Mars Sample Return ascent trajectory optimisation using Taylor Series integration

Master Thesis

S.D. Petrovic

Faculty Aerospace, Section Spaceflight

TU Delft Delft University of Technology

Challenge the future

PREFACE

Back in December 2013 Warren Gebbett gave a presentation on his work at the Jet Propulsion Laboratory (JPL) in Pasadena, California, USA and the opportunity for a new student to go and perform research at JPL. At this point I sent in my application together with eight other students. Then at the end of December I heard that I was invited for an interview in the first week of January 2014. In this interview it was concluded that I met all the requirements and that I was the perfect candidate to follow Warren up as the next student at JPL with financial backing of Dutch Space (now Airbus Defence and Space, the Netherlands). Financial backing was also going to be provided by the Stichting Prof.dr.ir. H.J. van der Maas Fonds (Aerospace Engineering Faculty, TU Delft) and the Stichting Universiteitsfonds Delft (TU Delft). Communication with JPL was thus started and in March 2015 it was clear that I would be working for the Mars Program Formulation Office under the supervision of Roby Wilson (Inner Solar System group, NASA JPL). He told me to focus on subjects that dealt with Mars missions. At that point I was doing my internship at DLR Bremen on Lunar rocket ascent and descent, which lasted till June 2015. When I came back to Delft me and my supervisors Erwin Mooij (rockets, trajectories, entry and descent, TU Delft) and Ron Noomen (mission design and orbit analysis, TU Delft) agreed that it would be best to perform a study on these Mars subjects to prepare for my visit to JPL and to formulate proposal thesis topics. The first week at JPL I presented these initial thesis topics to both people from the Inner Solar System group and the Mars program formulation office. The next few weeks were spent choosing and refining one of these topics. This document is the result of the two-month literature study on that topic to prepare for the thesis project.

S.D. Petrovic
Pasadena, California, February 2016

CONTENTS

ABBREVIATIONS

ACT	Advanced Concepts Team
DE	Differential Evolution
ESA	European Space Agency
GLOM	Gross Lift-Off Mass
GRAM	Global Reference Atmospheric Model
JPL	Jet Propulsion Laboratory
MAV	Mars Ascent Vehicle
MBH	Monotonic Basin Hopping
MOLA	Mars Orbiter Laser Altimeter
MSR	Mars Sample Return
PaGMO	Parallel Global Multi-objective Optimizer
RF	Frame of Reference
RKF45	Runge-Kutta-Fehlberg 4^{th} (5^{th}) order
RKF	Runge-Kutta-Fehlberg
RK4	Runge-Kutta 4^{th} order
s/c	Spacecraft
SNOPT	Sparse Nonlinear Optimizer
SQP	Sequential Quadratic Programming
TSI	Taylor Series integration
Tudat	TU Delft Astrodynamics Toolbox

1

INTRODUCTION

Mars Sample Return (MSR) has been a mission concept that has been proposed many times in the past two decades. Even today, research into this mission is still being done. And although it is not yet an official project proposal, NASAs Jet Propulsion Laboratory (JPL) is currently working on pre-cursor missions to eventually launch an MSR mission. To prepare for this, research is being conducted on different aspects of MSR, such as the Mars Ascent Vehicle (MAV) responsible for transporting the dirt and soil samples into a Martian orbit and the orbiter which will then transport the samples back to Earth. The current orbiter proposed by JPL is a low-thrust orbiter called Mars 2022. Such an MSR mission requires precise and optimum (optimised for lowest Gross Lift-Off Mass (GLOM)) trajectories to be able to bring back as many samples as possible. But how does one determine the optimum MAV trajectory? Especially when it is combined with the optimum trajectory of the low-thrust orbiter.

The proposed research would focus on the combined optimisation of an MAV trajectory and the trajectory of the low-thrust Mars 2022 orbiter. Also, one hypothesis is that great mass saving can be made if the orbiter and MAV would rendezvous within one single orbital revolution after MAV lift-off. Therefore, the question that should be answered is: what is the optimal trajectory solution for the combined trajectory problem of a high-thrust MAV and a low-thrust Mars orbiter performing a single-revolution rendezvous in Mars orbit? More information on the proposed topic is provided in **??**.

A mission such as MSR and the corresponding trajectories can be described in many different reference frames, or Frame of Reference (RF), and the motion of the MAV and the orbiter can be modelled in different ways. Therefore it is important to use the proper equations and environmental models. Also, the trajectory has to be determined or rather a prediction will have to be made. This can be done using integration methods. And finally, the optimum will have to be found using an optimisation method. All these different aspects are addressed in this literature study.

First however, it is important to determine the knowledge that already exists and the research that has already been performed. Therefore, **??** will describe previous sample return missions, low-thrust Spacecraft (s/c) missions, single-revolution rendezvous missions and the research performed in those fields. It will also describe the current MAV designs. Then before mathematically representing the problem it is important to understand in what kind of RF it has to be described. This will be done in **??**, followed by the MAV ascent and low-thrust Mars 2022 orbiter model descriptions in **????** respectively. Here, both chapters explain the assumptions and corresponding equations for each phase. One important aspect of the MAV ascent, which sets it apart from other sample return missions, is that Mars has an atmosphere which cannot be neglected. Accordingly **??** describes the different atmospheric models and the trade-off that was performed to decide which model to use in this thesis problem. Then the integration and optimisation are discussed in **??** and chapter 4 respectively. In the integrators chapter, different integration methods are described and a selection is made of the integration methods that will be used. The same is done for the different optimisers. All of this information will be used to define the final thesis topic, which is presented in **??**. For some of the aspects that will be treated in the final thesis problem, certain software is already available. A summary of this software is provided in **??**. Finally, a proposed schedule is presented in **??**, which shows the work which will have to be performed during the thesis work and the time that will have to be spend on each aspect of it. This literature study will serve as a guideline during the thesis project and provide background information for the final thesis report.

2

PROBLEM BACKGROUND

3

Models

4

OPTIMISATION

5

STANDARD INTEGRATION METHODS

6

TAYLOR SERIES INTEGRATION

6.1. GENERAL THEORY

6.1.1. WORKINGS OF TSI

6.1.2. STEP-SIZE

6.2. ASSOCIATED EQUATIONS

In order for TSI to be implemented, the state derivatives have to be modelled as a set of continuous functions which are a function of the state only. This can be done in different ways. In this case, three cases were tested: two Cartesian cases and one spherical case. For the Cartesian cases, the initial conditions first have to be transformed into Cartesian coordinates. The Cartesian equations themselves require reference frame transformations, which can be written in two different ways. Both cases are described in Section 6.2.1. The reason for testing the spherical case is that the initial conditions are provided in spherical coordinates and the intermediate computations can be easily interpreted and checked for errors. However, the equations are highly sensitive to singularities. The spherical equations are described in Section 6.2.2 and already include reference frame transformations.

6.2.1. CARTESIAN EQUATIONS

The Cartesian state is described in Equation (6.1), where m_{MAV} is the mass of the MAV and the subscript I refers to the inertial frame.

$$\mathbf{r} = \begin{pmatrix} x_I \\ y_I \\ z_I \end{pmatrix} \qquad \mathbf{V} = \begin{pmatrix} V_{x_I} \\ V_{y_I} \\ V_{z_I} \end{pmatrix} \qquad m_{MAV} \tag{6.1}$$

The corresponding state derivatives are then described by Equation (6.2).

$$\begin{aligned} \dot{x}_I &= V_{x_I} \\ \dot{y}_I &= V_{y_I} \\ \dot{z}_I &= V_{z_I} \end{aligned} \qquad \begin{aligned} \ddot{x}_I &= \dot{V}_{x_I} = a_{x_I} \\ \ddot{y}_I &= \dot{V}_{y_I} = a_{y_I} \\ \ddot{z}_I &= \dot{V}_{z_I} = a_{z_I} \end{aligned} \qquad \dot{m}_{MAV} = -\frac{T}{g_0 I_{sp}} \tag{6.2}$$

From this point on, the subscript I is omitted, because the state and the state derivatives are always presented in the inertial frame. If variables have to be presented in any other reference frame the corresponding subscripts will be provided and explained. The accelerations in the x-, y- and z-direction have three contributing components: gravitational acceleration, drag and thrust. The gravitational acceleration can be directly expressed in the inertial frame, however the drag is presented in the body frame and the thrust is expressed in the propulsion frame. Therefore, both the drag and thrust contributions have to be transformed to the inertial frame using transformation matrices. This then results in the expression for the acceleration vector as shown by Equation (6.3). The subscript G shows that the parameter is a function of the ground velocity.

$$\begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} = \begin{pmatrix} -\mu_M \dfrac{x}{r^3} \\ -\mu_M \dfrac{y}{r^3} \\ -\mu_M \dfrac{z}{r^3} \end{pmatrix} + \Big|_{\mathbf{I}} \mathbb{T}_{\mathbf{z}}(-\Omega_M t_O + \omega_P) \Big|_{\mathbf{R}} \mathbb{T}_{\mathbf{z}}(-\tau)\, \mathbb{T}_{\mathbf{y}}\left(\frac{\pi}{2} + \delta\right) \Big|_{\mathbf{V}} \mathbb{T}_{\mathbf{z}}(-\chi_G)\, \mathbb{T}_{\mathbf{y}}(-\gamma_G) \Big|_{\mathbf{B}} \left[\begin{pmatrix} -\dfrac{D}{m_{MAV}} \\ 0 \\ 0 \end{pmatrix} + \ldots \right.$$

$$\left. \ldots \Big|_{\mathbf{B}} \mathbb{T}_{\mathbf{z}}(-\psi_T)\, \mathbb{T}_{\mathbf{y}}(-\epsilon_T) \Big|_{\mathbf{P}} \begin{pmatrix} \dfrac{T}{m_{MAV}} \\ 0 \\ 0 \end{pmatrix} \right] \tag{6.3}$$

It can be seen that this set of equations is a function of the current position and many other parameters. These parameters will all have to be written as a function of the current state. This can be done by writing them into auxiliary equations, forming extra variables that then also require the auxiliary derivatives. This works for certain parameters, such as the gravity, because these equations are already expressed in the inertial frame. However, the transformation angles are defined in different reference frames, which means that finding the proper auxiliary derivatives can be tricky sometimes. Therefore it was decided to directly write these parameters as auxiliary functions. Each of the auxiliary functions performs one simple algebraic operation and the collection of these auxiliary functions then form the complete set of recurrence relations using the recurrence relations for the simple algebraic operations. This is all described in Section 6.2.3.

For the auxiliary equations, a similar notation will be used as shown by Scott and Martini (2008). Here the equations are denoted by x_{number} and the corresponding derivatives x'_{number}. This notation will also be used for the current state and the corresponding state derivatives. This way, Equation (6.1) can be written as Equation (6.4) and the corresponding derivatives can be written as presented by Equation (6.5).

$$\begin{aligned} x_1 &= x & x_4 &= \dot{x} = V_x \\ x_2 &= y & x_5 &= \dot{y} = V_y \qquad\qquad x_7 = m_{MAV} \\ x_3 &= z & x_6 &= \dot{z} = V_z \end{aligned} \tag{6.4}$$

$$\begin{aligned} x'_1 &= x_4 & x'_4 &= \dot{V}_x = a_x = a_{g,x} + a_{D,x} + a_{T,x} \\ x'_2 &= x_5 & x'_5 &= \dot{V}_y = a_y = a_{g,y} + a_{D,y} + a_{T,y} \qquad\qquad x'_7 = \dot{m}_{MAV} = -\frac{T}{g_0 I_{sp}} \\ x'_3 &= x_6 & x'_6 &= \dot{V}_z = a_z = a_{g,z} + a_{D,z} + a_{T,z} \end{aligned} \tag{6.5}$$

In this case the thrust and specific impulse are constant, which means that x'_7 is constant and any additional derivative will be zero. Also, neither one of the thrust angles is a function of the state, which means that a_T, in the body frame, is only a function of x_7 (also see Equation (6.3)). However, both a_g and a_D are a function of the position and velocity, where a_D is also a function of the MAV mass. Only a_g is rewritten using auxiliary equations as mentioned before. This is shown in Section 6.2.1.

GRAVITATIONAL ACCELERATION

For the gravitational acceleration two auxiliary equations were required since $r = \sqrt{x^2 + y^2 + z^2}$. The resulting expressions for the gravitational acceleration are shown in Equation (6.6) with the corresponding auxiliary equations and the derivatives defined in Equation (6.7).

$$\begin{aligned} a_{g,x} &= -\mu_M \frac{x_1}{r^3} = \frac{x_1}{\left(x_1^2 + x_2^2 + x_3^2\right)^{3/2}} = -\mu_M \frac{x_1}{x_9} \\ a_{g,y} &= -\mu_M \frac{x_2}{r^3} = \frac{x_2}{\left(x_1^2 + x_2^2 + x_3^2\right)^{3/2}} = -\mu_M \frac{x_2}{x_9} \\ a_{g,z} &= -\mu_M \frac{x_3}{r^3} = \frac{x_3}{\left(x_1^2 + x_2^2 + x_3^2\right)^{3/2}} = -\mu_M \frac{x_3}{x_9} \end{aligned} \tag{6.6}$$

$$\begin{aligned} x_8 &= x_1^2 + x_2^2 + x_3^2 & x'_8 &= 2x_1 x_4 + 2x_2 x_5 + 2x_3 x_6 \\ x_9 &= x_8^{3/2} & x'_9 &= \frac{3}{2} \frac{x_9 x'_8}{x_8} \end{aligned} \tag{6.7}$$

TRANSFORMATION ANGLES

The angles required for the transformation to go from the body frame to the reference frame all have to be written as a function of the state variables. The required angles are λ, δ, χ_G and γ_G. Here $\lambda = \tau + \Omega_M t_O - \omega_P$. This means that instead of first transforming to the rotating frame completely and then transforming to the inertial frame, an inertial longitude angle λ can be defined to directly transform to the inertial frame. The first two angles are the longitude and latitude and are spherical coordinates. Thus the relations for these angles can be found using the transformation from the Cartesian to the spherical system. However, the angles themselves are not required directly, because they are only used in transformation matrices. These matrices are comprised of the sines and cosines of these angles, which means that a direct relation between the state variables and the sines and cosines of the position angles can be used. These relations can be derived from Figure 6.1 and are described in Equation (6.8)

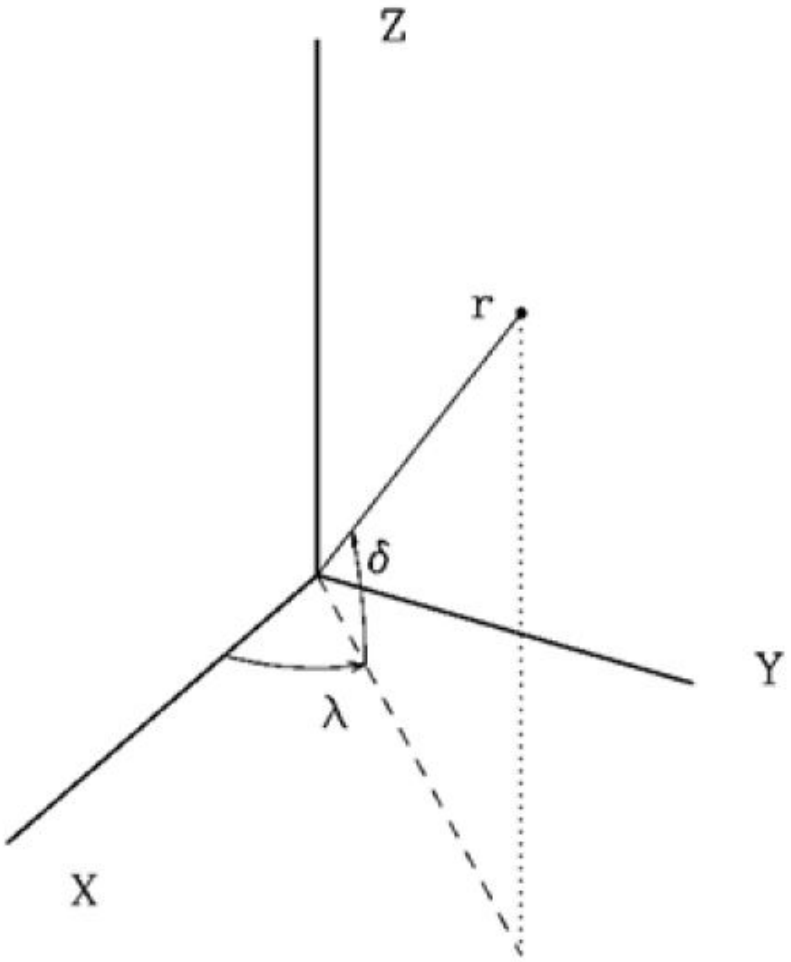

Figure 6.1: Spherical position variables in an inertial Cartesian frame (Noomen, 2013).

$$r = \sqrt{x^2 + y^2 + z^2} \qquad \begin{aligned} \sin\lambda &= \frac{y}{\sqrt{x^2+y^2}} \\ \cos\lambda &= \frac{x}{\sqrt{x^2+y^2}} \end{aligned} \qquad \begin{aligned} \sin\delta &= \frac{z}{r} \\ \cos\delta &= \frac{\sqrt{x^2+y^2}}{r} \end{aligned} \tag{6.8}$$

The corresponding auxiliary functions can then be described by **??** using the definitions provided in Equations (6.4) and (6.5).

$$
\begin{aligned}
w_{4,1} &= x_1^2 + x_2^2 \\
w_{4,2} &= w_{4,1} + x_3^2 \\
w_{4,3} &= r = \sqrt{w_{4,2}} \\
w_{4,4} &= \sqrt{w_{4,1}}
\end{aligned}
\qquad
\begin{aligned}
w_{4,5} &= s\lambda = \frac{x_2}{w_{4,4}} \\
w_{4,6} &= c\lambda = \frac{x_1}{w_{4,4}}
\end{aligned}
\qquad
\begin{aligned}
w_{4,7} &= s\delta = \frac{x_3}{w_{4,3}} \\
w_{4,8} &= c\delta = \frac{w_{4,4}}{w_{4,3}}
\end{aligned}
\tag{6.9}
$$

The latitude and longitude could be described using the position vector in the inertial frame, however the transformation from the body frame to the vertical frame is a function of the ground (underscore 'G') velocity in the rotational frame. Since the position and velocity in the inertial frame are known (current state), the ground velocity (V_G) can be written as a function of the inertial velocity (V_I). For this, the velocity components have to be transformed from the inertial frame to the vertical frame (which is the inverse of the first three transformations described in Equation (6.3)). This transformation is shown in Equation (6.10) and was described in Mooij (1994). Here, c stands for cosine and s stands for sine. This transformation also includes the rotational effect on the velocity due to the rotation of Mars.

$$
\mathbf{V_V} = \begin{pmatrix} V_{x_V} \\ V_{y_V} \\ V_{z_V} \end{pmatrix} = \begin{bmatrix} -c\lambda s\delta & -s\lambda s\delta & c\delta \\ -s\lambda & c\lambda & 0 \\ -c\lambda c\delta & -s\lambda c\delta & -s\delta \end{bmatrix} \left\{ \begin{pmatrix} V_x \\ V_y \\ V_z \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \\ \Omega_M \end{pmatrix} \times \begin{pmatrix} x \\ y \\ z \end{pmatrix} \right\} \tag{6.10}
$$

The ground velocity can now be computed as the norm of the vertical velocity vector as shown by Equation (6.11). The transformation matrices disappear because the norm of a transformation matrix is simply 1, which means that $V_G = V_V = V_R$.

$$
V_G = \|\mathbf{V_V}\| = \left\| \begin{pmatrix} V_x \\ V_y \\ V_z \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \\ \Omega_M \end{pmatrix} \times \begin{pmatrix} x \\ y \\ z \end{pmatrix} \right\| \tag{6.11}
$$

Equation (6.11) can be rewritten as Equation (6.12).

$$
V_G = \sqrt{\left(V_x + \Omega_M y\right)^2 + \left(V_y - \Omega_M x\right)^2 + V_z^2} \tag{6.12}
$$

The corresponding auxiliary functions are provided in Equation (6.13).

$$
\begin{aligned}
w_{4,9} &= V_x + \Omega_M y \\
w_{4,10} &= V_y - \Omega_M x
\end{aligned}
\qquad
\begin{aligned}
w_{4,11} &= w_{4,9}^2 + w_{4,10}^2 + x_6^2 \\
w_{4,12} &= V_G = \sqrt{w_{4,11}}
\end{aligned}
\tag{6.13}
$$

The spherical velocity angles can now be derifed from Figure 6.2 as described by Equation (6.14). These were also provided by Mooij (1994).

$$
\begin{aligned}
\sin\chi_G &= \frac{V_{y_V}}{\sqrt{V_{x_V}^2 + V_{y_V}^2}} \\
\cos\chi_G &= \frac{V_{x_V}}{\sqrt{V_{x_V}^2 + V_{y_V}^2}}
\end{aligned}
\qquad
\begin{aligned}
\sin\gamma_G &= \frac{-V_{z_V}}{V_G} \\
\cos\gamma_G &= \frac{\sqrt{V_{x_V}^2 + V_{y_V}^2}}{V_G}
\end{aligned}
\tag{6.14}
$$

Here, V_{x_V}, V_{y_V} and V_{z_V} are the velocities in the vertical frame. Expressions for these variables can be obtained by rewriting Equation (6.10). This then results in Equation (6.16).

$$
\begin{aligned}
V_{x_V} &= -\left(V_x + \Omega_M y\right) s\delta c\lambda - \left(V_y - \Omega_M x\right) s\delta s\lambda + V_z c\delta \\
V_{y_V} &= \left(V_y - \Omega_M x\right) c\lambda - \left(V_x + \Omega_M y\right) s\lambda \\
V_{z_V} &= -\left(V_x + \Omega_M y\right) - \left(V_y - \Omega_M x\right) c\delta s\lambda - V_z s\delta
\end{aligned}
\tag{6.15}
$$

The combined auxiliary functions for Equations (6.14) and (6.16) are described in Equation (6.13).

