.915	6.245	6.526	6.898
17	5.972	6.076	6.237	6.425	6.640	6.903	7.277
18	6.523	6.664	6.807	7.004	7.181	7.439	7.712
19	7.116	7.257	7.420	7.556	7.764	7.940	8.213
20	7.745	7.870	8.034	8.122	8.343	8.499	8.775

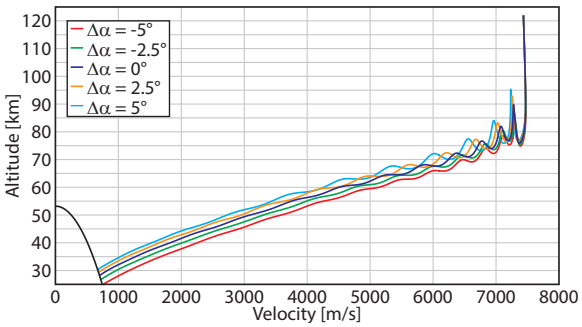
Appendix E

Sensitivity Analysis Graphs

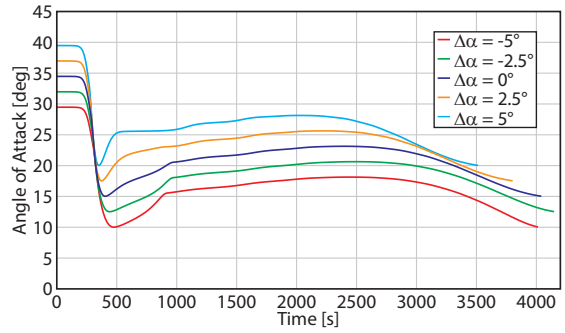
Here, the graphs of the control perturbation sensitivity analyses are shown. The wind perturbations only cause subtle changes that are hard to visually distinguish from each other in graphs, hence those graphs were left out.



