An Approach to Generalizing Taylor Series Integration for Low-Thrust Trajectories

Master's Thesis

Michael M. H. J. Van den Broeck

Delft University of Technology

TU Delft Delft University of Technology

Challenge the future

An Approach to Generalizing Taylor Series Integration for Low-Thrust Trajectories

Master's Thesis

by

Michael M. H. J. Van den Broeck

in partial fulfillment of the requirements for the degree of

Master of Science
in Aerospace Engineering

at the Delft University of Technology
to be defended publicly on Thursday July 20, 2017 at 13:00.

Cover image credit: 'Diophantine Flow' (2010) by Edward Belbruno who introduced new ideas in the field of low-energy transfer orbits and worked as an orbital analyst at the Jet Propulsion Laboratory on missions as Galileo, Magellan, Cassini and others.

July 6, 2017

Student number:	4152646	
Project duration:	March 28, 2016 – July 20, 2017	
Supervisor:	Kevin Cowan, MSc MBA	
Thesis committee:	Prof. dr. ir. Pieter Visser,	TU Delft, Dpt. of Astrodynamics and Space Missions
	Ir. Prem Sundaramoorthy,	TU Delft, Dpt. of Space Systems Engineering
	Dr. ir. Dominic Dirkx,	TU Delft, Dpt. of Astrodynamics and Space Missions
	Kevin Cowan, MSc MBA,	TU Delft, Dpt. of Astrodynamics and Space Missions

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft Delft University of Technology

ACKNOWLEDGMENTS

This thesis challenged me both personally and academically. I would not have been able to overcome these challenges if it wasn't from the support of many people.

First of all, I would like to thank my supervisor Kevin Cowan, MSc MBA for his academic and personal guidance and for making me feel comfortable from the first meeting onward, and Ir. Ron Noomen for the initial supervision during my the literature study. I would also like to express my gratitude to the following academics for taking the time to read this thesis and for being willing to serve on my graduation committee: Prof. dr. ir. Pieter Visser, Ir. Prem Sundaramoorthy and Dr. ir. Dominic Dirkx, which I like to thank in particular for his availability for questions and for his Tudat support. I'm also thankful to the following academics for making time to answer my questions: Ir. Jacco Geul, Dr. ir. Erwin Mooij, Dr. ir. Vivek Vittaldev, Dr. Richard Dwight and Dr. Steven Hulshoff.

Furthermore, I would like to thank the following people for enriching my life as a student:

My friends Pieter-Jan Deldycke, Sebastien Callens and Delphine De Tavernier for all the happy moments we shared and for their support during lesser moments. My fellow students René Hoogendoorn, Randy van Dooren, Alejandro González Puerta and Sander Vancraen for stimulating my scientific creativity and sharing my passion for spaceflight. My fellow Boulderites, Jessy Barreto, Bart Doekemeijer and Kay Fellows for making my internship an unforgettable experience, and Dr. Jay McMahon for his guidance at CCAR. The master students of the (removed) master room at the 9th floor, for the pleasant lunch breaks, the mutual moral support and the laughs at the coffee corner. My roommates for laughing at my silly jokes. And my Belgian friends Stefan Verbelen and Andries Saerens for the amusing bicycle rides which I needed to take my mind of things.

Last but not least, I would like thank my family:
My parents Ingrid and Walter for all the chances they gave me, for encouraging my curiosity and for their support during my entire educational career. And Naomi, for being such a wonderful sister!

Michael Van den Broeck
July 6, 2017

ABSTRACT

The determination of optimal low-thrust trajectories of spacecraft certainly is an important, yet challenging aspect of modern space missions. It is important because low-thrust propulsion is a very efficient way of propelling a spacecraft in space. The determination of the best trajectory is an optimization problem. For low-thrust trajectories this optimization is particularly challenging because of the large number of variables to be optimized. Indeed, low-thrust propulsion implies that the magnitude and direction of the thrust force have to be optimized for a large number of points on the trajectory.

For complex problems such as low-thrust trajectory optimization, evolutionary algorithms can be used as they only require a fitness or objective function, and are independent of the equations of motion. In evolutionary optimization many possible trajectories are computed. The value of the objective function is then calculated for each trajectory. These values are used to improve the population of the trajectories, generation after generation. Low-thrust trajectories are computed using numerical propagators, as their equations of motion cannot be integrated analytically.

Fast low-thrust trajectory optimization is important for several reasons. First, it allows to optimize the trajectory in real-time, allowing mission control to cope with deviations from the planned trajectory more easily. A second reason is that faster optimization makes on-board trajectory optimization on slower processors possible. On-board trajectory optimization increases the autonomy of spacecraft and takes away the problem of delay when sending commands from Earth to the spacecraft, especially for interplanetary missions.

As during the optimization process, the trajectories are computed a large number of times, the speed of the evolutionary optimization depends heavily on the speed of the numerical propagator. In turn, the speed of numerical propagation is mainly determined by the type of numerical integration and the used coordinate system. Regarding numerical integration, the literature study preceding this thesis concluded that Taylor Series Integration (TSI) is an accurate and computationally fast method. As for the coordinate system, expressing the spacecraft's state in Unified State Model (USM) elements allows large integration step-sizes due to the slow varying elements. These two methods can be combined to form the TSI-USM propagator. In his thesis, Dr. V. Vittaldev has shown that TSI-USM allows faster propagation of interplanetary low-thrust trajectories than an RK8(7)-based propagator [Vittaldev, 2010]. Unfortunately, the TSI-USM method is problem-specific as the use of a different thrust function would require a new derivation of the relations to compute the terms in the Taylor Series. The fact that TSI-USM is a problem-specific method is detrimental for its use in evolutionary algorithms. Indeed, when using evolutionary algorithms, the problem changes constantly due to variations in the low-thrust profile of the trajectories. Therefore, each low-thrust profile would require a specific derivation of the TSI relations.

In this thesis, a method was developed to generalize TSI-USM. It was decided to express the thrust profile in the form of a table, instead of opting for a fixed specific function. The table contains the values of the thrust in three dimensions for a discrete set of time points. The denser the table, the higher the resolution of the thrust profile. The developed method, referred to as the General Discrete Force Model (GDFM) uses the coefficients of the interpolation of the thrust profile to compute the recurrence relations of the derivatives in the Taylor Series terms. From a comparison of several interpolation methods, cubic spline interpolation was selected as it was found that it ensures an accurate representation of the thrust profile as well as a relatively simple computation of the TSI recurrence relations. The GDFM was implemented into the existing TSI-USM to form a generalized TSI-USM (GTSI-USM). The GTSI-USM was programmed in C++ to support future projects via the TU Delft Astrodynamics Toolbox as well as to increase its computational efficiency.

The accuracy and the computational efficiency of the GTSI-USM was tested for a range of different thrust profiles and compared to the performance of RK8(7)13M-based propagators. The results show that, although the GDFM indeed generalizes the TSI-USM, it is less accurate and requires more CPU time for a given accuracy than existing RK8(7)-based propagators. For constant tangential thrust, given a respectable accuracy of 2

km in position for interplanetary trajectories, the GTSI-USM is at least three orders of magnitude slower than a RK8(7)-based propagator with Cartesian coordinates (RK-Cart). Also for this case, the maximum achievable accuracy of the RK-Cart is at least one order of magnitude better than the GTSI-USM. Since a specific TSI-USM has been shown to be superior to RK8(7)-based propagators [Vittaldev, 2010], it seems that the improvements in generality of the TSI come with a increase in CPU time. Only in case of zero thrust, the GTSI-USM was capable of reaching a higher maximum accuracy than RK8(7)-based propagators. It was also found that the optimum order of the TSI depends on the type of orbit that is propagated. This is disadvantageous for the use of the GTSI-USM in evolutionary algorithms. Indeed, the CPU time required to optimize the order of the TSI has to be accounted for in the overall CPU time, hence making the method slower with respect to RK methods for which the order can not and should not be optimized.

The conclusions of this thesis are important for future research on generalizing the Taylor Series Integration. Using a general discrete force model is not preferred as the results have shown that there is a major inaccuracy in the method. Only for the zero thrust case and when used with the right order, the generalized TSI-USM can deliver a higher accuracy then RK8(7)-based propagators. Nevertheless, it can be expected that future attempts to generalize the TSI-USM propagator, without sacrificing its accuracy and computational speed, will remain challenging.

CONTENTS

List of Figures

List of Tables

List of Symbols

Notation

Symbol	Description
6.67408e-11	Equivalent to $6.67408 \cdot 10^{-11}$
$\boldsymbol{x}$	Vector
$\boldsymbol{x}^T$	Transpose of vector $\boldsymbol{x}$
$\hat{\boldsymbol{x}}$	Unit vector
$\dot{x}$	First derivative of x with respect to time
x'	First derivative of x with respect to time
$\ddot{x}$	Second derivative of x with respect to time
x''	Second derivative of x with respect to time
$\dddot{x}$	Third derivative of x with respect to time
x'''	Third derivative of x with respect to time
$x^{(k)}$	k^{th} derivative of x with respect to time
$x^{[k]}$	k^{th} normalized derivative of x with respect to time = $\frac{x^{(k)}}{k!}$
$X(k)$	Simplified notation of $x^{[k]}$

Roman letters

[-] means that the symbol is dimensionless. If the unit is not specified at all, the unit may vary depending on the use-case.

Symbol	Unit	Description
A	[-]	Coefficient of interpolating polynomial
a	m	Semi-major axis
B	[-]	Coefficient of interpolating polynomial
C	m/s	First velocity hodograph component of USM
C	[-]	Coefficient of interpolating polynomial
D	[-]	Coefficient of interpolating polynomial
$\boldsymbol{d}$		State derivative vector function
E	[-]	Eccentric anomaly
e	[-]	Eccentricity
F	N	Thrust force
$\mathcal{F}$		Reference frame
f	m/s^2	Thrust acceleration
G	$\text{m}^3/(\text{kg}\cdot\text{s}^2)$	Universal gravitational constant (6.67408e-11 $\text{m}^3/(\text{kg}\cdot\text{s}^2)$)
g_0	m/s^2	Standard gravity (9.80665 m/s^2)
h	s	Integration step-size
I_{sp}	s	Specific impulse
i	[-]	Inclination
J	[-]	Objective function
K	[-]	Order of Taylor series integration
k	[-]	Index
l	[-]	Variable used for simplification in the USM state derivative model
M	rad	Mean anomaly
m	kg	Mass of spacecraft
n	rad/s	Mean motion
O	[-]	Order of method
p	m	Semi-latus rectum
R_{f1}	m/s	Second velocity hodograph component of USM

R_{f2}	m/s	Third velocity hodograph component of USM
r	m	Radius or norm of spacecraft's position vector
s		Interpolating polynomial of thrust profile
T	[-]	Transformation matrix
T	s	Orbital period
t	s	Time
$U(k)_n$		k^{th} normalized derivative of n^{th} state variable
u	rad	Argument of latitude
u		Derivative of state variable x
V		Normalized derivative of auxiliary variable v in TSI recurrence relations
v	m/s	Velocity
v_{e1}	m/s	Additional USM state variable
v_{e2}	m/s	Additional USM state variable
W		Normalized derivative of auxiliary variable w in TSI recurrence relations
w		Auxiliary variable in TSI recurrence relations
$\boldsymbol{x}$		State vector of spacecraft
x	m	x-coordinate
x	[-]	x-axis of the velocity frame
x		General state variable
y	m	y-coordinate
y	[-]	y-axis of the velocity frame
Z	[-]	Rotation matrix
z	m	z-coordinate
z		z-axis of the velocity frame

Subscript	**Description**
0	Initial
A	Asteroid
abs	Absolute
curr	At current integration step
E	Earth
$e1$	x-direction of RTN frame
$e2$	y-direction of RTN frame
$e3$	z-direction of RTN frame
f	Final
J2000	J2000 orientation
i	Initial
i	Index for nodes in thrust profile
n	Index for state variable
O	Described with respect to the inertial frame
p	Pericenter
RMS	Root-Mean-Square
RTN	RTN frame
rel	Relative
S	Sun
VF	Velocity frame
virt	Virtual node
x	x-direction of VF frame
y	y-direction of VF frame
z	z-direction of VF frame

GREEK LETTERS

[-] means that the symbol is dimensionless.

Symbol	Unit	Description
α	rad	Thrust angle
γ	rad	Flight path angle
ε	[-]	Error
ϵ_{O1}	[-]	Fourth element of USM
ϵ_{O2}	[-]	Fifth element of USM
ϵ_{O3}	[-]	Sixth element of USM
η_O	[-]	Seventh element of USM
θ	rad	True anomaly
λ	rad	Sum of longitude of ascending node and argument of latitude
μ	m^3/s^2	Gravitational parameter
σ	[-]	Penalty coefficient
τ	s	Stay-time
τ	s	Alternative time scale
Ω	rad	Right ascension (or longitude) of ascending node
ω	rad	Argument of pericenter
ω_1	1/s	Additional USM variable
ω_3	1/s	Additional USM state variable

List of Abbreviations

Notation	Description
ASM	Astrodynamics and Space Missions. 4,
AU	Astronomical Unit (149 597 871 km). 67, 78, 104,
CNES	Centre Nationale d'Études Spatiales. 2,
CSI	cubic spline interpolation. xv, 50, 54, 55,
GDFM	General Discrete Force Model. 60, 103, 107, 108
GTOC	Global Trajectory Optimization Competition. xxi, 1, 2,
GTOC3	third edition of the Global Trajectory Optimization Competition (GTOC). 1, 2, 10, 14, 15,
GTSI-USM	generalized TSI-USM propagator for low-thrust trajectories. 59, 103, 107,
L2	second order Lagrange interpolation. xv, 54, 55,
L3	third order Lagrange interpolation. xv, 54, 55,
MEE	Modified Equinoctial Element. 5,
MMS	Method of Manufactured Solutions. 71, 97,
MRP	Modified Rodrigues Parameter. 6,
NEA	Near-Earth Asteroid. 2,
ODE	Ordinary Differential Equation. 9,
RAAN	right ascension of ascending node. 115, 116
RK	Runge-Kutta. 103
RK-Cart	Runge-Kutta 8(7)-13M propagator combined with Cartesian coordinates. 99, 103,
RK-Cart-TPS	version of RK-Cart propagator programmed completely within the Tudat propagation setup. xi, xii, 65, 66, 104, 107,
RK-USM	Runge-Kutta 8(7)-13M propagator combined with USM elements. 91, 99, 103, 107,
RK4	fourth order Runge-Kutta. 7,
RMS	Root Mean Square. 78, 91,
RTN	Radial Tangential Normal. 37, 55, 57, 60,
SMRP	Shadow Modified Rodrigues Parameter. 6,
TAI	International Atomic Time. 15,
TS	Taylor series. 40,
TSI	Taylor Series Integration. 9, 10, 27, 103, 121
TSI-USM	TSI propagator combined with USM elements. 103, 121
TT	Terrestial Time. 15,
TUDAT	TU Delft Astrodynamics Toolbox. 60, 67, 103, 104, 107, 109
USM	Unified State Model. 6, 17, 21, 60, 103, 121
VF	velocity frame. 16, 60,

1

INTRODUCTION

This chapter explains the process of establishing the research question for this thesis. This process has been performed as part of the literature study preceding this thesis. Therefore, this chapter also summarizes the most important parts of this literature study. The three chapter following this chapter will provide the required theory in order to arrive at the core of this thesis work, which will be described in Chapter 5 and of which the results are presented in Chapter 7. Each chapter will begin with an overview of its sections.

Contents

1.1. INTRODUCTION

This chapter explains the establishment of the research question of this thesis. The research question was a product of the literature study preceding this thesis. The topic of the literature study was the problem of the third edition of the Global Trajectory Optimization Competition (GTOC) (GTOC3). This problem comprises the optimization of a low-thrust interplanetary trajectory. In the literature study, GTOC3 has been used as a reference problem to study low-thrust interplanetary optimization problems in general. The characteristics of the GTOC3 problem will be revealed in Section 1.2. The conclusions of this study are presented in Section 1.2.3. These conclusions induced the study of different coordinate systems and numerical propagators, in Section 1.3 and Section 1.4 respectively, for the purpose of making the optimization of low-thrust interplanetary trajectories faster. Subsequently, an examination of useful software tools that will be used in this

thesis will be provided in Section 1.5. The research question for this thesis that followed from the literature study is presented in Section 1.6 and is accompanied by a rationale. Finally, Section 1.7 presents the outline of this thesis report.

1.2. GTOC3

The problem of the third edition of the Global Trajectory Optimization Competition (GTOC3) has been the topic of the literature study preceding this thesis. After a short background story of the GTOC in Section 1.2.1, the characteristics of the problem used for the third edition of this competition are presented in Section 1.2.2. The GTOC3 problem was used as a reference problem to study low-thrust interplanetary trajectory optimization problems in general. The conclusions of this study can be found Section 1.2.3.

1.2.1. BACKGROUND

GTOC is a contest for aerospace engineers and mathematicians all around the world. The objective is to solve a 'nearly-impossible' problem on the design of spacecraft trajectories. It is organized once every one or two years. Once the problem is defined by the winner of previous edition, the registered teams have four weeks to come up with a solution that maximizes a specified performance parameter or objective function. The submitted solution of each team is evaluated by the organizing team resulting in a score based on the value of the performance paramater [ESA, 2015]. The GTOC3 concerns a multiple sample return mission, which will be further explained in the next section. The problem was released on the 12th of November 2007 [ESA, 2015].

1.2.2. GTOC3 PROBLEM

The GTOC3 was organized by the Aerospace group of the *Dipartimento di Energetica* of the *Politecnico di Torino*, the winners of the GTOC2. The problem consists of the optimization of a multiple Near-Earth Asteroid (NEA) rendezvous mission. The spacecraft is supposed to be launched from the Earth and rendezvous with three different asteroids that are to be chosen out of a list of 140 NEAs. The asteroids can be considered as point masses and a rendezvous is defined as matching the asteroid's position and velocity. Each asteroid rendezvous is characterized by a stay-time, i.e. the period of time during which the spacecraft remains in a state of rendezvous with the asteroid. After three rendezvous' with the asteroids, the spacecraft should rendezvous with the Earth, where the mission ends. The trajectory should be optimized for the performance parameter or objective function J, which should be maximized. J is a non-dimensional quantity and is defined as follows [Casalino et al., 2007]:

$$J = \frac{m_f}{m_i} + K\frac{\min\limits_{j=1,2,3}\left(\tau_j\right)}{\tau_{\max}} \tag{1.1}$$

In Eq. (1.1) m_i and m_f are the spacecraft's initial and final mass, respectively. τ_j represents the stay-time at the j^{th} asteroid in the rendezvous sequence and τ_{max} is the total available trip time: 10 years. By looking at the definition of the objective function it can be seen that it favors low propellant consumption and long stay-times at the asteroids, thus increasing the scientific return. The constant K represents the relative importance of the stay-time and mass performance terms and equals 0.2. Back in 2007 the competition was won by the team of the Centre Nationale d'Études Spatiales (CNES) with an objective function value of 0.87.

Furthermore, the mission is subjected to a series of requirements which can best be given in a list. The requirements are subdivided into trajectory requirements, which address the constraints on the trajectory, limitations on the dynamic model, and spacecraft characteristics which define the specifications of the spacecraft at hand.

Trajectory Requirements The requirements on the trajectory of the spacecraft are:

- Launch

  L01 - The year of launch shall be in the range of 2016 to 2025, inclusive.

  L02 - The spacecraft shall be launched from the Earth with hyperbolic excess velocity of up to 0.5 km/s.

- Earth flybys

E01 - The spacecraft shall perform gravity assists around the Earth only.

- Asteroids rendezvous

RV01 - The spacecraft shall not perform a gravity assist around any asteroid.

RV02 - The spacecraft shall rendezvous with three different asteroids chosen from the provided list of 140 asteroids.

RV03 - The stay-time at each of the three asteroids shall be longer than or equal to 60 days.

- Arrival at Earth

A01 - The time between launch from and arrival at Earth shall not exceed 10 years.

Limitations on the Dynamic model The limitations on the dynamic model are:

DM01 - All bodies shall be treated as point masses.

DM02 - The Earth shall describe a Keplerian orbit around the Sun.

DM03 - Each asteroid shall follow a Keplerian orbit around the Sun.

DM04 - Outside the sphere of influence of the Earth, the only two forces acting on the spacecraft shall be the Sun's gravity force and the thrust from the propulsion system.

DM05 - Inside the sphere of influence of the Earth, the only force acting on the spacecraft shall be the Earth's gravity force.

DM06 - The coordinate frame used for reporting the solution shall be the J2000 heliocentric ecliptic reference frame.

DM07 - The constraints on position and velocity shall be satisfied with an accuracy of at least 1000 km and 1 m/s, respectively.

DM08 - The patched-conic method shall be used for the modelling of Earth flybys.

DM09 - The time spent inside the Earth's sphere of influence shall be neglected.

DM10 - The perigee radius of the hyperbolic flyby orbit around Earth shall be at least 6871 km.

Characteristics The characteristics of the spacecraft are:

SC01 - The spacecraft's initial mass shall be 2000 kg.

SC02 - The propulsion system shall have a specific impulse of 3000 s.

SC03 - The propulsion system shall have a maximum thrust level of 0.15 N.

SC04 - The spacecraft mass shall only vary because of the propellant consumption.

1.2.3. Conclusions from GTOC3

Since GTOC3 was a contest organized in 2007, there are many existing solutions available (Bertrand [2008]; Evertsz [2007]; L. Casalino and Sentinella [2007]; Belló and Cano [2008]). These solutions have been analyzed in the preceding literature study and the most important parts of optimizing GTOC3-like problems are listed below:

- Combinatorics: The selection of the best asteroid sequence consisting of three different asteroids to be chosen from a list of 140 asteroids.
- Global optimization: The identification and determination of the optimum trajectory. The detailed determination of the trajectory will be done during the local optimization.

- Numerical trajectory propagation: This is usually part of the global optimization. The global optimization requires a method to determine a trajectory for changing input conditions in order to optimize it. The computation of a single trajectory is performed by a numerical propagator, as trajectory subject to a low-thrust force cannot be integrated analytically.

- Local optimization: The finetuning of the optimum solution found by the global optimizer.

Of the above aspects it was decided to immediately omit the combinatorics' part. This is because it is more of a mathematical aspect that happens to be part of GTOC3, but which is generally not part of a spacecraft trajectory optimization problem and therefore less interesting for further research at the Astrodynamics and Space Missions (ASM) department of the TU Delft. After a literature study on global and local optimization it was decided to focus on global optimization. From an examination of global optimization and trajectory propagation, it was found that improving the speed of trajectory propagation has a large effect on the efficiency of the optimization process overall [Van den Broeck, 2016]. Surely, global optimization usually involves an evolutionary optimizer [L. Casalino and Sentinella, 2007]. In the optimization process of this class of optimizers, a generation is a set consisting of many possible trajectories. All these trajectories are propagated with a numerical propagator. Each trajectory will then get a score with the use of an objective function, such as the one in Eq. (1.1). Subsequently, this score is used to improve the set of possible trajectories of the next generation. If a population consists of 100 possible trajectories and 100 generations are used to converge to an optimum trajectory, then this means that during the evolutionary optimization 10000 trajectories have to be propagated. Therefore, if the trajectory can be propagated fast, the optimization will also be fast. A fast optimization of the trajectory is important for several reasons. First, it allows to optimize the trajectory in real-time, allowing mission control to cope with deviations from the planned trajectory more easily. A second reason is that faster optimization makes on-board trajectory optimization on slower processors possible. On-board trajectory optimization increases the autonomy of spacecraft and takes away the problem of delay when sending commands from Earth to the spacecraft, especially for interplanetary missions. The speed of numerical propagation is mainly determined by the type of numerical integration and the used coordinate system. In the literature study these two aspects were examined in detail and the findings are summarized in the next two sections.