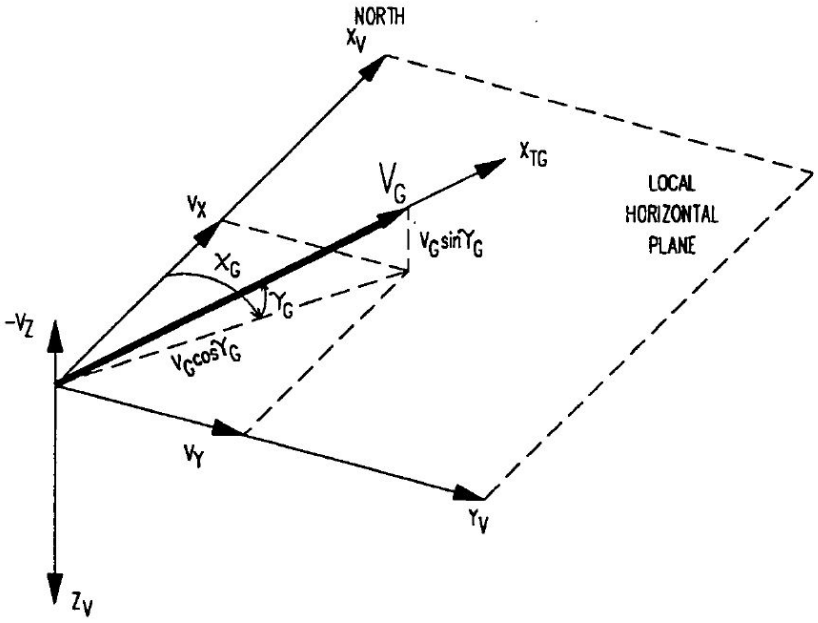

Figure 6.2: Spherical velocity variables in a vertical Cartesian frame (Mooij, 1994).

$$
\begin{aligned}
w_{4,13} &= -c\lambda s\delta = -w_{4,6}w_{4,7} \\
w_{4,14} &= -s\delta s\lambda = -w_{4,7}w_{4,5} \\
w_{4,15} &= -c\delta c\lambda = -w_{4,8}w_{4,6} \\
w_{4,16} &= -c\delta s\lambda = w_{4,8}w_{4,5}
\end{aligned}
\qquad
\begin{aligned}
w_{4,17} &= V_{x_V} = x_6 w_{4,8} + w_{4,9}w_{4,13} + w_{4,10}w_{4,14} \\
w_{4,18} &= V_{y_V} = w_{4,10}w_{4,6} - w_{4,9}w_{4,5} \\
w_{4,19} &= V_{z_V} = w_{4,9}w_{4,15} - x_6 w_{4,7} + w_{4,10}w_{4,16} \\
w_{4,20} &= V_{x_V}^2 + V_{y_V}^2 = w_{4,17}^2 + w_{4,18}^2 \\
w_{4,21} &= \sqrt{w_{4,20}}
\end{aligned}
\qquad
\begin{aligned}
w_{4,22} &= s\chi_G = \frac{w_{4,18}}{w_{4,21}} \\
w_{4,23} &= c\chi_G = \frac{w_{4,17}}{w_{4,21}} \\
w_{4,24} &= s\gamma_G = -\frac{w_{4,19}}{w_{4,12}} \\
w_{4,25} &= c\gamma_G = \frac{w_{4,21}}{w_{4,12}}
\end{aligned}
\tag{6.16}
$$

DRAG ACCELERATION

The drag acceleration is determined in the body frame by dividing the drag force (D) by the mass of the MAV (x_7). The drag force itself is a function of the position and velocity. The equations associated with the drag function are described in Equation (6.33) except for the C_D equations. The polynomial coefficients for the density equation are provided in Table 7.4 and are represented in Equation (6.33) by P_ρ.

$$
\begin{aligned}
h &= r - R_{MOLA} \\
D &= \frac{1}{2}\rho V_G^2 S C_D \\
\rho &= e^{P_{\rho 10}h^{10} + P_{\rho 9}h^9 + P_{\rho 8}h^8 + P_{\rho 7}h^7 + P_{\rho 6}h^6 + P_{\rho 5}h^5 + P_{\rho 4}h^4 + P_{\rho 3}h^3 + P_{\rho 2}h^2 + P_{\rho 1}h + P_{\rho 0}}
\end{aligned}
\tag{6.17}
$$

The numbering for the drag auxiliary functions start with 27 because it was added later on and because it used to be an auxiliary equation. The auxiliary functions for the density can then be described by Equation (6.23).

$$
\begin{aligned}
w_{27,1} &= h = w_{4,3} - R_{MOLA} \\
w_{27,2} &= w_{27,1}^2 \\
w_{27,3} &= w_{27,1}^3 \\
w_{27,4} &= w_{27,1}^4 \\
w_{27,5} &= w_{27,1}^5 \\
w_{27,6} &= w_{27,1}^6
\end{aligned}
\qquad
\begin{aligned}
w_{27,7} &= w_{27,1}^7 \\
w_{27,8} &= w_{27,1}^8 \\
w_{27,9} &= w_{27,1}^9 \\
w_{27,10} &= w_{27,1}^{10} \\
w_{27,11} &= P_{\rho 10}w_{27,10} + P_{\rho 9}w_{27,9} + \ldots + P_{\rho 1}w_{27,1} + P_{\rho 0} \\
w_{27,12} &= \rho = e^{w_{27,11}}
\end{aligned}
\tag{6.18}
$$

Since the drag coefficient is a function of Mach number as by Figure 7.10 and is not a continuous function, it has to be split into 6 different sections. Each section has a separate $C_D - M$ relation. Before these relations can be described, three additional expressions are required which are described in Equation (6.35).

$$\begin{aligned} M &= \frac{V_G}{a} \\ a &= \sqrt{\gamma_a R_a^* T_a} \quad \text{where} \quad R_a^* = \frac{R_a}{M_a} \end{aligned} \tag{6.19}$$

Where the corresponding auxiliary functions can be described by Equation (6.20).

$$\begin{aligned} w_{27,14} &= a = \sqrt{\gamma_a R_a^* w_{27,13}} \\ w_{27,15} &= M = \frac{w_{4,12}}{27,14} \end{aligned} \tag{6.20}$$

The conditional relations shown in Equation (6.36) describe the different equations that have to be used associated with the different sections of the drag coefficient plot. Here $P_{C_D number,section}$ are the polynomial fit coefficients as provided in Table 7.5.

$$C_D = \begin{cases} 0.2, & \text{for } 0 \leq M < 0.5 \\ P_{C_D 1,2} M + P_{C_D 0,2}, & \text{for } 0.5 \leq M < 1 \\ P_{C_D 1,3} M + P_{C_D 0,3}, & \text{for } 1 \leq M < 1.3 \\ P_{C_D 1,4} M + P_{C_D 0,4}, & \text{for } 1.3 \leq M < 2.5 \\ P_{C_D 1,5} M + P_{C_D 0,5}, & \text{for } 2.5 \leq M < 4 \\ 0.3, & \text{for } M \geq 4 \end{cases} \tag{6.21}$$

In this case, the auxiliary functions for $C_D \left(= w_{27,16}\right)$ is any of the conditional relations depending on $M \left(= w_{27,15}\right)$.

This only leaves the temperature $T_a \left(= w_{27,13}\right)$, which is a function of the altitude $h \left(= w_{27,1}\right)$ in km,Mars Orbiter Laser Altimeter (MOLA). But as described in Section 7.2.1, this parameter is split into different sections as well. The equations per section for the temperature is provided in Equation (6.39). Here $P_{T number,section}$ are the polynomial fit coefficients as provided in Table 7.2.

$$T_a = \begin{cases} P_{T1,1} h + P_{T0,1}, & \text{for } -0.6 \leq h < 5.04 \\ P_{T3,2} h^3 + P_{T2,2} h^2 + P_{T1,2} h + P_{T0,2}, & \text{for } 5.04 \leq h < 35.53 \\ P_{T6,3} h^6 + P_{T5,3} h^5 + P_{T4,3} h^4 + P_{T3,3} h^3 + \ldots & \\ \quad \ldots + P_{T2,3} h^2 + P_{T1,3} h + P_{T0,3}, & \text{for } 35.53 \leq h < 75.07 \\ P_{T8,4} h^8 + P_{T7,4} h^7 + P_{T6,4} h^6 + P_{T5,4} h^5 + \ldots & \\ \quad \ldots + P_{T4,4} h^4 + P_{T3,4} h^3 + P_{T2,4} h^2 + P_{T1,4} h + P_{T0,4}, & \text{for } 75.07 \leq h < 170.05 \\ 136.5, & \text{for } h \geq 170.05 \end{cases} \tag{6.22}$$

For both the conditional parameters C_D and T_a the required section has to be determined before the evaluation of the equations.

The drag can now also be described as an auxiliary function as shown by Equation (6.23).

$$\begin{aligned} w_{27,17} &= V_G^2 = w_{4,12}^2 \\ w_{27,18} &= V_G^2 C_D = w_{27,17} w_{27,16} \\ w_{27,19} &= D = \frac{1}{2} S w_{27,18} w_{27,12} \end{aligned} \tag{6.23}$$

THRUST ACCELERATION

The only acceleration component still missing is the thrust acceleration. To be able to write the auxiliary functions for the thrust, Equation (6.3) first has to be rewritten such that all the transformations are gathered into two transformation matrices (see Equation (6.24)). The transformation matrices are described by Equations (6.25) and (6.26) for $\mathbb{T}_{\mathbf{BP}}$ and $\mathbb{T}_{\mathbf{IB}}$ respectively.

$$\begin{pmatrix} x_4' \\ x_5' \\ x_6' \end{pmatrix} = \begin{pmatrix} -\mu_M \dfrac{x_1}{x_9} \\ -\mu_M \dfrac{x_2}{x_9} \\ -\mu_M \dfrac{x_3}{x_9} \end{pmatrix} + \mathbb{T}_{\mathbf{IB}} \left[\begin{pmatrix} -\dfrac{w_{27,19}}{x_7} \\ 0 \\ 0 \end{pmatrix} + \mathbb{T}_{\mathbf{BP}} \begin{pmatrix} \dfrac{T}{x_7} \\ 0 \\ 0 \end{pmatrix} \right] \tag{6.24}$$

$$\mathbb{T}_{\mathbf{BP}} = \begin{bmatrix} c\psi_T c\epsilon_T & -s\psi_T & c\psi_T s\epsilon_T \\ s\psi_T c\epsilon_T & c\psi_T & s\psi_T s\epsilon_T \\ -s\epsilon_T & 0 & c\epsilon_T \end{bmatrix} \tag{6.25}$$

$$\mathbb{T}_{\mathbf{IB}} = \begin{bmatrix} c\lambda\left(-s\delta c\chi c\gamma + c\delta s\gamma\right) - s\lambda s\chi c\gamma & c\lambda s\delta s\chi - s\lambda c\chi & c\lambda\left(-s\delta c\chi s\gamma - c\delta c\gamma\right) - s\lambda s\chi s\gamma \\ s\lambda\left(-s\delta c\chi c\gamma + c\delta s\gamma\right) + c\lambda s\chi c\gamma & s\lambda s\delta s\chi + c\lambda c\chi & s\lambda\left(-s\delta c\chi s\gamma - c\delta c\gamma\right) + c\lambda s\chi s\gamma \\ c\delta c\chi c\gamma + s\delta s\gamma & -c\delta s\chi & c\delta c\chi s\gamma - s\delta c\gamma \end{bmatrix} \tag{6.26}$$

Rest still has to be rewritten!!

Because the mass flow rate is constant, and not a function of the state, the first derivative of u_7 will be zero such that $u_7' = 0$. This means that only the first derivative of the mass flow rate has to be used and thus u_7 does not require any recurrence relations. This is not the case for the thrust accelerations because they are a function of the state (in this case x_7) and will require recurrence relations. Using an intermediate notation for the thrust accelerations in the x-, y- and z-directions, the last part of Equation (6.24) can be rewritten to Equation (6.27) using Equation (6.25).

$$\mathbb{T}_{\mathbf{BP}} \begin{pmatrix} \dfrac{T}{x_7} \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \dfrac{T}{x_7} c\psi_T c\epsilon_T \\ \dfrac{T}{x_7} s\psi_T c\epsilon_T \\ -\dfrac{T}{x_7} s\epsilon_T \end{pmatrix} = \begin{pmatrix} a_{T_B,x} \\ a_{T_B,y} \\ a_{T_B,z} \end{pmatrix} \tag{6.27}$$

The thrust accelerations are now written in the body frame, which means that they can be combined with the drag acceleration in the body frame. This is done in Equation (6.28). However, since both the thrust and drag accelerations include auxiliary equations (x_7 and x_{27}) the total acceleration in the body frame in the different directions have to be expressed as auxiliary functions. Each auxiliary function replaces one algebraic operation such as multiplication, division, power, exponential and trigonometric functions. Auxiliary functions will be represented by $w_{n,m}$ as per convention introduced by Scott and Martini (2008). Here n stands for the associating derivative and m represents the number of the auxiliary function for that particular derivative. These auxiliary functions will be used to write the different parameters into recurrence relations.

$$\begin{pmatrix} -\dfrac{x_{27}}{x_7} \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} a_{T_B,x} \\ a_{T_B,y} \\ a_{T_B,z} \end{pmatrix} = \begin{pmatrix} a_{T_B,x} - \dfrac{x_{27}}{x_7} \\ a_{T_B,y} \\ a_{T_B,z} \end{pmatrix} = \begin{pmatrix} a_{T,D_B,x} \\ a_{T_B,y} \\ a_{T_B,z} \end{pmatrix} = \begin{pmatrix} w_{4,2} \\ w_{4,33} \\ -w_{4,34} \end{pmatrix} \tag{6.28}$$

Now transforming the drag and thrust accelerations from the body frame to the inertial frame using $\mathbb{T}_{\mathbf{IB}}$ results in Equation (6.29).

$$\begin{pmatrix} a_{T,D_I,x} \\ a_{T_I,y} \\ a_{T_I,z} \end{pmatrix} = \mathbb{T}_{\mathbf{IB}} \begin{pmatrix} a_{T,D_B,x} \\ a_{T_B,y} \\ a_{T_B,z} \end{pmatrix} = \mathbb{T}_{\mathbf{IB}} \begin{pmatrix} w_{4,2} \\ w_{4,33} \\ -w_{4,34} \end{pmatrix}$$

$$= \begin{pmatrix} w_{4,2}\left(c\left(x_{10}+x_{11}\right)\left(-sx_{12}cx_{13}cx_{14} + cx_{12}sx_{14}\right) - s\left(x_{10}+x_{11}\right)sx_{13}cx_{14}\right) + \dots \\ w_{4,2}\left(s\left(x_{10}+x_{11}\right)\left(-sx_{12}cx_{13}cx_{14} + cx_{12}sx_{14}\right) + c\left(x_{10}+x_{11}\right)sx_{13}cx_{14}\right) + \dots \\ w_{4,2}\left(cx_{12}cx_{13}cx_{14} + sx_{12}sx_{14}\right) - \dots \end{pmatrix}$$

$$\begin{pmatrix} \dots w_{4,33}\left(c\left(x_{10}+x_{11}\right)sx_{12}sx_{13} - s\left(x_{10}+x_{11}\right)cx_{13}\right) - \dots \\ \dots w_{4,33}\left(s\left(x_{10}+x_{11}\right)sx_{12}sx_{13} + c\left(x_{10}+x_{11}\right)cx_{13}\right) - \dots \\ \dots w_{4,33}cx_{12}sx_{13} - \dots \end{pmatrix}$$

$$\left.\begin{array}{c} \ldots w_{4,34}\left(c\left(x_{10}+x_{11}\right)\left(-sx_{12}cx_{13}sx_{14}-cx_{12}cx_{14}\right)-s\left(x_{10}+x_{11}\right)sx_{13}sx_{14}\right) \\ \ldots w_{4,34}\left(s\left(x_{10}+x_{11}\right)\left(-sx_{12}cx_{13}sx_{14}-cx_{12}cx_{14}\right)+c\left(x_{10}+x_{11}\right)sx_{13}sx_{14}\right) \\ \ldots w_{4,34}\left(cx_{12}cx_{13}sx_{14}-sx_{12}cx_{14}\right) \end{array}\right) \tag{6.29}$$

Now including the gravity components as well, the (lengthy) expressions for u_4, u_5 and u_6 become Equations (6.30) to (6.32) respectively.

$$\begin{aligned} u_4 = -\mu_M \frac{x_1}{x_9} &+ w_{4,2}\left(c\left(x_{10}+x_{11}\right)\left(-sx_{12}cx_{13}cx_{14}+cx_{12}sx_{14}\right)-s\left(x_{10}+x_{11}\right)sx_{13}cx_{14}\right)+\ldots \\ &\ldots w_{4,33}\left(c\left(x_{10}+x_{11}\right)sx_{12}sx_{13}-s\left(x_{10}+x_{11}\right)cx_{13}\right)-\ldots \\ &\ldots w_{4,34}\left(c\left(x_{10}+x_{11}\right)\left(-sx_{12}cx_{13}sx_{14}-cx_{12}cx_{14}\right)-s\left(x_{10}+x_{11}\right)sx_{13}sx_{14}\right) \end{aligned} \tag{6.30}$$

$$\begin{aligned} u_5 = -\mu_M \frac{x_2}{x_9} &+ w_{4,2}\left(s\left(x_{10}+x_{11}\right)\left(-sx_{12}cx_{13}cx_{14}+cx_{12}sx_{14}\right)+c\left(x_{10}+x_{11}\right)sx_{13}cx_{14}\right)+\ldots \\ &\ldots w_{4,33}\left(s\left(x_{10}+x_{11}\right)sx_{12}sx_{13}+c\left(x_{10}+x_{11}\right)cx_{13}\right)-\ldots \\ &\ldots w_{4,34}\left(s\left(x_{10}+x_{11}\right)\left(-sx_{12}cx_{13}sx_{14}-cx_{12}cx_{14}\right)+c\left(x_{10}+x_{11}\right)sx_{13}sx_{14}\right) \end{aligned} \tag{6.31}$$

$$u_6 = -\mu_M \frac{x_3}{x_9} + w_{4,2}\left(cx_{12}cx_{13}cx_{14}+sx_{12}sx_{14}\right)-w_{4,33}cx_{12}sx_{13}-w_{4,34}\left(cx_{12}cx_{13}sx_{14}-sx_{12}cx_{14}\right) \tag{6.32}$$

6.2.2. SPHERICAL EQUATIONS

DRAG ACCELERATION

The drag acceleration is determined in the body frame by dividing the drag force (D) by the mass of the MAV (x_7). The drag force itself is a function of the position and velocity. The equations associated with the drag function are described in Equation (6.33) and the corresponding derivatives are presented by Equation (6.34) except for the C_D equations. The polynomial coefficients for the density equation are provided in Table 7.4 and are represented in Equation (6.33) by P_ρ.

$$\begin{aligned} h &= r - R_{MOLA} \\ D &= \frac{1}{2}\rho V_G^2 S C_D \\ \rho &= e^{P_{\rho 10}h^{10}+P_{\rho 9}h^9+P_{\rho 8}h^8+P_{\rho 7}h^7+P_{\rho 6}h^6+P_{\rho 5}h^5+P_{\rho 4}h^4+P_{\rho 3}h^3+P_{\rho 2}h^2+P_{\rho 1}h+P_{\rho 0}} \end{aligned} \tag{6.33}$$

$$\begin{aligned} x'_{27} &= \frac{1}{2}Sx_{15}\left(x_{15}\left(x_{29}x'_{28}+x_{28}x'_{29}\right)+2x_{28}x_{29}x'_{15}\right) \\ x'_{28} &= x'_{30}x_{28} \\ x'_{30} &= x'_{31}\left(10P_{\rho 10}x_{31}^9+9P_{\rho 9}x_{31}^8+8P_{\rho 8}x_{31}^7+7P_{\rho 7}x_{31}^6+6P_{\rho 6}x_{31}^5+\ldots\right. \\ &\quad \left.\ldots+5P_{\rho 5}x_{31}^4+4P_{\rho 4}x_{31}^3+3P_{\rho 3}x_{31}^2+2P_{\rho 2}x_{31}+P_{\rho 1}\right) \\ x'_{31} &= x'_{16} \end{aligned} \tag{6.34}$$

Since the drag coefficient is a function of Mach number as by Figure 7.10 and is not a continuous function, it had to be split into 6 different sections. Each section has a separate $C_D - M$ relation. Before these relations can be described, three additional auxiliary equations/expressions are required which are described in Equation (6.35).

$$\begin{aligned} x_{32} &= M = \frac{V}{a} = \frac{x_{15}}{x_{33}} \\ x_{33} &= a = \sqrt{\gamma_a R_a^* T_a} = \sqrt{\gamma_a R_a^* x_{34}} \quad \text{where} \quad R_a^* = \frac{R_a}{M_a} \\ x_{34} &= T_a \end{aligned} \tag{6.35}$$

The conditional relations shown in Equation (6.36) describe the different auxiliary equations that have to be used associated with the different sections. Here $P_{C_D number,section}$ are the polynomial fit coefficients as provided in Table 7.5.