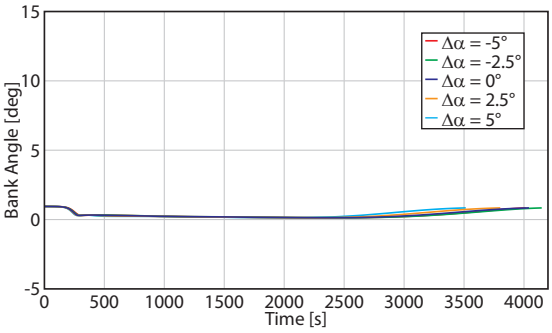
(a) Altitude versus time



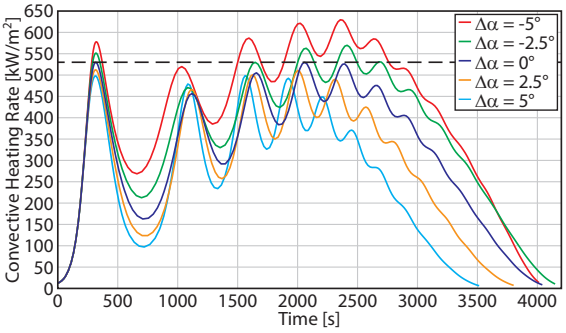
(b) Altitude versus velocity



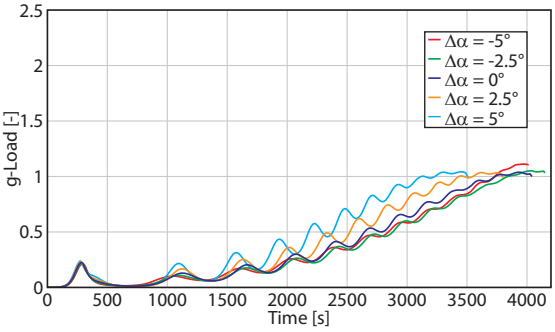
(c) Angle of attack versus time



(d) Bank angle versus time

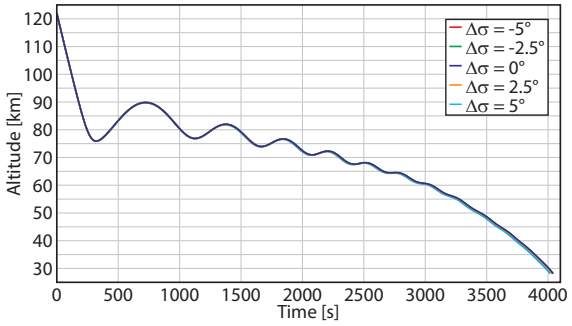


(e) Convective nose heating rate versus time

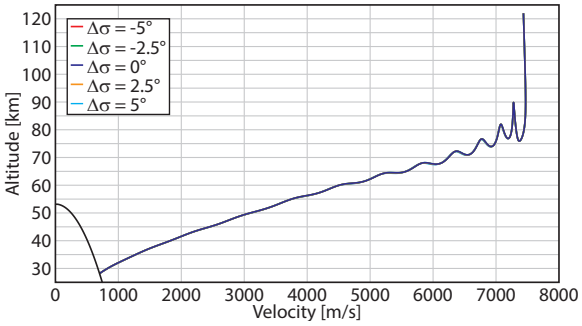


(f) Load factor versus time

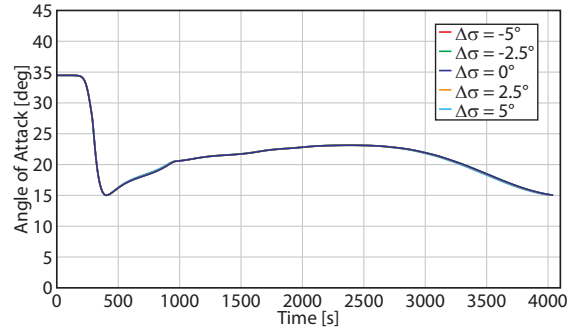
Figure E.1: Sensitivity analysis of the maximum downrange trajectory of the second optimization to perturbations in α_G.