1.3. COORDINATE SYSTEMS

In this section there will be looked at different coordinate systems to define the position and velocity of the spacecraft. First, the use of the Cartesian coordinates will be considered. Thereafter, the Kepler elements will be examined. Next, the the modified equinoctial elements will be discussed, which is followed by a discussion of the Unified State Model. Finally, in the last section, the coordinate system that can lead to the fastest numerical propagation of low-thrust trajectories will be selected.

1.3.1. CARTESIAN COORDINATES

Probably the best known form of a state vector for spacecraft is one that is expressed in Cartesian coordinates, invented in the 17th century by the French philosopher, mathematician and scientist René Descartes (1596-1650). The coordinates immediately follow from the definition of the J2000 heliocentric ecliptic reference frame (see Section 2.3) and have the following form:

$$\boldsymbol{x} = \begin{bmatrix} x & y & z & v_x & v_y & v_z \end{bmatrix}^T \tag{1.2}$$

The advantage of Cartesian coordinates is that they are well-established and have a high accuracy when integrating over linear paths. However, when in nonlinear regions, e.g. everywhere on a low eccentricity orbit or close to the pericenter and apocenter of highly eccentric orbits, the accuracy is rather low. Since the orbits of the asteroids contain few straight parts it seems that the Cartesian coordinates have an important disadvantage, see Fig. 1.1. A positive point about Cartesian coordinates is that they do not have any singularities as is the case with Kepler elements which will be explained in the next section.

1.3.2. KEPLER ELEMENTS

The Kepler elements are named after the famous German mathematician and astronomer Johannes Kepler (1571-1630) who used Tycho Brahe's observations of the planets to publish the book 'Astronomia nova' in 1609 in which he states his three laws of planetary motion. A three-dimensional Keplerian orbit is described

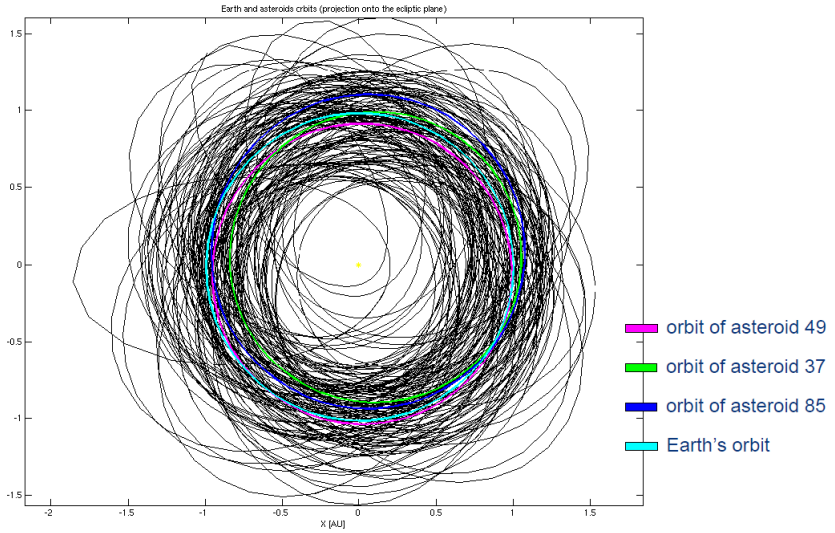


Figure 1.1: Projected orbits of the popular 49th (pink), 37th (green) and 85th (blue) asteroids along with the rest of the 140 asteroids (black) on the ecliptic plane. The orbit of the Earth around the Sun is indicated in electric blue. [Bertrand, 2008]

by five Kepler elements: $[a, e, i, \Omega, \omega]$, respectively the semi-major axis, eccentricity, inclination, longitude of ascending node and the argument of pericenter. A sixth Kepler element, the true anomaly θ indicates the position on the orbit. Because only one element is used to indicate the position on the two-dimensional orbit, Kepler elements are also a subset of the so-called canonical elements [Goldstein et al., 2002]. The Modified Equinocitial Elements, which will be discussed in the next subsection are also a subset of the canonical elements. For the transformation from Kepler elements to Cartesian coordinates and vice versa, see Appendix A.1 and Appendix A.2.
The advantages of Kepler elements are the ease of interpretation, the appearance of these elements in many mathematical equations, such as the Lagrange equations, and the availability of computer programs that can work with these elements. Disadvantages of the use of Kepler elements are however that they contain singularities. For instance when the eccentricity of an orbit is zero, the argument of pericenter is not defined. Also when the inclination is zero, the RAAN is not defined and since the argument of pericenter is measured with respect to the RAAN, it is also not defined. The common solution to this is to give the undefined elements a zero value, but special attention has to be given to these singularities while programming. The singularities in the Kepler elements are the reason of existence of the Modified Equinoctial Elements which will be discussed in the next section.

1.3.3. MODIFIED EQUINOCTIAL ELEMENTS

The equinoctial orbit elements were first investigated for the two body problem in 1972 by Roger Broucke and Paid Cefola in [Broucke and Cefola, 1972]. They also discussed the application of the equinoctial orbit elements to general and special perturbations. In 1985, M. Walker, B. Ireland and J. Owens introduced the Modified Equinoctial Elements (MEEs) in [Walker et al., 1985]. They derived the equations of motion in Lagrangian and Gaussian forms. As the name suggests, the MEEs can be seen as modified Kepler elements. The modifications are chosen such that the singularities in the inclination and eccentricity are eliminated [Mooij, 2012]. Although, the six MEEs have multiple different definitions, the following set is commonly used [Kéchichian, 2008]:

$$\boldsymbol{x} = \begin{bmatrix} a & g = e\sin(\Omega+\omega) & f = e\cos(\Omega+\omega) & k = \tan\left(\frac{i}{2}\right)\sin\Omega & h = \tan\left(\frac{i}{2}\right)\cos\Omega & L = \theta+\Omega+\omega \end{bmatrix}^T \tag{1.3}$$

Note that for the first and sixth elements different options exist. In this way instead of the semi-major axis a also the semi-latus rectum $p = a\left(1-e^2\right)$ can be used for the first element and for the sixth element also $F = E+\Omega+\omega$ is sometimes used [Walker et al., 1985].
As can be seen from Eq. (1.3), when for instance $i = 0$, it follows that $k = 0$ and $h = 0$ leaving no need to know Ω. Also, when the eccentricity $e = 0$ then $g = 0$ and $f = 0$ without the need of knowing ω. Although these elements do not have the singularities of the Kepler elements, one new singularity is created when $i = \pi$ which

causes k and h to go to infinity. An orbit with an inclination close to π rad or 180° is a retrograde orbit, i.e. a body in this orbit will evolve about the main gravitational body in clockwise direction when viewed from the North. Luckily, due to the rotational direction of the Solar System's planets around the Sun, these orbits are extremely rare for spacecraft of interplanetary missions. So it is highly unlikely that the spacecraft will ever reach an orbit with an inclination of $i = 180°$.

Thus, the MEEs advantage of having no singularities for the application to the GTOC3-like problems and the fact that they still have a strong link with Kepler elements and are therefore easier to interpret than Cartesian coordinates, make them a better choice.

1.3.4. UNIFIED STATE MODEL

A fourth type of coordinate system is the Unified State Model (USM). Unlike the above discussed elements, the USM is a method to express orbits using a set of seven elements, although efforts have been done to reduce it to six elements.

The USM was first proposed by Samuel Altman in 1972 and consists of a quaternion, which in turn consists of four fast varying elements, and three slowly varying velocity hodograph parameters [Altman, 1972]. The quaternion defines the orientation of a reference frame fixed to the orbiting body with respect to an inertial frame centered in the main body, e.g. the Sun. The shape and size of the orbit are indicated by the hodograph parameters which vary only slowly in case of perturbations. The origin of the USM will be further explained in Chapter 3.

The advantage of the USM with respect to Cartesian coordinates is that the period of variation is much longer, which results in a higher accuracy when the parameters are numerically integrated. Also, USM parameters are to be preferred over Kepler elements because they have singularities that are less obstructive, meaning that the singularities occur only in non-common trajectories. The first singularity occurs when the angular momentum per unit mass h is zero. This singularity can occur in the case for hyperbolic orbits when the true anomaly limit is reached. A second singularity occurs when the orbit is pure-retrograde.

Furthermore, in the USM the equations for the influence of perturbations are much simpler compared to a case where Kepler elements are used. Moreover, the dynamics of USM are simpler than that of Modified Equinoctial Elements [Vittaldev et al., 2010]. The complete set of USM elements is the following:

$$\boldsymbol{x} = \begin{bmatrix} C & R_{f1} & R_{f2} & \epsilon_{O1} & \epsilon_{O2} & \epsilon_{O3} & \eta_O \end{bmatrix}^T \tag{1.4}$$

where C is the magnitude of the spacecraft's velocity in the direction normal to the radius and in the orbital plain. R_{f1} and R_{f2} are the velocity components along two axes of a frame that will later be defined in Fig. 3.3. Vectors $\boldsymbol{C}$ (velocity-direction is positive) and $\boldsymbol{R}$ are also known as the hodograph parameters as they contain the information to set up the velocity hodograph of the spacecraft's orbit. $\epsilon_{O1}, \epsilon_{O2}, \epsilon_{O3}$ and η_O are the four elements of the quaternion. They describe the orientation of a body-centered frame with respect to the inertial frame centered in the main body. They are also called the Euler parameters. The conversion of the above elements from and to the Cartesian coordinates and Kepler elements is quite intensive and is therefore omitted in this section. The reader is referred to Appendix A for these conversions.

In [Schaub and Junkins, 2002] it is stated that any attitude parameter set that has less than 4 elements will have a singularity. This would mean that is not possible to reduce the amount of parameters needed in the USM to describe the orientation of the orbital plane from 4 to 3 elements. However, with the use of Modified Rodrigues Parameters (MRPs) it is possible to describe the orientation of the orbital frame in just 3 parameters. Although this causes a singularity for $\eta = 1$, the singularity can be avoided by the use of an additional alternative set of parameters known as the Shadow Modified Rodrigues Parameter (SMRP). The SMRPs have a singularity for $\eta = -1$. Since it is possible to alternate between the SMRPs and MRPs, all singularities can be avoided. With the use of the MRP, the USM can be reduced to six elements, which is referred to as USM6:

$$\boldsymbol{x} = \begin{bmatrix} \lambda & \gamma & \epsilon_{O1} & \epsilon_{O2} & \epsilon_{O3} & \eta_O \end{bmatrix}^T \tag{1.5}$$

1.3.5. SELECTION OF ORBITAL STATE VECTORS

From the comparative study performed in [Vittaldev, 2010] it can be concluded that based on criteria such as computation time and accuracy the Unified State Model is the best choice for the numerical propagation of low-thrust orbits. As the USM6 and USM7 coordinate systems have a similar computational efficiency for the propagation of low-thrust trajectories, the USM7 elements are selected for simplicity as they do not require an MRP and SMRP or any other technique to get rid of singularities. The fact that USM7, further referred to as USM, is not widely used yet is a disadvantage, but this can on the other hand be seen as an opportunity for this thesis to reintroduce this coordinate system in the optimization of interplanetary trajectories.
The reader is referred to Appendix B for a list of quotes on the USM in literature. Appendix A.3 contains the detailed description of the transformation from Kepler elements to USM elements. The reverse transformation is described in Appendix A.4.

1.4. NUMERICAL PROPAGATION

Numerical propagators are based on a numerical integration method. These methods are used to numerically integrate the equations of motion that describe the trajectory of the spacecraft. In the next few sections different numerical methods are discussed and the preferred method is selected in Section 1.4.4

1.4.1. EULER

The Euler integration method is the simplest integration method. To compute the state at time point $t = t_0 + h$ from a known state at time point t_0 with Euler integration, the first derivative of the state at t_0 is multiplied by the step-size h and added to the previous state at t_0:

$$\boldsymbol{x}(t) = \boldsymbol{x}(t_0) + \dot{\boldsymbol{x}}(t_0)h \tag{1.6}$$

Because this computation only requires one evaluation of the state derivative, i.e. at t_0, this is called a first order method. Although this method is fast for a single step of the integrator, it is not very accurate. This lack of accuracy can be compensated by taking more steps, i.e. decreasing the step-size h, but this slows down the propagation. Low-thrust trajectory optimization is known for having slow-varying dynamics, hence favoring large step-sizes. Unfortunately, Euler integration cannot benefit from the slow-varying dynamics due to the low order of integration. Therefore, Euler integration will be a slow method for low-thrust trajectory propagation.

1.4.2. RUNGE-KUTTA

The Runge-Kutta integration technique uses a weighted average of the state derivative at different time points to estimate the state vector at the next step. The method that is used to determine the weighted derivative Φ defines the type of Runge-Kutta integration method. Nowadays, plenty of different types of RK integration methods exist.

One method that is frequently used in orbit propagation is the fourth order Runge-Kutta (RK4) integrator which involves four state derivative evaluations per integration step. The state derivatives are computed at t_0, $t_0 + \frac{h}{2}$ and at t. The RK4 integration scheme is known for its combination of simplicity, accuracy and low computation time when propagating orbits [Haas, 2013]. A two-dimensional scheme is depicted in Fig. 1.2. The corresponding equations are the following [Kutta, 1901]:

$$\boldsymbol{x}(t_0 + h) \approx \boldsymbol{x}(t_0) + h\Phi \tag{1.7}$$

$$\Phi = \frac{1}{6}(\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4) \tag{1.8}$$

$$\boldsymbol{k}_1 = \boldsymbol{d}(t_0, \boldsymbol{x}(t_0)) \tag{1.9}$$

$$\boldsymbol{k}_2 = \boldsymbol{d}\left(t_0 + \frac{h}{2}, \boldsymbol{x}(t_0) + \frac{h\boldsymbol{k}_1}{2}\right) \tag{1.10}$$

$$\boldsymbol{k}_3 = \boldsymbol{d}\left(t_0 + \frac{h}{2}, \boldsymbol{x}(t_0) + \frac{h\boldsymbol{k}_2}{2}\right) \tag{1.11}$$

$$\boldsymbol{k}_4 = \boldsymbol{d}(t_0 + h, \boldsymbol{x}(t_0) + h\boldsymbol{k}_3) \tag{1.12}$$

In the above equations, $\boldsymbol{x}(t)$ is the state vector at time t. h is the integration step-size and Φ is the overall derivative that is used to propagate the initial state $\boldsymbol{x}(t_0)$ over time h to the new state $\boldsymbol{x}(t_0+h)$. In a fourth order Runge-Kutta integration scheme the overall derivative Φ comprises four different derivatives, which are computed with the state derivative function $\boldsymbol{d}(t,\boldsymbol{x})$, namely: $\boldsymbol{k}_1$ at time t_0, $\boldsymbol{k}_2$ and $\boldsymbol{k}_3$ at time $t_0+h/2$, and $\boldsymbol{k}_4$ at time t_0+h. These derivatives each have a different weight. The ones at time step $t_0+h/2$ have double the weight of the other two derivatives.

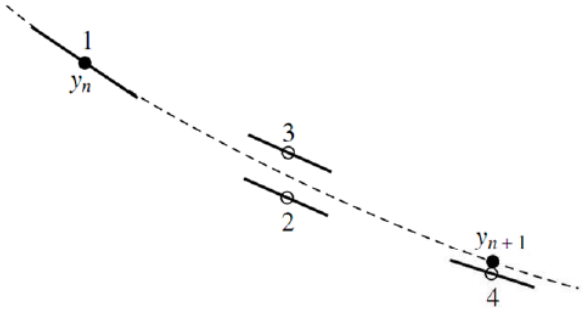


Figure 1.2: Two-dimensional illustration of the integration scheme of the fourth order Runge-Kutta method [Noomen, 2014b]. Note that y_n represents $x(t_0)$ and that y_{n+1} refers to $x(t_0+h)$.

Generalizing the above equations for any order RK method gives:

$$\boldsymbol{x}(t_0+h) = \boldsymbol{x}(t_0) + h\sum_{i=1}^{s} b_i \boldsymbol{k}_i \tag{1.13}$$

where

$$\boldsymbol{k}_1 = \boldsymbol{d}(t_0, \boldsymbol{x}(t_0)) \tag{1.14}$$
$$\boldsymbol{k}_2 = \boldsymbol{d}\left(t_0 + c_2 h, \boldsymbol{x}(t_0) + h(a_{21}\boldsymbol{k}_1)\right) \tag{1.15}$$
$$\boldsymbol{k}_3 = \boldsymbol{d}\left(t_0 + c_3 h, \boldsymbol{x}(t_0) + h(a_{31}\boldsymbol{k}_1 + a_{32}\boldsymbol{k}_2)\right) \tag{1.16}$$
$$\vdots$$
$$\boldsymbol{k}_s = \boldsymbol{d}\left(t_0 + c_s h, \boldsymbol{x}(t_0) + h(a_{s1}\boldsymbol{k}_1 + a_{s2}\boldsymbol{k}_2 + \cdots + a_{s,s-1}\boldsymbol{k}_{s-1})\right) \tag{1.17}$$

in which s is the number of stages, b_i are the weights and c_i are the time points or nodes expressed as a ratio of the step-size. The matrix $[a_{ij}]$ is known as the Runge-Kutta matrix. These coefficients can be arranged in a table known as the Butcher tableau [Butcher, 1963, 2008]:

$$\begin{array}{c|ccccc} 0 & & & & & \\ c_2 & a_{21} & & & & \\ c_3 & a_{31} & a_{32} & & & \\ \vdots & \vdots & & \ddots & & \\ c_s & a_{s1} & a_{s2} & \cdots & a_{s,s-1} & \\ \hline & b_1 & b_2 & \cdots & b_{s-1} & b_s \end{array} \tag{1.18}$$

In *adaptive* RK methods, the difference between the solutions produced with two consecutive orders is used to estimate the truncation error. The truncation error is then used to alter the step-size that will be used for the next integration step-size. In general, adaptive RK methods have the following form for the lower order, indicated with an asterisk (*) [Press et al., 2007]:

$$\boldsymbol{x}^*(t_0+h) = \boldsymbol{x}(t_0) + h\sum_{i=1}^{s} b_i^* \boldsymbol{k}_i \tag{1.19}$$

where $\boldsymbol{k}_i$ are the same for the higher-order solution. Now, the truncation error is of order $O(h^p)$ and is determined by:

$$\boldsymbol{\varepsilon}(t_0+h) = \boldsymbol{x}(t_0+h) - \boldsymbol{x}^*(t_0+h) = h\sum_{i=1}^{s}(b_i - b_i^*)\boldsymbol{k}_i \tag{1.20}$$

In comparison with Eq. (1.18), the Butcher tableau now gets an extra row for the weights of the lower order method:

$$\begin{array}{c|ccccc} 0 & & & & & \\ c_2 & a_{21} & & & & \\ c_3 & a_{31} & a_{32} & & & \\ \vdots & \vdots & & \ddots & & \\ c_s & a_{s1} & a_{s2} & \cdots & a_{s,s-1} & \\ \hline & b_1 & b_2 & \cdots & b_{s-1} & b_s \\ & b_1^* & b_2^* & \cdots & b_{s-1}^* & b_s^* \end{array} \tag{1.21}$$

In fact, the previously discussed Euler method is also a Runge-Kutta method with the following Butcher tableau:

$$\begin{array}{c|c} 0 & \\ \hline & 1 \end{array} \tag{1.22}$$

Of all RK methods, RK8(7)13M is selected because it is known to be fast and accurate for the propagation of low-thrust trajectories [Ghosh, 2013]. RK8(7)13M was proposed by Dormand and Prince in [Prince and Dormand, 1981] and is also known as DOPRI8. It is an 8^{th} order adaptive RK method in which a 7^{th} order RK is used to estimate the truncation error that is needed to determine the next step-size. Despite having 13 stages per integration, the method is fast because its high order allows very large step-sizes. The Butcher tableau of RK8(7) can be found on page 6 of [Prince and Dormand, 1981]. Note that Dormand and Prince used (ˆ) to denote the *higher* order solution as opposed to (*) that has been used in the above formulas to denote the *lower* order solution.

1.4.3. TAYLOR SERIES INTEGRATION

Taylor Series Integration (TSI) is a numerical integration technique that uses Taylor series expansions to integrate Ordinary Differential Equation (ODE)s that describe the variation of the state variables of the spacecraft over time. In other words, a Taylor series expansion is used to approximate the function that describes the value of a state variable over time. As can be seen in Eqs. (1.23) and (1.24), the Taylor series expansion of the n^{th} state variable at a certain time t_0 involves the calculation of the time derivatives of that state variable. The higher the order of the TSI, the more terms in the Taylor series expansion and the higher the accuracy of the TSI. A high order also means that a lot of high order derivatives need to be computed. This differentiation process can be facilitated by the use of recursive differentiation. This process is initiated by recasting the ODEs into the required *canonical* or *normal* form. Next, the higher order derivatives can be obtained from differentiating the next lower order derivative and they can in turn be obtained from differentiation of their next lower order derivative [Scott and Martini, 2008]. This process can be automated and implemented in a computer program, as will be seen in Chapter 4. One of the main features of this method is that it allows the user to choose the order of Taylor series expansion.

The n^{th} state variable x_n can be approximated by a finite Taylor series of K^{th} order. The error of the approximation is called the truncation error $\varepsilon_{n,K}$ for state variable n and order K. So for an initial condition at time t_0, the state variables can be expanded into a Taylor series as follows:

$$x_n(t) = \sum_{k=0}^{K} \frac{x_n^{(k)}(t_0)}{k!}(t-t_0)^k + \varepsilon_{n,K} \tag{1.23}$$

in which

$$x_n^{(k)} = \frac{d^k x_n}{dt^k} \tag{1.24}$$

When all the ODEs for the state variables are expanded according to Eq. (1.23) with a certain order K it is possible to evaluate the newly obtained expressions for the state variables at time $t_1 = t_0 + h_1$, in which h_1 is the

integration step-size which is determined to meet the local error tolerance. h_1 and $e_{n,1}$ are used to calculate the next step-size h_2. From t_1, the state variables are expanded in a new series to $t_2 = t_1 + h_2$. This process of 'analytical continuation' is repeated until the final integration time t_f is reached.

1.4.4. SELECTION OF NUMERICAL PROPAGATOR

It was decided to use TSI for the numerical integration of the low-thrust trajectory. The main motivation for using TSI is that the required CPU time is at least 1 to even 3 orders of magnitude less than traditional integration with Runge Kutta methods [Vittaldev, 2010], depending on the problem. Moreover, due to the possibility of using high order expansions during the integration, the step-sizes can become very large when compared to normal Runge Kutta integration methods which is especially beneficial for trajectories with slow-varying perturbations such as low-thrust trajectories. However, there is price attached to this increase in simulation speed. This price is the complexity of implementation.
The reader is referred to Appendix B for a list of quotes on the TSI in literature.

1.5. SOFTWARE

The TU Delft has its own astrodynamics software tool called the TU Delft Astrodynamics Toolbox (Tudat). Tudat was initiated in 2010 and its main contributors are Kartik Kumar, Dominic Dirkx and Jacco Geul [Tudat, 2017]. Recently, Tudat has made the transition from its own website to GitHub. The GitHub platform allows an easier interaction between the code written by the students and the core code which boosts new contributions. Many fellow students will be working with Tudat which guarantees sufficient support. For these reasons, the Tudat code library will be used whenever possible in the thesis work. The code will be edited in the free compiler Qt Creator. The output of the C++ modules will be exported to Matlab in order to make the data visually comprehensible using plots.

1.6. RESEARCH QUESTION

From the summary of the literature study in the previous sections of this chapter, this section will provide the rationale for the research question of this thesis.

The determination of optimal low-thrust trajectories certainly is a very important, yet challenging aspect of modern space missions. It is important because low-thrust propulsion is a very efficient way of propelling a spacecraft in space. The determination of the best trajectory is an optimization problem. For low-thrust trajectories this optimization is particularly challenging because of the large number of variables to be optimized. Indeed, low-thrust propulsion implies that the magnitude and direction of the thrust has to be optimized for every node of the trajectory, hence resulting in a large number of variables.