$$x_{29} = C_D = \begin{cases} x_{29,1} = 0.2, & \text{for } 0 \leq x_{32} < 0.5 \\ x_{29,2} = P_{C_D 1,2} x_{32} + P_{C_D 0,2}, & \text{for } 0.5 \leq x_{32} < 1 \\ x_{29,3} = P_{C_D 1,3} x_{32} + P_{C_D 0,3}, & \text{for } 1 \leq x_{32} < 1.3 \\ x_{29,4} = P_{C_D 1,4} x_{32} + P_{C_D 0,4}, & \text{for } 1.3 \leq x_{32} < 2.5 \\ x_{29,5} = P_{C_D 1,5} x_{32} + P_{C_D 0,5}, & \text{for } 2.5 \leq x_{32} < 4 \\ x_{29,6} = 0.3, & \text{for } x_{32} \geq 4 \end{cases} \tag{6.36}$$

The corresponding derivatives are presented Equation (6.37).

$$x'_{29} = \begin{cases} x'_{29,1} = 0, & \text{for } 0 \leq x_{32} < 0.5 \\ x'_{29,2} = P_{C_D 1,2} x'_{32}, & \text{for } 0.5 \leq x_{32} < 1 \\ x'_{29,3} = P_{C_D 1,3} x'_{32}, & \text{for } 1 \leq x_{32} < 1.3 \\ x'_{29,4} = P_{C_D 1,4} x'_{32}, & \text{for } 1.3 \leq x_{32} < 2.5 \\ x'_{29,5} = P_{C_D 1,5} x'_{32}, & \text{for } 2.5 \leq x_{32} < 4 \\ x'_{29,6} = 0, & \text{for } x_{32} \geq 4 \end{cases} \tag{6.37}$$

The Mach derivative and corresponding speed of sound derivative are described in Equation (6.38).

$$\begin{aligned} x'_{32} &= \frac{x_{33} x'_{15} - x_{15} x'_{33}}{x_{33}^2} \\ x'_{33} &= \frac{\gamma_a R_a^*}{2 x_{33}} x'_{34} \end{aligned} \tag{6.38}$$

This only leaves the temperature T_a, which is a function of the altitude h (x_{31}) in km,MOLA. But as described in Section 7.2.1, this parameter is split into different sections as well. The auxiliary equations per section for the temperature is provided in Equation (6.39). Here $P_{T number,section}$ are the polynomial fit coefficients as provided in Table 7.2.

$$x_{34} = T_a = \begin{cases} x_{34,1} = P_{T1,1} x_{31} + P_{T0,1}, & \text{for } -0.6 \leq x_{31} < 5.04 \\ x_{34,2} = P_{T3,2} x_{31}^3 + P_{T2,2} x_{31}^2 + P_{T1,2} x_{31} + P_{T0,2}, & \text{for } 5.04 \leq x_{31} < 35.53 \\ x_{34,3} = P_{T6,3} x_{31}^6 + P_{T5,3} x_{31}^5 + P_{T4,3} x_{31}^4 + P_{T3,3} x_{31}^3 + \ldots & \\ \qquad \ldots + P_{T2,3} x_{31}^2 + P_{T1,3} x_{31} + P_{T0,3}, & \text{for } 35.53 \leq x_{31} < 75.07 \\ x_{34,4} = P_{T8,4} x_{31}^8 + P_{T7,4} x_{31}^7 + P_{T6,4} x_{31}^6 + P_{T5,4} x_{31}^5 & \\ \qquad + P_{T4,4} x_{31}^4 + P_{T3,4} x_{31}^3 + P_{T2,4} x_{31}^2 + P_{T1,4} x_{31} + P_{T0,4}, & \text{for } 75.07 \leq x_{31} < 170.05 \\ x_{34,5} = 136.5, & \text{for } x_{31} \geq 170.05 \end{cases} \tag{6.39}$$

The corresponding derivatives are presented Equation (6.40).

$$x'_{34} = \begin{cases} x'_{34,1} = P_{T1,1} x'_{31}, & \text{for } -0.6 \leq x_{31} < 5.04 \\ x'_{34,2} = \left(3 P_{T3,2} x_{31}^2 + 2 P_{T2,2} x_{31} + P_{T1,2}\right) x'_{31}, & \text{for } 5.04 \leq x_{31} < 35.53 \\ x'_{34,3} = \left(6 P_{T6,3} x_{31}^5 + 5 P_{T5,3} x_{31}^4 + 4 P_{T4,3} x_{31}^3 + \ldots \right. & \\ \qquad \left. \ldots + 3 P_{T3,3} x_{31}^2 + 2 P_{T2,3} x_{31} + P_{T1,3}\right) x'_{31}, & \text{for } 35.53 \leq x_{31} < 75.07 \\ x'_{34,4} = \left(8 P_{T8,4} x_{31}^7 + 7 P_{T7,4} x_{31}^6 + 6 P_{T6,4} x_{31}^5 + 5 P_{T5,4} x_{31}^4 + \ldots \right. & \\ \qquad \left. \ldots + 4 P_{T4,4} x_{31}^3 + 3 P_{T3,4} x_{31}^2 + 2 P_{T2,4} x_{31} + P_{T1,4}\right) x'_{31}, & \text{for } 75.07 \leq x_{31} < 170.05 \\ x'_{34,5} = 0, & \text{for } x_{31} \geq 170.05 \end{cases} \tag{6.40}$$

For both the conditional parameters C_D (x_{29}) and T_a (x_{34}) the required section has to be determined before the evaluation of the auxiliary equations.

6.2.3. RECURRENCE RELATIONS AND AUXILIARY FUNCTIONS

To be able to determine the different auxiliary functions and eventually write the recurrence relations each of the derivatives presented in **??** will be rewritten using Equation (6.41) as by Scott and Martini (2008).

$$u_n = x_n', \quad n = 1, \ldots, 49 \tag{6.41}$$

These equations are then written as presented by Equations (6.42) and (6.43).

$$\begin{aligned}
u_1 &= x_4 & u_7 &= \dot{m}_{MAV} = -\frac{T}{g_0 I_{sp}} \\
u_2 &= x_5 & u_8 &= 2x_1x_4 + 2x_2x_5 + 2x_3x_6 \\
u_3 &= x_6 & u_9 &= \frac{3}{2}\frac{x_9 u_8}{x_8} \\
u_4 &= \dot{V}_x = a_x = a_{g,x} + a_{D,x} + a_{T,x} & u_{10} &= \Omega_M \\
u_5 &= \dot{V}_y = a_y = a_{g,y} + a_{D,y} + a_{T,y} & u_{11} &= \dot{\tau} = \frac{x_{15} s x_{13} c x_{14}}{x_{16} c x_{12}} \\
u_6 &= \dot{V}_z = a_z = a_{g,z} + a_{D,z} + a_{T,z}
\end{aligned} \tag{6.42}$$

$$\begin{aligned}
u_{12} &= \dot{\delta} = \frac{x_{15} c x_{13} c x_{14}}{x_{16}} \\
u_{13} &= \dot{\chi}_G = 2\frac{\Omega_M}{c x_{14}}\left(s x_{12} c x_{14} - c x_{12} s x_{14} c x_{13}\right) + \frac{x_{15}}{x_{16}} c x_{14} t x_{12} s x_{13} + \frac{\Omega_M^2}{x_{15} c x_{14}} x_{16} c x_{12} s_{x12} s x_{13} - \frac{T}{x_{15} c x_{14} x_7} s\psi_T c\epsilon_T \\
u_{14} &= \dot{\gamma}_G = 2\Omega_M c x_{12} s x_{13} + \frac{x_{15}}{x_{16}} c x_{14} + \frac{\Omega_M^2}{x_{15}} x_{16} c x_{12}\left(c x_{12} c x_{14} + s x_{14} s x_{12} c x_{13}\right) + \frac{T}{x_{15} x_7} s\epsilon_T - \frac{\mu_M}{x_{15} x_{16}^2} c x_{14} \\
u_{15} &= \dot{V}_G = \Omega_M^2 x_{16} c x_{12}\left(s x_{14} c x_{12} - c x_{14} s x_{12} c x_{13}\right) + \frac{T}{x_7} c\psi_T c\epsilon_T - \frac{x_{27}}{x_7} - \frac{\mu_M}{x_{16}^2} s x_{14} \\
u_{16} &= \dot{r}_G = x_{15} s x_{14}
\end{aligned} \tag{6.43}$$

$$
\begin{aligned}
u_{27} &= \frac{1}{2} S x_{15} \left(x_{15} \left(x_{29} u_{28} + x_{28} u_{29} \right) + 2 x_{28} x_{29} u_{15} \right) \\
u_{28} &= u_{30} x_{28} \\
u_{29} &= \begin{cases} u_{29,1} = 0, & \text{for } 0 \leq x_{32} < 0.5 \\ u_{29,2} = P_{C_D 1,2} u_{32}, & \text{for } 0.5 \leq x_{32} < 1 \\ u_{29,3} = P_{C_D 1,3} u_{32}, & \text{for } 1 \leq x_{32} < 1.3 \\ u_{29,4} = P_{C_D 1,4} u_{32}, & \text{for } 1.3 \leq x_{32} < 2.5 \\ u_{29,5} = P_{C_D 1,5} u_{32}, & \text{for } 2.5 \leq x_{32} < 4 \\ u_{29,6} = 0, & \text{for } x_{32} \geq 4 \end{cases} \\
u_{30} &= u_{31} \left(10 P_{\rho 10} x_{31}^9 + 9 P_{\rho 9} x_{31}^8 + 8 P_{\rho 8} x_{31}^7 + 7 P_{\rho 7} x_{31}^6 + 6 P_{\rho 6} x_{31}^5 + 5 P_{\rho 5} x_{31}^4 \ldots \right. \\
&\quad \left. \ldots + 4 P_{\rho 4} x_{31}^3 + 3 P_{\rho 3} x_{31}^2 + 2 P_{\rho 2} x_{31} + P_{\rho 1} \right) \\
u_{31} &= u_{16} \\
u_{32} &= \frac{x_{33} u_{15} - x_{15} u_{33}}{x_{33}^2} \\
u_{33} &= \frac{\gamma_a R_a^*}{2 x_{33}} u_{34} \\
u_{34} &= \begin{cases} u_{34,1} = P_{T1,1} u_{31}, & \text{for } -0.6 \leq x_{31} < 5.04 \\ u_{34,2} = \left(3 P_{T3,2} x_{31}^2 + 2 P_{T2,2} x_{31} + P_{T1,2} \right) u_{31}, & \text{for } 5.04 \leq x_{31} < 35.53 \\ u_{34,3} = \left(6 P_{T6,3} x_{31}^5 + 5 P_{T5,3} x_{31}^4 + 4 P_{T4,3} x_{31}^3 + \ldots \right. & \\ \qquad \left. \ldots + 3 P_{T3,3} x_{31}^2 + 2 P_{T2,3} x_{31} + P_{T1,3} \right) u_{31}, & \text{for } 35.53 \leq x_{31} < 75.07 \\ u_{34,4} = \left(8 P_{T8,4} x_{31}^7 + 7 P_{T7,4} x_{31}^6 + 6 P_{T6,4} x_{31}^5 + 5 P_{T5,4} x_{31}^4 + \ldots \right. & \\ \qquad \left. \ldots + 4 P_{T4,4} x_{31}^3 + 3 P_{T3,4} x_{31}^2 + 2 P_{T2,4} x_{31} + P_{T1,4} \right) u_{31}, & \text{for } 75.07 \leq x_{31} < 170.05 \\ u_{34,5} = 0, & \text{for } x_{31} \geq 170.05 \end{cases}
\end{aligned}
\tag{6.44}
$$

To be able to compute the initial values of the auxiliary equations (x_n) and their derivatives(u_n) they all have to be computed in a certain order. Unfortunately, because of the manner in which these were defined, the order is not simply the order in which they were numbered. Therefore Table 6.1 shows the required order in which they have to be computed. In order to avoid confusion the auxiliary equations are computed first and then the derivatives.

Table 6.1: Order of auxiliary equations and derivatives computations

Auxiliary equations				**Auxiliary derivatives**			
Order	$\mathbf{x_n}$	**Order**	$\mathbf{x_n}$	**Order**	$\mathbf{u_n}$	**Order**	$\mathbf{u_n}$
0	$x_1, x_2, x_3, x_4, x_5, x_6, x_7$	5	x_{32}	9	$u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8,$ $u_{10}, u_{11}, u_{12}, u_{13}, u_{14}, u_{15}, u_{16}$	12	u_{28}, u_{33}
1	$x_8, x_9, x_{10}, x_{15}, x_{16}$	6	x_{29}			13	u_{32}
2	x_{11}, x_{12}, x_{31}	7	x_{27}	10	u_9, u_{31}	14	u_{29}
3	$x_{13}, x_{14}, x_{30}, x_{34}$	8	$w_{4,2}, w_{4,33},$ $w_{4,34}$	11	u_{30}, u_{34}	15	u_{27}
4	x_{28}, x_{33}			12	u_{28}, u_{33}		

The first auxiliary functions were already introduced in Equation (6.28), however, many more are required to be able to write the recurrence relations. All auxiliary functions are presented in Table 6.2, where the convention was used as introduced earlier. In this case, the entire equation was already written out when the auxiliary function numbers were assigned, which is why these can simply be computed in the order as presented. Except for $w_{4,2}$, because this is the only subtraction and requires $w_{4,32}$, $w_{4,25}$, because $w_{4,37}$ has to be multiplied with the constant thrust, and $w_{4,33}$ and $w_{4,34}$. This is due to the number assignment and the later addition of the thrust auxiliary functions.

Table 6.2: Auxiliary functions as a function of the auxiliary equations and derivatives.

Auxiliary function	**Equation**	**Category**
$w_{4,0} =$	$\frac{x_{27}}{x_7}$	Division
$w_{4,1} =$	$\frac{x_1}{x_9}$	Division
$w_{4,2} =$	$w_{4,32} - w_{4,0}$	Subtraction
$w_{4,3} =$	$c\left(x_{10} + x_{11}\right)$	Cosine
$w_{4,4} =$	sx_{12}	Sine
$w_{4,5} =$	cx_{13}	Cosine
$w_{4,6} =$	cx_{12}	Cosine
$w_{4,7} =$	sx_{14}	Sine
$w_{4,8} =$	$s\left(x_{10} + x_{11}\right)$	Sine
$w_{4,9} =$	sx_{13}	Sine
$w_{4,10} =$	$w_{4,4}\, w_{4,5}$	Multiplication
$w_{4,11} =$	$w_{4,6}\, w_{4,7}$	Multiplication
$w_{4,12} =$	$w_{4,9}\, w_{4,38}$	Multiplication
$w_{4,13} =$	$w_{4,4}\, w_{4,9}$	Multiplication
$w_{4,14} =$	$w_{4,8}\, w_{4,5}$	Multiplication
$w_{4,15} =$	$w_{4,6}\, w_{4,38}$	Multiplication
$w_{4,16} =$	$w_{4,9}\, w_{4,7}$	Multiplication
$w_{4,17} =$	$w_{4,10}\, w_{4,38}$	Multiplication
$w_{4,18} =$	$w_{4,8}\, w_{4,12}$	Multiplication
$w_{4,19} =$	$w_{4,3}\, w_{4,13}$	Multiplication
$w_{4,20} =$	$w_{4,10}\, w_{4,7}$	Multiplication
$w_{4,21} =$	$w_{4,8}\, w_{4,16}$	Multiplication
$w_{4,22} =$	$w_{4,3}\left(w_{4,11} - w_{4,17}\right)$	Multiplication
$w_{4,23} =$	$w_{4,3}\left(-w_{4,20} - w_{4,15}\right)$	Multiplication
$w_{4,24} =$	$w_{4,2}\left(w_{4,22} - w_{4,18}\right)$	Multiplication
$w_{4,25} =$	$Tw_{4,37}$	Constant Multiplication
$w_{4,26} =$	$c\psi_T$	Cosine
$w_{4,27} =$	$c\epsilon_T$	Cosine
$w_{4,28} =$	$s\psi_T$	Sine
$w_{4,29} =$	$s\epsilon_T$	Sine
$w_{4,30} =$	$w_{4,26}\, w_{4,27}$	Multiplication
$w_{4,31} =$	$w_{4,28}\, w_{4,27}$	Multiplication
$w_{4,32} =$	$w_{4,25}\, w_{4,30}$	Multiplication
$w_{4,33} =$	$w_{4,25}\, w_{4,31}$	Multiplication
$w_{4,34} =$	$w_{4,25}\, w_{4,29}$	Multiplication
$w_{4,35} =$	$w_{4,33}\left(w_{4,19} - w_{4,14}\right)$	Multiplication
$w_{4,36} =$	$w_{4,34}\left(w_{4,23} - w_{4,21}\right)$	Multiplication
$w_{4,37} =$	$\frac{1}{x_7}$	Power
$w_{4,38} =$	cx_{14}	Cosine
$w_{5,1} =$	$\frac{x_2}{x_9}$	Division
$w_{5,2} =$	$w_{4,8}\left(w_{4,11} - w_{4,17}\right)$	Multiplication
$w_{5,3} =$	$w_{4,3}\, w_{4,12}$	Multiplication
$w_{5,4} =$	$w_{4,8}\, w_{4,13}$	Multiplication
$w_{5,5} =$	$w_{4,3}\, w_{4,5}$	Multiplication
$w_{5,6} =$	$w_{4,8}\left(-w_{4,20} - w_{4,11}\right)$	Multiplication
$w_{5,7} =$	$w_{4,3}\, w_{4,16}$	Multiplication
$w_{5,8} =$	$w_{4,2}\left(w_{5,2} + w_{5,3}\right)$	Multiplication

$w_{5,9} =$	$w_{4,33}\left(w_{5,4} + w_{5,5}\right)$	Multiplication
$w_{5,10} =$	$w_{4,34}\left(w_{5,6} + w_{5,7}\right)$	Multiplication
$w_{6,0} =$	$w_{4,4}\, w_{4,7}$	Multiplication
$w_{6,1} =$	$\dfrac{x_3}{x_9}$	Division
$w_{6,2} =$	$w_{4,5}\, w_{4,15}$	Multiplication
$w_{6,3} =$	$w_{4,6}\, w_{4,9}$	Multiplication
$w_{6,4} =$	$w_{4,5}\, w_{4,11}$	Multiplication
$w_{6,5} =$	$w_{4,4}\, w_{4,38}$	Multiplication
$w_{6,6} =$	$w_{4,2}\left(w_{6,2} + w_{6,0}\right)$	Multiplication
$w_{6,7} =$	$w_{4,33}\, w_{6,3}$	Multiplication
$w_{6,8} =$	$w_{4,34}\left(w_{6,4} - w_{6,5}\right)$	Multiplication
$w_{8,1} =$	$x_1\, x_4$	Multiplication
$w_{8,2} =$	$x_2\, x_5$	Multiplication
$w_{8,3} =$	$x_3\, x_6$	Multiplication
$w_9 =$	$\dfrac{x_9\, u_8}{x_8}$	Multiplication and Division
$w_{11,1} =$	$\dfrac{x_{15}}{x_{16}}$	Division
$w_{11,2} =$	$\dfrac{w_{4,12}}{w_{4,6}}$	Division
$w_{11,3} =$	$w_{11,1}\, w_{11,2}$	Multiplication
$w_{12,1} =$	$w_{4,5}\, w_{4,38}$	Multiplication
$w_{12,2} =$	$w_{11,1}\, w_{12,1}$	Multiplication
$w_{13,0} =$	$\dfrac{x_{16}}{x_{15}}$	Division
$w_{13,1} =$	$w_{4,5}\, w_{4,11}$	Multiplication
$w_{13,2} =$	$\dfrac{w_{4,4}}{w_{4,6}}$	Division
$w_{13,3} =$	$\dfrac{w_{4,4}}{w_{4,38}}$	Division
$w_{13,4} =$	$x_{15}\, w_{4,38}$	Multiplication
$w_{13,5} =$	$w_{13,0}\, w_{4,12}$	Multiplication
$w_{13,6} =$	$w_{11,1}\, w_{13,3}$	Multiplication
$w_{13,7} =$	$\dfrac{w_{6,5} - w_{13,1}}{w_{4,38}}$	Division
$w_{13,8} =$	$w_{13,5}\, w_{13,2}$	Multiplication
$w_{13,9} =$	$w_{13,6}\, w_{6,3}$	Multiplication
$w_{13,10} =$	$\dfrac{w_{4,33}}{w_{13,4}}$	Division
$w_{14,1} =$	x_{16}^2	Power
$w_{14,2} =$	$w_{11,1}\, w_{4,38}$	Multiplication
$w_{14,3} =$	$w_{13,0}\, w_{4,6}$	Multiplication
$w_{14,4} =$	$w_{6,0}\, w_{4,5}$	Multiplication
$w_{14,5} =$	$\dfrac{w_{4,34}}{x_{15}}$	Division
$w_{14,6} =$	$\dfrac{w_{4,38}}{x_{15}}$	Division
$w_{14,7} =$	$w_{14,3}\left(w_{4,15} + w_{14,4}\right)$	Multiplication
$w_{14,9} =$	$\dfrac{w_{14,6}}{w_{14,1}}$	Division
$w_{15,1} =$	$x_{16}\, w_{4,6}$	Multiplication
$w_{15,2} =$	$w_{4,38}\, w_{4,10}$	Multiplication
$w_{15,3} =$	$\dfrac{w_{4,7}}{w_{14,1}}$	Division
$w_{15,4} =$	$w_{15,1}\left(w_{4,11} - w_{15,2}\right)$	Multiplication
$w_{16} =$	$x_{15}\, w_{4,7}$	Multiplication