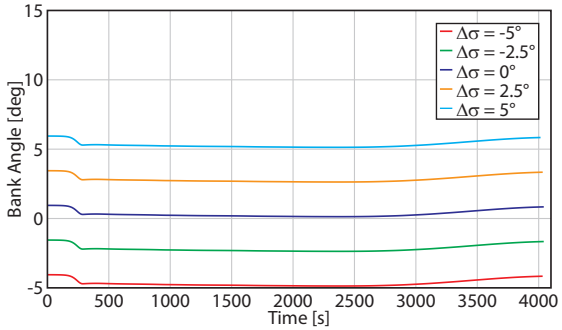
(a) Altitude versus time



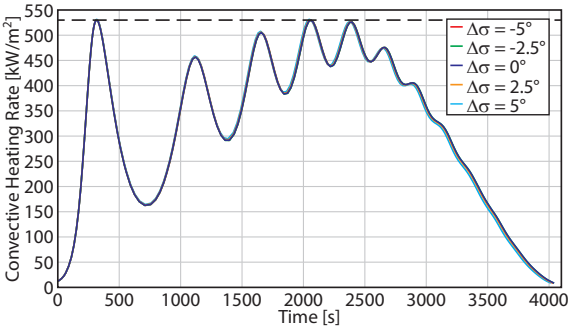
(b) Altitude versus velocity



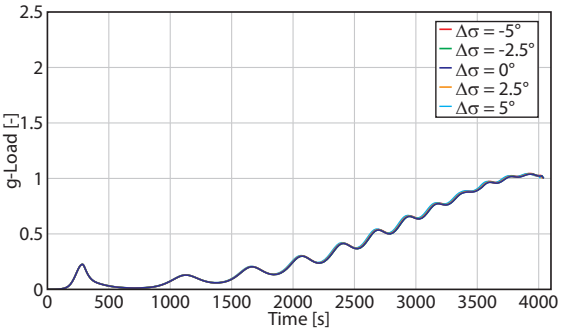
(c) Angle of attack versus time



(d) Bank angle versus time

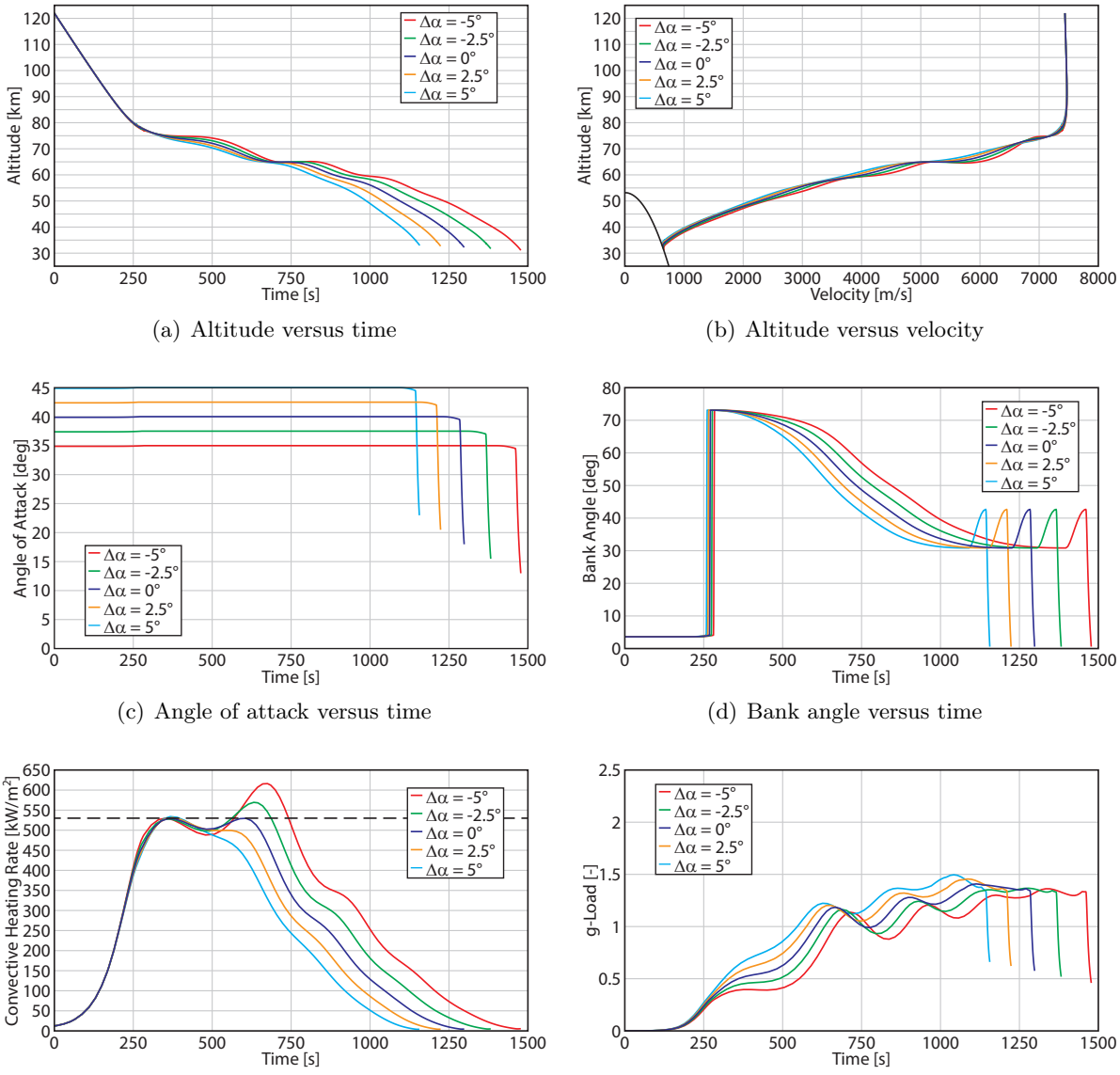


(e) Convective nose heating rate versus time



(f) Load factor versus time

Figure E.2: Sensitivity analysis of the maximum downrange trajectory of the second optimization to perturbations in σ_G.



(a) Altitude versus time

(b) Altitude versus velocity

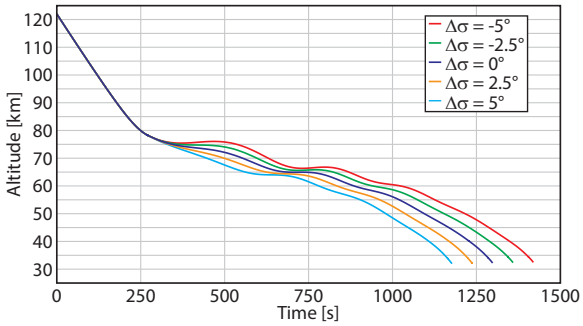
(c) Angle of attack versus time

(d) Bank angle versus time

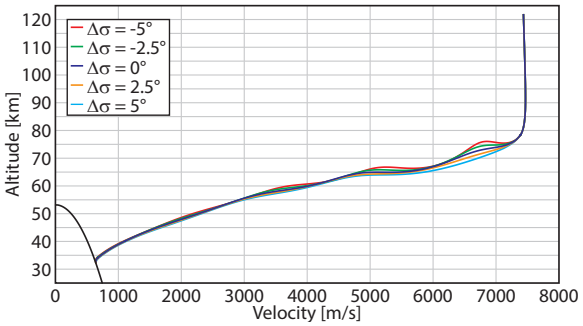
(e) Convective nose heating rate versus time