For complex problems such as low-thrust trajectory optimization, evolutionary algorithms can be used as they only require a fitness or objective function and are independent of the equations of motion. In evolutionary optimization many possible trajectories are computed. The value of the objective function is then calculated for each trajectory. These values are used to improve the population of the trajectories, generation after generation. The low-thrust trajectories are computed using numerical propagators, as their equations of motion cannot be integrated analytically.

A fast optimization of the trajectory is important for several reasons. First, it allows to optimize the trajectory in real-time, allowing mission control to cope with deviations from the planned trajectory more easily. A second reason is that faster optimization makes on-board trajectory optimization on slower processors possible. On-board trajectory optimization increases the autonomy of spacecraft and takes away the problem of delay when sending commands from Earth to the spacecraft, especially for interplanetary missions.

As the trajectories are computed a large number of times, the speed of the evolutionary optimization depends heavily on the speed of the numerical propagator. To investigate ways to lower the CPU time without jeopardizing the accuracy of numerical propagators, the GTOC3 problem is used as framework in this thesis.

The speed of numerical propagation is mainly determined by the type of numerical integration and the used coordinate system. Regarding numerical integration, the literature study preceding this thesis pointed out

that Taylor Series Integration (TSI) is a fast and promising method. As for the coordinate system, the Unified State Model (USM) allows fast propagation. Therefore, the combination of TSI and USM is used. Unfortunately, this method is very problem-specific as the equations of the TSI vary per problem. Once the TSI is set up for a particular problem, it can be much faster than common Runge-Kutta (RK) methods. However, when using evolutionary algorithms, the problem changes constantly due to variations in the low-thrust profile of the trajectories. Therefore, each low-thrust profile would require a specific setup of the TSI, which would be detrimental for its use in evolutionary algorithms.

Therefore, the main research question of this thesis, is the following:

"Can the Taylor Series Integration method be generalized to allow fast evolutionary optimization of low-thrust spacecraft trajectories?"

LIST OF ADDITIONAL RESEARCH QUESTIONS

The method that is used to answer the research question involves different aspects which lead to the following additional research questions and sub-questions:

- 1. What is the influence of the increase in generality of the TSI on its computational efficiency?
  - 1.1 Does the use of USM elements provide a faster propagation of trajectories than Cartesian coordinates?
  - 1.2 How does the computational efficiency of the generalized TSI compare against generalized Runge-Kutta based propagators for a fixed step-size?
  - 1.3 How does the computational efficiency of the generalized TSI compare against generalized Runge-Kutta based propagators for a variable step-size?
  - 1.4 How does the accuracy of the generalized TSI compare against generalized Runge-Kutta based propagators?
- 2. What influence does the type of orbit have on the CPU time?
- 3. What is the effect of the Tudat propagation setup on the CPU time?
- 4. Does the optimum order of the TSI vary per problem?
  - 4.1. Does this affect the overall CPU time of the generalized TSI?

1.7. THESIS OUTLINE

First, the astrodynamic models and the reference frames used for this thesis will be defined in Chapter 2. Next, the Unified State Model will be studied in detail in Chapter 3. When the characteristics of the USM are understand completely, the combination of TSI and USM, forming the TSI-USM propagator, will be explained in Chapter 4. It will become clear why TSI is a problem-specific method, and why this would be detrimental for its use in evolutionary algorithms for the global optimization of low-thrust trajectories. In Chapter 5 a model, referred to as the General Discrete Force Model, will be developed that, when implemented in TSI-USM, will allow the propagation of any low-thrust profile, as long as it is specified in advance and with respect to time. This generalized TSI-USM will be referred to as the GTSI-USM. Subsequently, Chapter 6 will discuss the verification of the GTSI-USM and describe three reference propagators that will be used to determine and judge the accuracy and the computational efficiency of the GTSI-USM. Chapter 7 will present the validation of the GTI-USM and show the results of its performance with respect to the reference propagators. The conclusions of the results will be presented Chapter 8. Finally, the last chapter of this thesis will contain the author's recommendations for future work as well as some valuable tips for prospective master students. Fig. 1.3 presents the chapter outline of the thesis.

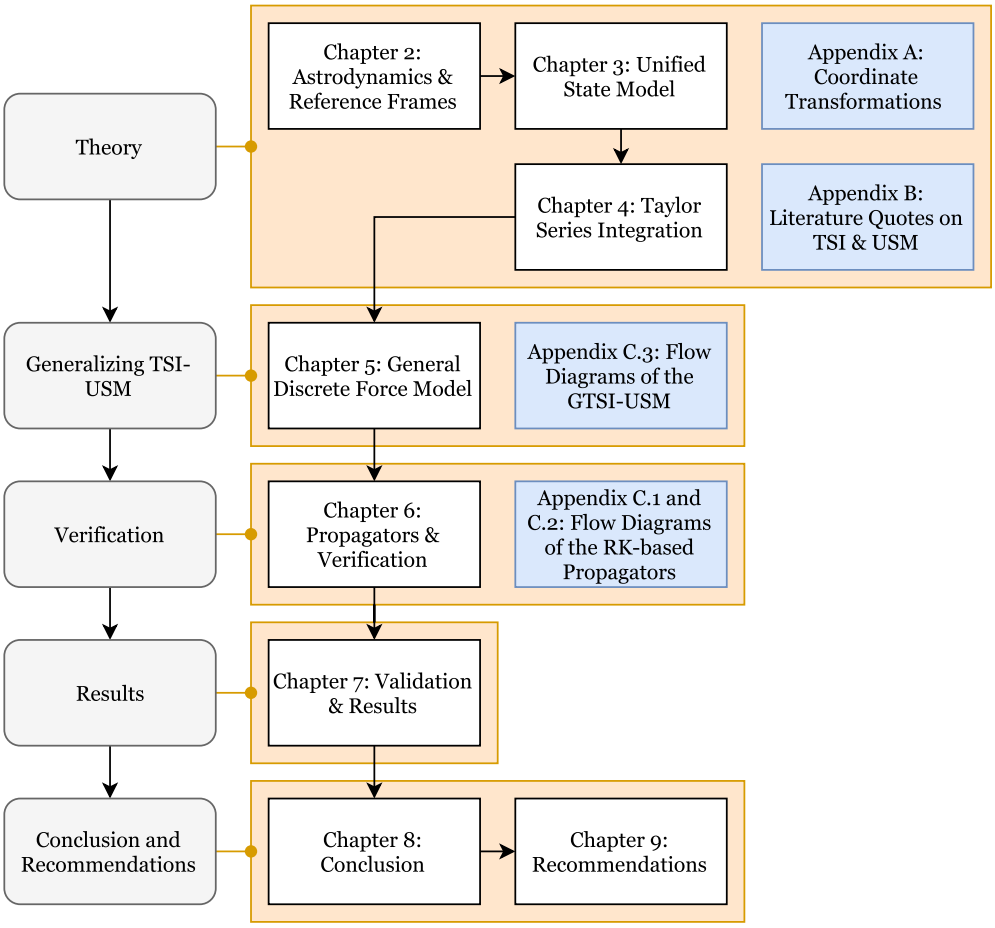


Figure 1.3: Thesis outline

2

Astrodynamics & Reference Frames

The previous chapter presented the process that lead to the establishment of the research question of this thesis. To answer the question whether Taylor Series Integration (TSI) can be generalized for low-thrust trajectories, the TSI method needs to be analyzed in detail. This analysis will be carried out in Chapter 4 and will be preceded by a discussion of the selected coordinate system, the Unified State Model (USM), in Chapter 3. The developed method to generalize the combination of TSI and USM will be described in Chapter 5. Its verification and performance tests will be presented in Chapter 6 and Chapter 7, respectively. Before going into the theory of TSI and the USM, the definition of the astrodynamic models and reference frames used throughout this thesis will be provided in this chapter.

Contents

2.1. Introduction

This chapter contains the definition of the astrodynamic model and the reference frames that are used throughout this thesis. First, in Section 2.2, the gravity model, thrust model and the combination of these two models in the equations of motion for the low-thrust trajectories are presented. Section 2.3 provides an overview of the definition of the inertial reference frame as well as the spacecraft-centered reference frames that are used for the specification of the thrust force.

2.2. Astrodynamics

This section will provide an overview of the astrodynamic models used in this thesis. Since the focus of this thesis is on low-thrust trajectories, the complexity of perturbations other than the thrust force is limited as much as possible. First, the model that is used to incorporate the gravitational acceleration will be discussed Section 2.2.1. Secondly, the thrust model is explained in Section 2.2.2. Finally, Section 2.2.3 will show the combination of both models in the equations of motion that will be used to propagate low-thrust trajectories in this thesis.

2.2.1. GRAVITY MODEL

As the GTOC3 problem is used as the reference low-thrust trajectory optimization problem of this thesis, its gravity model will be used in for the implementation of the general TSI-USM. According to the GTOC3 problem definition, the only two considered forces acting on the spacecraft are the gravity and thrust force [Casalino et al., 2007]. During the coasting phases, e.g. when the spacecraft has a rendezvous with an asteroid, the thruster is turned off and the spacecraft experiences no perturbations what so ever. Outside the Earth's sphere of influence, the spacecraft will be completely subjected to the central gravity field of the Sun only, as all existing perturbations should be neglected (see requirement DM04 in Section 1.2.2). During propelled phases of the trajectory, the perturbing thrust force will be included in the equations of motion of the spacecraft. The model of the perturbed orbit will be discussed further in Section 2.2.2.

A central gravitational field is used. There is no need to make the model more complicated to test the general TSI-USM. Because this thesis focuses on interplanetary trajectories, the strength of the gravitational field is determined by gravitational parameter of the Sun: $\mu_S = 1.32712440018\text{e}20\ \text{m}^3/\text{s}^2$. The gravitational accelerations in Cartesian coordinates are:

$$\ddot{x} = \frac{-\mu_S}{r^3} x \tag{2.1}$$

$$\ddot{y} = \frac{-\mu_S}{r^3} y \tag{2.2}$$

$$\ddot{z} = \frac{-\mu_S}{r^3} z \tag{2.3}$$

with

$$r = \sqrt{x^2 + y^2 + z^2} \tag{2.4}$$

2.2.2. THRUST MODEL

The model to implement the thrust is also adapted from the GTOC3 problem description [Casalino et al., 2007]. The thrust accelerations are define as:

$$f_x = \frac{F_x}{m} \tag{2.5}$$

$$f_y = \frac{F_y}{m} \tag{2.6}$$

$$f_z = \frac{F_z}{m} \tag{2.7}$$

with

$$\dot{m} = \frac{-F}{g_0 I_{sp}} \tag{2.8}$$

in which $g_0 = 9.80665\ \text{m/s}^2$ is the standard gravity and I_{sp} is the specific impulse. The following constants are adapted from the GTOC3 problem description:

Table 2.1: Definition of constants

Symbol	Name	Value	Unit
I_{sp}	specific impulse	3000	s
m_0	initial mass	2000	kg

2.2.3. EQUATIONS OF MOTION

The Cartesian state vector of the spacecraft consists of the position and velocity vectors and the mass:

$$\boldsymbol{x} = \begin{bmatrix} x \\ y \\ z \\ v_x \\ v_y \\ v_z \\ m \end{bmatrix} \tag{2.9}$$

The equations of motion can be obtained by taking the sum of the previously defined gravitational accelerations and thrust accelerations. The Cartesian state vector of the spacecraft consists

$$\ddot{x} = \frac{-\mu_S}{r^3} x + \frac{F}{m} \tag{2.10}$$

$$\ddot{y} = \frac{-\mu_S}{r^3} x + \frac{F}{m} \tag{2.11}$$

$$\ddot{z} = \frac{-\mu_S}{r^3} x + \frac{F}{m} \tag{2.12}$$

$$\dot{m} = \frac{-F}{g_0 I_{sp}} \tag{2.13}$$

Now the complete state derivative model becomes:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} v_x \\ v_y \\ v_z \\ \frac{-\mu_S}{r^3} x + \frac{F}{m} \\ \frac{-\mu_S}{r^3} x + \frac{F}{m} \\ \frac{-\mu_S}{r^3} x + \frac{F}{m} \\ \frac{-F}{g_0 I_{sp}} \end{bmatrix} \tag{2.14}$$

2.3. REFERENCE FRAMES

The three reference frames used in this thesis as well as the transformations between them are discussed in the next following sections.

2.3.1. INERTIAL REFERENCE FRAME

Since the GTOC3 problem deals with heliocentric trajectories it is best to use a heliocentric reference frame. Usually the x-axis of the inertial reference frame is defined to point in the direction of the vernal equinox. However, the position of the vernal equinox in the celestial sphere of the Sun is not constant. The solution to this is the J2000 heliocentric ecliptic reference frame. In this right-handed orthogonal frame with the center of the Sun at the origin, the x-axis points toward the position of the vernal equinox on January 1st, 2000 at 12:00h Terrestial Time (TT). The TT runs parallel with the International Atomic Time (TAI) to millisecond accuracy with an offset of 32.184 s [IAU, 1991]. The y-axis of the J2000 heliocentric ecliptic reference frame lies in the ecliptic. The z-axis is perpendicular to the x-y-plane and points toward the North. This reference frame is also used in the problem definition of GTOC3 and the Kepler elements of the asteroids are given in this frame. For these reasons, the J2000 heliocentric ecliptic reference frame will serve as the inertial reference frame in this thesis. Fig. 2.1 illustrates the selected reference frame.

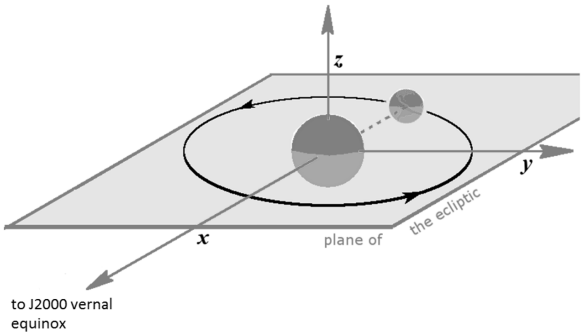


Figure 2.1: Definition of the J2000 heliocentric ecliptic reference frame. [Wikipedia, 2012] (edited)

Note that the J2000 heliocentric reference frame is in fact not an inertial reference frame since its center is not moving through space on a straight line with a constant velocity. Surely, the Sun rotates around the center of the Milky Way with a period of approximately 250 million years and with an orbital velocity of around 220 km/s. And even the entire Milky Way is moving through space with respect to other galaxies. If each of these motions would be accounted for, there would only be one inertial reference frame: the one that is centered at the center of the Universe, but the Universe does not even have a center. Nevertheless, for most applications the J2000 heliocentric ecliptic reference frame can be regarded as an inertial reference frame. This assumption is also made in this thesis.

The J2000 heliocentric reference frame will be used to define the position and velocity of the spacecraft. The orientation of this reference frame can also be used to define the thrust force exerted by the spacecraft's engine. In that case the reference frame would not be heliocentric anymore, instead it will become the J2000 spacecraft-centered reference frame. However, sometimes it is more useful to define the thrust force in another spacecraft-centered reference frame. Two additional reference frames are used in this thesis to define the thrust force. These reference frames are discussed in the next two sections.

2.3.2. VELOCITY FRAME

The velocity frame (VF) is a right-handed orthogonal spacecraft-centered reference frame of which the x-axis is parallel to the spacecraft's velocity vector, the y-axis lies in the orbital plane and the z-axis points in the direction of the positive angular momentum vector. The frame is depicted in Figs. 2.2 and 2.3. The advantage of this frame is the ease of interpretation. Therefore, this frame will be used to define the input thrust profile, as will become clear in Chapter 5. The frame is similar to the NTW coordinate system described in [Vallado, 2007]. The difference is an opposite direction of the N-axis and a different order of the N and T axes.

The following names, which are based on the Frenet-Serret frame, will be used to indicate the three axes of the VF [Frenet, 1852; Serret, 1851].

Table 2.2: Nomenclature of the VF axes used in this thesis

VF axis	Name	Explanation
x-axis	tangential	tangent to the trajectory, pointing in the direction of motion
y-axis	normal	normal to the tangent, in the orbital plane and in the direction of the positive derivative of the tangential unit vector with respect to the arclength parameter of the trajectory
z-axis	binormal	normal to the tangential and normal axes

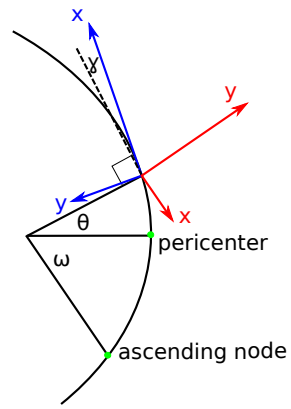


Figure 2.2: 2D illustration of the velocity frame (blue) and the J2000 frame (red) centered at the spacecraft.

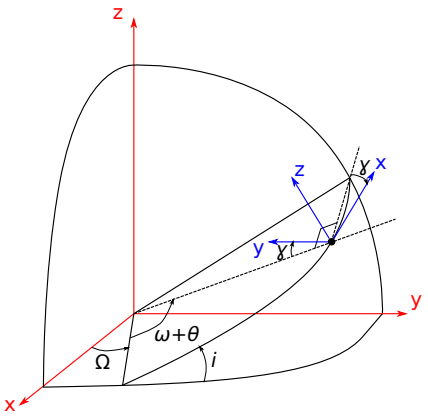


Figure 2.3: 3D illustration of the velocity frame (blue) and the J2000 frame (red).

2.3.3. RTN FRAME

As will be seen in Chapter 4, the state derivative model in USM elements requires the use of a reference frame different from the previously defined velocity frame for the definition of the thrust force. It is a right-handed orthogonal spacecraft-centered reference frame of which the x-axis is parallel to the spacecraft's radius vector, the y-axis is in the orbital plane, and the z-axis points in the direction of the positive angular momentum vector. In literature, this reference frame is also referred to as the RSW or local vertical, local horizontal (LVLH) frame. The frame is illustrated in Fig. 2.4. Note that, to comply with the notation used in literature, the x, y and z axes of the RTN frame will be referred to as the $e1$, $e2$ and $e3$ axes in the context of the USM.

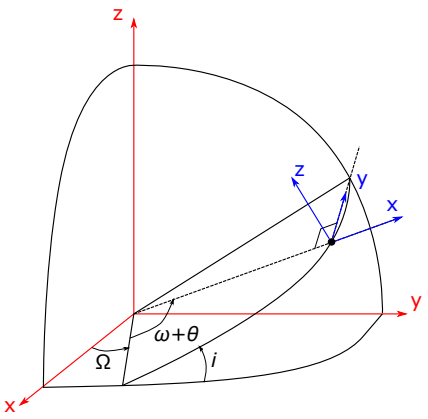


Figure 2.4: 3D illustration of the RTN frame (blue) with respect to the J2000 frame (red)

2.3.4. FRAME TRANSFORMATIONS

In this section, the frame transformation matrices to and from the above stated reference frames will be given.

VF TO J2000 SPACECRAFT-CENTERED FRAME TRANSFORMATION

If $\boldsymbol{F}_{\text{J2000}}$ is the thrust vector in the J2000 spacecraft-centered reference frame and $\boldsymbol{F}_{\text{VF}}$ is the thrust vector in the velocity frame, then the transformation between the two can be computed with the use of either the Keplerian state or the Cartesian state.

Keplerian state

For the Keplerian state $[a, e, i, \Omega, \omega, \theta]$ the following method can be used:

$$\boldsymbol{F}_{\text{J2000}} = Z_{\text{VF}}(-\Omega) \cdot X_{\text{VF}}(-i) \cdot Z_{\text{VF}}\left(-\frac{\pi}{2} + \gamma - (\theta + \omega)\right) \cdot \boldsymbol{F}_{VF} \tag{2.15}$$

in which [Mulder et al., 2013]:

$$Z_{\text{VF}}(-\Omega) = \begin{bmatrix} \cos(-\Omega) & \sin(-\Omega) & 0 \\ -\sin(-\Omega) & \cos(-\Omega) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.16}$$

$$X_{\text{VF}}(-i) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(-i) & \sin(-i) \\ 0 & -\sin(-i) & \cos(-i) \end{bmatrix} \tag{2.17}$$

$$Z_{\text{VF}}\left(-\frac{\pi}{2} + \gamma - (\theta + \omega)\right) = \begin{bmatrix} \cos\left(-\frac{\pi}{2} + \gamma - (\theta + \omega)\right) & \sin\left(-\frac{\pi}{2} + \gamma - (\theta + \omega)\right) & 0 \\ -\sin\left(-\frac{\pi}{2} + \gamma - (\theta + \omega)\right) & \cos\left(-\frac{\pi}{2} + \gamma - (\theta + \omega)\right) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.18}$$

Of course, the reversed operation can be done by reversing the order of multiplication of the transformation matrices in Eq. (2.15) and taking the additive inverses of the rotation angles [Mulder et al., 2013].

Cartesian state

The transformation can also be done with the use the Cartesian radius $\boldsymbol{r}$ and velocity $\boldsymbol{v}$ vectors in the J2000 heliocentric frame. First, the unit column vectors of the velocity frame are calculated:

$$\hat{\boldsymbol{x}}_{\text{VF}} = \frac{\boldsymbol{v}}{|\boldsymbol{v}|} \tag{2.19}$$

$$\hat{\boldsymbol{z}}_{\text{VF}} = \boldsymbol{r} \times \boldsymbol{v} \tag{2.20}$$

$$\hat{\boldsymbol{y}}_{\text{VF}} = \hat{\boldsymbol{z}}_{\text{VF}} \times \hat{\boldsymbol{x}}_{\text{VF}} \tag{2.21}$$

Next, the transformation matrix can be filled:

$$T = \begin{bmatrix} \hat{\boldsymbol{x}}_{\text{VF}} & \hat{\boldsymbol{y}}_{\text{VF}} & \hat{\boldsymbol{x}}_{\text{VF}} \end{bmatrix} \tag{2.22}$$

And finally:

$$\boldsymbol{F}_{\text{J2000}} = T\boldsymbol{F}_{\text{VF}} \tag{2.23}$$

Of course, the inverse transformation would be:

$$\boldsymbol{F}_{\text{VF}} = T^{-1}\boldsymbol{F}_{\text{J2000}} \tag{2.24}$$

VF TO RTN FRAME TRANSFORMATION

As can be seen from Fig. 2.3 and Fig. 2.4 the difference between the VF and RTN frames is a rotation around their z-axis by the flight path angle γ, so the rotation matrix becomes:

$$Z(\gamma) = \begin{bmatrix} \cos(\gamma) & \sin(\gamma) & 0 \\ -\sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.25}$$

The transformation now is:

$$\boldsymbol{F}_{\mathrm{VF}} = Z(\gamma)\boldsymbol{F}_{\mathrm{RTN}} \tag{2.26}$$

Of course, the inverse transformation can be done by:

$$\boldsymbol{F}_{\mathrm{RTN}} = Z(-\gamma)\boldsymbol{F}_{\mathrm{VF}} \tag{2.27}$$

RTN TO J2000 SPACECRAFT-CENTERED FRAME TRANSFORMATION

The transformation from the RTN to the J2000 spacecraft-centered frame can simply be done by first performing the transformation from RTN to VF followed by the transformation from VF to J2000.

3

UNIFIED STATE MODEL

In Chapter 1 the Unified State Model (USM) has been selected as the coordinate system to be combined with Taylor Series Integration (TSI) because of its positive contribution to the the computational efficiency when propagating trajectories of spacecraft. This chapter is the first of two chapters which explain the necessary theory of the TSI-USM propagator. This chapter will address the USM coordinate system and Chapter 4 will explain the TSI method and the combination of USM and TSI to form the TSI-USM propagator. It will be followed by the development of the method to generalize the TSI-USM in Chapter 5. The verification and the performance of the generalized TSI-USM are addressed in Chapter 6 and Chapter 7 respectively.