$w_{27,1} =$	$x_{29}\,u_{28}$	Multiplication
$w_{27,2} =$	$x_{28}\,u_{29}$	Multiplication
$w_{27,3} =$	$x_{28}\,x_{29}$	Multiplication
$w_{27,4} =$	$x_{15}\left(w_{27,1} + w_{27,2}\right)$	Multiplication
$w_{27,5} =$	$w_{27,3}\,u_{15}$	Multiplication
$w_{27,6} =$	$x_{15}\left(w_{27,4} + w_{27,5}\right)$	Multiplication
$w_{28} =$	$u_{30}\,x_{28}$	Multiplication
$w_{30,1} =$	x_{31}^{9}	Power
$w_{30,2} =$	x_{31}^{8}	Power
$w_{30,3} =$	x_{31}^{7}	Power
$w_{30,4} =$	x_{31}^{6}	Power
$w_{30,5} =$	x_{31}^{5}	Power
$w_{30,6} =$	x_{31}^{4}	Power
$w_{30,7} =$	x_{31}^{3}	Power
$w_{30,8} =$	x_{31}^{2}	Power
$w_{30,9} =$	$u_{31}\left(10P_{\rho 10}\,w_{30,1} + 9P_{\rho 9}\,w_{30,2} + \cdots + 3P_{\rho 3}\,w_{30,8} + 2P_{\rho 2}\,x_{31} + P_{\rho 1}\right)$	Multiplication
$w_{32,1} =$	$x_{33}\,u_{15}$	Multiplication
$w_{32,2} =$	$x_{15}\,u_{33}$	Multiplication
$w_{32,3} =$	x_{33}^{2}	Power
$w_{32,4} =$	$\dfrac{w_{32,1} - w_{32,2}}{w_{32,3}}$	Division
$w_{33} =$	$\dfrac{u_{34}}{x_{33}}$	Division
$w_{34,2} =$	$\left(3P_{T3,2}\,w_{30,8} + 2P_{T2,2}\,x_{31} + P_{T1,2}\right)u_{31}$	Multiplication
$w_{34,3} =$	$\left(6P_{T6,3}\,w_{30,5} + 5P_{T5,3}\,w_{30,6} + 4P_{T4,3}\,w_{30,7} + 3P_{T3,3}\,w_{30,8} + 2P_{T2,3}\,x_{31} + P_{T1,3}\right)u_{31}$	Multiplication
$w_{34,4} =$	$\left(8P_{T8,4}\,w_{30,3} + 7P_{T7,4}\,w_{30,4} + \cdots + 3P_{T3,4}\,w_{30,8} + 2P_{T2,4}\,x_{31} + P_{T1,4}\right)u_{31}$	Multiplication

Now that all the auxiliary functions are known, they can be used to determine the required recurrence relations. In order to write these concisely and following the same convention as used by Scott and Martini (2008), a number of extra parameters will have to be introduced. The Taylor series coefficients can be written as described by Equation (6.45). Where n is the variable number and k is the order of the derivative.

$$\frac{x_n^{(k)}}{k!} \triangleq X_n(k) \quad \text{where} \quad k \geq 1 \tag{6.45}$$

A similar expression is defined for u_n and w_n as well as the place-holder functions f_n and g_n which are all shown in Equation (6.46).

$$U_n(k) \triangleq \frac{u_n^{(k)}}{k!} \qquad W_n(k) \triangleq \frac{w_n^{(k)}}{k!} \qquad F_n(k) \triangleq \frac{f_n^{(k)}}{k!} \qquad G_n(k) \triangleq \frac{g_n^{(k)}}{k!} \tag{6.46}$$

Then using Equations (6.41) and (6.45) a relation can be described between X_n and U_n as shown in Equation (6.47) (Scott and Martini, 2008).

$$\begin{aligned} u_n^{(k-1)} = x_n^{(k)} \quad &\Rightarrow \quad \frac{u_n^{(k-1)}}{(k-1)!} = \frac{x_n^{(k)}}{(k-1)!} \quad \Rightarrow \\ U_n(k-1) = kX_n(k) \quad &\Rightarrow \quad X_n(k) = \frac{U_n(k-1)}{k} \end{aligned} \tag{6.47}$$

As was mentioned before, all the auxiliary functions have been defined such that they incorporate one of the following operations: multiplication, division, power, exponential or trigonometric. This has been done because recurrence relations exist for these simple operations as provided by Jorba and Zou (2005). These recurrence relations are written using the definitions from Equation (6.46) and are described in Equations (6.48) to (6.54).

$$\text{for} \quad f_n \pm g_n \quad \Rightarrow \quad W_{n,\pm}(k) = F_n(k) \pm G_n(k) \tag{6.48}$$

$$\text{for} \quad f_n g_n \quad \Rightarrow \quad W_{n,mult}(k) = \sum_{j=0}^{k} F_n(j)\, G_n(k-j) \tag{6.49}$$

$$\text{for} \quad \frac{f_n}{g_n} \quad \Rightarrow \quad W_{n,div}(k) = \frac{1}{G_n(0)} \left[F_n(k) - \sum_{j=1}^{k} G_n(j)\, W_{n,div}(k-j) \right] \tag{6.50}$$

$$\text{for} \quad f_n^{\alpha} \quad \Rightarrow \quad W_{n,pow}(k) = \frac{1}{kF_n(0)} \sum_{j=0}^{k-1} \left[k\alpha - j(\alpha+1) \right] F_n(k-j)\, W_{n,pow}(j) \tag{6.51}$$

$$\text{for} \quad e^{f_n} \quad \Rightarrow \quad W_{n,exp}(k) = \frac{1}{k} \sum_{j=0}^{k-1} (k-j)\, W_{n,exp}(j)\, F_n(k-j) \tag{6.52}$$

$$\text{for} \quad \cos f_n \quad \Rightarrow \quad W_{n,cos}(k) = -\frac{1}{k} \sum_{j=1}^{k} j\, W_{n,sin}(k-j)\, F_n(j) \tag{6.53}$$

$$\text{for} \quad \sin f_n \quad \Rightarrow \quad W_{n,sin}(k) = \frac{1}{k} \sum_{j=1}^{k} j\, W_{n,cos}(k-j)\, F_n(j) \tag{6.54}$$

Checking the presented equations it can be seen that Equations (6.53) and (6.54) are interdependent. This means that whenever a cosine recurrence relation has to be computed, the same recurrence relation for sine (with at least order $k-1$) has to be computed at the same time (and vice versa). All these equations have been derived by Jorba and Zou (2005) using the general Leibniz rule for the k^{th} derivative of a multiplication as portrayed in Equation (6.55).

$$\left(f_n g_n\right)^{(k)} = \sum_{j=0}^{k} \binom{k}{j} f_n^{(j)}\, g_n^{(k-j)} \tag{6.55}$$

Combining Equation (6.55) and the definition of the reduced derivative for the auxiliary function (see Equation (6.46)) results in Equation (6.56). This equation, when rewritten to include the reduced derivatives for f_n and g_n, results in Equation (6.49).

$$W_{n,mult}(k) = \frac{1}{k!} \sum_{j=0}^{k} \binom{k}{j} f_n^{(j)}\, g_n^{(k-j)} \tag{6.56}$$

Using the basic recurrence relations, the auxiliary function provided in Table 6.2 can be written to form the required recurrence relations as is shown by Equation (6.57). Here a sample recurrence relation is provided for each of the basic relations as well as the recurrence relation for w_9 since this involves both a multiplication as well as a division.

$$
\begin{aligned}
W_{8,1}(k) = x_1 x_4 &= \sum_{j=0}^{k} X_1(j) X_4(k-j) = x_1 \frac{U_4(k-1)}{k} + x_4 \frac{U_1(k-1)}{k} + \sum_{j=1}^{k-1} \frac{U_1(j-1)}{j} \frac{U_4(k-j-1)}{k-j} \\
W_{4,1}(k) = \frac{x_1}{x_9} &= \frac{1}{x_9}\left[X_1(k) - \sum_{j=1}^{k} X_9(j) W_{4,1}(k-j)\right] = \frac{1}{x_9}\left[\frac{U_1(k-1)}{k} - \sum_{j=1}^{k} \frac{U_9(j-1)}{j} W_{4,1}(k-j)\right] \\
W_{14,1}(k) = x_{16}^2 &= \frac{1}{k x_{16}} \sum_{j=0}^{k-1} [2k - j(2+1)] X_{16}(k-j) W_{14,1}(j) \\
&= \frac{1}{k x_{16}} \sum_{j=0}^{k-1} [2k - j(2+1)] \frac{U_{16}(k-j-1)}{k-j} W_{14,1}(j) \\
X_{28}(k) = \frac{U_{28}(k-1)}{k} = e^{x_{30}} &= \frac{1}{k} \sum_{j=0}^{k-1} (k-j) X_{28}(j) X_{30}(k-j) = \frac{1}{k} \sum_{j=0}^{k-1} (k-j) \frac{U_{28}(j-1)}{j} \frac{U_{30}(k-j-1)}{k-j} \\
&= x_{28} \frac{U_{30}(k-1)}{k} + \frac{1}{k} \sum_{j=1}^{k-1} (k-j) \frac{U_{28}(j-1)}{j} \frac{U_{30}(k-j-1)}{k-j} \\
W_{4,6}(k) = \cos x_{12} &= -\frac{1}{k} \sum_{j=0}^{k} j W_{4,4}(k-j) X_{12}(j) = -\frac{1}{k} \sum_{j=1}^{k} j W_{4,4}(k-j) \frac{U_{12}(j-1)}{j} \\
W_{4,4}(k) = \sin x_{12} &= \frac{1}{k} \sum_{j=0}^{k} j W_{4,6}(k-j) X_{12}(j) = \frac{1}{k} \sum_{j=1}^{k} j W_{4,6}(k-j) \frac{U_{12}(j-1)}{j} \\
W_9(k) = \frac{x_9 u_8}{x_8} &= \frac{1}{x_8}\left[\sum_{j=0}^{k} X_9(j) U_8(k-j) - \sum_{j=1}^{k} X_8(j) W_9(k-j)\right] \\
&= \frac{1}{x_8}\left[x_9 U_8(k) + \sum_{j=1}^{k} \frac{U_9(j-1)}{j} U_8(k-j) - \sum_{j=1}^{k} \frac{U_8(j-1)}{j} W_9(k-j)\right]
\end{aligned}
\tag{6.57}
$$

Notice how all $X_n(k)$ have been replaced by $\frac{U_n(k-1)}{k}$. This way, all recurrence relations are a function of the previous recurrence relations and the initial conditions only. Using this same notation and provided the auxiliary functions, the equations presented in Equations (6.42) and (6.43) can be written as recurrence relations. These are shown by Equations (6.58) to (6.60) and hold for $k \geq 1$.

$$
\begin{aligned}
U_1(k) &= X_4(k) = \frac{U_4(k-1)}{k} \\
U_2(k) &= X_5(k) = \frac{U_5(k-1)}{k} \\
U_3(k) &= X_6(k) = \frac{U_6(k-1)}{k} \\
U_4(k) &= -\mu_M W_{4,1}(k) + W_{4,24}(k) + W_{4,35}(k) - W_{4,36}(k) \\
U_5(k) &= -\mu_M W_{5,1}(k) + W_{5,8}(k) + W_{5,9}(k) - W_{5,10}(k) \\
U_6(k) &= -\mu_M W_{6,1}(k) + W_{6,6}(k) - W_{6,7}(k) - W_{6,8}(k) \\
U_7(k) &= 0 \\
U_8(k) &= 2W_{8,1}(k) + 2W_{8,2}(k) + 2W_{8,3}(k) \\
U_9(k) &= \frac{3}{2} W_9(k) \\
U_{10}(k) &= 0
\end{aligned}
\tag{6.58}
$$

$$
\begin{aligned}
U_{11}(k) &= W_{11,3}(k) \\
U_{12}(k) &= W_{12,2}(k) \\
U_{13}(k) &= 2\Omega_M W_{13,7}(k) + W_{13,8}(k) + \Omega_M^2 W_{13,9}(k) - W_{13,10}(k) \\
U_{14}(k) &= 2\Omega_M W_{6,3}(k) + W_{14,2}(k) + \Omega_M^2 W_{14,7}(k) + W_{14,5} - \mu_M W_{14,9}(k) \\
U_{15}(k) &= \Omega_M^2 W_{15,4}(k) + W_{4,2}(k) - \mu_M W_{15,3}(k) \\
U_{16}(k) &= W_{16}(k)
\end{aligned}
\tag{6.59}
$$

$$
\begin{aligned}
U_{27}(k) &= \frac{1}{2} S W_{27,6}(k) \\
U_{28}(k) &= W_{28}(k) \\
U_{29}(k) &= \begin{cases}
U_{29,1}(k) = 0, & \text{for } 0 \le x_{32} < 0.5 \\
U_{29,2}(k) = P_{C_D 1,2} U_{32}(k), & \text{for } 0.5 \le x_{32} < 1 \\
U_{29,3}(k) = P_{C_D 1,3} U_{32}(k), & \text{for } 1 \le x_{32} < 1.3 \\
U_{29,4}(k) = P_{C_D 1,4} U_{32}(k), & \text{for } 1.3 \le x_{32} < 2.5 \\
U_{29,5}(k) = P_{C_D 1,5} U_{32}(k), & \text{for } 2.5 \le x_{32} < 4 \\
U_{29,6}(k) = 0, & \text{for } x_{32} \ge 4
\end{cases} \\
U_{30}(k) &= W_{30,9}(k) \\
U_{31}(k) &= U_{16}(k) \\
U_{32}(k) &= W_{32,4}(k) \\
U_{33}(k) &= \frac{\gamma_a R_a^*}{2} W_{33}(k) \\
U_{34}(k) &= \begin{cases}
U_{34,1}(k) = P_{T1,1} U_{31}(k), & \text{for } -0.6 \le x_{31} < 5.04 \\
U_{34,2}(k) = W_{34,2}(k), & \text{for } 5.04 \le x_{31} < 35.53 \\
U_{34,3}(k) = W_{34,3}(k), & \text{for } 35.53 \le x_{31} < 75.07 \\
U_{34,4}(k) = W_{34,4}(k), & \text{for } 75.07 \le x_{31} < 170.05 \\
U_{34,5}(k) = 0, & \text{for } x_{31} \ge 170.05
\end{cases}
\end{aligned}
\tag{6.60}
$$

These equations are now all a function of the recurrence relations corresponding to the rest of the auxiliary functions, which are all basic recurrence relations as mentioned in Table 6.2.

With all the recurrence relations now known, and using the definition of Equation (6.47) the updated state can be described using the Taylor series expansion as described in Equation (6.61).

$$
x_n(t+h) = \sum_{k=0}^{K} \frac{x_n^{(k)}(t)}{k!} h^k + T_{n,K} = \sum_{k=0}^{K} X_n(k) h^k + T_{n,K} \quad \text{with } n = 1, \ldots, 7 \tag{6.61}
$$

Here K is the order of the series to which it has to be evaluated and $T_{n,K}$ is the truncation error. Please note that all the Taylor series coefficients computed for the auxiliary equations are only needed to determine the Taylor series coefficients of the state variables (which is why they are auxiliary).

7

PROGRAM OPTIMISATION TOOL

To perform the analysis associated with this thesis, a simulation and optimisation program is used. This optimisation tool is comprised of both existing (Section 7.1) and newly developed software (Section 7.2). It is written in C++ and is based on the Tudat structure. The purpose of the software is to simulate the trajectory of the MAV and optimise this trajectory with respect to the lowest propellant mass required. This tool is written such that the performance of Runge-Kutta-Fehlberg 4^{th} (5^{th}) order (RKF45) and TSI can be compared.

7.1. EXISTING SOFTWARE

The use of existing software can greatly improve the performance of the final tool and save time as well. Another important reason to use existing software is that this will make it easier for other people to use and incorporate into their software as well. The existing software used for this thesis is software that is currently being used by the space department of the TU Delft and (in case of SNOPT and Mars-GRAM) by the mission design section at JPL.

7.1.1. TUDAT

Tudat is, as the name suggests, a toolbox that can be used to solve numerous astrodynamic problems (Dirkx *et al.*, 2016). It was and still is being developed by students and staff of the Delft University of Technology. Specifically by the section Astrodynamics and Space missions of the Aerospace Engineering faculty. It is programmed in C++ and consists of a number of libraries. These libraries can be called upon by the user to invoke different Tudat functionalities such as standard reference frame transformations or often used integrators. The available software is completely validated and comes with its own tests to make sure that everything is working properly. It itself uses two external libraries: Eigen and Boost. Both these libraries will be discussed in Sections 7.1.2 and 7.1.3 respectively. Figure 7.1 shows Tudat with the different libraries that are used within the Tudat Bundle including the core functions (Tudat Core). In this thesis, the Tudat libraries are used for all standard mathematical and astrodynamic operations.

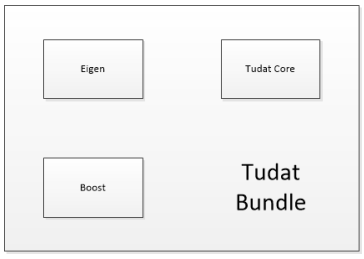

Figure 7.1: Tudat structure

7.1.2. EIGEN

Eigen is an external C++ library that was written to perform linear algebra computations [1]. The software is free and easy to use, which is why it is widely used by the C++ community and thus also within Tudat (Dirkx *et al.*, 2016). Another advantage is that because it does not use any source files, it does not need to be build before using it. The Eigen libraries contain a number of standardized matrices and vectors, each with its own characteristics. An example of an often used vector is *Vector3d* (or *Eigen::Vector3d*), which can for instance be used to store the Cartesian position of a satellite. Here the *3* shows that it can store 3 values/parameters and the *d* shows that these are of the type *double*. It is mentioned on the Tudat wiki (Dirkx *et al.*, 2016) that these Eigen vectors and matrices should only be used if required for linear algebra computations. For ordinary storage, the C++ arrays, vectors and matrices should be used to save both storage and computation time.

7.1.3. BOOST

Boost is a slightly more complicated set of C++ libraries, where compared to the Eigen library, Boost first has to be compiled before being able to use all of its functionalities. Fortunately, this compiling is performed by Tudat automatically when setting it up for the first time. Boost is described as an addition to the standard C++ libraries, thus adding more functionalities (Dirkx *et al.*, 2016) [2]. Within Tudat, Boost is used to pass free and class functions as an argument to another object and also for dynamic allocation using so-called pointers. Four libraries that are often used within Tudat are *boost::function*, *boost::bind*, *boost::shared_ptr* and *boost::make_shared*. The first two libraries are used to pass functions (a function is pointed to by *function* and called by *bind*) and the last two are used in case of dynamic allocation (*shared_ptr* is the pointer and *make_shared* is the object creator that returns a shared pointer to the created object).

7.1.4. PAGMO

PaGMO is a free optimisation tool developed by European Space Agency (ESA)s Advanced Concepts Team (ACT). It uses parallel computations to perform the optimisation and can even optimise for multi-objective problems. Parallel computation is the act of performing multiple computations on the same machine using different CPU cores. This allows the cost function to be computed for different sets of optimisation parameters at the same time and thus reducing the total CPU time required. However, this only works if the cost function evaluations are independent, which is not always the case (e.g. Dynamic Differential Evolution (DE) described by Qing (2009)). The tool itself incorporates many different local and global optimisation methods as mentioned by Izzo (2012), among which the optimisation method used in this thesis Monotonic Basin Hopping (MBH). This method has been written in PaGMO in such a way that it can use any of the provided local optimisers. PaGMO is written in C++ and requires the shared libraries of Boost to run [3]. Interfaces to external libraries are also provided, which can incorporate for instance SNOPT as a local optimisation method. In this thesis SNOPT is used as the local optimiser for MBH as implemented by PaGMO. More information on SNOPT is provided in Section 7.1.5. For SNOPT to be recognised by PaGMO, it has to be installed separately. Figure 7.2 shows PaGMO with the internally used Boost library and the externally called SNOPT software.

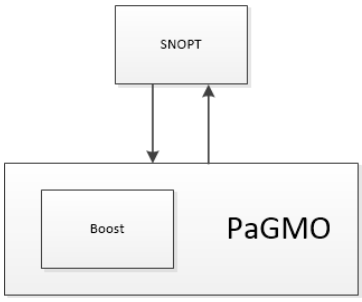

Figure 7.2: PaGMO structure

[1] More documentation on Eigen can be found on eigen.tuxfamily.org/dox/ [Accessed 8 March 2016]
[2] More documentation on Boost can be found on http://www.boost.org/ [Accessed 8 March 2016]
[3] More documentation on PaGMO can be found on https://esa.github.io/pagmo/ [Accessed 9 March 2016]

7.1.5. SNOPT

SNOPT was introduced by Gill *et al.* (2002) as a Sequential Quadratic Programming (SQP) method. It uses the first function derivatives and is very effective with highly constrained problems such as trajectory optimisation. Because it is based on SQP it is only able to find the local optimum and it is thus not guaranteed that this is also the global optimum. By combining SNOPT and MBH the global optimum can indeed be found or approached. The tool itself does not require that many evaluations, which is why it is very useful for complex problems with many optimisation variables (Gill *et al.*, 2008). The code for SNOPT has been written in Fortran, but can easily be translated to C,C++ using *f2c* which is provided with SNOPT as well [4]. This way it can be called by PaGMO. It should be noted that SNOPT is not free and can only be used under a licence agreement.