(f) Load factor versus time

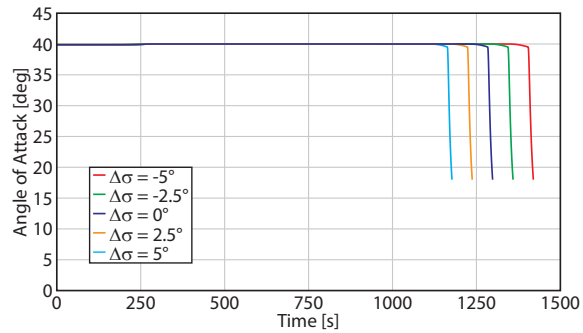
Figure E.3: Sensitivity analysis of the minimum heat load trajectory of the second optimization to perturbations in α_G.



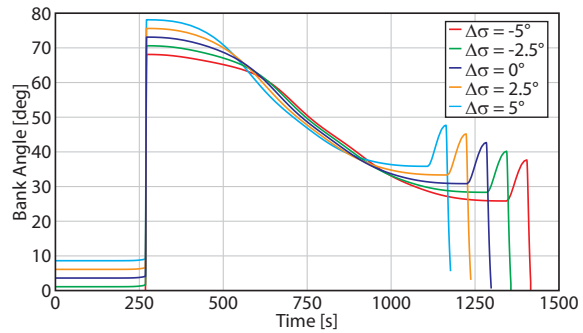
(a) Altitude versus time



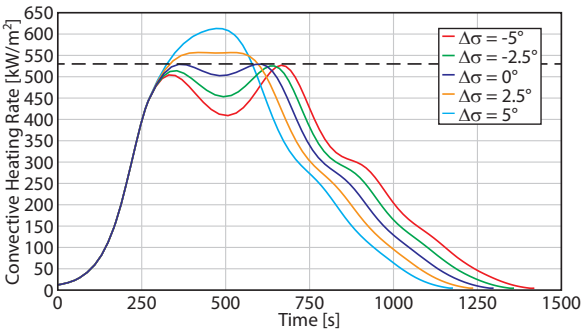
(b) Altitude versus velocity



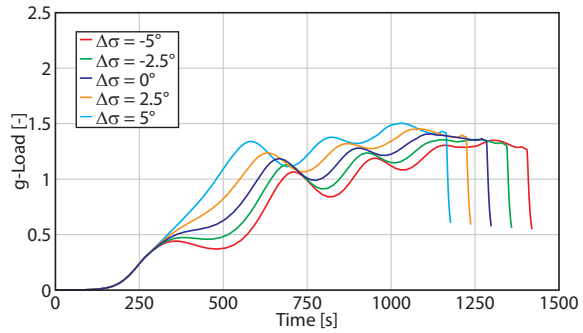
(c) Angle of attack versus time



(d) Bank angle versus time

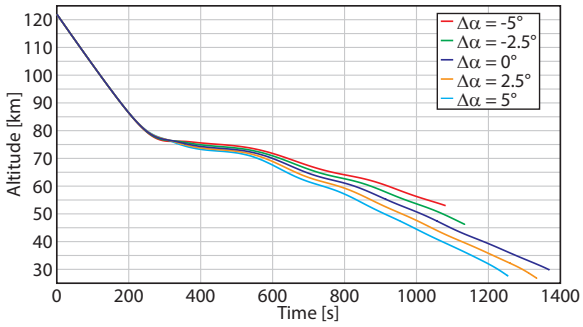


(e) Convective nose heating rate versus time

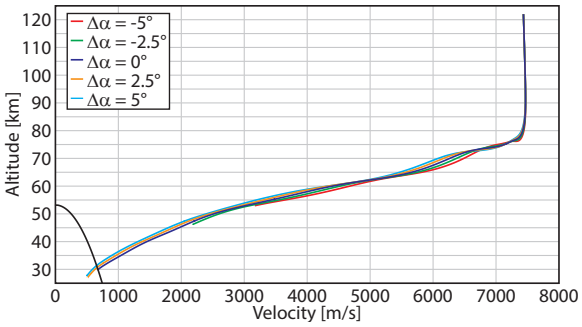


(f) Load factor versus time

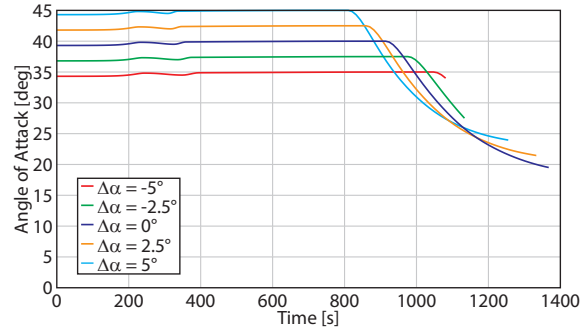
Figure E.4: Sensitivity analysis of the minimum heat load trajectory of the second optimization to perturbations in σ_G.