Contents

3.1. INTRODUCTION

The USM is a relatively unknown coordinate system used to express the position and velocity of a spacecraft in space. Section 3.2 presents its origin. Unlike Cartesian coordinates and Kepler elements, the state vector of a spacecraft expressed in USM elements contains seven state variables. The first three elements are velocity hodograph parameters defining the size and shape of the spacecraft's orbit. These parameters will be explained in Section 3.3. The remaining four elements complete a quaternion that defines the orientation of a spacecraft-fixed frame with respect to the inertial frame as can be read in Section 3.4. Finally, Section 3.5 will provide the set of equations that describe the change in USM elements over time for a spacecraft in a central gravitational field subject to a perturbing force.

For a detailed description of the transformation from Kepler elements to USM elements and vice versa, the reader is referred to Appendix A.3 and Appendix A.4, respectively.

3.2. ORIGIN OF THE USM

The USM was first proposed by Samuel Altman in 1972 and consists of a quaternion, which in turn consists of four fast varying elements, and three slowly varying velocity hodograph parameters [Altman, 1972]. The quaternion defines the orientation of a reference frame fixed to the orbiting body with respect to an inertial frame centered in the main body, e.g. the Sun. The shape and size of the orbit are indicated by the hodograph parameters which vary only slowly in case of perturbations.

There has not been much research carried out on the USM since its creation. After its inception in 1972, Altman wrote an article about the application of the USM for satellite state estimation in 1975 [Altman, 1975].

Almost six years later, in 1981, Dr. Paul Chodas, the current manager of NASA's Near-Earth Object office at the Jet Propulsion Laboratory in Pasadena, California [NASA, 2015], published an article in which he presents the USM in a manner which facilitates its implementation. However, the author claims that the original USM dynamics equations from Altman are not entirely correct [Chodas, 1981]. Chodas corrected this and his correction has been confirmed by Vivek Vittaldev in his Master's thesis [Vittaldev, 2010]. Between Chodas and Vittaldev only one more article on the USM has been published, this time by J. Raol and N. Sinha [Raol and Sinha, 1985]. In the article an orbit determination was carried out using the USM model as given in [Chodas, 1981]. More recently, in 2010, TU Delft Master student Vivek Vittaldev picked up the USM again and explains in detail how the USM should be implemented [Vittaldev, 2010].

The complete set of USM elements is the following:

$$\boldsymbol{x} = \begin{bmatrix} C & R_{f1} & R_{f2} & \epsilon_{O1} & \epsilon_{O2} & \epsilon_{O3} & \eta_O \end{bmatrix}^T \tag{3.1}$$

The meaning of each of the seven elements will be provided in the following sections.

3.3. HODOGRAPH PARAMETERS

The orbital velocity vector of a spacecraft can be decomposed in many different vectors. To arrive at the first three elements of the Unified State Model, the velocity vector is composed into an R component and a C component, as shown in Fig. 3.1.

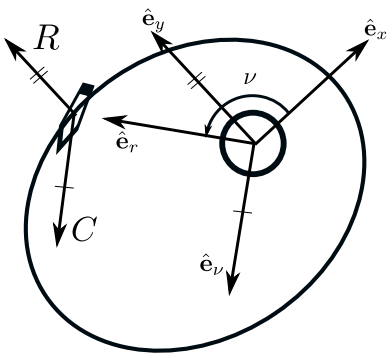


Figure 3.1: Definition of the R and C velocity components in the two-dimensional orbital plane. Note that ν denotes the true anomaly θ. [Vittaldev, 2010]

As can be seen from the figure, R points in the direction of $\hat{\boldsymbol{e}}_y$ which is in turn perpendicular to $\hat{\boldsymbol{e}}_x$ pointing in the direction of the orbit's pericenter. C points in the angular velocity direction $\hat{\boldsymbol{e}}_\nu$ which is perpendicular to the radius vector. R and C are called velocity hodograph components because they define the velocity hodograph of an orbit, shown in Fig. 3.2. R represents the distance from the origin to the center of the velocity hodograph circle, while C represents the radius of the circle. For its use as a USM elements R is decomposed into two components R_{f1} and R_{f2} along the $\hat{\boldsymbol{f}}_1$ and $\hat{\boldsymbol{f}}_2$ axes in Fig. 3.3. Due to the combination of R_{f1}, R_{f2} and C it is possible to unambiguously define the shape and size of a two-dimensional orbit. The orientation of the two-dimensional orbit in three-dimensional space as well as the location of a spacecraft along the orbit is defined by a quaternion, as will be explained in the next section.

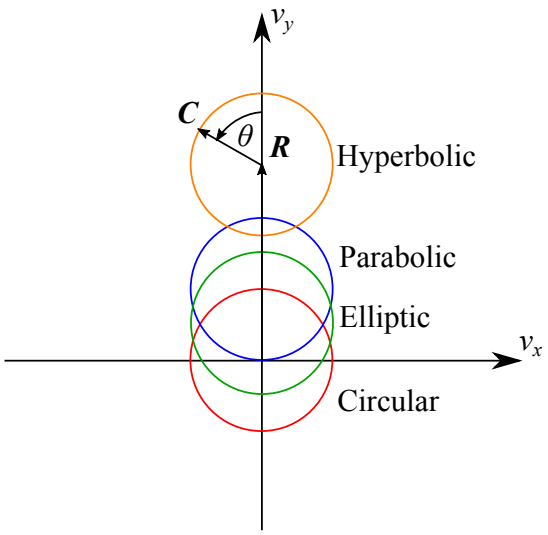


Figure 3.2: Velocity hodograph for various types of orbits in the v_x-v_y plane (see Fig. 3.1 for the definition of the x and y direction), based on [Vittaldev, 2010]

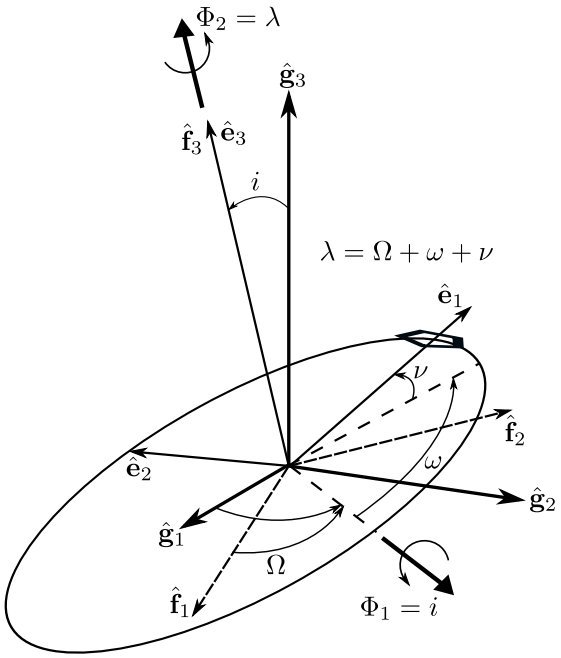


Figure 3.3: Definition of the $\hat{\boldsymbol{f}}_1$ and $\hat{\boldsymbol{f}}_2$ axes and the relationship between the inertial reference frame $\mathcal{F}_g$, the orbital frame $\mathcal{F}_e$ and the $\mathcal{F}_f$ frame. Note that axes $\hat{\boldsymbol{e}}_1$ and $\hat{\boldsymbol{e}}_2$ are identical to axes $\hat{\boldsymbol{e}}_r$ and $\hat{\boldsymbol{e}}_v$ respectively in Fig. 3.1. Note that ν denotes the true anomaly θ in the figure. Adapted from [Vittaldev, 2010]

3.4. QUATERNION

A quaternion is a four dimensional hyper-complex number. This means that it consists of one real number η and an imaginary part containing three elements $\epsilon_1 i$, $\epsilon_2 j$ and $\epsilon_3 k$. Quaternions were invented in 1843 by William Rowan Hamilton in his search for a method to multiply three-dimensional vectors [Hamilton, 1844]. The three imaginary numbers have the following properties:

$$i^2 = j^2 = k^2 = ijk = -1 \tag{3.2}$$

$$\begin{array}{lll} ij = k & jk = i & ki = j \\ ji = -k & kj = -i & ik = -j \end{array} \tag{3.3}$$

And for the norm of the quaternion, the following holds:

$$\sqrt{\epsilon_1^2+\epsilon_2^2+\epsilon_3^2+\eta^2}=1 \tag{3.4}$$

The properties in Eq. (3.3) are best known to engineers when taking the cross product of three dimensional vectors. The property in Eq. (3.4) shows that a quaternion has a unit magnitude. Furthermore, in the same way that complex numbers can represent a rotation in the complex plane, see Fig. 3.4, quaternions can represent a rotation in three dimensional space. This allows for the quaternion to be used as an Euler parameter for the representation of a rotation between two three-dimensional reference frames [Vittaldev, 2010]. Usually an Euler parameter contains three elements, however since the quaternion consists of four elements, it does not have singularities. The imaginary part of a quaternion $\boldsymbol{\epsilon} = [\epsilon_1, \epsilon_2, \epsilon_3]$ represents the three-dimensional vector around which a vector, or in this case a reference frame, will be rotated. The real part η represents the amount of rotation. More specifically:

$$\eta = \cos\left(\frac{\theta}{2}\right) \tag{3.5}$$

$$\boldsymbol{\epsilon} = [\epsilon_1, \epsilon_2, \epsilon_3] = \sin\left(\frac{\theta}{2}\right)\hat{\boldsymbol{a}} \tag{3.6}$$

in which θ is the Euler angle of rotation and $\hat{\boldsymbol{a}}$ is the Euler axis of rotation. In the USM, the quaternion is used to define the orientation of a spacecraft-fixed reference frame with respect to the inertial frame. Therefore the quaternion defines both the orientation of the orbital plane in three dimensional space (defined by $\mathcal{F}_e$ in Fig. 3.3) with respect to the inertial frame ($\mathcal{F}_g$ in Fig. 3.3) as well as the location of the spacecraft along the orbit. To comply with the notation used in literature, the quaternion elements of the USM are indicated with the subscript O which indicates that the orientation is described with respect to the inertial frame. The state of the spacecraft can thus be represented by the following USM state vector:

$$\boldsymbol{x} = \begin{bmatrix} C & R_{f1} & R_{f2} & \epsilon_{O1} & \epsilon_{O2} & \epsilon_{O3} & \eta_O \end{bmatrix}^T \tag{3.7}$$

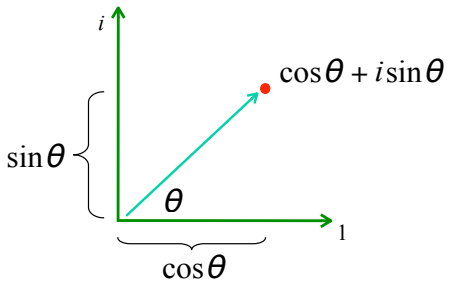


Figure 3.4: A vector in the complex plane with one real and one imaginary axis, can represent a rotation in two-dimensions. The same concept holds for a quaternion in three-dimensional space. [Van Verth, 2013]

3.5. VARIATION OF USM ELEMENTS

It is important to find out how the USM elements change during an orbit. Similar to the Gauss planetary equations, the equations below express the variation in the USM elements with time.

Using USM elements as the state vector for the propagation of low-thrust trajectories requires an expression for the variation of the USM state with time similar to the Gauss planetary equations. Given in [Vittaldev et al., 2010] is the variation of USM elements with time under the influence of a perturbing force in a right-handed reference frame with components along the position vector ($\hat{e}_1$), perpendicular to the position vector and in the orbital plane ($\hat{e}_2$), and along the positive orbit normal ($\hat{e}_3$). The components are identical to the x, y and z axes of the RTN frame defined in Section 2.3.3.

$$\frac{dC}{dt} = -pf_{e2} \tag{3.8a}$$
$$\frac{dR_{f1}}{dt} = f_{e1}\cos\lambda - f_{e2}(1+p)\sin\lambda - f_{e3}l\frac{R_{f2}}{v_{e2}} \tag{3.8b}$$
$$\frac{dR_{f2}}{dt} = f_{e1}\sin\lambda + f_{e2}(1+p)\cos\lambda + f_{e3}l\frac{R_{f1}}{v_{e2}} \tag{3.8c}$$
$$\frac{d\epsilon_{O1}}{dt} = \frac{1}{2}(\omega_3\epsilon_{O2} + \omega_1\eta_O) \tag{3.8d}$$
$$\frac{d\epsilon_{O2}}{dt} = \frac{1}{2}(-\omega_3\epsilon_{O1} + \omega_1\epsilon_{O3}) \tag{3.8e}$$
$$\frac{d\epsilon_{O3}}{dt} = \frac{1}{2}(-\omega_1\epsilon_{O2} + \omega_3\eta_O) \tag{3.8f}$$
$$\frac{d\eta_O}{dt} = \frac{1}{2}(-\omega_1\epsilon_{O1} - \omega_3\epsilon_{O3}) \tag{3.8g}$$

With

$$p = \frac{C}{v_{e2}} \tag{3.9}$$
$$l = \frac{\epsilon_{O1}\epsilon_{O3} - \epsilon_{O2}\eta_O}{\epsilon_{O3}^2 + \eta_O^2} \tag{3.10}$$
$$\omega_1 = \frac{f_{e3}}{v_{e2}} \tag{3.11}$$
$$\omega_3 = \frac{Cv_{e2}^2}{\mu} \tag{3.12}$$
$$\cos\lambda = \frac{\eta_O^2 - \epsilon_{O3}^2}{\epsilon_{O3}^2 + \eta_O^2} \tag{3.13}$$
$$\sin\lambda = \frac{2\epsilon_{O3}\eta_O}{\epsilon_{O3}^2 + \eta_O^2} \tag{3.14}$$
$$v_{e2} = C - R_{f1}\sin\lambda + R_{f2}\cos\lambda \tag{3.15}$$

The thrust acceleration components in Eq. (3.8) f_{e1}, f_{e2} and f_{e3} can be computed from the thrust components in the velocity frame (see Section 2.3.2) with the use of the following equation:

$$\begin{bmatrix} f_{e1} \\ f_{e2} \\ f_{e3} \end{bmatrix} = \begin{bmatrix} f_x\sin\gamma - f_y\cos\gamma \\ f_x\cos\gamma + f_y\sin\gamma \\ f_z \end{bmatrix} \tag{3.16}$$

4

TAYLOR SERIES INTEGRATION

With the Unified State Model (USM) fully introduced in the previous chapter, the Taylor Series Integration (TSI) and its combination with the USM to form the TSI-USM propagator can be explained in this chapter. After this final chapter of theory, Chapter 5 will address the development of a method to generalize the TSI-USM propagator so that it can propagate the trajectory of a spacecraft subject to any type of thrust profile to allow its use in evolutionary algorithms for the optimization of low-thrust trajectories.

Contents

4.1. INTRODUCTION

This chapter will explain the Taylor Series (TSI) method and how it can be combined with the Unified State Model (USM) to form the TSI-USM propagator. After an introduction to Taylor series in Section 4.2. The technique of using recurrence relations to compute the terms of the Taylor series in a recursive way to allow variable order TSI will be explained in Section 4.3. This section will also contain the derivation of the recurrence relations of a simple example before addressing the derivation of the recurrence relations for the USM in Section 4.4. Finally Section 4.5 will discuss the limitations of the TSI and the areas of concern when using TSI.

4.2. TAYLOR SERIES

To understand TSI, one first has to understand Taylor series. Taylor series are used to represent a function by an infinite sum of terms. The infinite series of terms is calculated from the derivatives of the function at a single point. In practice the series is cut off at a certain order K, leaving a finite series of $K+1$ terms which approximates the original function. The difference between the Taylor series approximation and the original function is called the truncation error ε_K. The general definition of a Taylor series of a function f around at $x = a$ is the following:

$$f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \ldots \tag{4.1}$$
$$= \sum_{k=0}^{\infty} \frac{f^{(k)}(a)}{k!}(x-a)^k \tag{4.2}$$
$$= \sum_{k=0}^{K} \frac{f^{(k)}(a)}{k!}(x-a)^k + \varepsilon_K \tag{4.3}$$
$$= g(x) + \varepsilon_K \tag{4.4}$$

The higher the order K, the better $g(x)$ will approximate $f(x)$ in the neighborhood of $x = a$.
Next is an example in which a fifth-order function will be approximated by a third order Taylor series at $x = 0$. The function is:

$$f(x) = \frac{1}{25}x^5 + x^2 - x + 2 \tag{4.5}$$

With time derivatives

$$f'(x) = \frac{1}{5}x^4 + 2x - 1 \tag{4.6}$$
$$f''(x) = \frac{4}{5}x^3 + 2 \tag{4.7}$$
$$f'''(x) = \frac{12}{5}x^2 \tag{4.8}$$

Now, the Taylor series approximation $g_1(x)$ of order $K = 3$ for the above function evaluated at $x = 0$ is:

$$f(x) = f(0) + f'(0)(x-0) + \frac{f''(0)}{2!}(x-0)^2 + \frac{f'''(0)}{3!}(x-0)^3 + \varepsilon_3 \tag{4.9a}$$
$$= 2x^2 - x + 2 + \varepsilon_3 \tag{4.9b}$$
$$g_1(x) = 2x^2 - x + 2 \tag{4.9c}$$

Evaluating the Taylor series approximation at a different point, e.g. at $x = 1$, gives:

$$f(x) = f(1) + f'(1)(x-1) + \frac{f''(1)}{2!}(x-1)^2 + \frac{f'''(1)}{3!}(x-1)^3 + \varepsilon_3 \tag{4.10a}$$
$$= \frac{51}{25} + \frac{6}{5}(x-1) + \frac{14}{5}(x-1)^2 + \frac{12}{5}(x-1)^3 + \varepsilon_3 \tag{4.10b}$$
$$= \frac{1}{5}\left(12x^3 - 22x^2 + 14x + \frac{31}{5}\right) + \varepsilon_3 \tag{4.10c}$$
$$g_2(x) = \frac{1}{5}\left(12x^3 - 22x^2 + 14x + \frac{31}{5}\right) \tag{4.10d}$$

As can be seen in Fig. 4.1, g_1 approximates $f(x)$ well in the neighborhood of $x = 0$. When taking a time step of size 1 s (referred to as the *step-size*)and moving to $x = 1$, g_2 is a better approximation. Imagine that it is required to continue approximating $f(x)$ for increasing x until $x = 10$. This would require nine more approximating g-functions when the same step-size of 1 s is used.

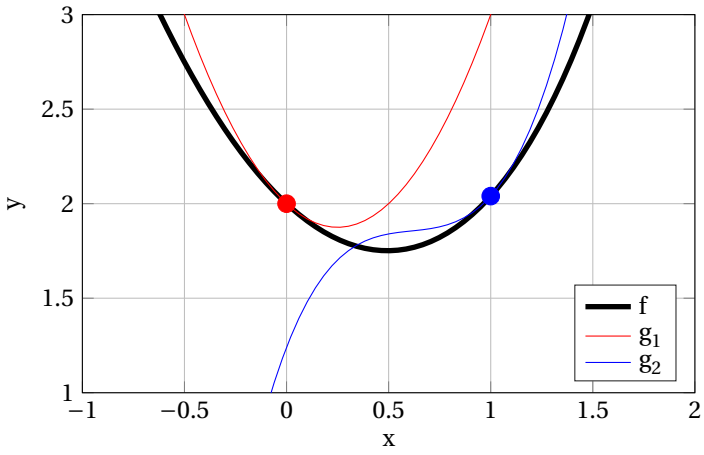


Figure 4.1: Taylor Series approximation of a function at two different points

When for a certain function f the step-size between the approximating g functions is sufficiently small, the consecutive g functions can be used as a good approximation for the entire function, as long as they are used in the small neighborhood in which they are a good approximation of f. If the function value of the point around which the Taylor series is expanded (i.e. $f(a)$) is given, then from Eq. (4.1) it can be seen that this method does not require the original function to be known, it only requires the derivatives of the original function to be known. This makes this method very useful for integration of differential equations. Surely, differential equations in Eq. (3.8) express the derivative of the state variables with respect to time. Integration of these differential equations using Taylor series allows to formulate an expression for the state variable as a function of time. If a high order Taylor series is chosen, then $g_1(x)$ will approximate $f(x)$ very well in the neighborhood of $x = 0$. Therefore, integration of differential equations using Taylor series can be very accurate and the accuracy depends on the amount of computed terms in the series. The accuracy decreases strongly when moving away from the point at which the Taylor series has been evaluated. If we move outside of the neighborhood of this point, and the accuracy becomes insufficient, a new point is picked at which a new Taylor series is computed. The difference between the time-coordinate of the new point and the previous points is called a step-size. As a result of this method, many approximating g-functions have to be computed, i.e. one for every step. If the computation of those functions can be programmed efficiently, it allows for a fast and accurate integration, called Taylor Series Integration.

4.3. TAYLOR SERIES INTEGRATION

Ideally, the equations representing the state variables as a function of time are available. However in reality, there is only information available on the forces acting on the spacecraft. Therefore, there is only information available on the acceleration of the spacecraft, i.e. the derivative of the velocity and the second order derivative of the position. The set of equations describing the derivative of the state variables are differential equations. One way to compute the variation of the state variables with respect to time, is to integrate the differential equations. The integration cannot be done analytically for complex problems. The fourth order Runge-Kutta (RK4) integrator is an example of a numerical integrator. As seen in the previous section, another way to compute the state variables is to expand the unknown equations for the state variables in Taylor series. The components of the Taylor series can be calculated using the differential equations, which are known. This results in an approximation of the unknown function describing the state variable w.r.t. time, that is valid in the neighborhood of a specific time corresponding to a specific value of the state variables. When the Taylor series is used to compute the state variables at the next time-step, a new Taylor series can be set up around the last obtained state variables. This procedures can be repeated until the values of the state variables at the final time are known. The process is known as *Taylor Series Integration*.

The advantage of Taylor Series Integration is that when its order of approximation is sufficiently high, the approximation becomes very good and the step-size between two consecutive Taylor series expansions can be large. So, although computing a single step of in a TSI algorithm usually requires more CPU time than that of an RK4 integrator, its overall CPU time can be lower due to larger step-sizes. According to [Vittaldev, 2010]

"TSI is a very effective technique to accurately integrate dynamic equations with very little CPU time. The required CPU time is at least 1 to even 3 orders of magnitude less than traditional integration with RK methods.". The downside of TSI is that it is more difficult to implement. This section will explain how the Taylor series can be derived using the differential equations of the USM elements from Chapter 3 and how the Taylor series will be used to propagate the perturbed trajectory of the spacecraft.

In the previous section it has been explained that the Taylor series of a function has to be computed at a specific point. When propagating the trajectory of a spacecraft, this point moves continuously. Therefore, the Taylor series of a state variable should be recomputed at every time step of the integrator. A Taylor series consisting of a many terms, ensures a high accuracy and allows for large step-sizes. However, the computation of the Taylor series requires a lot of effort. A solution to this is the use of the recurrence relations. These are general relations that can be used to compute the Taylor series easily at different points. Setting up the recurrence relations for calculating the derivatives of the Taylor series requires a lot of work, but once it is done the computation of the Taylor series at different points becomes relatively simple. Prior to providing the recurrence relations for the set of differential equations given in Chapter 3, the theory behind the recurrence relations as well as a simple example will be given in the next two sections.

4.3.1. RECURRENCE RELATIONS THEORY

As explained in the previous section, a Taylor series has to be computed for each state variable at every step of the integration. This means that the derivatives that are part of the Taylor series need to be computed a large number of times. To this in a time efficient way, it is important to simplify these computations as much as possible. Simplifying the computations allows the recursive computation of the derivatives. Here is where recurrence relations come in. Recurrence relations are needed to provide a general calculation scheme to compute the coefficients of a Taylor series. The idea is to use auxiliary variables to rewrite the differential equations into simple multiplications, divisions, additions and subtractions. Problem independent recurrence relations exist for those basic mathematical operations. The simplification process of the differential equations for the USM elements and the computation of higher order derivatives using recurrence relations will be explained in this section. First, some notation standards used throughout this section will be introduced.