7.1.6. MARS-GRAM

Mars-GRAM is a high-fidelity atmospheric model developed by NASA to simulate the global atmospheric conditions on Mars (Justh and Justus, 2008) [5]. The model is based on NASA Ames Mars General Circulation Model (for altitudes between 0-80 km) and Mars Thermospheric General Circulation model (for altitude above 80 km). It can provide density, temperature and pressure data (among other data) with respect to the current altitude, latitude and longitude on Mars. Seasonal variations are taken into account in the model as well, which is why different calender dates will result in different atmospheric compositions. The tool can be used within a simulation tool or as a separate executable. Unfortunately, because it is so detailed, each computation requires a lot of CPU time. This is why it was decided to use the stand-alone Mars-GRAM executable to generate a detailed table with atmospheric data as a function of altitude, latitude and longitude at the start of the optimisation. Even generating this table required a lot of CPU time (on average a single computation using the stand-alone executable took 67.9 seconds to complete). The starting altitude was set at -0.6 km MOLA and advanced with a step-size of 0.1 km to 320 km altitude to cover the entire range that the MAV would have to cover. Also, the latitude and longitude were varied within 10 degrees from the launch site with a step-size of 1 degree. A Matlab script was written to extract the relevant atmospheric data from the Mars-GRAM output files and write them into a .csv file, thus creating the required atmospheric data table. The atmospheric data in this table was then interpolated to provide an estimate of the atmospheric characteristics at every point along the ascent trajectory, which is required to compute the drag at each time step. Some of the earlier versions of Mars-GRAM are available for free (such as the Mars-GRAM 2005 version used in this thesis), however, the latests versions (such as the Mars-GRAM 2010 version used as a back-up in this thesis) require a licence agreement.

7.2. DEVELOPED SOFTWARE

This section of the software chapter describes the software that either had to be developed around existing software/libraries or had to be developed from scratch (the TSI propagator). Each piece of software is accompanied by the corresponding software architecture. Every next piece of software then indirectly incorporates the previous architecture through the use of the completed tool.

7.2.1. ATMOSPHERIC TABLE FUNCTION FIT

Using Mars-GRAM 2005, a table containing altitude, latitude and longitude dependent temperature and density data was produced. The altitude range was -0.6 to 320 km MOLA with a step-size of 0.01 km, the latitude and longitude ranges were centred around the launch site (21.0 °N and 74.5 °E) with a 10 degree range in each direction and a step-size of 1 degree. The rest of the input parameters were constant and can be seen in Appendix A. The temperature and density data produced is shown in Figures 7.3 and 7.4 respectively for 9 latitude and longitude combinations including the launch site itself.

Unfortunately discontinuous data tables cannot be used when integrating using TSI, which is why both these data tables had to be fitted with continuous functions. The temperature data could not be smoothly fit with one continuous function. Therefore, depending on the altitude range, a different approximation function is required. The condition to be met for a proper fit came from the differences in the temperature-altitude and density-altitude curves, where the maximum difference with respect to the launch site curve was taken. The requirement for the standard deviation of the polynomial curve fit was then to be (at least) one order

[4] More documentation on SNOPT can be found on http://www.sbsi-sol-optimize.com/asp/sol_products_snopt_desc.htm [Accessed 9 March 2016]

[5] NASA website: https://see.msfc.nasa.gov/model-Marsgram [Accessed 9 March 2016]

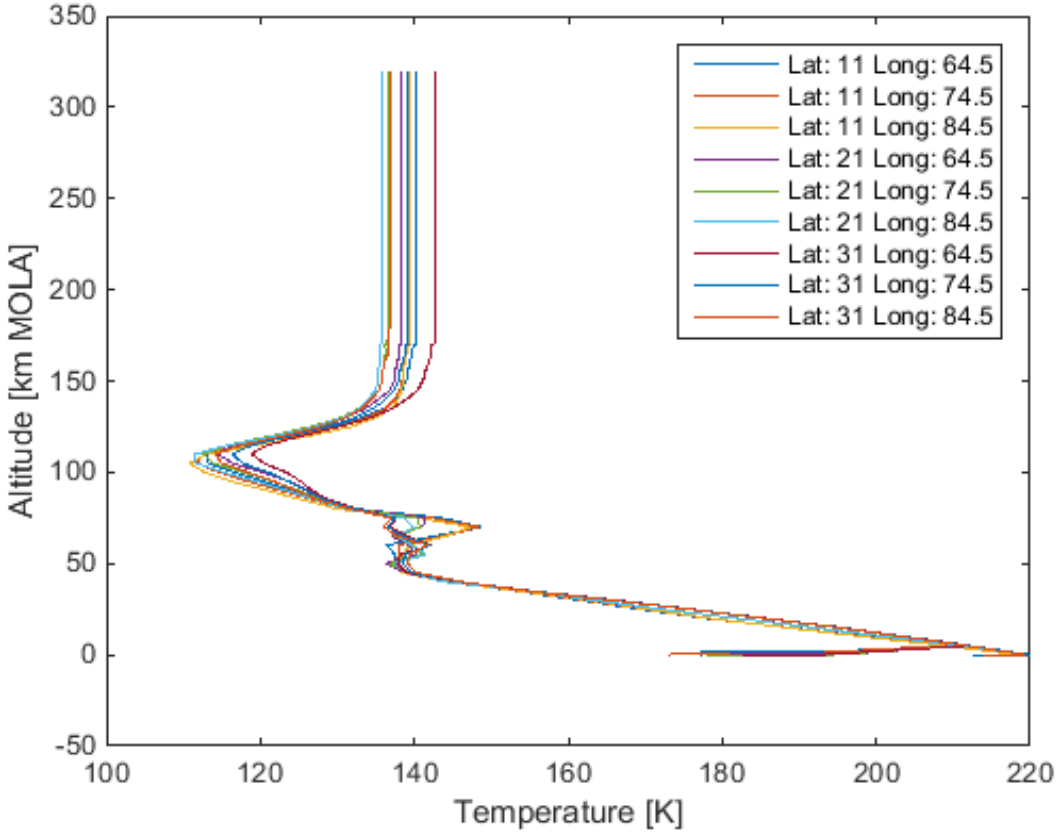

Figure 7.3: Temperature data generated with Mars-GRAM 2005 showing 9 different latitude and longitude combinations

lower than this maximum difference and that the maximum difference between the fit and the launch site curve was lower than the maximum difference. The temperature-altitude curve was split into 5 sections as roughly visualised in Figure 7.5. The number of sections come from both the shape of the curves and the requirement for accuracy and maximum order of the polynomial, which is set at 8 because otherwise the polynomial would get too long. Also, the number of sections were to be kept at a minimum. More information on the fitting process and early results is provided in Appendix B.

Each section was fit with a polynomial function of the n^{th} order where the function is represented by Equation (7.1). The last section shows a constant temperature, thus the temperature of the launch site curve was chosen to represent this final section, which is equal to 136.5 K.

$$y = p_n x^n + p_{n-1} x^{n-1} + \cdots + p_1 x + p_0 \tag{7.1}$$

A lower order is preferred, because then the fitted function will be simpler to evaluate and contain fewer terms. However the order has to be high enough to meet the accuracy requirements. Table 7.1 shows the orders that were required and the deviations to the launch site temperature-altitude curve. The actual corresponding parameters are provided in Table 7.2. It should be noted that the first few temperature data values were so different from the rest of the curve that it was assumed that this is a lack of the Mars-GRAM program and were thus treated as outliers.

The complete polynomial fit for the launch site curve for the temperature is shown in Figure 7.6.

The density fit was slightly more difficult because the curves are all very similar and thus result in a higher accuracy requirement for the fit. At first glance it looks like a natural logarithmic function, unfortunately an ordinary exponential did not fit the curve. This is why a more extensive exponential fit was required. The natural logarithm of the data has been plotted in Figure 7.7.

With the data represented in the logarithmic domain, again a polynomial function can be fit. The total fit would then satisfy Equation (7.2).

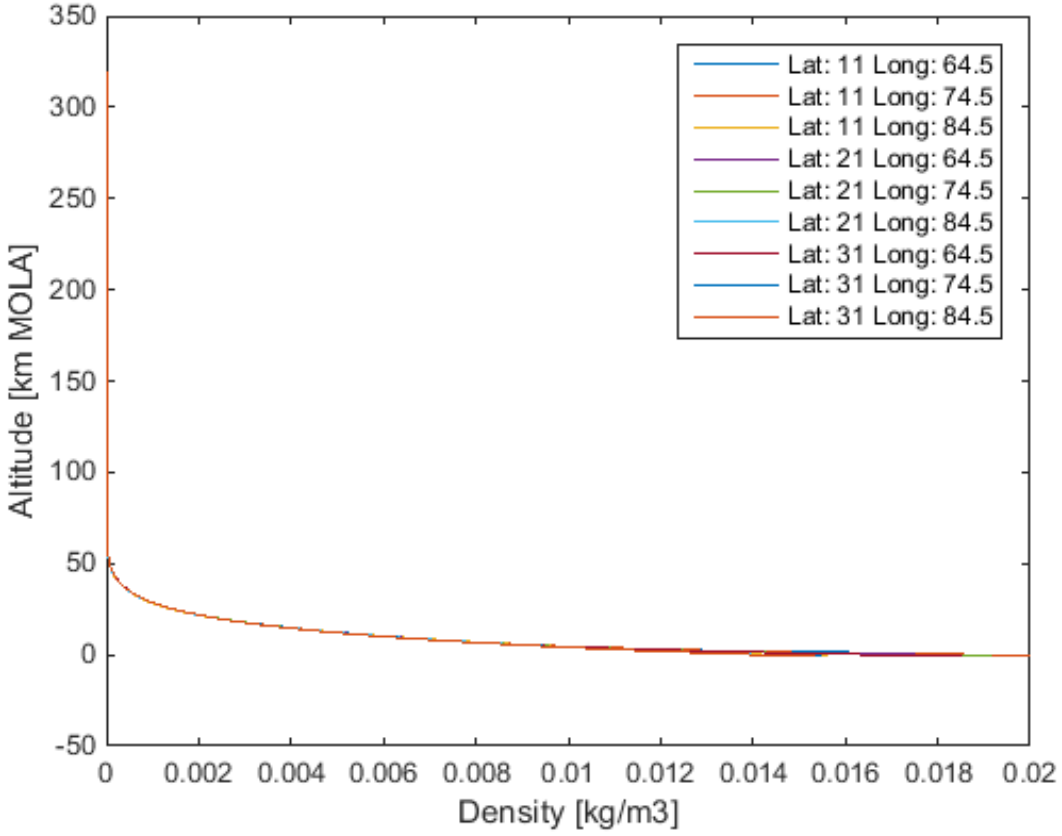

Figure 7.4: Density data generated with Mars-GRAM 2005 showing 9 different latitude and longitude combinations

Table 7.1: Temperature curve fit data all with respect to the launch site curve (Latitude and longitude of the launch site)

Section	**Altitude range [km MOLA]**	**Order**	**Maximum polynomial standard deviation [K]**	**Maximum polynomial difference [K]**	**Maximum data curves difference [K]**
1	-0.6 to 5.04	1	0.0312	25.8	0.177
2	5.04 to 35.53	2	0.287	3.90	0.7056
3	35.53 to 75.07	6	0.624	8.00	1.69
4	75.07 to 170.05	8	0.523	6.60	2.45

Table 7.2: Temperature curve fit parameters (rounded to 3 decimal points)

Section	p_8	p_7	p_6	p_5	p_4	p_3	p_2	p_1	p_0
1								3.415	194.165
2							0.006	-2.130	222.052
3			$-5.388 \cdot 10^{-7}$	$1.785 \cdot 10^{-4}$	-0.0243	1.733	-68.294	$1.407 \cdot 10^{3}$	$-1.167 \cdot 10^{4}$
4	$4.1942 \cdot 10^{-12}$	$-4.328 \cdot 10^{-9}$	$1.931 \cdot 10^{-6}$	$-4.862 \cdot 10^{-4}$	0.076	-7.405	447.378	$-1.523 \cdot 10^{4}$	$2.236 \cdot 10^{5}$

$$y = exp\left(p_n x^n + p_{n-1} x^{n-1} + \cdots + p_1 x + p_0\right) \tag{7.2}$$

The same polynomial requirements as for the temperature curve were enforced for the density curve as

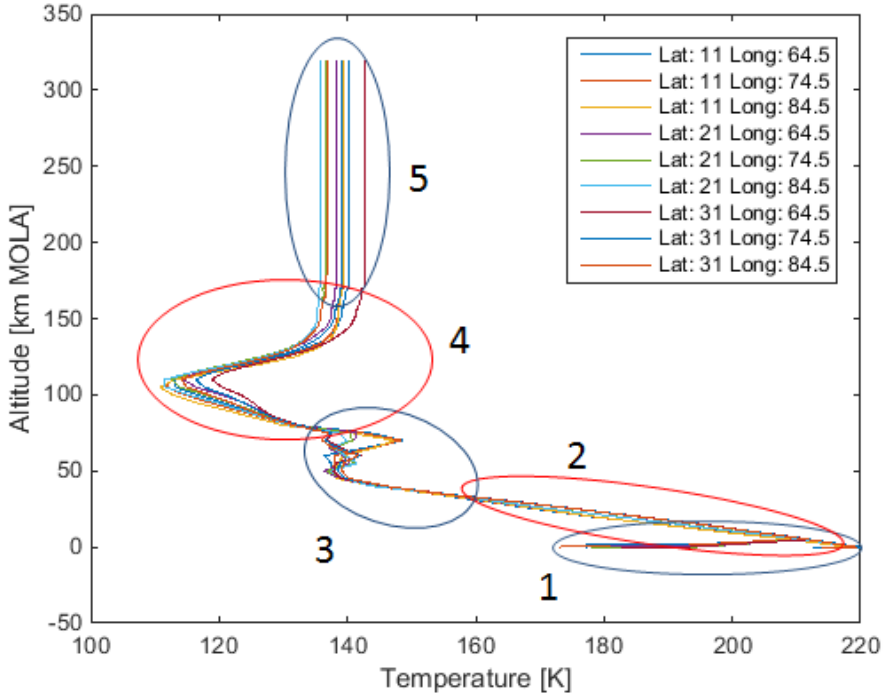

Figure 7.5: Different temperature curve sections

well. However, because the polynomial is used in an exponential, some extra requirements are needed to assure the accuracy of the fit. One requirement is that the maximum difference between the final exponential fit and the normal launch site density curve is smaller than the maximum difference between all the data curves. Also, in this case the standard deviation of the difference between the exponential fit and the normal launch site curve had to be within the range of standard deviations of the difference between the different data curves. This meant that even though an 8^{th} order polynomial fit could be achieved for the natural logarithmic data with the required accuracy, when converted to the exponential fit, the last two requirements were not met. Before it was mentioned that an order higher than 8 was not desirable. However, in this case, a single exponential fit could be achieved using a 10^{th} order polynomial. This fit meant that the density curve did not have to be split up at all, which makes the integration slightly easier. Therefore, it was decided that a 10^{th} order polynomial was acceptable in this case. The results of the fit is presented in Tables 7.3 and 7.4 and the polynomial and exponential fit curves are shown in Figures 7.8 and 7.9 respectively.

Table 7.3: Density curve fit data (10^{th} order polynomial) with respect to the launch site curve (Latitude and longitude of the launch site)

Maximum polynomial standard deviation [kg/m^3]	0.0501
Maximum polynomial difference [kg/m^3]	0.160
Maximum natural logarithmic data curves difference [kg/m^3]	0.460
Maximum exponential difference with launch site curve [kg/m^3]	$2.826 \cdot 10^{-3}$
Maximum data curves difference [kg/m^3]	$3.910 \cdot 10^{-3}$
Standard deviation exponential fit difference [kg/m^3]	$1.167 \cdot 10^{-4}$
Maximum standard deviation data curves difference [kg/m^3]	$2.106 \cdot 10^{-4}$

7.2.2. DRAG COEFFICIENT GRAPH FUNCTION FIT

Similar to the temperature and density curves, the relation between Mach number and drag coefficient, as depicted in Figure 7.10, had to be modelled as a continuous function as well. Again, it could not be fitted using one continuous function, but instead had to be modelled by different functions.

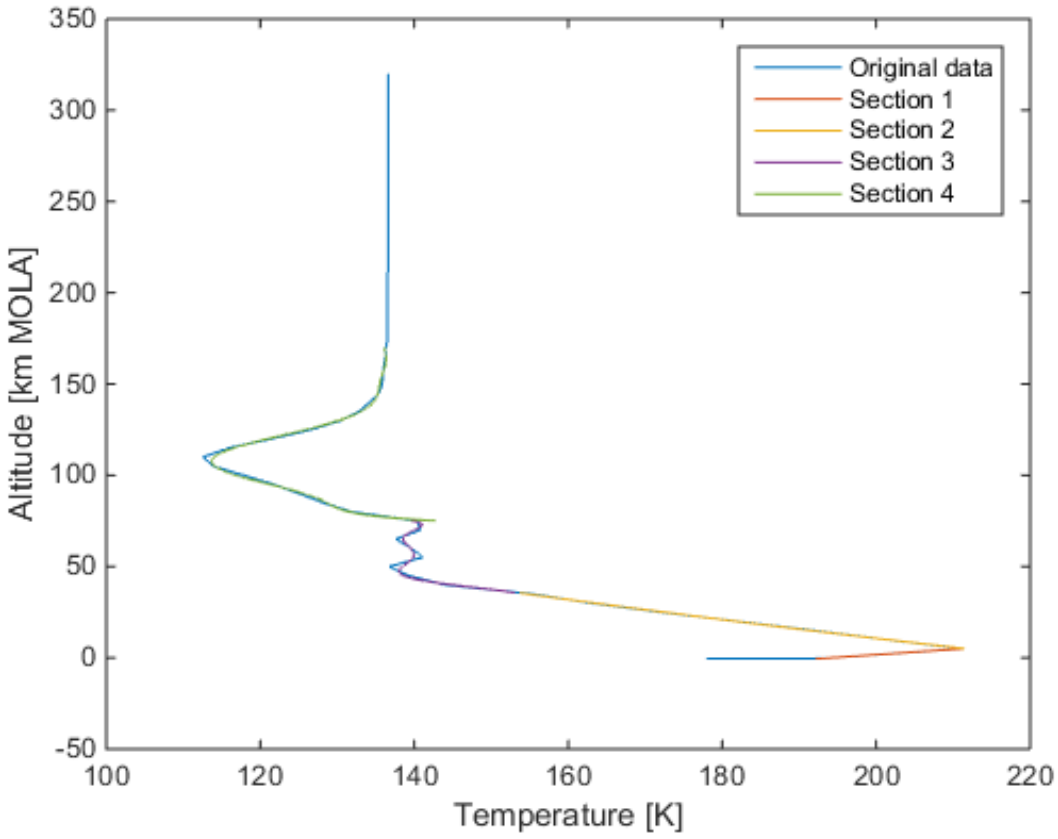

Figure 7.6: All section fits for the launch site temperature data curve

Table 7.4: Density curve fit parameters (rounded to 3 decimal points)

p_{10}	p_9	p_8	p_7	p_6	p_5	p_4	p_3	p_2	p_1	p_0
2.287 $\cdot 10^{-21}$	-3.724 $\cdot 10^{-18}$	2.559 $\cdot 10^{-15}$	-9.620 $\cdot 10^{-13}$	2.146 $\cdot 10^{-10}$	-2.884 $\cdot 10^{-8}$	2.273 $\cdot 10^{-6}$	-9.604 $\cdot 10^{-5}$	1.414 $\cdot 10^{-3}$	-0.0962	-4.172

Fortunately, this curve is already an approximation and thus consists of linear elements only. It can be split up into 6 different sections where the first and last section are constant (C_D is 0.2 and 0.3 respectively). Using a similar polynomial fit as before, but now for 1 order only, a linear fit could be made for each of the remaining 4 sections. The corresponding parameters are shown in Table 7.5 and the curve fit is shown in Figure 7.11.

Table 7.5: Drag coefficient curve fit parameters (rounded to 3 decimal points)

Section	Mach range	p_1	p_0
2	0.5 to 1	0.400	-2.483 $\cdot 10^{-16}$
3	1 to 1.3	0.567	-0.167
4	1.3 to 2.5	-0.142	0.754
5	2.5 to 4	-0.0667	0.567

7.2.3. RK4 AND RKF PROPAGATOR

The RK4 and RKF (or traditional) propagator architecture is described in Figure 7.12. It starts with the current state, which is then passed on to the state derivative function. The state derivative function is used by the RK4

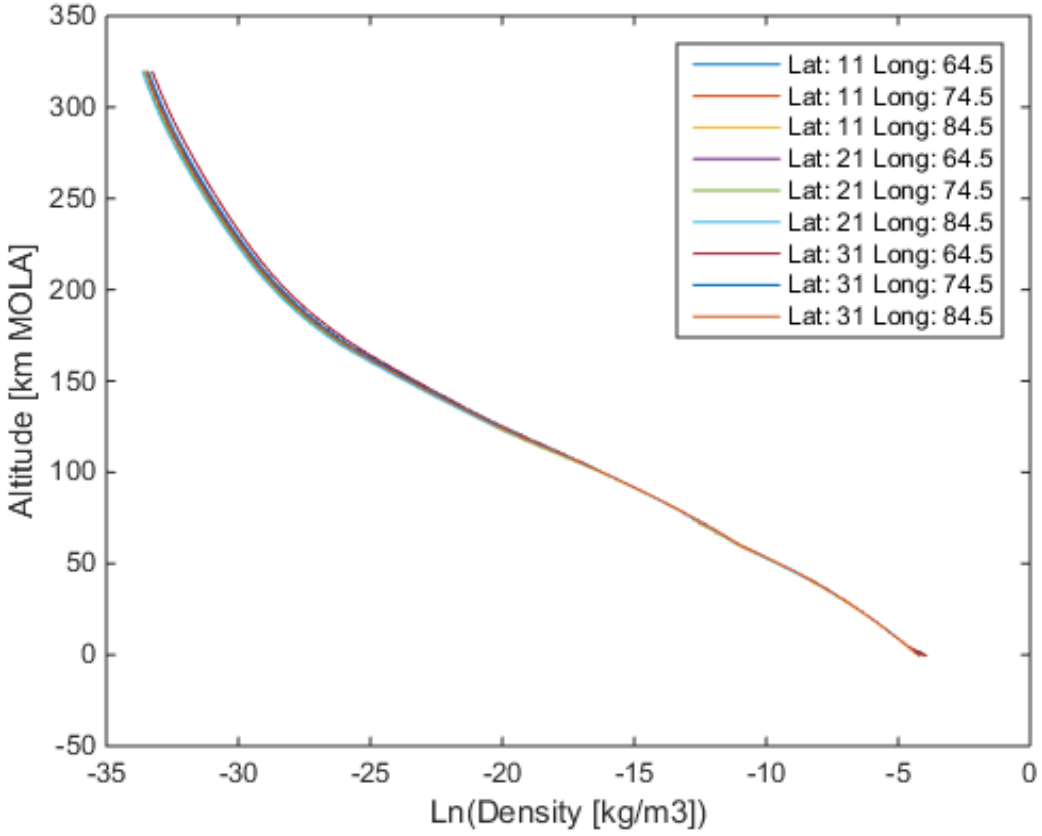

Figure 7.7: Natural logarithmic plot of the density data

and RKF integrators to determine the next state by calling the function a number of times depending on the used method. Both RK4 and RKF45 (and higher order RKF integrators) are already available through the Tudat libraries. RK4 can be called by including the rungeKutta4Integrator.h header file, and RKF45 can be called by including the rungeKuttaVariableStepSizeIntegrator.h header file. This integration process is repeated until the final condition is met. Within the state derivative function all the sate derivatives are updated and stored. The current position is used to update the gravitational acceleration on the MAV, the current mass is used to determine the accelerations caused by the thrust and finally the complete state is required to determine the accelerations caused by the drag. Both the drag and thrust accelerations have to be transformed to the inertial frame using the updated angles from the current state. The function also computes the current mass flow rate, however since the thrust is constant, this does not change over time. In the state derivative function, all the transformations are governed by pre-developed functions within the Tudat library, which includes the state transformations and the frame transformation from the body frame to the inertial frame. The transformation from the propulsion frame to the body frame is however not included in Tudat and had to be written.