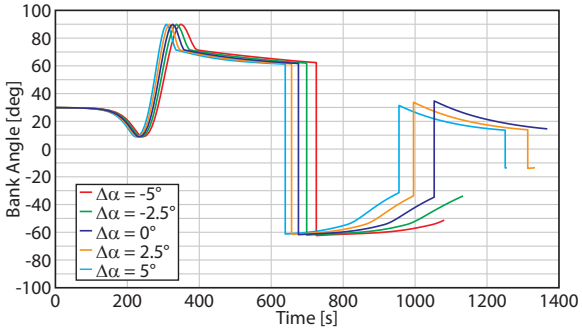
(a) Altitude versus time



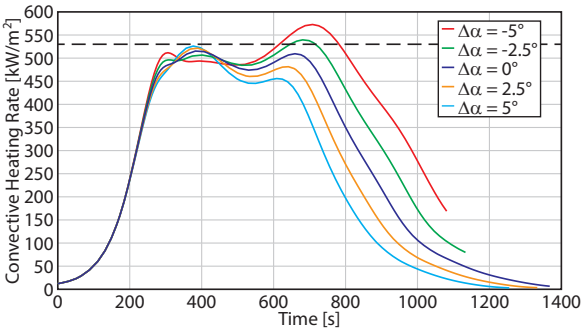
(b) Altitude versus velocity



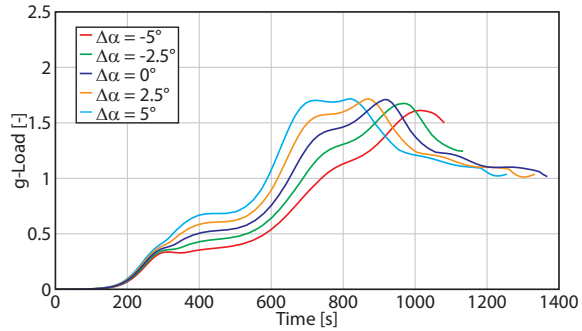
(c) Angle of attack versus time



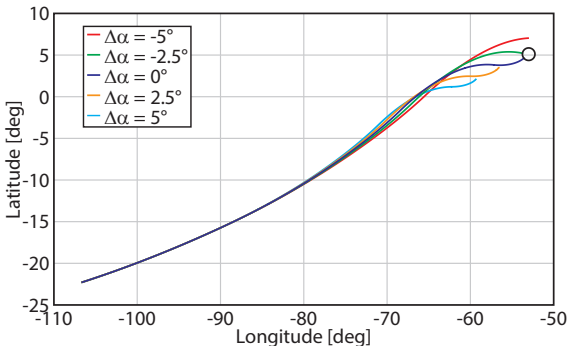
(d) Bank angle versus time



(e) Convective nose heating rate versus time

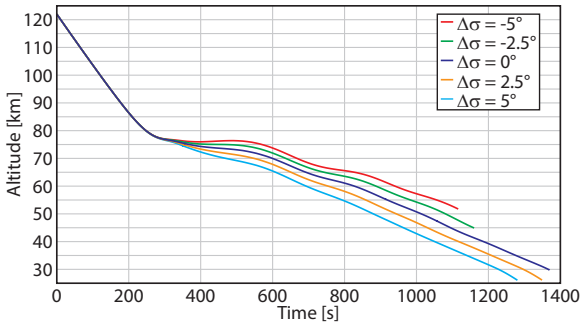


(f) Load factor versus time

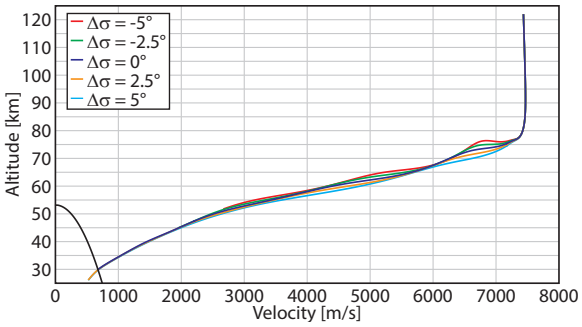


(g) Longitude versus latitude

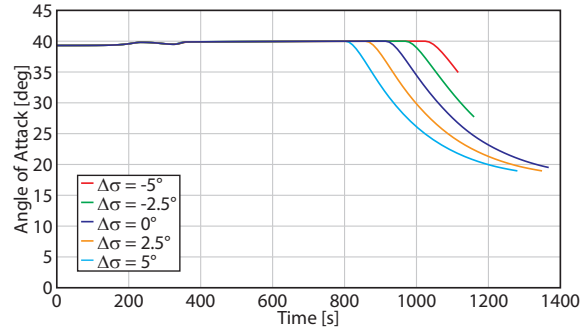