The following mathematical notation is adapted from [Scott and Martini, 2008]. In all of the below expressions, the function for the n^{th} state variable is w.r.t. to time, so that: $x_n = x_n(t)$. The k^{th} derivative of the n^{th} state variable is written as:

$$x_n^{(k)} = \frac{d^k x_n}{dt^k} \tag{4.11}$$

Now $u_n = x_n' = \frac{dx_n}{dt}$ is introduced. And the *normalized derivatives* X_n and U_n as a function of the order k are defined as:

$$X_n(k) = x_n^{[k]} = \frac{x_n^{(k)}}{k!} \tag{4.12}$$

$$U_n(k) = u_n^{[k]} = \frac{u_n^{(k)}}{k!} \tag{4.13}$$

So that:

$$x_n^{(k)} = u_n^{(k-1)} \quad \text{with} \quad k \geq 1 \tag{4.14}$$

$$\iff \frac{x_n^{(k)}}{k(k-1)!} = \frac{u_n^{(k-1)}}{k(k-1)!} \tag{4.15}$$

$$\iff \frac{x_n^{(k)}}{k!} = X_n(k) = \frac{1}{k} U_n(k-1) \tag{4.16}$$

The recurrence relations for three basic mathematical operations are shown below. These relations will be used in Sections 4.4.1 and 4.4.2. They are:

$$w_{\text{mult}} = x_m x_n \Longrightarrow W_t(k) = x_m \frac{U_n(k-1)}{k} + x_n \frac{U_m(k-1)}{k} + \sum_{j=1}^{k-1} \left(\frac{U_m(j-1)}{j} \frac{U_n(k-j-1)}{k-j} \right) \tag{4.17}$$

$$w_{\text{div}} = \frac{x_m}{x_n} \Longrightarrow W_u(k) = \frac{1}{x_n} \left[\frac{U_m(k-1)}{k} - \sum_{j=1}^{k} \left(\frac{U_n(j-1)}{j} W_u(k-j) \right) \right] \tag{4.18}$$

$$w_{\text{multdiv}} = \frac{x_n u_m}{x_m} \Longrightarrow W_v(k) = \frac{1}{x_m} \left[x_n U_m(k) + \sum_{j=1}^{k} \left(\frac{U_m(j-1)}{j} U_m(k-j) \right) - \sum_{j=1}^{k} \left(\frac{U_m(j-1)}{j} W_v(k-j) \right) \right] \tag{4.19}$$

In the above equations, W is the normalized derivative of w. w itself should be seen as an auxiliary variable that is introduced to simplify an equation. For instance, whenever $\frac{x_n u_m}{x_m}$ appears in an equation, it can be replaced by just w_{multdiv}, and its k^{th} derivative will be defined by Eq. (4.19).

Using the notation presented earlier in this chapter, the expression for the Taylor Series of the n^{th} state variable $x_n = x_n(t)$ around $t = t_0$ is:

$$x_n = x_n(t) \approx \sum_{k=0}^{K} x_n^{[k]}(t_0)(t-t_0)^k \tag{4.20a}$$

$$\approx \sum_{k=0}^{K} X_n(k)(t-t_0)^k \tag{4.20b}$$

$$\approx \underbrace{x_n(t_0)}_{k=0} + \underbrace{\sum_{k=1}^{K} \frac{1}{k} U_n(k-1)(t-t_0)^k}_{k \geq 1} \tag{4.20c}$$

It is important to notice that in Eq. (4.20c) only the value of the previous normalized derivative $U_n(k-1)$ is required to calculate the k^{th} term in the Taylor series expansion, hence the name *recurrence relations*.

In the following sections the complexity increases gradually. The following section contains the derivation of the recurrence relations for a simple example problem. Next, there is a section on the derivation of the recurrence relations for the USM without perturbing forces and finally, the last section contains the detailed description of the TSI-USM combination for the propagation of the trajectory of a spacecraft subject to perturbing thrust force.

4.3.2. SIMPLE EXAMPLE

This section contains the derivation of the recurrence relations for a simple example problem. Despite it being a simple problem, it nevertheless contains every aspect of the more complex recurrence relations that are to come in the following sections. Therefore, this section can also be regarded as a summary of the derivations in the following sections.

Imagine a dynamic model where the state of a spacecraft is described by two variables x_1 and x_2. The time variation of the two state variables x_1 and x_2 are described by the following differential equations:

$$\frac{dx_1}{dt} = x_1 x_2 + f_x \tag{4.21a}$$

$$\frac{dx_2}{dt} = x_1^2 + 2x_2^2 + f_y \tag{4.21b}$$

in which f_x and f_y are perturbing accelerations in the x and y direction of a spacecraft-centered orthogonal frame. With the use of the notation $u_n = x_n' = \frac{dx_n}{dt}$ with n denoting the state variable, Eq. (4.21) becomes:

$$u_1 = x_1 x_2 + f_x \tag{4.22a}$$

$$u_2 = x_1^2 + 2x_2^2 + f_y \tag{4.22b}$$

Let us now simplify the above equations into basic multiplications and additions by carefully using auxiliary variables w_n:

$$w_{1,1} = x_1 x_2 \tag{4.23a}$$
$$w_{1,2} = f_x \tag{4.23b}$$
$$w_{2,1} = x_1 x_1 \tag{4.23c}$$
$$w_{2,2} = x_2 x_2 \tag{4.23d}$$
$$w_{2,3} = f_y \tag{4.23e}$$

The normalized derivatives $W_{1,1}$, $W_{2,1}$ and $W_{2,2}$ of $w_{1,1}$, $w_{2,1}$ and $w_{2,2}$ can be determined using Eq. (4.17). Note that when $k = 1$, then $U_n(k-1) = \frac{dx_n}{dt}$ which can be calculated from Eq. (4.21). To obtain the normalized derivatives of $w_{1,2}$ and $w_{2,3}$, the derivatives of the perturbing accelerations are required:

$$W_{1,2}(k) = \frac{f_x^{(k)}}{k!} \tag{4.24a}$$
$$W_{2,3}(k) = \frac{f_y^{(k)}}{k!} \tag{4.24b}$$

The normalized derivatives of the state variables now are:

$$U_1(k) = W_{1,1}(k) + W_{1,2}(k) \tag{4.25a}$$
$$U_2(k) = W_{2,1}(k) + 2W_{2,2}(k) + W_{2,3}(k) \tag{4.25b}$$

The Taylor series expansion of each variable can now be used to obtain the value of the state variable as a function of time, which allows propagating the trajectory. Unfortunately the Taylor series expansion around the initial (known) state at t_0 gets less accurate for increasing time. Therefore after some later time $t_1 > t_0$, a new Taylor series expansion has to be executed. The difference between t_1 and t_0 is defined as the initial step-size of the TSI $h_0 = t_1 - t_0$. The step-size does not need to be constant. When the orbit varies slowly with time, the step-size can be large and vice versa. The Taylor series approximation of $x_1(t)$ and $x_2(t)$ for $t \in [t_0, t_1)$ is:

$$x_1(t) = x_1(t_0) + \sum_{k=1}^{K} \frac{1}{k} U_1(k-1)(t-t_0)^k \tag{4.26a}$$
$$x_2(t) = x_2(t_0) + \sum_{k=1}^{K} \frac{1}{k} U_2(k-1)(t-t_0)^k \tag{4.26b}$$

Similarly, for $t = [t_1, t_2]$ the Taylor series expansion is:

$$x_1(t) = x_1(t_1) + \sum_{k=1}^{K} \frac{1}{k} U_1(k-1)(t-t_1)^k \tag{4.27a}$$
$$x_2(t) = x_2(t_1) + \sum_{k=1}^{K} \frac{1}{k} U_2(k-1)(t-t_1)^k \tag{4.27b}$$

Note that every new Taylor series expansion requires the recalculation of the U_n's. This is because the initial conditions have changed from $x_n(t_0)$ to $x_n(t_1)$ meaning that all the derivatives have to be recalculated at t_1. The computation of the derivatives of the perturbing accelerations f_x and f_y requires an additional explanation. These derivatives are needed to in Eq. (4.24). To allow any thrust acceleration as input, it is decided to make use of a discrete profile for the perturbing accelerations. An example of an acceleration profile is:

Table 4.1: Example profile of the perturbing accelerations

t [s]	f_x [m/s^2]	f_y [m/s^2]
0	1.0	2.5
1	0.5	5.0
2	3.2	8.6
3	2.5	7.3
4	5.2	2.0

Several methods exist to determine the derivatives of a discrete profile. As will be described in the next chapter, the developed method to generalize the TSI will make use of the coefficients of an interpolating polynomial to calculate the derivatives of the thrust accelerations.

In the following two sections, the recurrence relations for the USM state derivatives are derived in way similar to the example problem.

4.4. RECURRENCE RELATIONS FOR THE USM

As explained before, the use of recurrence relations provides a general calculation scheme to compute the derivatives in the Taylor series. For this purpose, the differential equations expressed in USM elements that describe a perturbed motion of a spacecraft in a central body gravitational field should first be reduced to simple multiplications, divisions, additions and subtractions. This simplification can be achieved by introducing auxiliary variables and adding some of these variables to the state vector. For this purpose, the mass of the spacecraft m, the variable ω_3 (a variable that is added to the state vector for the purpose of improving the computational efficiency) and velocity components v_{e1} and v_{e3} are added resulting in the following set of eleven state variables:

$$x_1 = C \tag{4.28a}$$
$$x_2 = R_{f1} \tag{4.28b}$$
$$x_3 = R_{f2} \tag{4.28c}$$
$$x_4 = \epsilon_{O1} \tag{4.28d}$$
$$x_5 = \epsilon_{O2} \tag{4.28e}$$
$$x_6 = \epsilon_{O3} \tag{4.28f}$$
$$x_7 = \eta_O \tag{4.28g}$$
$$x_8 = m \tag{4.28h}$$
$$x_9 = v_{e1} \tag{4.28i}$$
$$x_{10} = v_{e2} \tag{4.28j}$$
$$x_{11} = \omega_3 \tag{4.28k}$$

Next, the governing differential equations are [Vittaldev et al., 2012]:

$$\frac{dC}{dt} = -p f_{e2} \tag{4.29a}$$
$$\frac{dR_{f1}}{dt} = f_{e1}\cos\lambda - f_{e2}\left(1+p\right)\sin\lambda - f_{e3}\, l \frac{R_{f2}}{v_{e2}} \tag{4.29b}$$
$$\frac{dR_{f2}}{dt} = f_{e1}\sin\lambda + f_{e2}\left(1+p\right)\cos\lambda + f_{e3}\, l \frac{R_{f1}}{v_{e2}} \tag{4.29c}$$
$$\frac{d\epsilon_{O1}}{dt} = \frac{1}{2}\left(\omega_3\epsilon_{O2} + \omega_1\eta_O\right) \tag{4.29d}$$
$$\frac{d\epsilon_{O2}}{dt} = \frac{1}{2}\left(-\omega_3\epsilon_{O1} + \omega_1\epsilon_{O3}\right) \tag{4.29e}$$
$$\frac{d\epsilon_{O3}}{dt} = \frac{1}{2}\left(-\omega_1\epsilon_{O2} + \omega_3\eta_O\right) \tag{4.29f}$$
$$\frac{d\eta_O}{dt} = \frac{1}{2}\left(-\omega_1\epsilon_{O1} - \omega_3\epsilon_{O3}\right) \tag{4.29g}$$
$$\frac{dm}{dt} = \frac{-F}{g_0 I_{sp}} \tag{4.29h}$$
$$\frac{dv_{e1}}{dt} = f_{e1} + \omega_3\left(v_{e2} - C\right) \tag{4.29i}$$
$$\frac{dv_{e2}}{dt} = f_{e2} - \omega_3 v_{e1} \tag{4.29j}$$
$$\frac{d\omega_3}{dt} = \frac{\omega_3\dot{C}}{C} + \frac{2\omega_3\dot{v}_{e2}}{v_{e2}} \tag{4.29k}$$

With [Vittaldev et al., 2012]:

$$p = \frac{C}{v_{e2}} \tag{4.30}$$
$$l = \frac{\epsilon_{O1}\epsilon_{O3} - \epsilon_{O2}\eta_O}{\epsilon_{O3}^2 + \eta_O^2} \tag{4.31}$$
$$\omega_1 = \frac{f_{e3}}{v_{e2}} \tag{4.32}$$
$$\omega_3 = \frac{C v_{e2}^2}{\mu} \tag{4.33}$$
$$\cos\lambda = \frac{\eta_O^2 - \epsilon_{O3}^2}{\epsilon_{O3}^2 + \eta_O^2} \tag{4.34}$$
$$\sin\lambda = \frac{2\epsilon_{O3}\eta_O}{\epsilon_{O3}^2 + \eta_O^2} \tag{4.35}$$
$$v_{e1} = R_{f1}\cos\lambda + R_{f2}\sin\lambda \tag{4.36}$$
$$v_{e2} = C - R_{f1}\sin\lambda + R_{f2}\cos\lambda \tag{4.37}$$
$$R_{f1} = v_{e1}\cos\lambda - (v_{e2} - C)\sin\lambda \tag{4.38}$$
$$R_{f2} = v_{e1}\sin\lambda + (v_{e2} - C)\cos\lambda \tag{4.39}$$
$$\dot{\lambda} = l\omega_1 + \omega_3 \tag{4.40}$$

Because it is difficult to integrate Eq. (4.29) directly to find equations of the form $C(t) = \ldots$, $R_{f1}(t) = \ldots$, $R_{f2}(t) = \ldots$ etc., and because the first order derivatives are known, i.e. the equations themselves, it is possible to compute the second, third and higher order derivatives which allow the computation of all the terms that

are required in the Taylor series expansion for C, R_{f1}, R_{f2}, etc. as a function of t. Unfortunately the calculation of the derivatives of Eq. (4.29) with respect to time becomes complicated for higher orders. For example the second order derivative of the Eq. (4.29a) becomes:

$$\frac{d^2C}{dt^2} = -f_{e2}\frac{dp}{dt} - p\frac{df_{e2}}{dt} = -f_{e2}\frac{d}{dt}\frac{C}{v_{e2}} - p\frac{df_{e2}}{dt} \tag{4.41a}$$

$$= f_{e2}\frac{\dot{C}v_{e2} - \dot{v}_{e2}C}{v_{e2}^2} - p\dot{f}_{e2} \tag{4.41b}$$

In the above equation, $\dot{C}$ and $\dot{v}_{e2}$ are known and listed in Eq. (4.29). $\dot{f}_{e2}$ can be found by either differencing two consecutive values of the thrust force or differentiating the underlying function. At this point, the first and second order derivatives of C are known, and the third order derivative can be calculated. Thereafter, the fourth order derivatives should be calculated and so on. Because of the nature of the Taylor series integration, the n^{th} order derivative will never need to be used without the $(n-1)^{th}$ derivative being known. Therefore the expressions for the higher order derivatives can be of recursive nature, hence the recurrence relations.
If the effort of setting up the recurrence relations for the derivatives of the eleven state derivatives is done once, the derivatives can be calculated up to the desired order, which, when implemented in a computer program, allows for a user specified order for the Taylor series approximations.
The recurrence relations for the USM elements without a perturbing force will be derived in the next section.

4.4.1. TSI OF USM STATE WITHOUT PERTURBING FORCE

Before deriving the recurrence relations of the combination of TSI and USM (TSI-USM) for the perturbed case, the case without a perturbing force will be considered first. No perturbing force means a zero thrust acceleration:

$$\boldsymbol{f} = \begin{bmatrix} f_{e1} \\ f_{e2} \\ f_{e3} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.42}$$

In that case Eq. (4.29) becomes:

$$\frac{dC}{dt} = 0 \tag{4.43a}$$

$$\frac{dR_{f1}}{dt} = 0 \tag{4.43b}$$

$$\frac{dR_{f2}}{dt} = 0 \tag{4.43c}$$

$$\frac{d\epsilon_{O1}}{dt} = \frac{1}{2}\omega_3\epsilon_{O2} \tag{4.43d}$$

$$\frac{d\epsilon_{O2}}{dt} = -\frac{1}{2}\omega_3\epsilon_{O1} \tag{4.43e}$$

$$\frac{d\epsilon_{O3}}{dt} = \frac{1}{2}\omega_3\eta_O \tag{4.43f}$$

$$\frac{d\eta_O}{dt} = -\frac{1}{2}\omega_3\epsilon_{O3} \tag{4.43g}$$

$$\frac{dm}{dt} = 0 \tag{4.43h}$$

$$\frac{dv_{e1}}{dt} = \omega_3\left(v_{e2} - C\right) \tag{4.43i}$$

$$\frac{dv_{e2}}{dt} = -\omega_3 v_{e1} \tag{4.43j}$$

$$\frac{d\omega_3}{dt} = \frac{2\omega_3\dot{v}_{e2}}{v_{e2}} \tag{4.43k}$$

Using the notation from Eq. (4.28) and $u_n = \dot{x}_n$, Eq. (4.43) becomes:

$$u_1 = 0 \tag{4.44a}$$
$$u_2 = 0 \tag{4.44b}$$
$$u_3 = 0 \tag{4.44c}$$
$$u_4 = \frac{1}{2} x_5 x_{11} \tag{4.44d}$$
$$u_5 = -\frac{1}{2} x_4 x_{11} \tag{4.44e}$$
$$u_6 = \frac{1}{2} x_7 x_{11} \tag{4.44f}$$
$$u_7 = -\frac{1}{2} x_6 x_{11} \tag{4.44g}$$
$$u_8 = 0 \tag{4.44h}$$
$$u_9 = x_{10} x_{11} - x_1 x_{11} \tag{4.44i}$$
$$u_{10} = -x_9 x_{11} \tag{4.44j}$$
$$u_{11} = 2 \frac{x_{11} u_{10}}{x_{10}} \tag{4.44k}$$

Next, a couple of auxiliary variables are defined to further simplify the derivatives:

$$w_4 = x_5 x_{11} \tag{4.45a}$$
$$w_5 = x_4 x_{11} \tag{4.45b}$$
$$w_6 = x_7 x_{11} \tag{4.45c}$$
$$w_7 = x_6 x_{11} \tag{4.45d}$$
$$w_{9,1} = x_{10} x_{11} \tag{4.45e}$$
$$w_{9,2} = x_1 x_{11} \tag{4.45f}$$
$$w_{10} = x_9 x_{11} \tag{4.45g}$$
$$w_{11} = \frac{x_{11} u_{10}}{x_{10}} \tag{4.45h}$$

Now the normalized derivatives are:

$$U_1(k) = 0 \tag{4.46a}$$
$$U_2(k) = 0 \tag{4.46b}$$
$$U_3(k) = 0 \tag{4.46c}$$
$$U_4(k) = \frac{1}{2} W_4(k) \tag{4.46d}$$
$$U_5(k) = -\frac{1}{2} W_5(k) \tag{4.46e}$$
$$U_6(k) = \frac{1}{2} W_6(k) \tag{4.46f}$$
$$U_7(k) = -\frac{1}{2} W_7(k) \tag{4.46g}$$
$$U_8(k) = 0 \tag{4.46h}$$
$$U_9(k) = W_{9,1}(k) - W_{9,2}(k) \tag{4.46i}$$
$$U_{10}(k) = -W_{10}(k) \tag{4.46j}$$
$$U_{11}(k) = 2 W_{11}(k) \tag{4.46k}$$

At this point it is possible to construct the recurrence relations by the computing Eq. (4.46) using Eq. (4.45) and Eqs. (4.17) to (4.19). In the next section, the recurrence relations for the perturbed case will be derived.

4.4.2. TSI OF USM STATE WITH PERTURBING FORCE

Similar to the unperturbed case, the derivations start with the differential equations for the motion of a spacecraft in a central gravitational field expressed in USM elements, now with a perturbing thrust acceleration expressed in the Radial Tangential Normal (RTN) frame (see Figure 2.4) $\boldsymbol{f} = [f_{e1}, f_{e2}, f_{e3}]$.

$$\frac{dC}{dt} = -p f_{e2} \tag{4.47a}$$
$$\frac{dR_{f1}}{dt} = f_{e1}\cos\lambda - f_{e2}\left(1+p\right)\sin\lambda - f_{e3}\,k\frac{R_{f2}}{v_{e2}} \tag{4.47b}$$
$$\frac{dR_{f2}}{dt} = f_{e1}\sin\lambda + f_{e2}\left(1+p\right)\cos\lambda + f_{e3}\,l\frac{R_{f1}}{v_{e2}} \tag{4.47c}$$
$$\frac{d\epsilon_{O1}}{dt} = \frac{1}{2}\left(\omega_3\epsilon_{O2} + \omega_1\eta_O\right) \tag{4.47d}$$
$$\frac{d\epsilon_{O2}}{dt} = \frac{1}{2}\left(-\omega_3\epsilon_{O1} + \omega_1\epsilon_{O3}\right) \tag{4.47e}$$
$$\frac{d\epsilon_{O3}}{dt} = \frac{1}{2}\left(-\omega_1\epsilon_{O2} + \omega_3\eta_O\right) \tag{4.47f}$$
$$\frac{d\eta_O}{dt} = \frac{1}{2}\left(-\omega_1\epsilon_{O1} - \omega_3\epsilon_{O3}\right) \tag{4.47g}$$
$$\frac{dm}{dt} = \frac{-F}{g_0 I_{sp}} \tag{4.47h}$$
$$\frac{dv_{e1}}{dt} = f_{e1} + \omega_3\left(v_{e2} - C\right) \tag{4.47i}$$
$$\frac{dv_{e2}}{dt} = f_{e2} - \omega_3 v_{e1} \tag{4.47j}$$
$$\frac{d\omega_3}{dt} = \frac{\omega_3\dot{C}}{C} + \frac{2\omega_3\dot{v}_{e2}}{v_{e2}} \tag{4.47k}$$

Using the notation from Eq. (4.28), the expression in Eq. (4.32), and $u_n = \dot{x}_n$, Eq. (4.47) becomes:

$$u_1 = -p f_{e2} \tag{4.48a}$$
$$u_2 = f_{e1}\cos\lambda - f_{e2}\left(1+p\right)\sin\lambda - f_{e3}\,l\frac{x_3}{x_{10}} \tag{4.48b}$$
$$u_3 = f_{e1}\sin\lambda + f_{e2}\left(1+p\right)\cos\lambda + f_{e3}\,l\frac{x_2}{x_{10}} \tag{4.48c}$$
$$u_4 = \frac{1}{2}x_5 x_{11} + \frac{1}{2}\frac{f_{e3}}{x_{10}}x_7 \tag{4.48d}$$
$$u_5 = -\frac{1}{2}x_4 x_{11} + \frac{1}{2}\frac{f_{e3}}{x_{10}}x_6 \tag{4.48e}$$
$$u_6 = \frac{1}{2}x_7 x_{11} - \frac{1}{2}\frac{f_{e3}}{x_{10}}x_5 \tag{4.48f}$$
$$u_7 = -\frac{1}{2}x_6 x_{11} - \frac{1}{2}\frac{f_{e3}}{x_{10}}x_4 \tag{4.48g}$$
$$u_8 = \frac{-F}{g_0 I_{sp}} \tag{4.48h}$$
$$u_9 = f_{e1} - x_{10}x_{11} - x_1 x_{11} \tag{4.48i}$$
$$u_{10} = f_{e2} - x_9 x_{11} \tag{4.48j}$$
$$u_{11} = \frac{x_{11}u_1}{x_1} + 2\frac{x_{11}u_{10}}{x_{10}} \tag{4.48k}$$

Next, some auxiliary variables are introduced which allow writing Eq. (4.48) in a simpler way [Vittaldev, 2010]. Note that Eqs. (4.30) to (4.35) and (4.37) are used to break down Eq. (4.48) into smaller parts.

$$v_1 = \epsilon_{O3}^2 + \eta_O^2 = x_6^2 + x_7^2 \tag{4.49a}$$
$$v_2 = \epsilon_{O3}\eta_O = x_6 x_7 \tag{4.49b}$$
$$v_3 = \eta_O^2 - \epsilon_{O3}^2 = x_7^2 - x_6^2 \tag{4.49c}$$
$$v_4 = \epsilon_{O1}\epsilon_{O3} - \epsilon_{O2}\eta_O = x_4 x_6 - x_5 x_7 \tag{4.49d}$$
$$v_5 = \sin\lambda = 2\frac{v_2}{v_1} \tag{4.49e}$$
$$v_6 = \cos\lambda = \frac{v_3}{v_1} \tag{4.49f}$$
$$v_7 = l = \frac{v_4}{v_1} \tag{4.49g}$$
$$v_8 = p = \frac{x_1}{x_{10}} \tag{4.49h}$$
$$v_9 = \frac{l}{v_{e2}} = \frac{v_7}{x_{10}} \tag{4.49i}$$
$$v_{10} = \frac{R_{f1} l}{v_{e2}} = x_2 v_9 \tag{4.49j}$$
$$v_{11} = \frac{R_{f2} l}{v_{e2}} = x_3 v_9 \tag{4.49k}$$
$$v_{12} = f_{e1} \tag{4.49l}$$
$$v_{13} = f_{e2} \tag{4.49m}$$
$$v_{14} = f_{e3} \tag{4.49n}$$
$$v_{15} = \omega_1 = \frac{v_{14}}{x_{10}} \tag{4.49o}$$
$$v_{16} = f_{e2}(1+p) = v_{13} + v_{13} v_8 \tag{4.49p}$$
$$v_{17} = \frac{f_{e3} l R_{f1}}{v_{e2}} = v_{14} v_{10} \tag{4.49q}$$
$$v_{18} = \frac{f_{e3} l R_{f2}}{v_{e2}} = v_{14} v_{11} \tag{4.49r}$$

with

$$v_{1,1} = x_6 x_6 \tag{4.50a}$$
$$v_{1,2} = x_7 x_7 \tag{4.50b}$$
$$v_{4,1} = x_4 x_6 \tag{4.50c}$$
$$v_{4,2} = x_5 x_7 \tag{4.50d}$$
$$v_{16,1} = v_{13} v_8 \tag{4.50e}$$

With the use of Eqs. (4.17) to (4.19) the normalized derivatives $V_n(k)$ of the auxiliary variables v_n can now be calculated. This works for all v_n's except for v_{12}, v_{13} and v_{14}. From Eq. (4.13) it can be deduced that:

$$V_{12}(k) = \frac{v_{14}^{(k)}}{k!} = \frac{f_{e1}^{(k)}}{k!} \tag{4.51a}$$
$$V_{13}(k) = \frac{v_{13}^{(k)}}{k!} = \frac{f_{e2}^{(k)}}{k!} \tag{4.51b}$$
$$V_{14}(k) = \frac{v_{14}^{(k)}}{k!} = \frac{f_{e3}^{(k)}}{k!} \tag{4.51c}$$

The issue of calculating the k^{th} derivative of the accelerations f_{e1}, f_{e2} and f_{e3} will be handled in the next chapter, as it is part of generalizing the TSI-USM. (see Eq. (5.22)).