All blocks represent a different action. These actions might be performed in classes, header files and/or source files. More information on the classification of the different blocks can be found in Appendix C.

7.2.4. TSI PROPAGATOR

The TSI propagator has a significantly different architecture compared to the traditional propagator as can be seen in Figure 7.13. TSI requires an initial order and step-size to start the integration process. In this thesis it has been decided to keep the order the same throughout the entire integration. The step-size will change during the integration depending on the Taylor series evaluations. The initial state is set as the current state and is fed into the TSI block. Within this block, first the auxiliary equations and functions are called, which were set-up for this particular problem. They are evaluated using the current state. These auxiliary equations and functions already include all the reference frame and coordinate transformations, as well as approximate atmospheric parameter functions. This is required to set-up the recurrence relations, which is where TSI

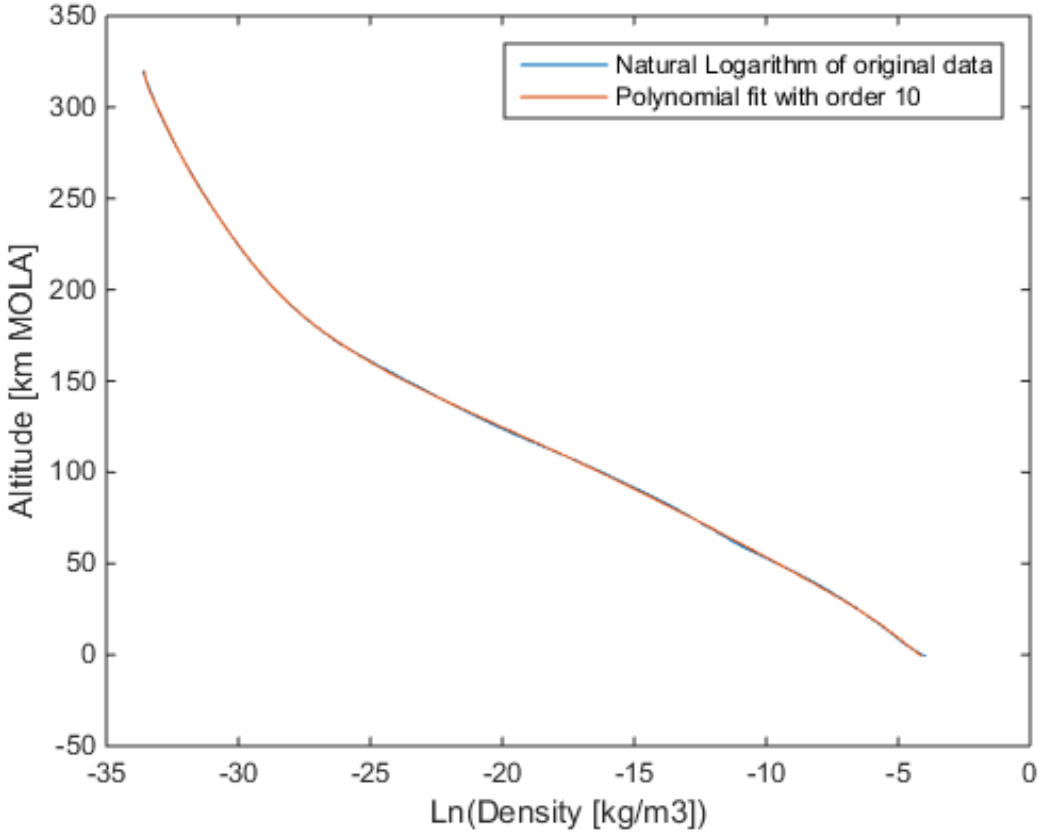

Figure 7.8: Polynomial fit for the launch site density data curve

differs from the traditional propagator. Once the auxiliary equations and functions have been computed they are used to compute the Taylor coefficients through the recurrence relations set-up for the thesis problem. These coefficients are then stored for later use and are also passed to the block creating the Taylor series expansion for every state variable thus creating he updated state. The last two coefficients are then used to determine the next step-size. This continues until the final integration condition has been met.

7.2.5. OPTIMISER

The optimisation software is a combination of the SNOPT local optimisation tool and PaGMOs MBH. Even though both these tools were already available and did not have to be developed, it is still important to understand how the rest of the software interacts with the optimiser. This is why Figure 7.14 shows the architecture of the MBH optimiser. It starts with the initial generation of the optimisation parameters, after which the 'Number of not improved iterations' is set to zero. This is then fed into the local optimiser, where the trajectory is integrated using the previously described tools. Once a local optimised trajectory is found, it is stored if it is better than the previous local optimised trajectory and the counter is set to zero again. If the newly found trajectory is not better than the current best the 'Number of not improved iterations' is increased by one. Once the maximum number of not improved iterations is met, the current best optimal trajectory (which is the optimum for the current "funnel") is stored and the process is repeated till the final global optimisation condition is met. At this point the global optimum is the best optimal trajectory from all the funnels computed at that time, which is then returned as the program solution.

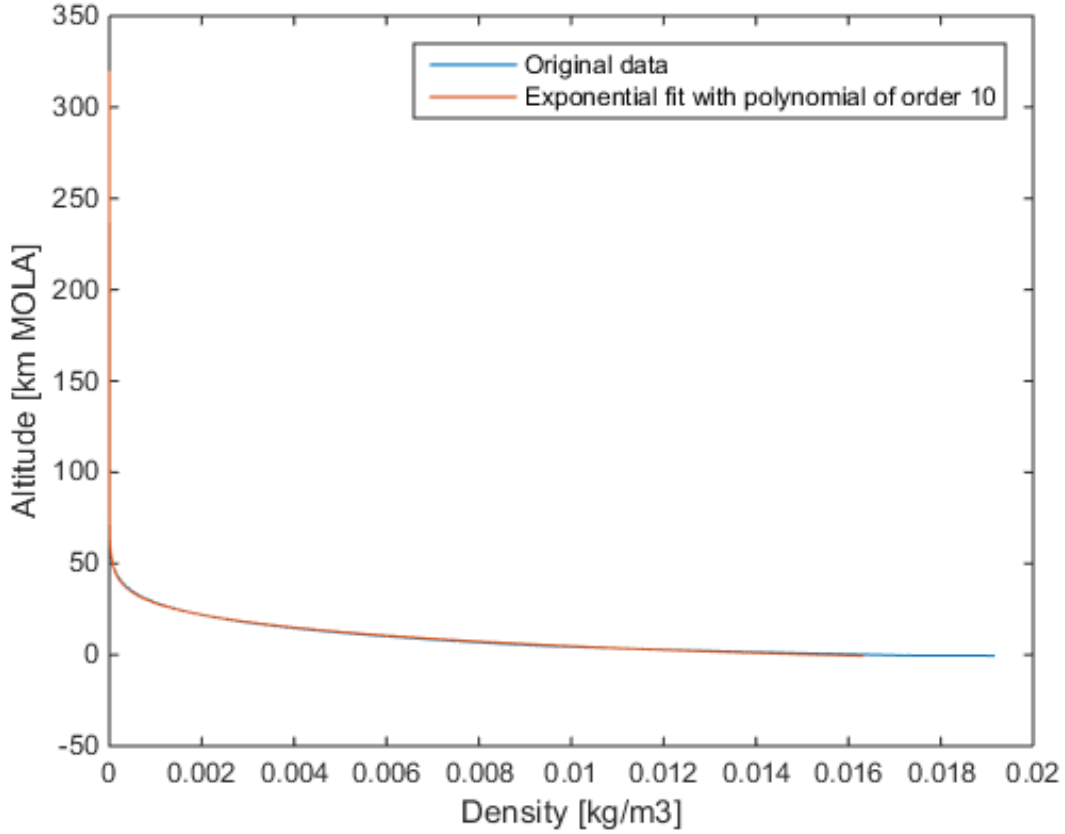

Figure 7.9: Exponential fit for the launch site density data curve

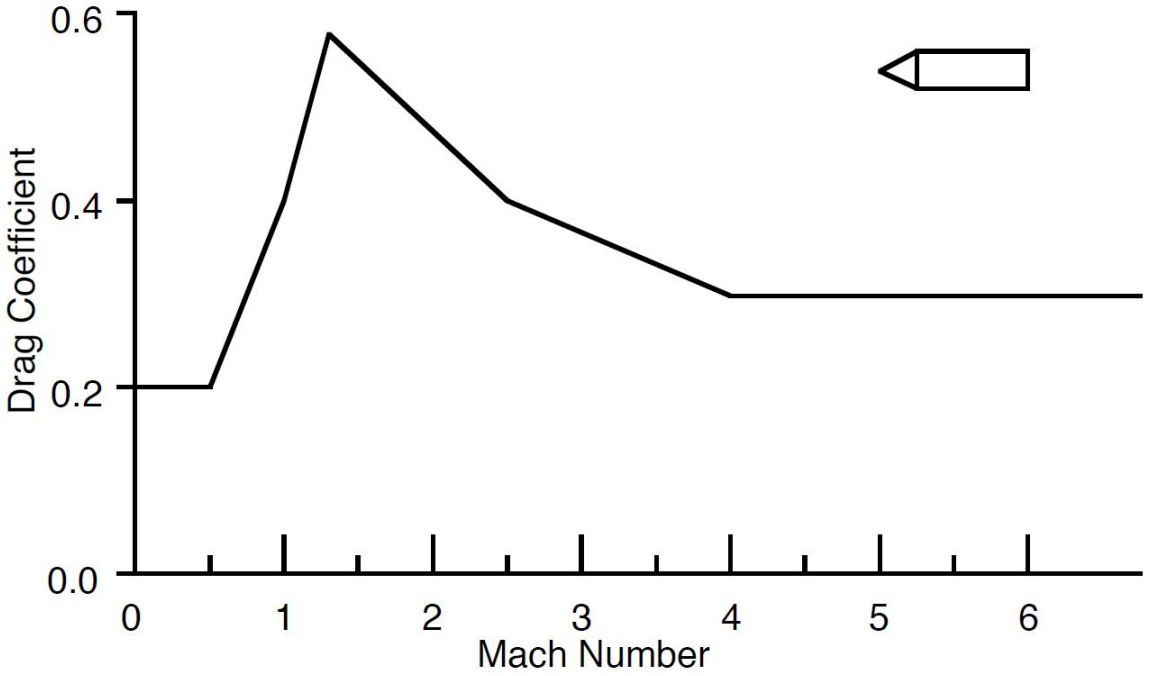

Figure 7.10: Drag coefficient as a function of Mach number Whitehead (2004)