Figure E.5: Sensitivity analysis of the lowest heat load trajectory of the first optimization to perturbations in α_G.



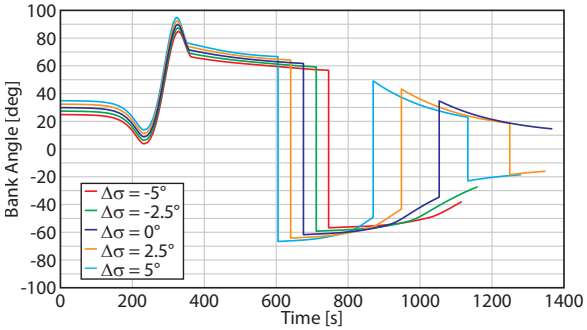
(a) Altitude versus time



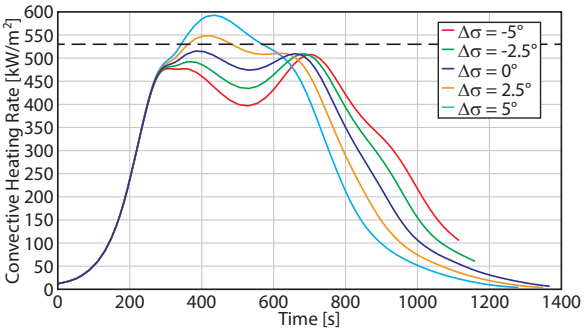
(b) Altitude versus velocity



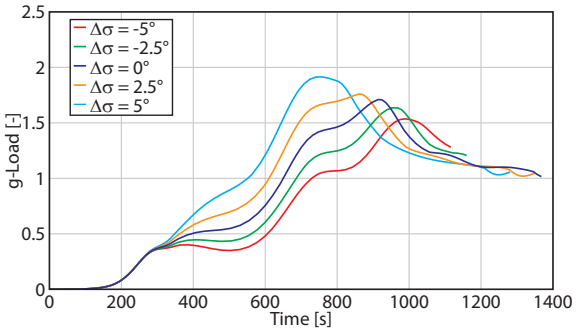
(c) Angle of attack versus time



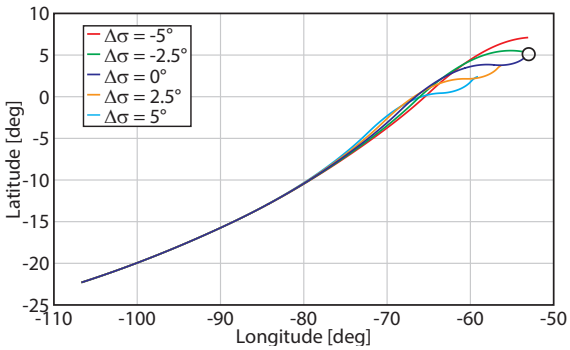
(d) Bank angle versus time



(e) Convective nose heating rate versus time



(f) Load factor versus time



(g) Longitude versus latitude

Figure E.6: Sensitivity analysis of the lowest heat load trajectory of the first optimization to perturbations in σ_G.