Subsequently, with the use of Eq. (4.49), Eq. (4.48) can be written as:

$$u_1 = -v_8 v_{13} \tag{4.52a}$$
$$u_2 = v_{12} v_6 - v_{16} v_5 - v_{18} \tag{4.52b}$$
$$u_3 = v_{12} v_5 + v_{16} v_6 + v_{17} \tag{4.52c}$$
$$u_4 = \frac{1}{2} x_{11} x_5 + \frac{1}{2} v_{15} x_7 \tag{4.52d}$$
$$u_5 = -\frac{1}{2} x_{11} x_4 + \frac{1}{2} v_{15} x_6 \tag{4.52e}$$
$$u_6 = \frac{1}{2} x_{11} x_7 - \frac{1}{2} v_{15} x_5 \tag{4.52f}$$
$$u_7 = -\frac{1}{2} x_{11} x_6 - \frac{1}{2} v_{15} x_4 \tag{4.52g}$$
$$u_8 = \frac{-F}{g_0 I_{sp}} \tag{4.52h}$$
$$u_9 = v_{12} - x_1 x_{11} + x_{10} x_{11} \tag{4.52i}$$
$$u_{10} = v_{13} - x_9 x_{11} \tag{4.52j}$$
$$u_{11} = \frac{x_{11} u_1}{x_1} + 2\frac{x_{11} u_{10}}{x_{10}} \tag{4.52k}$$

Eq. (4.52) can again be simplified using the following variables:

$$w_1 = v_8 v_{13} \tag{4.53a}$$
$$w_{2,1} = v_{12} v_6 \tag{4.53b}$$
$$w_{2,2} = v_{16} v_5 \tag{4.53c}$$
$$w_{3,1} = v_{12} v_5 \tag{4.53d}$$
$$w_{3,2} = v_{16} v_6 \tag{4.53e}$$
$$w_{4,1} = x_{11} x_5 \tag{4.53f}$$
$$w_{4,2} = v_{15} x_7 \tag{4.53g}$$
$$w_{5,1} = x_{11} x_4 \tag{4.53h}$$
$$w_{5,2} = v_{15} x_6 \tag{4.53i}$$
$$w_{6,1} = x_{11} x_7 \tag{4.53j}$$
$$w_{6,2} = v_{15} x_5 \tag{4.53k}$$
$$w_{7,1} = x_{11} x_6 \tag{4.53l}$$
$$w_{7,2} = v_{15} x_4 \tag{4.53m}$$
$$w_{9,1} = x_1 x_{11} \tag{4.53n}$$
$$w_{9,2} = x_{10} x_{11} \tag{4.53o}$$
$$w_{10} = x_9 x_{11} \tag{4.53p}$$
$$w_{11,1} = \frac{x_{11} u_1}{x_1} \tag{4.53q}$$
$$w_{11,2} = \frac{x_{11} u_{10}}{x_{10}} \tag{4.53r}$$

From Eq. (4.53), the normalized derivatives $W_n(k)$ can be determined using Eqs. (4.17) to (4.19). Subsequently, the normalized derivatives $U_n(k)$ of u_n are computed in the following way. Note that Eq. (4.54h)

is obtained by applying the relation $U_n(k) = \frac{u_n^{(k)}}{k!}$ to Eq. (4.52h).

$$U_1(k) = -W_1(k) \tag{4.54a}$$
$$U_2(k) = W_{2,1}(k) - W_{2,2}(k) - V_{18}(k) \tag{4.54b}$$
$$U_3(k) = W_{3,1}(k) + W_{3,2}(k) + V_{17}(k) \tag{4.54c}$$
$$U_4(k) = \frac{1}{2}W_{4,1}(k) + \frac{1}{2}W_{4,2}(k) \tag{4.54d}$$
$$U_5(k) = -\frac{1}{2}W_{5,1}(k) + \frac{1}{2}W_{5,2}(k) \tag{4.54e}$$
$$U_6(k) = \frac{1}{2}W_{6,1}(k) - \frac{1}{2}W_{6,2}(k) \tag{4.54f}$$
$$U_7(k) = -\frac{1}{2}W_{7,1}(k) - \frac{1}{2}W_{7,2}(k) \tag{4.54g}$$
$$U_8(k) = -\frac{F^{(k)}}{g_0 I_{sp} k!} \tag{4.54h}$$
$$U_9(k) = V_{12}(k) - W_{9,1}(k) + W_{9,2}(k) \tag{4.54i}$$
$$U_{10}(k) = V_{13}(k) - W_{10} \tag{4.54j}$$
$$U_{11}(k) = W_{11,1}(k) + 2W_{11,2} \tag{4.54k}$$

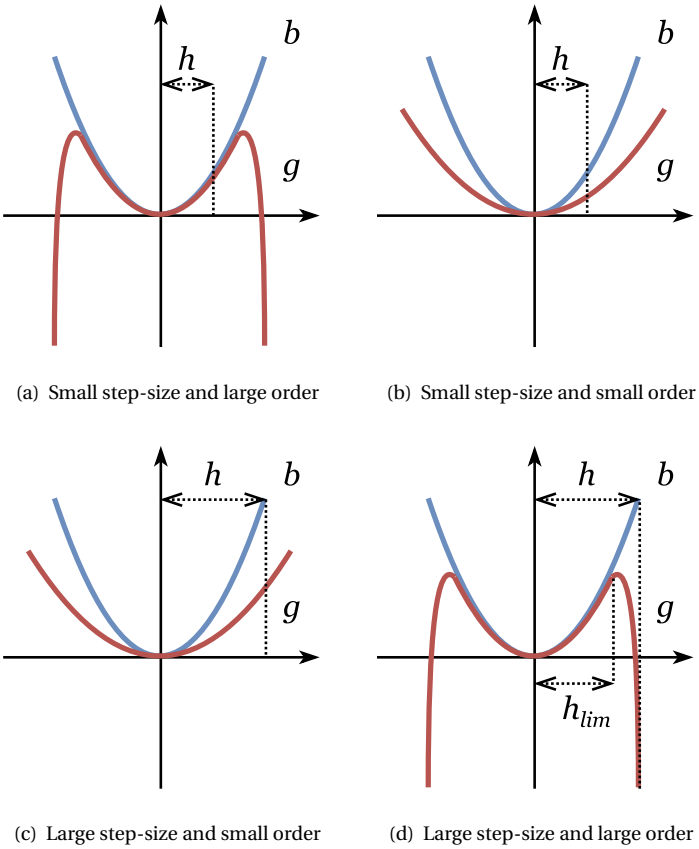
Eq. (4.54h) shows that, other than in previous research on the combination of USM with TSI, the normalized derivatives of the mass $U_8(k)$ should be calculated using the derivatives of the thrust magnitude in order not to pose any constraints on the thrust profile. This ensures that the thrust profile can be arbitrary chosen, i.e. it is not subject to any pre-determined function.

4.5. LIMITATIONS OF TSI

This section will disclose some limitations of TSI. The most important limitation of TSI for its use in evolutionary algorithms is its the problem-specificity. This limitation will be discussed in Section 4.5.2 and forms the rationale for developing a method to generalize the TSI-USM propagator, which will be done in the next chapter. The remnants of the limitations discussed below will be observed in some of the results provided in Chapter 7.

4.5.1. COMBINATION OF LARGE STEP-SIZE AND ORDER

One limitation of TSI-USM and, by extension, every TSI-based propagator, occurs for a combination of a large step-size and a large order of the Taylor series. For this combination of integration settings, the TSI can become very inaccurate to the point that the solution will explode, i.e. the error of the propagated trajectory becomes very large over only a few integration steps. The cause of this limitation can be understand by looking at Fig. 4.2 in which function b is approximated by a Taylor series expansion g around a point on the origin for four different settings. First, in (a) a case is shown for which the Taylor series (TS) approximation is very accurate due to the combination of a large TS order and a small step-size. Secondly, in (b), it can be seen that when the order of the TS is decreased, the approximation becomes less accurate but certainly the difference between the exact function b and the TS approximation g (i.e. the error) does not explode. A similar behavior can be observed for a combination of a small TS order and a large step-size in (c); here the error is larger, but it does not explode. (d) shows a combination of a large step-size with a large order. Although the order is larger then in (c), the approximation is worse for the displayed step-size h. The error has suddenly become very large and exploded. If the step-size would have been smaller than h_{lim}, there would have been no problem. The accuracy would even have been higher than that of case (c). Apparently, for a certain order of a TS approximation, there exists a step-size limit h_{lim} which, when exceeded, causes the error to explode. The value of this step-size limit will vary depending the order. It can be expected that the larger the order, the smaller the step-size limit will be.



(a) Small step-size and large order

(b) Small step-size and small order

(c) Large step-size and small order

(d) Large step-size and large order

Figure 4.2: Illustration of the limitation of TSI for the combination of high order and a large step-size (h). The blue function b is approximated by a TS expansion around point $(0,0)$ represented by the red function g for which the step-size and order are varied.

4.5.2. PROBLEM-SPECIFICITY OF TSI

When looking at Eq. (4.54h) and Eq. (4.51), it can be seen that the derivatives of the thrust accelerations in the RTN frame $\boldsymbol{f} = [f_{e1}, f_{e2}, f_{e3}]$ as well as the derivatives of the magnitude of the thrust force F need to be computed. In order to calculate those derivatives in a recursive manner, as the TSI recurrence relations require, the previous application of the TSI-USM in [Vittaldev, 2010] used a specific function to represent the thrust profile. In [Vittaldev, 2010], the recurrence relations of the function describing the applied thrust acceleration were derived. Although it is possible to represent different thrust profiles with a single function without the need to re-derive the recurrence relations for its derivatives (i.e. with the use of general coefficients that also appear in the recurrence relations), the generality of these functions will be limited. For example, it would be difficult to find a thrust function that can represent both a smooth variation of the thrust acceleration and a non-smooth, discontinuous variation of the thrust acceleration, while only varying the coefficients of that function. And even in the case such a function could be found, it would be complicated to derive its recurrence relations. Therefore, it can be concluded that the TSI-USM propagator is a problem-specific propagation method since a specific set of recurrence relations can only be used for the corresponding specific thrust profile.

In [Vittaldev, 2010], the process of re-deriving the recurrence relations was repeated for several different functions that described the thrust acceleration. The method of re-deriving the recurrence relations for every different thrust function is slow and requires a lot of effort. Furthermore, it is detrimental for the use of the TSI-USM propagator inside an evolutionary algorithm. When optimizing a low-thrust spacecraft trajectory using an evolutionary optimization algorithm, the thrust profile itself is optimized to result in the best possible trajectory. This means that the initial population of the optimization algorithm consists of a range of different low-thrust profiles. The trajectories corresponding to each of these low-thrust profiles are found by a numerical propagator. If that propagator would be the TSI-USM, each of the functions representing a thrust profile would require a separate derivation of the recurrence relations. In practice this would lead to a very

slow optimization process. So, although the TSI-USM propagator is computationally fast, its combination with evolutionary algorithms to optimize a low-thrust trajectory is not.

This is a major limitation of the TSI-USM propagator and of TSI in general. Therefore, in the following chapter, a method will be developed that tries to generalize the TSI-USM so that it can be used in evolutionary algorithms without the need to re-derive the recurrence relations of the derivatives of the thrust accelerations and thrust force. The method will be referred to as the General Discrete Force Model.

5

General Discrete Force Model

From the introduction in Chapter 1 and the discussion of the Taylor Series Integration method in Chapter 4 it followed that the combination of TSI and USM results in a fast propagator for low-thrust trajectories. However, it was also found that the TSI-USM is a problem-specific method which would lead to computationally slow optimization of low-thrust trajectories when used in evolutionary algorithms. In this chapter, a model, called the General Discrete Force Model (GDFM) is developed that provides a solution to this very problem. The GDFM will be implemented in the existing TSI-USM propagator to form a generalized TSI-USM (GTSI-USM) propagator. The contents of this chapter are shown below. In the following Chapter 6, the GTSI-USM as well as three other reference propagators will be verified.

Contents

5.1. Introduction

As was explained in Section 4.5.2 of the previous chapter, the TSI-USM is a problem-specific method. This means that the recurrence relations of the TSI would have to be re-derived for any different thrust profile. This chapter presents the development of a method that would allow TSI to work without the need for adjusting the TSI recurrence relations. The developed method will be referred to as the General Discrete Force Model (GDFM). As will be seen in this chapter, this method involves the interpolation of a pre-set discrete thrust profile. The coefficients of the interpolating function will be used to compute the derivatives of the thrust acceleration which are required in the recurrence relations of the TSI. Several interpolating functions have been considered in Section 5.2 of which three methods are discussed in more detail. The best interpolation method will be selected in Section 5.3. An improved design of the GTSI-USM is presented in Section 5.4 in which the assumption of a constant flight path angle will be corrected. Another design iteration led to a correction of the interpolation of the thrust magnitude, which will be explained in Section 5.5.

For a flow diagram of the program structure of the implemented GTSI-USM, the reader is referred to Appendix C.

5.2. Determination of the Thrust Derivatives

In the previous chapter it has been shown how the recurrence relations of the TSI are derived for the USM. In the last section of that chapter, Section 4.5.2, it has been explained why the TSI is a problem-specific method.

To allow the TSI-USM propagator to work with different types of thrust profiles without the need to adapt the re-derive its recurrence relations, the user should be able to input any given thrust profile in the form of a table or matrix, in which the number of rows n (i.e. one row = one node of the thrust profile) determines the fineness of the thrust profile. The general form of the thrust profile is shown in Table 5.1. Note that it is required that the time values are ranked from smallest to largest, but it is not required that the difference between two consecutive time values is constant: $t_{i+1} - t_i > 0$.

Table 5.1: Example of a general thrust profile. Each row represents one node in the thrust profile.

Time [s]	F_x **[N]**	F_y **[N]**	F_z **[N]**
t_0	$F_x(t_0)$	$F_y(t_0)$	$F_z(t_0)$
t_1	$F_x(t_1)$	$F_y(t_1)$	$F_z(t_1)$
t_2	$F_x(t_2)$	$F_y(t_2)$	$F_z(t_2)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
t_n	$F_x(t_n)$	$F_y(t_n)$	$F_z(t_n)$

The question now is how to compute the derivatives of the thrust force $F^{(k)}$ and the derivatives of the thrust acceleration $f^{(k)}$ from the above thrust profile. It is desired to be able to calculate those derivatives without posing constraints on the thrust profile. Although posing constraints on the thrust force profile is done regularly to solve specific problems, e.g. limit the profile to constant tangential thrust, it would sacrifice its generality. In this thesis a method is developed that allows propagating a general thrust profile. The thrust profile will be general in the sense that the user will be able to fill in a table containing the thrust value in three dimensions of a body-centered frame at specified times, leading to a discrete thrust profile. It is desirable to have the user input the thrust in the velocity frame because this frame has a comprehensible definition. For the recurrence relations, the thrust should then be transformed to the RTN frame, i.e. thrust in e_1-, e_2- and e_3-directions at discrete time-steps (see the definition of the frame in Figs. 2.4 and 3.3). Until now, this functionality has never been implemented before.

The thrust force at every time-step of the integrator can be found by interpolating the thrust input vector. However as the value of $F^{(k)}$ with $k > 0$ is required, the thrust force needs to be differentiated. The coefficients of the interpolating function can be used to determine the derivatives of the thrust force. Several types of interpolating functions are considered and a list is given below. In general, these functions should meet two requirements: (1) the function must be an accurate interpolation of the thrust profile and (2) the function must be differentiable.

List of Interpolation Methods

- **Step function**: This function would consists of a superposition of multiple heaviside step functions [Weisstein, 2017]. The height of every heaviside step function would be chosen such that the overall step function would equal the value of the thrust force, specified in the thrust profile at every part of the domain. The overall step function remains constant around the known point to instantaneously switch to the take the value of the thrust value at the next discrete point in time. A big advantage of using a step function is that its derivatives become zero, hence simplifying the recurrence relations. However, the gain in simplicity comes at a reduction of realism. The low-thrust propulsion system should change its thrust level instantaneously, which is a far from realistic. Another disadvantage is that in case a smooth thrust function is required, a large number of thrust values will be necessary. The use of a step function for the approximation of the thrust force in continuous time will therefore result in trajectories that are not achievable in real life.

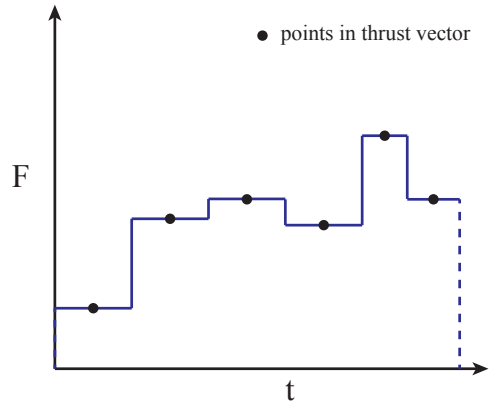


Figure 5.1: Block functions at the points where the thrust is defined cause the overall function to switch values right in between two points

- **Radial basis functions**: Similar to the step function's scenario, here a function is created by superimposing the same amount of radial basis functions as the amounts of points in the thrust vector. A radial basis function is a function that only depends on the distance from the origin [Buhmann, 2003]. Different types of radial basis functions exist to approximate functions. Preferably, the function is easy to differentiate. Unfortunately even the simplest radial basis function, the Gaussian $f(x) = e^{-(er)^2}$, has complex higher order derivatives.

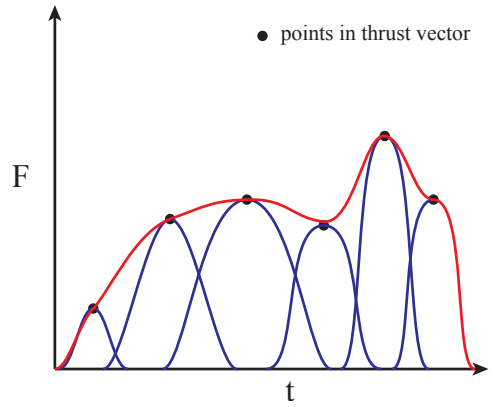


Figure 5.2: Summation of Gaussian radial basis functions at the points where the thrust is defined result in the overall function (red)

- **Lagrange interpolation**: A Lagrange polynomial is a n^{th} order polynomial that goes exactly through a set of $n+1$ points, in this case the thrust value at the discrete time-steps. The advantage of a Lagrange polynomial is that the function can easily be differentiated and that the differentiation shows a recurrent pattern [Hazewinkel, 2001]. Furthermore, depending on the required order of integration, a lot of the derivatives become zero. The first n derivatives of an n^{th} order Lagrange polynomial are non-zero, from the $(n+1)^{\text{th}}$ the derivatives are zero. When the amount of points in the thrust vector is large, so is the order of the approximating Lagrange polynomial. However, it is not needed for a single Lagrange polynomial to go through all the points. When three successive points are approximated by a second order Lagrange polynomial, and the next three successive points by another second order Lagrange polynomial (indicated with the green, red and blue colors in Fig. 5.3), the central parts of each polynomial can be annexed to form a spline. The spline in itself is continuous, but its derivatives are generally not. The discontinuity can cause the second derivative to change sign. An example of a location where this happens is indicated with the arrows in the figure. It is rather difficult to predict the effect of the discontinuous derivatives on the accuracy of the TSI. Therefore this method will be implemented and its accuracy will be compared with Hermite cubic spline interpolation which will be discussed next.

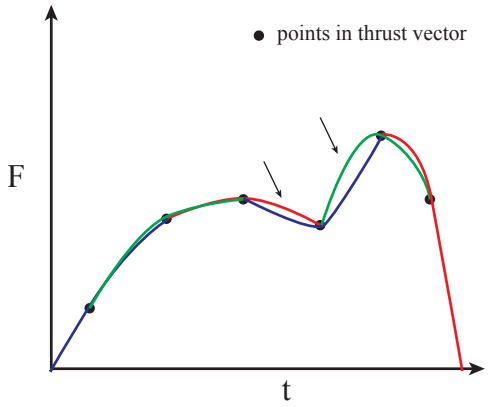


Figure 5.3: Third order Lagrange interpolation through each set of three consecutive points (polynomials have different colors). The center parts of each polynomial are connected to form to overall function (not shown).

- **Hermite interpolation**: Hermite interpolation is a method that is designed such that the interpolating polynomial also matches the first n derivatives at the given points. Hermite polynomials perform better than Lagrange polynomials when a large number of points need to be interpolated. The downside is that the derivatives at the given points need to be known in advance. This is not the case in this application (TSI). The fact that only three consecutive points are being interpolated and that the derivatives at these points are not known, means that Hermite interpolation is not a valid option for this application. Fortunately, it possible to use the concept of Hermite interpolation without the need to define the first n derivatives. This is the case in *cubic spline interpolation*, which will be discussed in more detail later in this chapter [de Boor, 1978; Dwight, 2011].