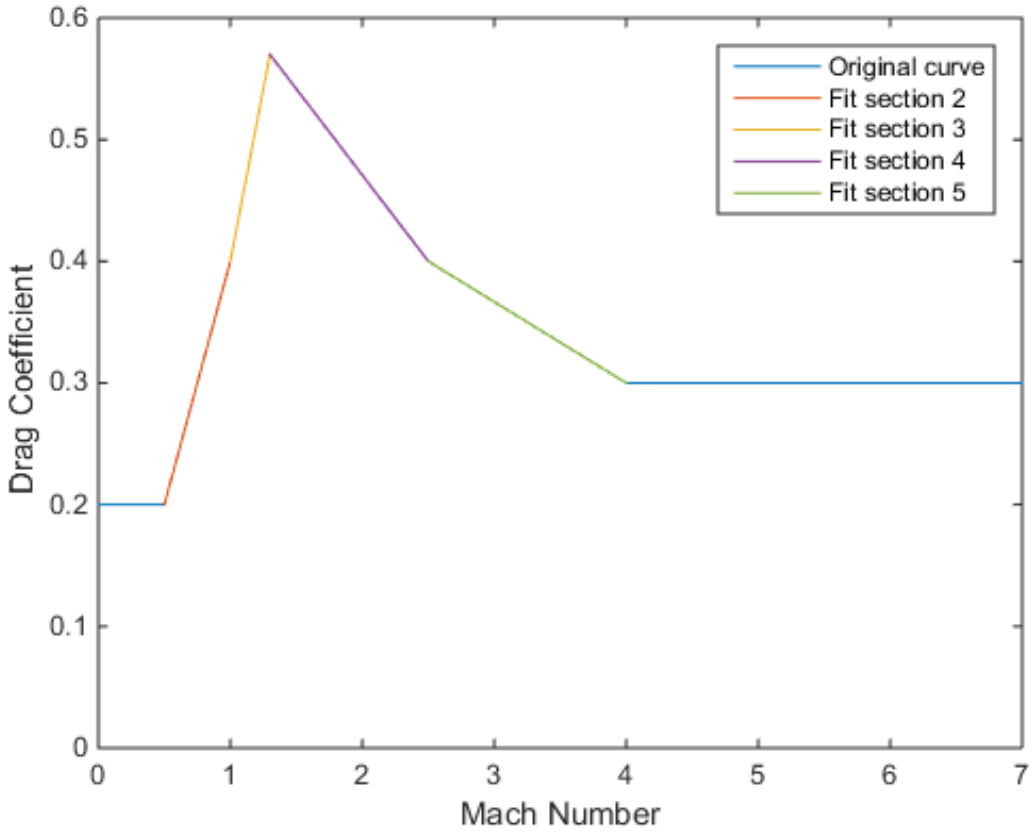

Figure 7.11: All section fits for the drag coefficient - Mach curve

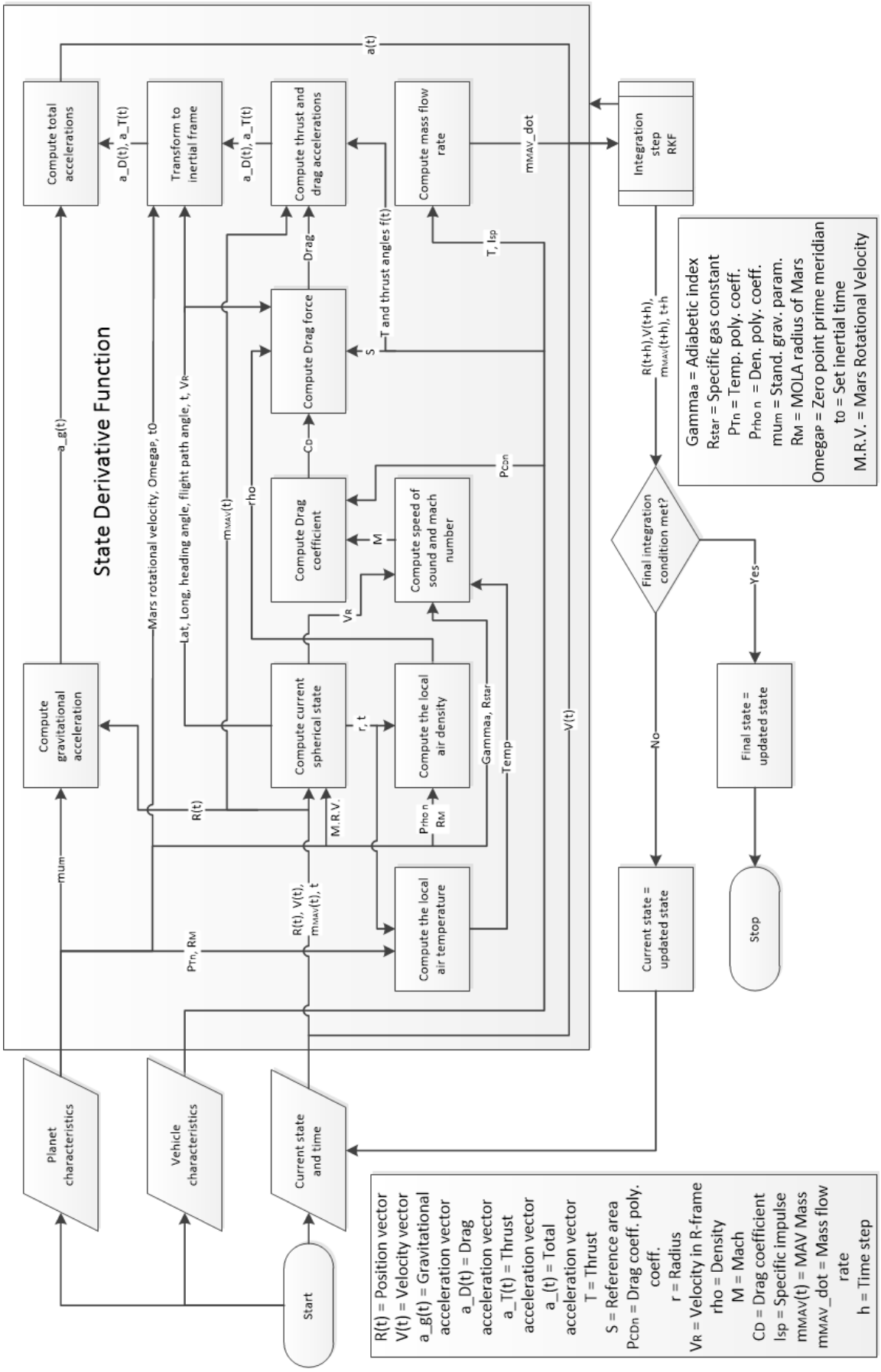

Figure 7.12: RK4 and RKF interface architecture

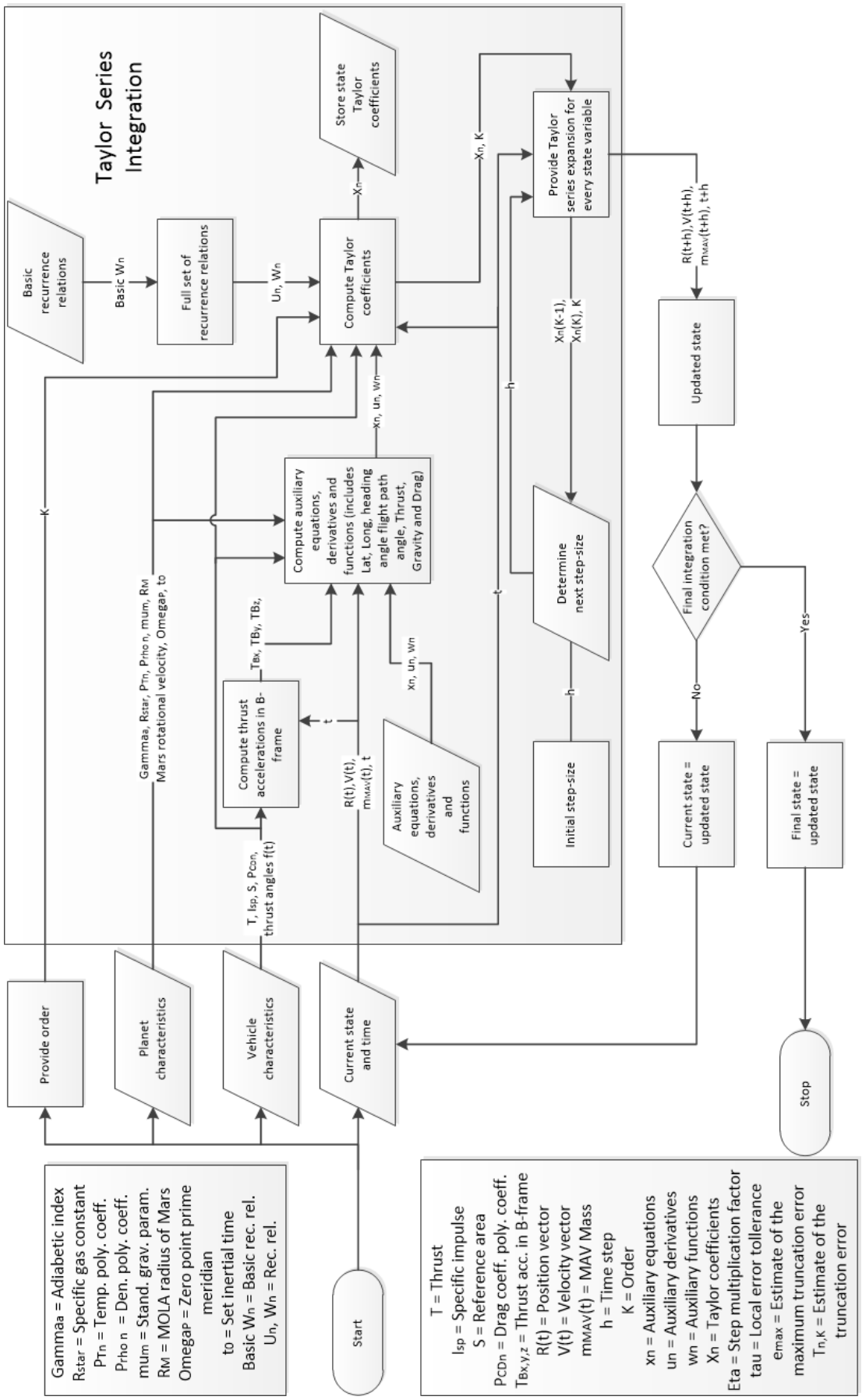

Figure 7.13: TSI architecture

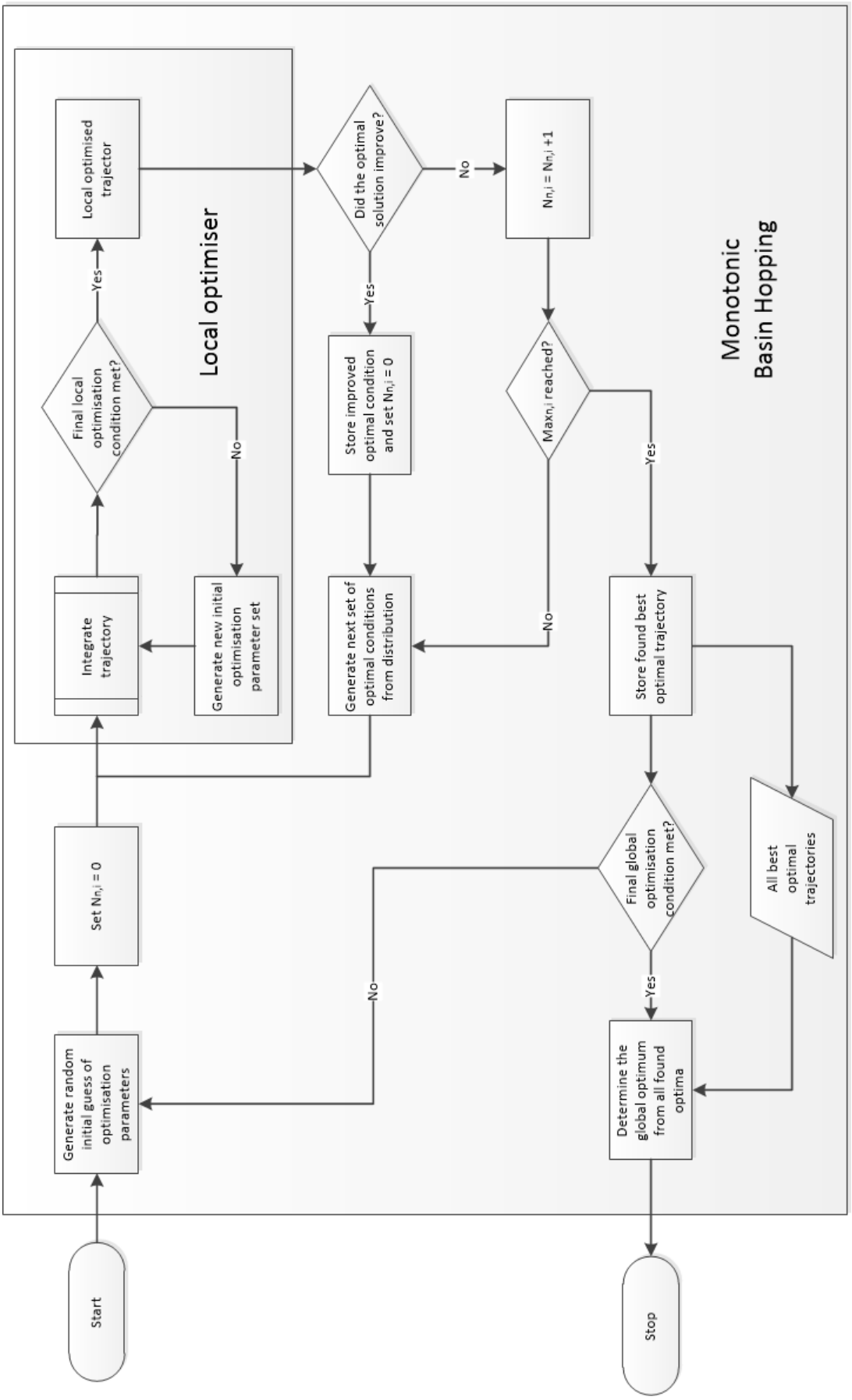

Figure 7.14: Optimiser interface architecture

8

VERIFICATION AND VALIDATION

Verification is the process of determining whether a program meets the requirements or not. Is it working the way it is suppose to? As soon as it works and produces output (verified), these outputs can be compared to other data from which it is known that it is correct. This is called validation. If a program is verified and validated, the outputs should be correct. Fortunately, all the existing software has already been validated, which means that they only still have to be verified to make sure everything is working properly on the simulation computer. Each of the software packages comes with tests which can be used to do this. If all the tests are passed it means that the software is verified for the computer and ready to use. Since Tudat shipped with Eigen and Boost, the Tudat test files also test these libraries. All the test were passed, which means that the Tudat, Eigen and Boost libraries are working properly. However, because the integrators are an intricate part of this thesis, it was decided to perform a separate verification for the Runge-Kutta integrators. This verification is described in Section 8.2. Similarly, PaGMO was verified using its test files.

SNOPT verification still has to be done and added!! As soon as I can get it to work....

Mars-GRAM also came with its own verification test, where three delivered output files had to be replicated. These verification tests were also successful.

8.1. INTERPOLATION

8.2. RK4 AND RKF INTEGRATORS

The tutorial page of the Tudat website (Dirkx *et al.*, 2016) offers two integration tutorials: one involving RK4 and another involving variable step-size Runge-Kutta methods including RKF45. The objective of the tutorial is to get familiar with the different integration methods available in Tudat. At the same time, a small data table has to be reproduced, which also serves as a verification test. For each of the integrators the same problem was addressed: the computation of the velocity of a falling body after a certain amount of time assuming no drag. The results that had to be reproduced are presented in Table 8.1.

Table 8.1: Verification data for the standard integrators

End time [s]	1.0	5.0	15.0	25.0
Velocity [m/s]	-9.81	-49.05	-147.15	-245.25

The first script was written using the instructions from the tutorial and is called numericalintegrators.cpp. This script uses the rungeKutta4Integrator.h header file and the RungeKutta4IntegratorXd function from this header file. This function requires three inputs: the state derivative function (problem specific), en initial time and the initial state. Using the .integrateTo extension the end state at a certain end time can be integrated. This requires the end time and the step-size. For RK4 the step-size is constant. This resulted in the same values as presented in Table 8.1.

The second script was used to test the variable step-size integrators (including RKF45). This script is called rungekuttavariable.cpp and uses the rungeKuttaCoefficients.h and rungeKuttaVariableStepSizeIntegrator.h

header files. In this case the RungeKuttaVariableStepSizeIntegratorXd function was used which requires the Runge-Kutta coefficients (from the respective header file), the state derivative function, the initial time, initial state, zero minimum step-size, infinite maximum step-size, relative tolerance and the absolute tolerance as inputs. In this case the integration can be done in individual steps using .performIntegrationStep with the current step-size as the only input. The current step-size is computed in the integration method itself, but in this case it is checked to make sure that the step-size does not take the function beyond the specified end time. The integration steps are then repeated until the end time is met. Running this script resulted in the same results as presented in Table 8.1 as well.

These results, combined with the fact that the test files for these two methods produced no errors is proof that they are working accordingly. Thus it can be said that the standard integration methods used in this thesis are verified and ready for use in the optimisation tool. However, during the development of the trajectory propagation tools, the entire tool (including the integrators) will be verified again to determine the performance of the integrators.

8.3. TAYLOR SERIES INTEGRATION
8.4. COMPLETE TRAJECTORY PROPAGATION
8.5. OPTIMISER
8.6. COMPLETE OPTIMISATION TOOL

9

RESULTS

10

Analysis

11

CONCLUSIONS AND RECOMMENDATIONS

A

MARS-GRAM 2005 INPUT FILE

```
$INPUT
 LSTFL    = 'LIST.txt'
 OUTFL    = 'OUTPUT.txt'
 TRAJFL   = 'TRAJDATA.txt'
 profile  = 'null'
 WaveFile = 'null'
 DATADIR  = '/home/stachap/MarsGram/binFiles_TUDelft/'
 GCMDIR   = '/home/stachap/MarsGram/binFiles_TUDelft/'
 IERT     = 0
 IUTC     = 1
 MONTH    = 2
 MDAY     = 28
 MYEAR    = 2025
 NPOS     = 32061
 IHR      = 12
 IMIN     = 0
 SEC      = 0.0
 LonEW    = 1
 Dusttau  = 0
 Dustmin  = 0.3
 Dustmax  = 1.0
 Dustnu   = 0.003
 Dustdiam = 5.0
 Dustdens = 3000.
 ALS0     = 0.0
 ALSDUR   = 48.
 INTENS   = 0.0
 RADMAX   = 0.0
 DUSTLAT  = 0.0
 DUSTLON  = 0.0
 MapYear  = 0
 F107     = 68.0
 STDL     = 0.0
 NR1      = 1234
 NVARX    = 1
 NVARY    = 0
 LOGSCALE = 0
 FLAT     = 21
 FLON     = 74.5
 FHGT     = -0.6
 MOLAhgts = 1
 hgtasfcm = 0.
 zoffset  = 0.
 ibougher = 0
 DELHGT   = 0.01
 DELLAT   = 0.0
 DELLON   = 0.0
 DELTIME  = 0.0
 ΔTEX = 0.0
```

```
 profnear = 0.0
 proffar  = 0.0
 rpscale  = 1.0
 rwscale  = 1.0
 wlscale  = 1.0
 wmscale  = 0.0
 blwinfac = 0.0
 NMONTE   = 1
 iup      = 13
 WaveA0   = 1.0
 WaveDate = 0.0
 WaveA1   = 0.0
 Wavephi1 = 0.0
 phi1dot  = 0.0
 WaveA2   = 0.0
 Wavephi2 = 0.0
 phi2dot  = 0.0
 WaveA3   = 0.0
 Wavephi3 = 0.0
 phi3dot  = 0.0
 iuwave   = 0
 Wscale   = 20.
 corlmin  = 0.0
 ipclat   = 1
 requa    = 3396.19
 rpole    = 3376.20
 idaydata = 1
$END

Explanation of variables:
LSTFL    =  List file name (CON for console listing)
OUTFL    =  Output file name
TRAJFL   =  (Optional) Trajectory input file. File contains time (sec)
             relative to start time, height (km), latitude (deg),
             longitude (deg W if LonEW=0, deg E if LonEW=1, see below)
profile  =  (Optional) auxiliary profile input file name
WaveFile =  (Optional) file for time-dependent wave coefficient data.
             See file description under parameter iuwave, below.
DATADIR  =  Directory for COSPAR data and topographic height data
GCMDIR   =  Directory for GCM binary data files
IERT     =  1 for time input as Earth-Receive time (ERT) or 0 Mars-event
             time (MET)
IUTC     =  1 for time input as Coordinated Universal Time (UTC), or 0
             for Terrestrial (Dynamical) Time (TT)
MONTH    =  (Integer) month of year
MDAY     =  (Integer) day of month
MYEAR    =  (Integer) year (4-digit; 1970-2069 can be 2-digit)
NPOS     =  max # positions to evaluate (0 = read data from trajectory
             input file)
IHR      =  Hour of day (ERT or MET, controlled by IERT and UTC or TT,
             controlled by IUTC)
IMIN     =  minute of hour (meaning controlled by IERT and IUTC)
SEC      =  seconds of minute (meaning controlled by IERT and IUTC).
             IHR:IMIN:SEC is time for initial position to be evaluated
LonEW    =  0 for input and output West longitudes positive; 1 for East
             longitudes positive
Dusttau  =  Optical depth of background dust level (no time-developing
             dust storm, just uniformly mixed dust), 0.1 to 3.0, or use
             0 for assumed seasonal variation of background dust
Dustmin  =  Minimum seasonal dust tau if input Dusttau=0 (≥0.1)
Dustmax  =  Maximum seasonal dust tau if input Dusttau=0 (≤1.0)
Dustnu   =  Parameter for vertical distribution of dust density (Haberle
             et al., J. Geophys. Res., 104, 8957, 1999)
Dustdiam =  Dust particle diameter (micrometers, assumed monodisperse)
Dustdens =  Dust particle density (kg/m**3)
ALS0     =  starting Ls value (degrees) for dust storm (0 = none)
ALSDUR   =  duration (in Ls degrees) for dust storm (default = 48)
INTENS   =  dust storm intensity (0.0 - 3.0)
RADMAX   =  max. radius (km) of dust storm (0 or >10000 = global)
DUSTLAT  =  Latitude (degrees) for center of dust storm
DUSTLON  =  Longitude (degrees) (West positive if LonEW=0, or East
```

```
                positive if LonEW = 1) for center of dust storm
MapYear   =  1 or 2 for TES mapping year 1 or 2 GCM input data, or 0 for
                Mars-GRAM 2001 GCM input data sets
F107      =  10.7 cm solar flux (10**-22 W/cm**2 at 1 AU)
NR1       =  starting random number (0 < NR1 < 30000)
NVARX     =  x-code for plotable output (1=hgt above MOLA areoid).
                See file xycodes.txt
NVARY     =  y-code for 3-D plotable output (0 for 2-D plots)
LOGSCALE  =  0=regular SI units, 1=log-base-10 scale, 2=percentage
                deviations from COSPAR model, 3=SI units, with density
                in kg/km**3 (suitable for high altitudes)
FLAT      =  initial latitude (N positive), degrees
FLON      =  initial longitude (West positive if LowEW = 0 or East
                positive if LonEW = 1), degrees
FHGT      =  initial height (km); ≤-10 means evaluate at surface height;
                > 3000 km means planeto-centric radius
MOLAhgts  =  1 for input heights relative to MOLA areoid, otherwise
               input heights are relative to reference ellipsoid
hgtasfcm  =  height above surface (0-4500 m); use if FHGT ≤ -10. km
zoffset   =  constant height offset (km) for MTGCM data or constant
               part of Ls-dependent (Bougher) height offset (0.0 means
               no constant offset).  Positive offset increases density,
               negative offset decreases density.
ibougher  =  0 for no Ls-dependent (Bougher) height offset term; 1
               means add Ls-dependent (Bougher) term, -A*Sin(Ls) (km),
               to constant term (zoffset) [offset amplitude A = 2.5 for
               MapYear=0 or 0.5 for MapYear > 0]; 2 means use global mean
               height offset from data file hgtoffst.dat; 3 means use
               daily average height offset at local position; 4 means
               use height offset at current time and local position.
               Value of zoffset is ignored if ibougher = 2, 3, or 4.
DELHGT    =  height increment (km) between steps
DELLAT    =  Latitude increment (deg) between steps (Northward positive)
DELLON    =  Longitude increment (deg) between steps (Westward positive
                if LonEW = 0, Eastward positive if LonEW = 1)
DELTIME   =  time increment (sec) between steps
ΔTEX =  adjustment for exospheric temperature (K)
profnear  =  Lat-lon radius (degrees) within which weight for auxiliary
                profile is 1.0 (Use profnear = 0.0 for no profile input)
proffar   =  Lat-lon radius (degrees) beyond which weight for auxiliary
                profile is 0.0
rpscale   =  random density perturbation scale factor (0-2)
rwscale   =  random wind perturbation scale factor (≥0)
wlscale   =  scale factor for perturbation wavelengths (0.1-10)
wmscale   =  scale factor for mean winds
blwinfac  =  scale factor for boundary layer slope winds (0 = none)
NMONTE    =  number of Monte Carlo runs
iup       =  0 for no LIST and graphics output, or unit number for output
WaveA0    =  Mean term of longitude-dependent wave multiplier for density
WaveDate  =  Julian date for (primary) peak(s) of wave (0 for no traveling
                component)
WaveA1    =  Amplitude of wave-1 component of longitude-dependent wave
                multiplier for density
Wavephi1  =  Phase of wave-1 component of longitude-dependent wave
                multiplier (longitude, with West positive if LonEW = 0,
                East positive if LonEW = 1)
phi1dot   =  Rate of longitude movement (degrees per day) for wave-1
                component (Westward positive if LonEW = 0, Eastward
                positive if LonEW = 1)
WaveA2    =  Amplitude of wave-2 component of longitude-dependent wave
                multiplier for density
Wavephi2  =  Phase of wave-2 component of longitude-dependent wave
                multiplier (longitude, with West positive if LonEW = 0,
                East positive if LonEW = 1)
phi2dot   =  Rate of longitude movement (degrees per day) for wave-2
                component (Westward positive if LonEW = 0, Eastward
                positive if LonEW = 1)
WaveA3    =  Amplitude of wave-3 component of longitude-dependent wave
                multiplier for density
Wavephi3  =  Phase of wave-3 component of longitude-dependent wave
                multiplier (longitude, with West positive if LonEW = 0,
```

```
192                East positive if LonEW = 1)
193   phi3dot   =  Rate of longitude movement (degrees per day) for wave-3
194                component (Westward positive if LonEW = 0, Eastward
195                positive if LonEW = 1)
196   iuwave    =  Unit number for (Optional) time-dependent wave coefficient
197                data file "WaveFile" (or 0 for none).
198                WaveFile contains time (sec) relative to start time, and
199                wave model coefficients (WaveA0 thru Wavephi3) from the
200                given time to the next time in the data file.
201   Wscale    =  Vertical scale (km) of longitude-dependent wave damping
202                at altitudes below 100 km (10≤Wscale≤10,000 km)
203   corlmin   =  minimum relative step size for perturbation updates
204                (0.0-1.0); 0.0 means always update perturbations, x.x
205                means only update perturbations when corlim > x.x
206   ipclat    =  1 for Planeto-centric latitude and height input,
207                0 for Planeto-graphic latitude and height input
208   requa     =  Equatorial radius (km) for reference ellipsoid
209   rpole     =  Polar radius (km) for reference ellipsoid
210   idaydata  =  1 for daily max/min data output; 0 for none
```

B

ATMOSPHERIC DATA FITTING

In Section 7.2.1 the final results of the atmospheric data fits were presented. However, before those results could be obtained, a few iterations of trial-and-error fits were required. These intermediate trials are presented for both the temperature data, in Appendix B.1, as well as for the density data, in Appendix B.2. The fit requirements for the polynomial function were set such that the errors created by the fit were less than the uncertainty of the latitude and longitude dependent data. This way the desired accuracy could still be reached. This uncertainty is caused by the difference in atmospheric data curves for the different latitudes and longitudes. The atmospheric data curve for the latitude and longitude corresponding to the launch site (21.0 °N and 74.5 °E) was taken as the reference curve and called the launch site curve. The differences between this launch site curve and the 8 other curves were used to define the maximum data curves difference. The *polyfit.m* function within Matlab was used to fit a polynomial to the data. This function also directly provides a standard deviation of the fit for each data point (in combination with the *polyval.m* function). One of the requirements was for the maximum standard deviation of the polynomial fit to be one order less than the maximum difference of the data curves with respect to the launch site curve. The second requirements for a proper fit was for the absolute maximum difference between the polynomial fit and the launch site curve to be less than the absolute maximum difference between the data curves and that same launch site curve.

B.1. TEMPERATURE

Initially, the temperature data curve was split into 6 different sections as portrayed in Figure B.1. The sections were selected on the basis of their individual shapes and the maximum order that was required, where the maximum order for temperature was set at 8. Fewer sections were however always preferred.

Unfortunately, the fourth quarter of section 3 caused this section to not meet the maximum standard deviation requirement because in that quarter the linear behaviour changes directions slightly and the differences between the different data curves become a lot smaller. Therefore section 3 was split into two different sections resulting in a total of 7 sections as shown in Figure B.2.

These sections all met the requirements and thus provided a proper fit to the data. This fit is shown in Figure B.3.

However, because fewer sections were preferred, an attempt was made to reduce the number of sections. Because of the inaccuracy in the Mars-GRAM model close to the planet surface, it was decided to delete the outliers near the surface (section 1) and stretch the outcome of the second polynomial fit to the surface instead. Also, it turned out that section 4 and 5 could be combined in such a way that the requirements were still met. This reduced the number of section from 7 to 5. The final fit was presented in Section 7.2.1 as well as the corresponding errors and polynomial coefficients.

B.2. DENSITY

For the density curve, because the data represents a logarithmic curve, initially an exponential atmosphere was fit. However, this did not meet any of the requirements for a proper fit as can be seen in . This is why a polynomial fit was attempted for the density curve as well with the same requirement as the temperature fits. The first fit was attempted with two sections as presented in Figure B.4.