- **Chebychev interpolation**: Although interpolation using the Chebychev polynomial is very effective in some domains, it is not useful for application in this thesis. The thing is that the approximating polynomial needs to be differentiated recursively many times. The more effective interpolation using Chebychev polynomials does not weigh up against the increased difficulty in computing its derivatives, when compared to Lagrange polynomials [Sarra, 2006].

Of the above listed methods, 2$^{\text{nd}}$ order Lagrange interpolation, 3$^{\text{rd}}$ order Lagrange interpolation and cubic spline interpolation were selected and implemented in the TSI for the purpose of interpolating the thrust profile. Below, the implementation of these three methods will be explained in detail. After implementation of these methods, the interpolation method that leads to the most accurate propagation of a low-thrust trajectory will be selected for implementation into the GDFM in Section 5.3.

5.2.1. SECOND ORDER LAGRANGE INTERPOLATION

The second order Lagrange interpolation is considered because of the combination of its simple derivatives and its sufficiently accurate interpolation. The information from three consecutive rows in the thrust profile will be used to find the coefficients of a 'moving' Lagrange polynomial. The term 'moving' refers to the fact that every three consecutive thrust points result in an a different interpolating Lagrange polynomial. Below, the computation of the derivatives of the thrust force and thrust acceleration from the Lagrange polynomials will be explained.

As mentioned before, the thrust force at discrete points in time is known from the thrust profile. The coefficients of the interpolating Lagrange polynomial can be used to compute the derivatives of the thrust force. To determine the derivatives of the thrust accelerations, the derivatives of the thrust force cannot simply be divided by the mass. Because the mass is a state variable, the mass at the current integration step is known. Furthermore, the current mass flow can be calculated using the current resulting thrust force, which can in turn be determined by Lagrange interpolation using the surrounding three points from the input thrust vector. In the following formulas, the current time, indicated with the subscript 'curr', is the time of the current step of the integrator.

$$\dot{m}_{curr} = -\frac{F_{curr}}{g_0 I_{sp}} \tag{5.1}$$

In the above equation, the current resulting thrust force F_{curr} is not known in general, as the time at the current integration step can fall in between two nodes (e.g. rows) of the thrust profile. Fortunately, the value of the resulting thrust force at the nearest node is known. The nearest node will be referred to as the i^{th} node. This value will be used, leading to the following assumption:

$$\dot{m}_{curr} \approx \dot{m}_i = -\frac{F_i}{g_0 I_{sp}} \tag{5.2}$$

In order to determine the derivatives of the thrust acceleration, Lagrange interpolation is used again. The Lagrange polynomials are created using three consecutive points in time at which the acceleration is known. The thrust acceleration at three consecutive nodes is calculated as follows:

$$\boldsymbol{f}_{i-1} = \frac{\boldsymbol{F}_{i-1}}{m_{curr} + \dot{m}_{curr}(t_{i-1} - t_{curr})} \tag{5.3a}$$

$$\boldsymbol{f}_{i} = \frac{\boldsymbol{F}_{i}}{m_{curr} + \dot{m}_{curr}(t_{i} - t_{curr})} \tag{5.3b}$$

$$\boldsymbol{f}_{i+1} = \frac{\boldsymbol{F}_{i+1}}{m_{curr} + \dot{m}_{curr}(t_{i+1} - t_{curr})} \tag{5.3c}$$

With the thrust acceleration known at three consecutive points, a second order Lagrange polynomial is used to interpolate the thrust acceleration. The derivatives of the thrust acceleration can now be calculated from the coefficients of the interpolating Lagrange polynomial.

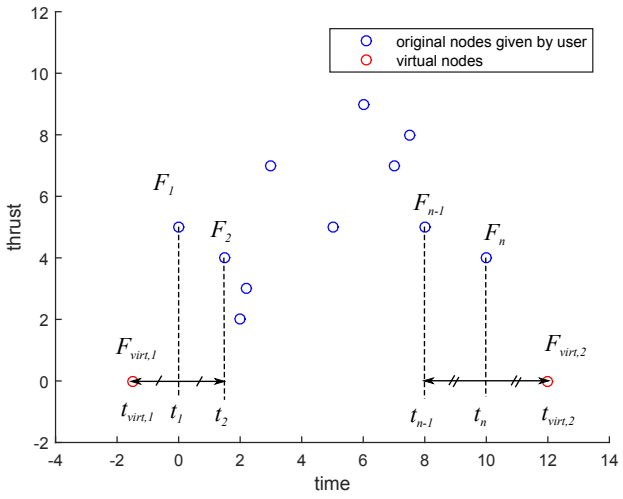
5.2.2. THIRD ORDER LAGRANGE INTERPOLATION

As the third order Lagrange interpolation is expected to perform better than the second order Lagrange polynomial its implementation in the TSI will be described even more detailed. While a second order Lagrange polynomial requires three nodes of the thrust profile to be uniquely defined, a third order Lagrange polynomial requires four nodes. Similar to second order Lagrange inerpolation, the coefficients of the interpolating third order Lagrange polynomial will be used to compute the derivatives of the thrust force and thrust acceleration. The description of this computation is subdivided into four steps.

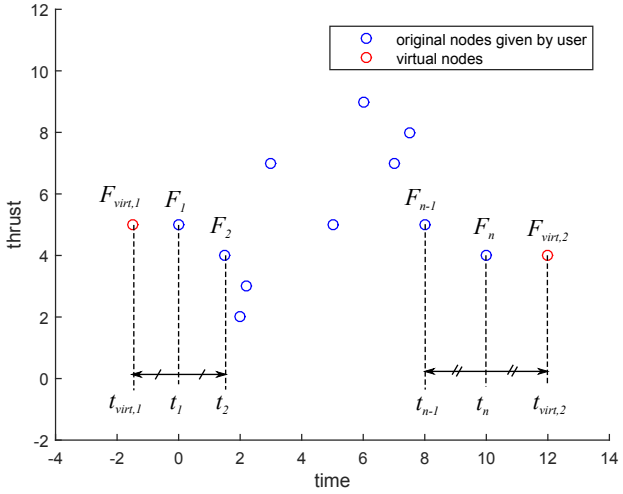
1 Calculate the time value at the four nodes required for the 3$^{\text{rd}}$ order Lagrange interpolation. The following names will be used to indicate the four consecutive nodes in the thrust profile that will be interpolated by the same polynomial: the first, left, right, and last node. Note that the left node is defined as the node that is closest (in time) to the time at the current integration step t_{curr}. The left node can be found using the Tudat `computeNearestLeftNeighborUsingBinarySearch` function. This is a function that searches in the thrust profile for the listed time that is closest to the current time. When the time of the left node is found, the time value of the first, right, and last node can simply be determined by accessing thrust profile table. A problem arises when the left node is a boundary node, i.e. if it refers to either the first row or last row of the thrust profile, because in that case the first node will be undefined. Therefore, at the lower and upper boundaries two additional nodes are required, as will be further explained in the next paragraph.

2 Calculation of the thrust values at the four nodes.

In the previous paragraph it has been explained that two additional nodes need to be created: one to the left of the lower boundary node and one to the right of the upper boundary node, as is depicted in Fig. 5.4. These nodes will be referred to as *virtual* nodes as they are not specified by the user and are therefore not part of the thrust profile. Two definitions are considered for the time and thrust values of the virtual nodes:



(a) Definition 1



(b) Definition 2

Figure 5.4: Visualization of the two options for the definition of the virtual nodes added at the boundaries of the thrust profile

Definition 1: (see Fig. 5.4(a))

- first virtual node: $\left(t_{virt,1}, F_{virt,1}\right) = \left(t_1 - (t_2 - t_1), 0\right)$
- second virtual node: $\left(t_{virt,2}, F_{virt,2}\right) = \left(t_n + (t_n - t_{n-1}), 0\right)$

Definition 2: (see Fig. 5.4(b))

- first virtual node: $\left(t_{virt,1}, F_{virt,1}\right) = \left(t_1 - (t_2 - t_1), F_1\right)$
- second virtual node $\left(t_{virt,2}, F_{virt,2}\right) = \left(t_n + (t_n - t_{n-1}), F_n\right)$

The two definitions only differ in the value of the thrust force. Comparison showed that the second definition is preferred because of its smoother interpolating polynomial for the first and last nodes, especially for the 3^{rd} order Lagrange interpolation.

3 Interpolating the thrust values
Interpolation of the thrust profile is required to obtain:

- the value of the thrust force magnitude F_{curr} at the current time of the integration step t_{curr}.
- the coefficients of the interpolating polynomial are required to calculate the derivatives of the thrust force magnitude and thrust accelerations at the current time.

At this point, the 3rd order Lagrange interpolator kicks in. It requires four points with a t-coordinate (time) and an F-coordinate (thrust) to calculate the coefficients of the continuous interpolating Lagrange polynomial that goes exactly through all given points.
Next, the coefficients can be used to determine the first three derivatives of the interpolating polynomial using this scheme:

$$F(t) = A_F t^3 + B_F t^2 + C_F t + D_F \tag{5.4a}$$
$$F'(t) = 3A_F t^2 + 2B_F t + C_F \tag{5.4b}$$
$$F''(t) = 6A_F t + 2B_F \tag{5.4c}$$
$$F'''(t) = 6A_F \tag{5.4d}$$

The following matrix multiplication results in the value of the F and its derivative at the current time t_{curr}:

$$\begin{bmatrix} F_{curr} F'_{curr} F''_{curr} F'''_{curr} \end{bmatrix} = \begin{bmatrix} A_F & B_F & C_F & D_F \end{bmatrix} \begin{bmatrix} t_{curr}^3 & 3t_{curr}^2 & 6t_{curr} & 6 \\ t_{curr}^2 & 2t_{curr} & 2 & 0 \\ t_{curr} & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{5.5}$$

And, of course, all higher order derivatives will evaluate to zero.

4 Calculation of the thrust accelerations and their derivatives.
Since the equations of motion in the Unified State Model need the thrust accelerations in the RTN frame, they need to be computed using the thrust force values at the four nodes. The thrust values are expressed in the velocity frame, so they first need to be transformed to the RTN frame. As the derivatives of the thrust accelerations are required too, the values of the thrust acceleration at four nodes are calculated and they are used to obtain the coefficients of a 3rd order Lagrangian interpolating polynomial.

$$f_{first} = \frac{F_{first}}{m_{first}} \tag{5.6}$$
$$f_{left} = \frac{F_{left}}{m_{left}} \tag{5.7}$$
$$f_{right} = \frac{F_{right}}{m_{right}} \tag{5.8}$$
$$f_{last} = \frac{F_{last}}{m_{last}} \tag{5.9}$$

F_{first} to F_{last} are already known, but the masses are not known yet. They can be determined using the mass flow in the following equations:

$$m_{first} = m_{curr} + \dot{m}_{curr}(t_{first} - t_{curr}) \tag{5.10}$$
$$m_{left} = m_{curr} + \dot{m}_{curr}(t_{left} - t_{curr}) \tag{5.11}$$
$$m_{right} = m_{curr} + \dot{m}_{curr}(t_{right} - t_{curr}) \tag{5.12}$$
$$m_{last} = m_{curr} + \dot{m}_{curr}(t_{last} - t_{curr}) \tag{5.13}$$

In which:

$$\dot{m}_{curr} = -\frac{F_{curr}}{g_0 I_{sp}} \tag{5.14}$$

Of course, m_{curr} is known because the mass is an element of the propagated state vector.
Next, the values of f_{first}, f_{left}, f_{right} and f_{last} together with their corresponding time values are sent to the 3rd order Lagrangian interpolator, which calculates the coefficients A_f, B_f, C_f and D_f of the interpolating polynomial. Applying a similar matrix multiplication as done for the determination of F_{curr} and its derivatives gives:

$$\begin{bmatrix} f_{curr} & f'_{curr} & f''_{curr} & f'''_{curr} \end{bmatrix} = \begin{bmatrix} A_f & B_f & C_f & D_f \end{bmatrix} \begin{bmatrix} t_{curr}^3 & 3t_{curr}^2 & 6t_{curr} & 6 \\ t_{curr}^2 & 2t_{curr} & 2 & 0 \\ t_{curr} & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{5.15}$$

Higher order derivatives of the acceleration will be zero.

Note that the force and acceleration has three components, i.e. one in e_1, one in e_2 and one in e_3 direction of the RTN frame. This means that the last three of the above described steps need to be executed for each direction separately. After doing that, all variables required for the TSI have been calculated.

5.2.3. CUBIC SPLINE INTERPOLATION

In cubic spline interpolation (CSI) a third order interpolating polynomial is used to interpolate between two consecutive points. The i-th interpolating polynomial $s_i(t)$ active in the subinterval between two nodes $[t_i, t_{i+1}]$ can be denoted as:

$$s_i(t) = A_i + B_i(t - t_i) + C_i(t - t_i)^2 + D_i(t - t_i)^3 \tag{5.16}$$

in which $i = 0, 1, ..., n$ with n being the index of the last node. Besides the standard interpolating conditions ($s_i(t_i) = F_i$ and $s_i(t_{i+1}) = F_{i+1}$ using two consecutive nodes (t_i, F_i) and (t_{i+1}, F_{i+1})), two additional constraints need to be given in order for the third order polynomial to be uniquely defined. These additional constraints can be stated as follows [Dwight, 2011]:

- The first derivative at each node must be continuous:

$$s'_i(t_i) = s'_{i-1}(t_i) \tag{5.17}$$

  and

$$s'_i(t_i) = s'_{i+1}(t_i) \tag{5.18}$$

- When $i = 0$ the constraint expressed in Eq. (5.17) cannot be evaluated. In that case, the following constraint will be used instead:

$$s''_0(t_0) = 0 \tag{5.19}$$

  Similarly when $i = n$, Eq. (5.18) cannot be evaluated and is replaced by:

$$s''_n(t_n) = 0 \tag{5.20}$$

The first two constraints are chosen such that the piecewise polynomial approximation has continuous first and second derivatives at all interior nodes, i.e. all nodes except the boundary nodes. The third and fourth constraint are required to define the second derivative at the boundary nodes. Defining the second derivative to be zero at the boundary nodes is the simplest choice. When making this choice, the cubic spline is called a *natural cubic spline*.

With four active constraints for each polynomial in the spline, the coefficients A_i, B_i, C_i and D_i can be calculated. The value of the thrust force F and its derivatives for $t \in [t_i, t_{i+1}]$ can then be calculated:

$$F(t) = s_i(t) = A_i + B_i(t-t_i) + C_i(t-t_i)^2 + D_i(t-t_i)^3 \tag{5.21a}$$
$$F'(t) = s_i'(t) = B_i + 2C_i(t-t_i) + 3D_i(t-t_i)^2 \tag{5.21b}$$
$$F''(t) = s_i''(t) = 2C_i + 6D_i(t-t_i) \tag{5.21c}$$
$$F^{(3)}(t) = s_i^{(3)}(t) = 6D_i \tag{5.21d}$$
$$F^{(k)}(t) = s_i^{(k)}(t) = 0 \quad \text{for} \quad k > 3 \tag{5.21e}$$

Note that the above procedure is carried out four times: three times for the three components of the thrust force vector and a fourth time for the thrust magnitude. The thrust magnitude is interpolated separately because the k^{th} derivative of the thrust magnitude is not simply equal to the norm of the k^{th} derivative of the three components of the thrust force vector. Now that the required background knowledge of cubic spline interpolation has been introduced, the following step-by-step plan can be followed to obtain all variables required in the TSI-USM.

STEP-BY-STEP PLAN OF THE APPLICATION OF CSI TO TSI

1 The entire discrete thrust profile is interpolated and the four coefficients of every third order polynomial of the spline are stored before the loop over the integration time of the TSI is started. This is done four times, i.e. once for each of the three thrust components and once for the thrust magnitude.

2 Within the loop over the integration time, the stored coefficients are used to calculate the current value of each component of the thrust force as well as the thrust magnitude using Eq. (5.21). Next, using the same set of equations, the derivatives of the thrust magnitude are determined. Subsequently, the thrust acceleration in the RTN frame is determined by converting the thrust force from the velocity frame to the RTN frame using the flight path angle γ (see Section 2.3.4) and dividing this thrust force by the mass.

3 This leaves the calculation of the derivatives of the thrust acceleration. First, the derivatives of the three components of the thrust force in the velocity frame are determined using Eq. (5.21) three times. Next, the derivatives of the thrust acceleration f for $t \in [t_i, t_{i+1}]$ can be calculated using the quotient rule in Eq. (5.22). Note that, to reduce complexity, it is assumed that $f^{(k)} = 0$ for $k > 3$. This assumption is responsible because Eq. (5.21e) shows that from the fourth order derivatives the thrust force is zero as well. Although this does not directly mean that the fourth and higher order derivatives of the thrust accelerations are zero, but they will at least be small because the derivative of the mass is expected to be small as the spacecraft's mass does not change very fast with time.

$$\begin{aligned}
f &= \frac{F}{m} \\
f' &= \frac{F'm - m'F}{m^2} \\
f'' &= \frac{1}{m^3}\left(F''m^2 - m''Fm - 2mm'F' + 2m'm'F\right) \\
f''' &= \frac{1}{m^4}\Big(\left(F'''m^2 - m'''Fm - m''F'm - m''Fm' - 2mm''F' + 4m'm''F\right)m \\
&\qquad -3F''m^2 + 3m''Fm + 6mm'F' - 6m'm'F\Big)
\end{aligned} \tag{5.22}$$

in which

$$m' = -\frac{F}{g_0 I_{sp}} \tag{5.23a}$$

$$m'' = -\frac{F'}{g_0 I_{sp}} \tag{5.23b}$$

$$m''' = -\frac{F''}{g_0 I_{sp}} \tag{5.23c}$$

with F being the thrust magnitude. Now there is still one problem left. The derivatives of the accelerations are required in the three components of the RTN frame: $f_{e1}^{(k)}$, $f_{e2}^{(k)}$ and $f_{e3}^{(k)}$. The following expression represents the calculation of the first derivative of the accelerations in the RTN frame. Remember that f_x', f_y' and f_z' are the accelerations in the velocity frame and are already known from Eq. (5.22).

$$\begin{bmatrix} f_{e1}' \\ f_{e2}' \\ f_{e3}' \end{bmatrix} = \frac{d}{dt}\left(\underbrace{\begin{bmatrix} \sin\gamma & -\cos\gamma & 0 \\ \cos\gamma & \sin\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\substack{\text{transformation matrix from} \\ \text{velocity frame to RTN frame}}} \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} \right) \tag{5.24a}$$

$$= \frac{d}{dt}\left(\begin{bmatrix} \sin\gamma & -\cos\gamma & 0 \\ \cos\gamma & \sin\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} + \begin{bmatrix} \sin\gamma & -\cos\gamma & 0 \\ \cos\gamma & \sin\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \frac{d}{dt}\left(\begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} \right) \tag{5.24b}$$

$$= \begin{bmatrix} \dot\gamma\cos\gamma & \dot\gamma\sin\gamma & 0 \\ -\dot\gamma\sin\gamma & \dot\gamma\cos\gamma & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} + \begin{bmatrix} \sin\gamma & -\cos\gamma & 0 \\ \cos\gamma & \sin\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_x' \\ f_y' \\ f_z' \end{bmatrix} \tag{5.24c}$$

Now in Eq. (5.24c) the time derivative of the flight path angle $\dot\gamma$ is required. Unfortunately this derivative cannot readily be calculated. Even though low-thrust trajectories will not be circular in general, the flight path angle will be small and change only slowly over time for most low-thrust trajectories as is the case for the reference GTOC3 problem. Therefore $\dot\gamma = 0$ is assumed which avoids a lot of complexity. Note that this assumption is corrected later on in Section 5.4 because it was found that it had a negative impact on the accuracy of the TSI. The assumption allows to swap the order of operations around. Instead of transforming the accelerations to the USM frame first and then computing its derivative, the derivatives can be calculated first and then be transformed into the right frame. This simplifies the calculation of the derivatives of the acceleration in the USM frame to:

$$\begin{bmatrix} f_{e1}^{(n)} \\ f_{e2}^{(n)} \\ f_{e3}^{(n)} \end{bmatrix} = \begin{bmatrix} \sin\gamma & -\cos\gamma & 0 \\ \cos\gamma & \sin\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_x^{(n)} \\ f_y^{(n)} \\ f_z^{(n)} \end{bmatrix} \tag{5.25}$$

Eq. (5.25) completes the calculation of all the variables that are needed to make Taylor Series Integration work with cubic spline interpolation of a discrete thrust profile.

5.3. Comparison of Interpolation Methods

In general, the advantage of third order Lagrangian interpolation is that the interpolating polynomial is continuous, which is not the case for a 2nd order Lagrangian interpolating polynomial. This can be seen in Fig. 5.5. Since for 2$^{\text{nd}}$ order Lagrange interpolation the i^{th} polynomial connects an odd number of nodes, i.e. the $(i-1)^{\text{th}}$, i^{th} and $(i+1)^{\text{th}}$ node, each polynomial is valid in the interval $[\frac{1}{2}(t_i - t_{i-1}), \frac{1}{2}(t_{i+1} - t_i)]$. This means that the point at which there is switched from one interpolating polynomial to the next, will always be in the midpoint between two nodes, hence resulting in a discontinuous spline.

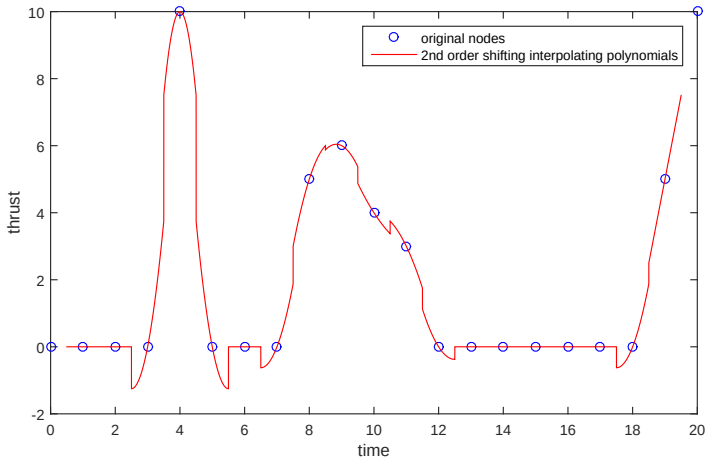


Figure 5.5: Lagrange interpolation of the thrust profile with a spline consisting of second order polynomials

When the same nodes are subjected to a moving 3rd order Lagrangian interpolation, the switching points between two consecutive interpolating polynomials are located at the nodes themselves which results in a continuous function. However, just like the 2nd order Lagrangian interpolation, the derivatives are not continuous.

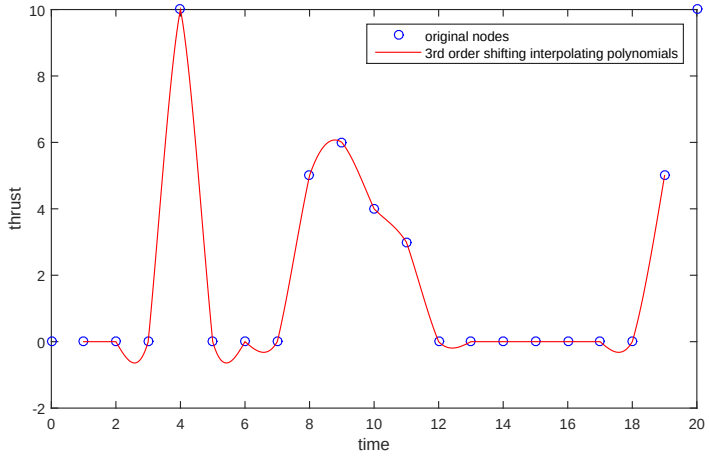


Figure 5.6: Lagrange interpolation of the thrust profile with a spline consisting of third order polynomials

Lastly, Fig. 5.7 shows cubic spline interpolation of those same nodes again. The biggest difference with the 3rd order Lagrange interpolating spline is that the first derivatives are continuous, which is due to the constraints expressed in Eqs. (5.17) and (5.18).