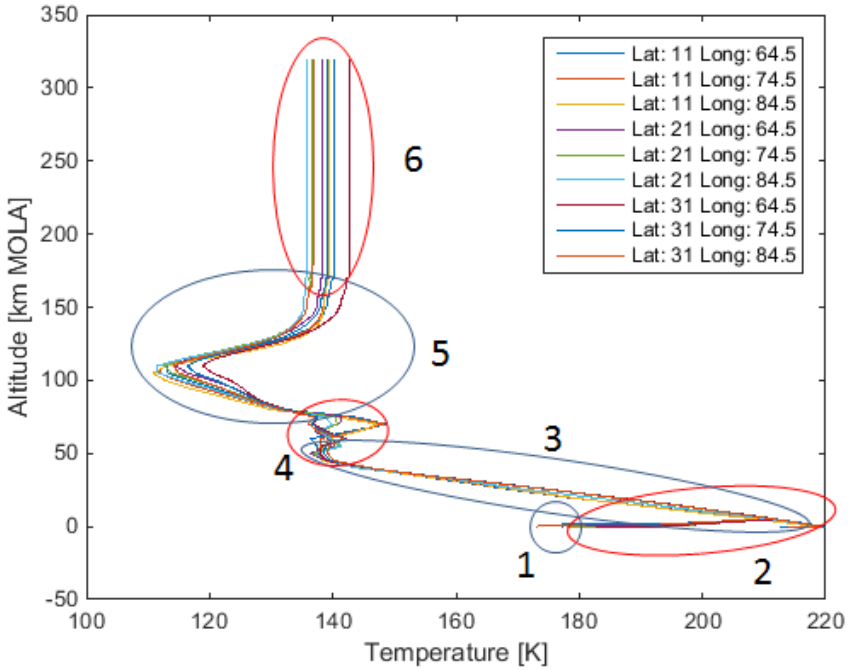

Figure B.1: Six different temperature curve sections

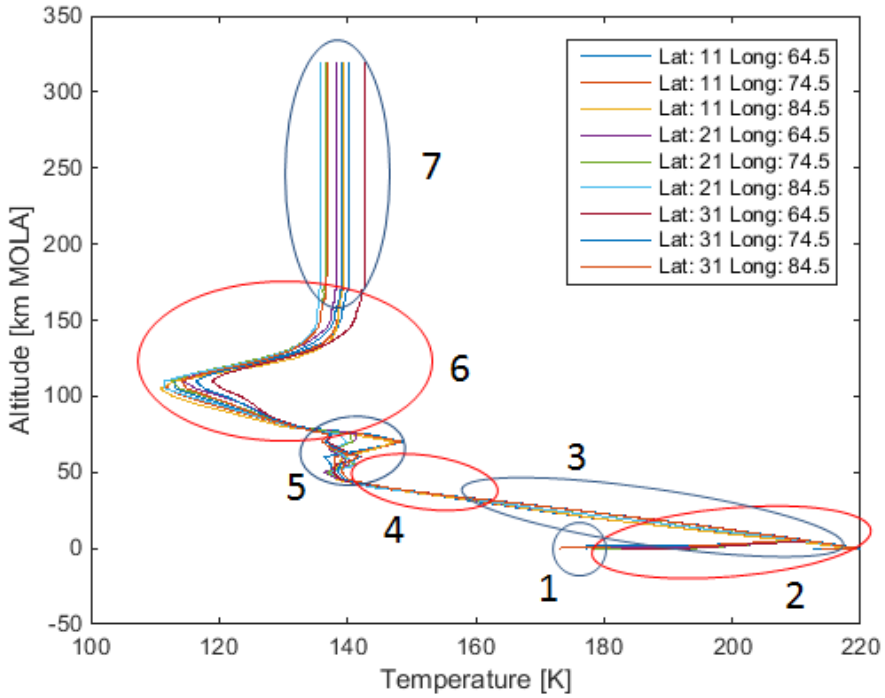

Figure B.2: Seven different temperature curve sections

However, the second section could not meet the requirements because of the initial part. This is why it was split into two sections as presented in Figure B.5.

This did meet the error requirements and resulted in the fit as shown in Figure B.6 with section 1 being

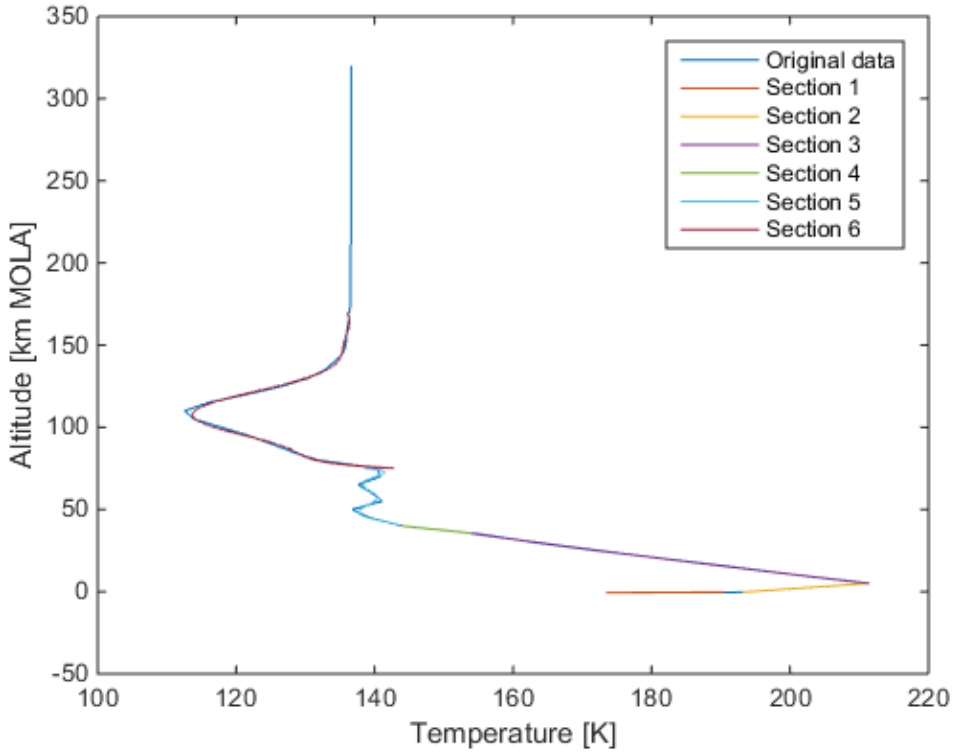

Figure B.3: All section fits for the launch site temperature data curve

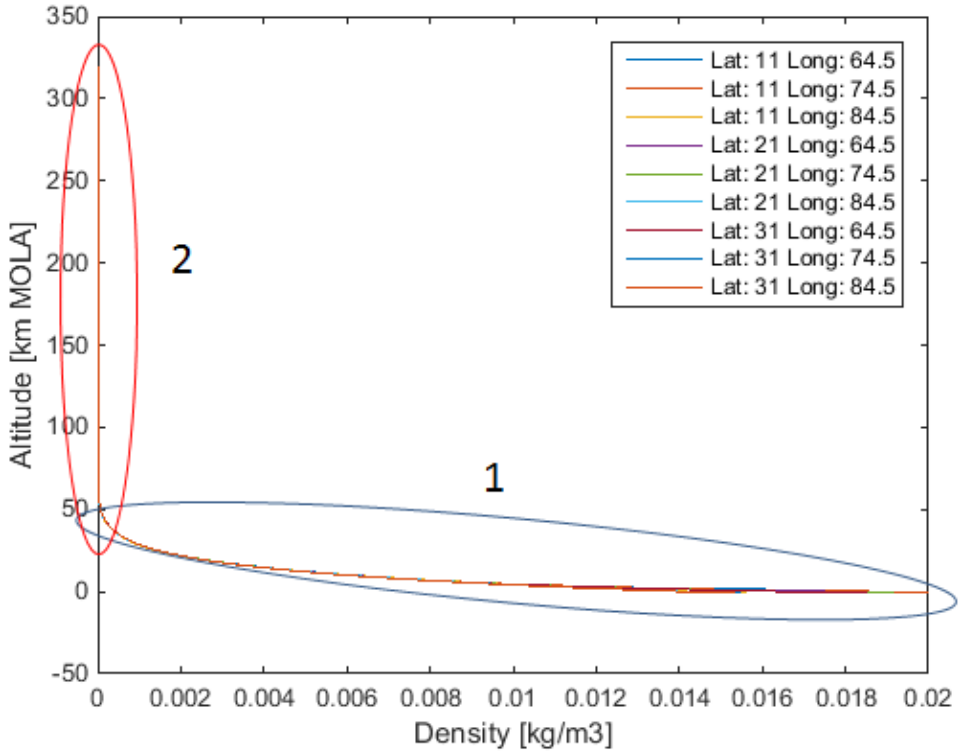

Figure B.4: Two different density curve sections

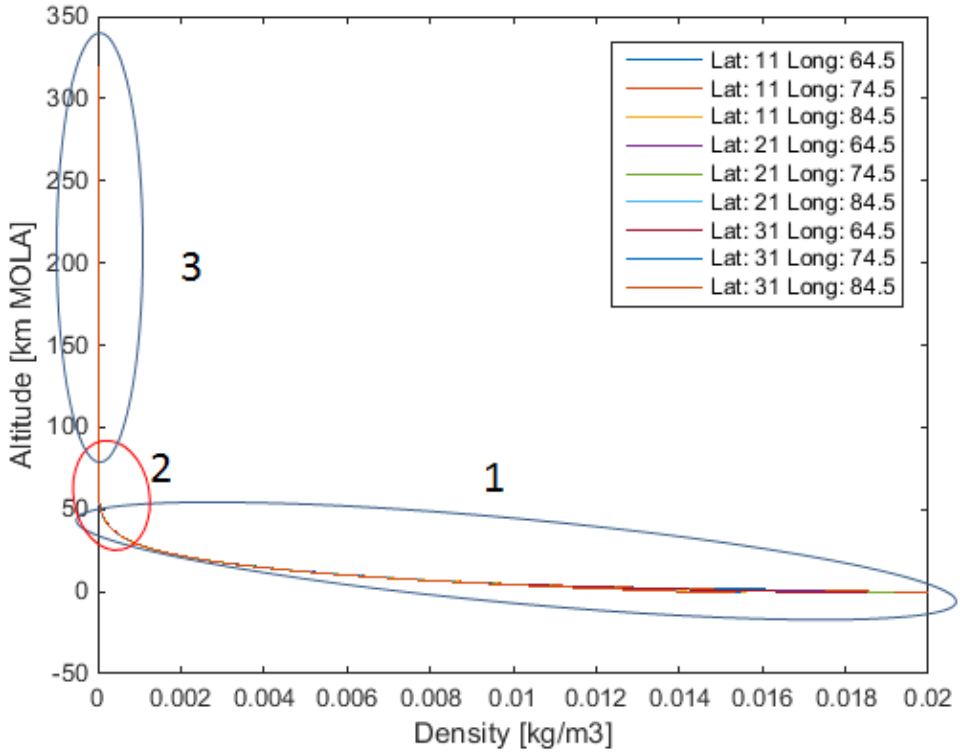

Figure B.5: Three different density curve sections

the lower part, section 2 the middle and section 3 the upper part.

However, in this case the fit for section three resulted in an oscillating behaviour around the actual data which caused the density values to drop below zero as shown in Figure B.7. Since this is unrealistic, an exponential atmospheric fit for only the third section was attempted (also shown in Figure B.7) however, this resulted in the same behaviour as the attempt for the entire curve. Therefore, it was decided to try a different exponential approach which eventually resulted in the fit described in Section 7.2.1.

Figure B.8 clearly shows that both the full exponential atmospheric fit and the polynomial fits were not a proper choice here. The figure shows the curved part of the curve.

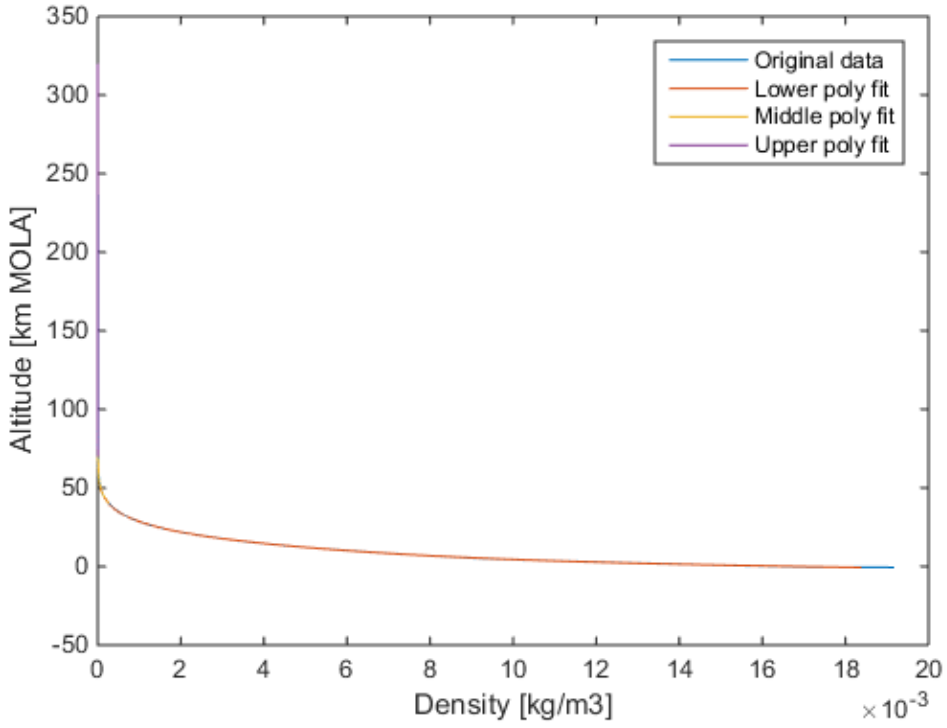

Figure B.6: All section fits for the launch site density data curve

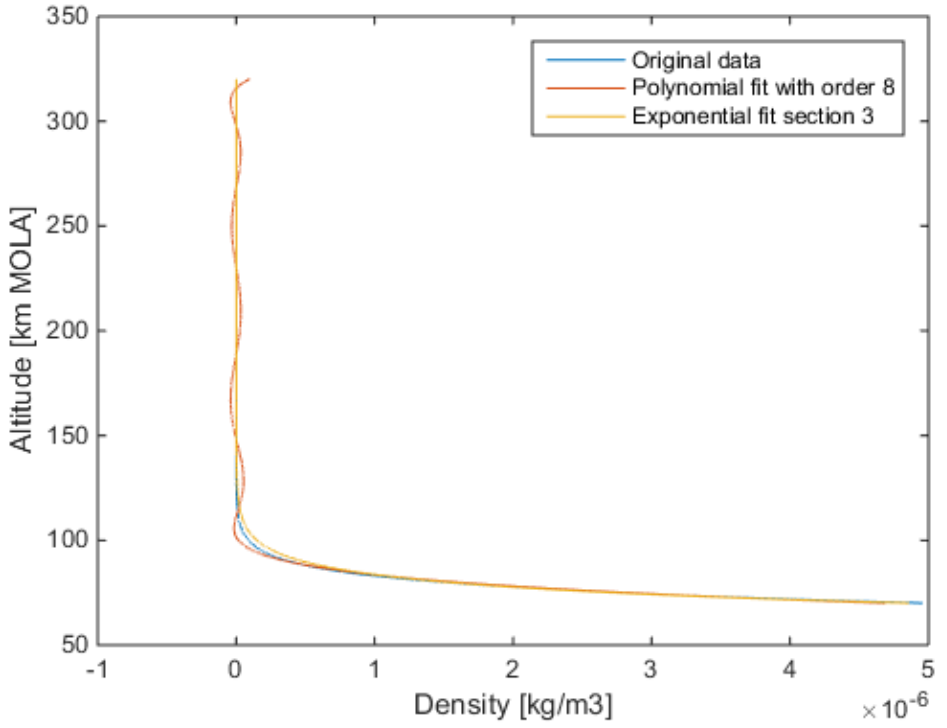

Figure B.7: Section 3 fit for the launch site density data curve

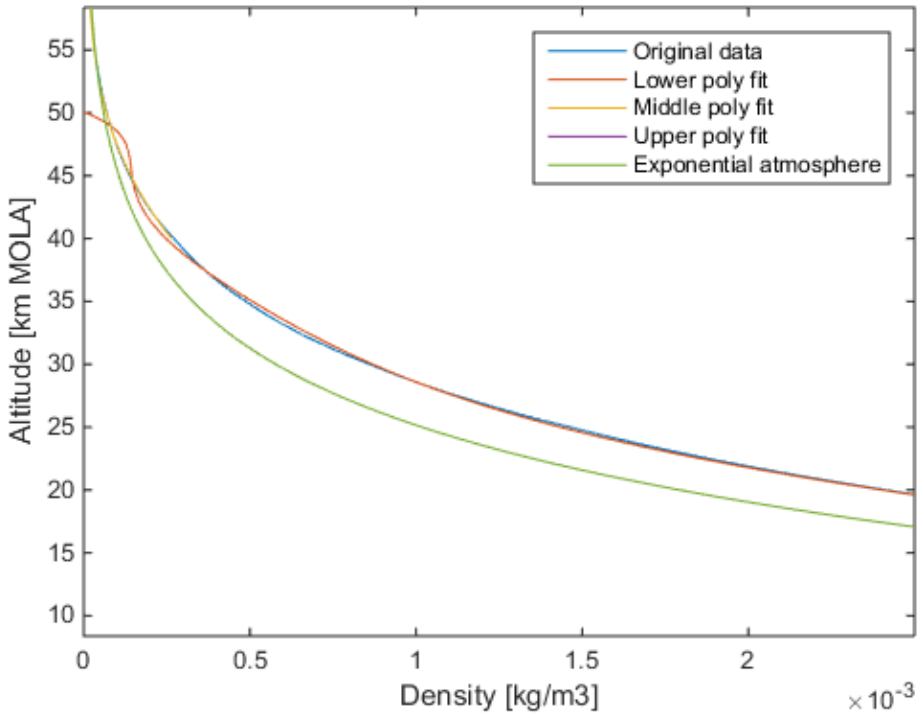

Figure B.8: Zoom in of the end of section 1 with the polynomial and exponential atmospheric fits

C

PROGRAM FILE DEFINITIONS

Both the RKF and TSI propagator architectures were presented in Chapter 7(Figures 7.12 and 7.13 respectively). These included many different operation blocks. Each block is either a separate class, a combination of a header and source file (without it being a class), a function of either of these files or a function in the main file (MAVPropagator). The different files that include several blocks are MAVPropagator.h/.cpp, TaylorSeriesIntegration.h/.cpp, ascentDragForce.h/.cpp and ascentStateDerivativeFunction.h/.cpp, where this last file is also a class. The functions that will be implemented in these files are mentioned in Table C.1 all other blocks are described in Table C.2.

Table C.1: Large program files and included functions.

MAVPropagator	**TaylorSeriesIntegration**	**ascentStateDerivativeFunction**
Update current state	Compute auxiliaries	Compute current spherical state
Integration step	Thrust acceleration in B-frame	Compute gravitational acceleration
	Compute Taylor coefficients	Compute thrust and drag acceleration
ascentDragForce	Store state Taylor coefficients	Transfer to inertial frame
Compute speed of sound and Mach number	Initial step-size	Compute total acceleration
	Provide Taylor series expansion	Compute mass flow rate
	Estimate the max. trunc. error	
Compute drag force	Taylor series expansion incl. trunc. error	

Table C.2: Separate function files

Block	**Kind of file**
Planet characteristics	Class
Vehicle characteristics	Class
Auxiliary equations, derivatives and functions	Class
Current state and time	Class
Basic recurrence relations	Header and source
Full set of recurrence relations	Header and source
Determine next step-size	Header and source
Compute local air temperature	Header and source
Compute local air density	Header and source
Compute drag coefficient	Header and source

BIBLIOGRAPHY

J. R. Scott and M. C. Martini, *High speed solution of spacecraft trajectory problems using Taylor series integration,* in *AIAA/AAS Astrodynamics Specialist Conference and Exhibit, Honolulu, Hawaii* (2008).

R. Noomen, *ae4-878.basics.v4-14,* Lecture slides, Delft University of Technology (2013), [Internal publication] Course: Mission Geometry and Orbit Design.

E. Mooij, *The motion of a vehicle in a planetary atmosphere* (Delft University Press, Delft, 1994).

A. Jorba and M. Zou, *A Software Package for the Numerical Integration of ODEs by Means of High-Order Taylor Methods,* in *Experimental Mathematics*, Vol. 14 (Taylor & Francis, 2005) pp. 99–117.

D. Dirkx, K. Kumar, E. Doornbos, E. Mooij, and R. Noomen, *TUDAT,* (2016), [online database], URL: tudat.tudelft.nl [cited 8 March 2016] (Only available from the TU Delft network).

A. Qing, *Differential Evolution: Fundamentals and Applications in Electrical Engineering* (John Wiley & Sons, Singapore, 2009).

D. Izzo, *PyGMO and PyKEP: Open source tools for massively parallel optimization in astrodynamics (the case of interplanetary trajectory optimization),* in *5th International Conference on Astrodynamics Tools and Techniques (ICATT)* (2012).

P. E. Gill, W. Murray, and M. A. Saunders, *SNOPT: An SQP algorithm for large-scale constrained optimization,* in *SIAM journal on optimization*, Vol. 12 (2002) pp. 979–1006.

P. E. Gill, W. Murray, and M. A. Saunders, *Users guide for SNOPT version 7 - Software for large-scale nonlinear programming,* (2008), [online database], URL: http://web.stanford.edu/group/SOL/guides/ [cited 23 October 2015].

H. L. Justh and C. G. Justus, *Utilizing Mars global reference atmospheric model (Mars-GRAM 2005) to evaluate entry probe mission sites,* Presentation (2008), [online database], URL: https://smartech.gatech.edu/bitstream/handle/1853/26375/34-186-1-PB.pdf?sequence=1 [cited 10 December 2015].

J. C. Whitehead, *Mars Ascent Propulsion Trades with Trajectory Analysis,* in *40th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit* (American Institude of Aeronautics and Astronautics, 2004).