Note that in Figs. 5.5 and 5.6 the first and last nodes are not connected by the interpolating polynomial. This is caused by a limitation of the interpolation method. In order to connect the first and last nodes, two virtual nodes should be created. This can be done using one of the two previously explained definitions for virtual nodes. The cubic spline interpolation does not need these virtual points as it uses the constraints expressed in Eqs. (5.19) and (5.20) to define the polynomials at the left and right boundary of the time domain.

Now that the differences between the different three interpolation methods have been shown, it is time to compare the results of the TSI obtained with these three interpolation methods.

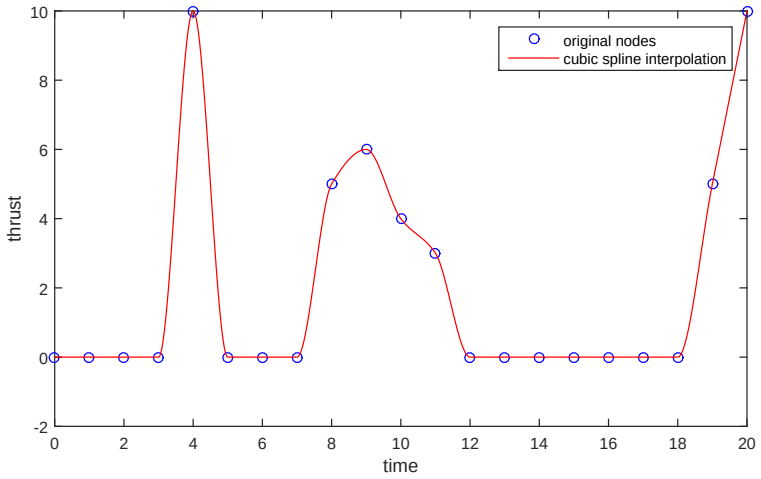


Figure 5.7: Cubic spline interpolation of the thrust profile

ACCURACY RESULTS

In this paragraph, the difference in the propagated trajectory of the following combinations will be compared against the trajectory produced with an RK4 propagator with third order Lagrange interpolation (L3) with cubic spline interpolation for the boundary handling.

- GTSI-USM with second order Lagrange interpolation (L2) with definition 1 for the virtual points
- GTSI-USM L2 with definition 2 for the virtual points
- GTSI-USM L3 with definition 2 for the virtual points
- GTSI-USM with CSI

The order of the relative differences in propagated mass, thrust, position and velocity for different thrust profiles are given in Table 5.3. Note that the table displays the order of magnitude of the relative differences in these parameters. The settings for the TSI and RK-based propagators are shown in Table 5.2.

Table 5.2: Propagation settings for TSI and RK-based propagators

Initial orbit	
semi-major axis	72130e3 [m]
eccentricity	0.6
inclination	169 [deg]
longitude of ascending node	80 [deg]
argument of pericenter	45 [deg]
true anomaly	15 [deg]
Spacecraft settings	
Specific impulse (constant)	3000 [s]
Initial mass	2000 [kg]
Integration settings	
duration	10 [days]
step size (fixed)	60 [s]
TSI order	30
refinement factor	0

Table 5.3: Comparison of TSI L2 (both with definition 1 and 2 for the virtual nodes), TSI L3 (definiton 2 for the virtual nodes) and TSI CSI for different thrust profiles (see Table 2.2 for the definition of the thrust directions) with respect to L3 RK with CSI boundary handling. The numbers show the order of the relative difference between two trajectories. (green = high accuracy, orange = medium accuracy, red = low accuracy)

Thrust profile	Parameter	L2 Def1 TSI vs L3 RK	L2 Def2 TSI vs L3 RK	L3 Def2 TSI vs L3 RK	CSI TSI vs L3 RK
1N constant normal thrust	Mass [kg]	-5	-13	no difference	no difference
	Thrust [N]	-5	-7	-7	-7
	Position [km]	-2	-2	-2	-2
	Velocity [m/s]	-3	-3	-3	-3
1N constant binormal thrust	Mass [kg]	-5	-13	no difference	no difference
	Thrust [N]	-5	-11	-11	-11
	Position [km]	-3	-4	-4	-4
	Velocity [m/s]	-4	-5	-5	-5
1N constant tangential thrust	Mass [kg]	-5	-13	no difference	no difference
	Thrust [N]	-5	-9	-9	-9
	Position [km]	-1	-4	-4	-4
	Velocity [m/s]	-2	-5	-5	-5
10N constant tangential thrust	Mass [kg]	-4	-9	-10	no difference
	Thrust [N]	-4	-8	-8	-8
	Position [km]	1	-3	-3	-3
	Velocity [m/s]	-2	-5	-5	-5
non-constant thrust	Mass [kg]	-4	-5	-5	-5
	Thrust [N]	-4	-5	-5	-5
	Position [km]	-1	-1	-1	-1
	Velocity [m/s]	-2	-2	-2	-2
zero thrust	Mass [kg]	no difference	no difference	no difference	no difference
	Thrust [N]	no difference	no difference	no difference	no difference
	Position [km]	-4	-4	-4	-4
	Velocity [m/s]	-5	-5	-5	-5

From the above table, it can be concluded that CSI is the preferred interpolation method for the interpolation of the thrust profile. This means that the method described in Section 5.2.3 will be used in the GDF for the remainder of this report.

Besides the values in the above table, two general observations can be done: (1) the difference in the thrust, position and velocity is largest at the pericenter of the orbit and (2) for L2 with zero virtual points (definition 2), the larger difference for the mass and thrust is only caused by the larger differences at the first and last node. For the rest of the domain, the difference is equal to the observed difference for L2 TSI.

5.4. CORRECTION OF THE CONSTANT FLIGHT PATH ANGLE ASSUMPTION

Previously, at the end of Section 5.2, it was assumed that the flight path angle remained constant during the entire low-thrust trajectory. This results in a zero derivative of the flight path angle which simplifies the transformation of the thrust force derivatives from the velocity frame to the RTN frame.

However, it was found that this assumption led to an unacceptable decrease in the accuracy of TSI. Therefore, in this section, the calculation of the higher order derivatives of the thrust accelerations in the RTN frame will be explained.

First, Eq. (5.24c) is repeated below:

$$\begin{bmatrix} f'_{e1} \\ f'_{e2} \\ f'_{e3} \end{bmatrix} = \begin{bmatrix} \dot{\gamma}\cos\gamma & \dot{\gamma}\sin\gamma & 0 \\ -\dot{\gamma}\sin\gamma & \dot{\gamma}\cos\gamma & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} + \begin{bmatrix} \sin\gamma & -\cos\gamma & 0 \\ \cos\gamma & \sin\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f'_x \\ f'_y \\ f'_z \end{bmatrix} \tag{5.26}$$

The second derivative of the thrust acceleration vector in the RTN frame becomes:

$$\begin{bmatrix} f_{e1}'' \\ f_{e2}'' \\ f_{e3}'' \end{bmatrix} = \begin{bmatrix} \ddot{\gamma}\cos\gamma - \dot{\gamma}^2\sin\gamma & \ddot{\gamma}\sin\gamma + \dot{\gamma}^2\cos\gamma & 0 \\ -\ddot{\gamma}\sin\gamma - \dot{\gamma}^2\cos\gamma & \ddot{\gamma}\cos\gamma - \dot{\gamma}^2\sin\gamma & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} + 2 \begin{bmatrix} \dot{\gamma}\cos\gamma & \dot{\gamma}\sin\gamma & 0 \\ -\dot{\gamma}\sin\gamma & \dot{\gamma}\cos\gamma & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} f_x' \\ f_y' \\ f_z' \end{bmatrix} + \begin{bmatrix} \sin\gamma & -\cos\gamma & 0 \\ \cos\gamma & \sin\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_x'' \\ f_y'' \\ f_z'' \end{bmatrix} \tag{5.27}$$

The third derivative becomes:

$$\begin{aligned} \begin{bmatrix} f_{e1}''' \\ f_{e2}''' \\ f_{e3}''' \end{bmatrix} = & \begin{bmatrix} \dddot{\gamma}\cos\gamma - 3\ddot{\gamma}\dot{\gamma}\sin\gamma - \dot{\gamma}^3\cos\gamma & \dddot{\gamma}\sin\gamma + 3\ddot{\gamma}\dot{\gamma}\cos\gamma - \dot{\gamma}^3\sin\gamma & 0 \\ -\dddot{\gamma}\sin\gamma - 3\ddot{\gamma}\dot{\gamma}\cos\gamma + \dot{\gamma}^3\sin\gamma & \dddot{\gamma}\cos\gamma - 3\ddot{\gamma}\dot{\gamma}\sin\gamma - \dot{\gamma}^3\cos\gamma & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} \\ & + 3 \begin{bmatrix} \ddot{\gamma}\cos\gamma - \dot{\gamma}^2\sin\gamma & \ddot{\gamma}\sin\gamma + \dot{\gamma}^2\cos\gamma & 0 \\ -\ddot{\gamma}\sin\gamma - \dot{\gamma}^2\cos\gamma & \ddot{\gamma}\cos\gamma - \dot{\gamma}^2\sin\gamma & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} f_x' \\ f_y' \\ f_z' \end{bmatrix} \\ & + 3 \begin{bmatrix} \dot{\gamma}\cos\gamma & \dot{\gamma}\sin\gamma & 0 \\ -\dot{\gamma}\sin\gamma & \dot{\gamma}\cos\gamma & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} f_x'' \\ f_y'' \\ f_z'' \end{bmatrix} \\ & + \begin{bmatrix} \sin\gamma & -\cos\gamma & 0 \\ \cos\gamma & \sin\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_x''' \\ f_y''' \\ f_z''' \end{bmatrix} \end{aligned} \tag{5.28}$$

To be exact, this derivation has to be continued until the order of the derivatives reaches the order the TSI. However, to reduce the CPU time and complexity the following assumption is made:

$$\frac{d^k}{dt^k}\left(\begin{bmatrix} f_{e1} \\ f_{e2} \\ f_{e3} \end{bmatrix}\right) = 0 \qquad \text{for } k > 3 \tag{5.29}$$

This assumption is based on the fact that from the fourth derivative of the thrust acceleration in the velocity frame, the derivatives become zero. Of course, this does not mean that the fourth and higher order derivatives of the thrust acceleration in the RTN frame are zero, but at least they will be small. Indeed, the assumption of a constant flight path angle has been replaced by the assumption of zero fourth and higher order derivatives of the thrust acceleration in the RTN frame to be zero. However, the latter assumption poses much less threat to the accuracy of the TSI.

If the calculation of the derivatives is limited to the first, second and third order derivatives in Eqs. (5.28) to (5.28) respectively, the first three derivatives of γ are required. These can be calculated in the following way.

The flight path angle is defined as

$$\gamma = \operatorname{atan}\left(\frac{v_r}{v_{tr}}\right) \tag{5.30}$$

in which v_r is the radial and v_{tr} is the transverse velocity (i.e. normal to the radius vector, in the orbital plane and in the direction of positive velocity) of the spacecraft. The derivative of the flight path angle becomes

$$\begin{aligned} \dot{\gamma} &= \frac{1}{1 + \left(\frac{v_r}{v_{tr}}\right)^2} \frac{\dot{v}_r v_{tr} - \dot{v}_{tr} v_r}{v_{tr}^2} \\ &= \frac{\dot{v}_r v_{tr} - \dot{v}_{tr} v_r}{v_{tr}^2 + v_r^2} \end{aligned} \tag{5.31}$$

The second derivative now is

$$\ddot{\gamma} = \frac{\frac{\ddot{v}_r}{v_{tr}} - \frac{2\dot{v}_r\dot{v}_{tr}}{v_{tr}^2} + \frac{2v_r\dot{v}_{tr}^2}{v_{tr}^3} - \frac{v_r\ddot{v}_{tr}}{v_{tr}^2}}{\frac{v_r^2}{v_{tr}^2} + 1} - \frac{\left(\frac{\dot{v}_r}{v_{tr}} - \frac{v_r\dot{v}_{tr}}{v_{tr}^2}\right) \cdot \left(\frac{2v_r\dot{v}_r}{v_{tr}^2} - \frac{2v_r^2\dot{v}_{tr}}{v_{tr}^3}\right)}{\left(\frac{v_r^2}{v_{tr}^2} + 1\right)^2} \tag{5.32}$$

And, finally, the third derivative is

$$\begin{aligned}\dddot{\gamma} = &\frac{2\cdot\left(\frac{\dot{v}_r}{v_{tr}}-\frac{v_r\dot{v}_{tr}}{v_{tr}^2}\right)\cdot\left(\frac{2v_r\dot{v}_r}{v_{tr}^2}-\frac{2v_r^2\dot{v}_{tr}}{v_{tr}^3}\right)^2}{\left(\frac{v_r^2}{v_{tr}^2}+1\right)^3}\\ &-\frac{2\cdot\left(\frac{2v_r\dot{v}_r}{v_{tr}^2}-\frac{2v_r^2\dot{v}_{tr}}{v_{tr}^3}\right)\cdot\left(\frac{\ddot{v}_r}{v_{tr}}-\frac{2\dot{v}_r\dot{v}_{tr}}{v_{tr}^2}+\frac{2v_r\dot{v}_{tr}^2}{v_{tr}^3}-\frac{v_r\ddot{v}_{tr}}{v_{tr}^2}\right)}{\left(\frac{v_r^2}{v_{tr}^2}+1\right)^2}\\ &-\frac{\left(\frac{\dot{v}_r}{v_{tr}}-\frac{v_r\dot{v}_{tr}}{v_{tr}^2}\right)\cdot\left(\frac{2\dot{v}_r^2}{v_{tr}^2}+\frac{6v_r^2\dot{v}_{tr}^2}{v_{tr}^4}-\frac{2v_r^2\ddot{v}_{tr}}{v_{tr}^3}+\frac{2v_r\ddot{v}_r}{v_{tr}^2}-\frac{8v_r\dot{v}_r\dot{v}_{tr}}{v_{tr}^3}\right)}{\left(\frac{v_r^2}{v_{tr}^2}+1\right)^2}\\ &-\frac{\left(\frac{v_r\dddot{v}_{tr}}{v_{tr}^2}-\frac{\dddot{v}_r}{v_{tr}}-\frac{6\dot{v}_r\dot{v}_{tr}^2}{v_{tr}^3}+\frac{3\dot{v}_r\ddot{v}_{tr}}{v_{tr}^2}+\frac{3\dot{v}_{tr}\ddot{v}_r}{v_{tr}^2}+\frac{6v_r\dot{v}_{tr}^3}{v_{tr}^4}-\frac{6v_r\dot{v}_{tr}\ddot{v}_{tr}}{v_{tr}^3}\right)}{\left(\frac{v_r^2}{v_{tr}^2}+1\right)}\end{aligned} \tag{5.33}$$

The values of the radial and transverse velocities directly follow from the TSI state vector elements v_{e1} and v_{e2} as can be seen from Eqs. (4.28i) and (4.28j). The derivatives of the two velocity components can be obtained from Eqs. (4.54i) and (4.54j).

To wrap up, during every integration step of the GTSI-USM propagator, the radial and transverse velocities as well as their first and second derivatives are obtained from the state vector (Eqs. (4.28i) and (4.28j)) and its derivatives (Eqs. (4.54i) and (4.54j)) at that step. Subsequently, the values of γ (Eq. (5.30)), $\dot{\gamma}$ (Eq. (5.31)), $\ddot{\gamma}$ (Eq. (5.32)) and $\dddot{\gamma}$ (Eq. (5.33)) are calculated. Next, these values are used in Eqs. (5.26) to (5.28) to calculate the first three derivatives of the thrust acceleration in the RTN frame. Finally, these derivatives are used to compute the auxiliary parameters V_{12}, V_{13} and V_{14} in Eq. (4.51). These values are in turn used to calculate the next order derivative of U in Eqs. (4.54i) and (4.54j) and indirectly in Eqs. (4.54b) and (4.54c) through the use of Eqs. (4.49q) and (4.49r).

5.5. CORRECTION OF THE INTERPOLATION OF THE THRUST MAGNITUDE

As has been explained before, the thrust values for the three dimensions of the thrust vector and the thrust magnitude are interpolated separately. This is because the coefficients of the interpolating polynomials are required to separately calculate the derivatives of the three thrust components and the thrust magnitude.

A problem arises when interpolating thrust profiles such as the one on the left side in Fig. 5.8. In the bottom left of the figure, it can be seen that the CSI fit crosses the time axis at some point. This means that after that point, the thrust magnitude would be negative. This is physically impossible and it causes problems for the propagation of the mass. From Eq. (5.23a) it can clearly be seen that a negative thrust magnitude F causes the mass flow m' to be positive resulting in the spacecraft to gain mass while thrusting.

To avoid this problem the thrust magnitude F is calculated by taking the Euclidean norm of the interpolated F_x, F_y and F_z as can be seen in the graphs on the right hand side. This guarantees F to be positive at all times. This value of F will be used in Eq. (5.23a). For the derivatives of F (e.g. in Eqs. (5.23b) and (5.23c)), the coefficients of the interpolating polynomials of the CSI from the bottom left graph will still be used. This does not form a problem because the derivatives of the thrust magnitude can physically be positive or negative.

As expected, the correction of the interpolation of the thrust magnitude resolved a major error in the mass propagation.

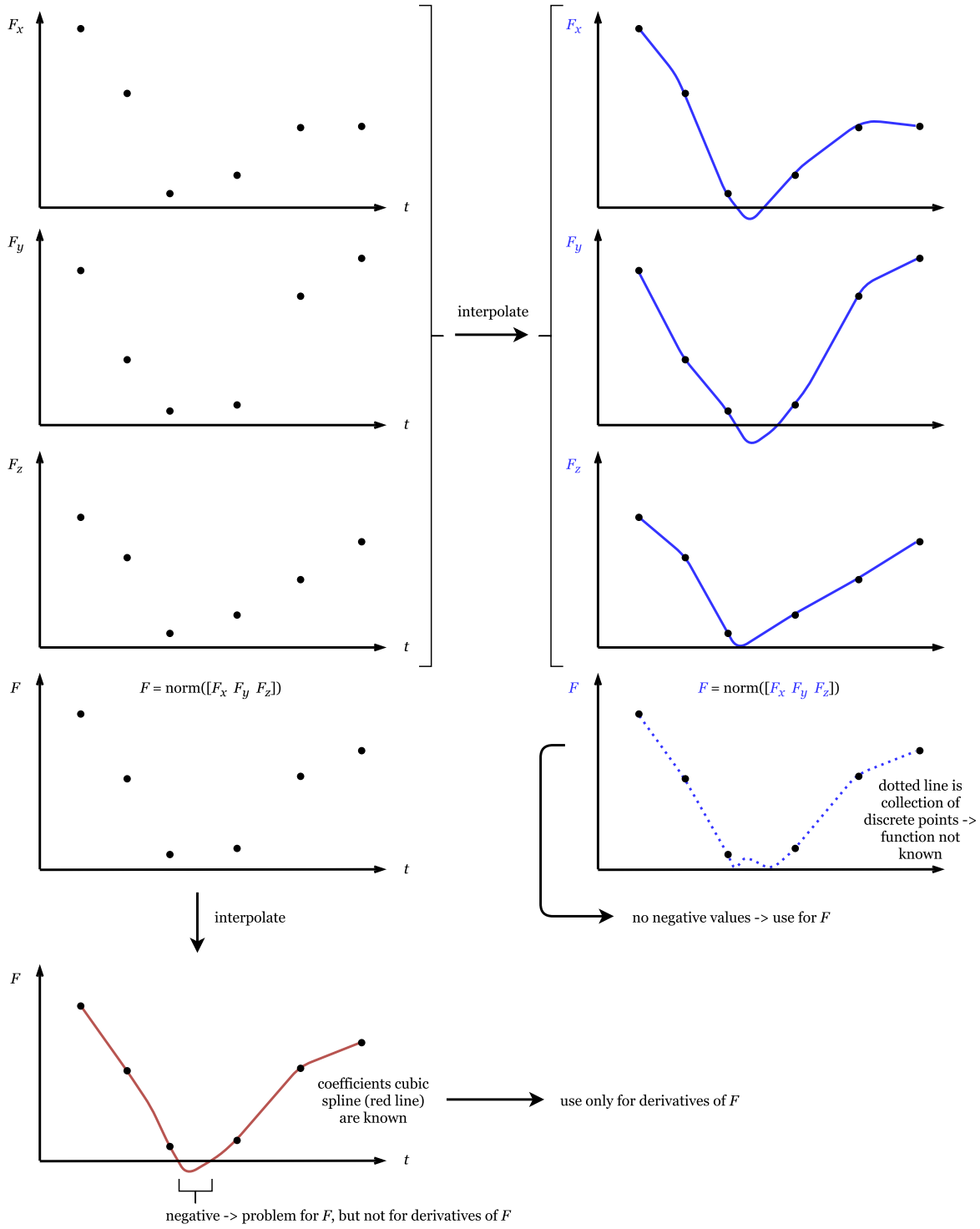


Figure 5.8: Diagram of the difference in the method used to determine the thrust magnitude F and its derivatives.

6

PROPAGATORS & VERIFICATION

The introduction in Chapter 1 and the discussion of the TSI in Chapter 4 indicated that the combination of TSI and USM is a fast propagator for low-thrust trajectories. However, it was also found that the TSI-USM is a problem-specific method which is detrimental for its use in evolutionary algorithms. In Chapter 5 the General Discrete Force Model (GDFM) was developed that, when implemented in TSI-USM, would allow the propagation of any low-thrust profile. This generalized TSI-USM is referred to as GTSI-USM. This chapter will discuss the verification of the GTSI-USM and describe the reference propagators that will be used in the next chapter to determine and judge the accuracy and the computational efficiency of the GTSI-USM. The latter task can be regarded as the validation of the GTSI-USM and will, among other things, contribute to the results of this thesis. The validation and results will be explained Chapter 7.

Contents

6.1. INTRODUCTION

To determine the accuracy and the computational efficiency of the developed generalized TSI-USM propagator for low-thrust trajectories (GTSI-USM) propagator, three RK8(7)13M based propagators are implemented to act as reference propagators to compare the performance of GTSI-USM with: RK8(7)13M combined with Cartesian coordinates (RK-Cart), RK8(7)13M combined with USM elements and a version of the RK-Cart in the Tudat propagation setup (RK-Cart-TPS). The first section of this chapter will give a summary of these implemented propagators, including the GTSI-USM.

In the second part of this chapter, the verification of the propagators will be described. The RK-Cart-TPS is completely built up of validated Tudat modules, which ensures its validation as will be described in Section 6.3. In Section 6.4, the order of accuracy of the RK-Cart-TPS will be determined. This is important to know to what accuracy level the propagator can be used to produce reference trajectories for the validation of the GTSI-USM. Subsequently, in Section 6.5, the C++ version of the GTSI-USM is verified by comparing it to the output of the Matlab version. Also in this section, is a qualitative test of the thrust direction is presented. As will be seen in the next chapter, the RK-USM and RK-Cart will not be used for the non-zero thrust case, so they only need to be verified for the zero thrust case. The verification of the GTSI-USM, RK-Cart and RK-USM for the zero thrust case is discussed in Section 6.6. The accuracy test of the GTSI-USM for the non-zero thrust case will be referred to as the validation of the GTSI-USM and, as this contributes to the results of this thesis, it will be discussed in the next chapter.

The third part of this chapter will discuss a useful technique that can be used to generate reference solutions for perturbed trajectories. Unfortunately, this method became available quite late in the process of this thesis which explains why the method was not part of the verification process of this thesis. However, as the method can be very useful for future students it will be discussed in Section 6.7.

During the process of programming the GTSI-USM, several design iterations were performed. This resulted in a number of corrections that are presented in Section 6.8. These corrections were implemented prior to starting the validation process of GTSI-USM which will be the topic of the next chapter.

6.2. PROPAGATORS

This section contains a summary of the four different propagators that are implemented in C++ and that are used to produce the results of this thesis.

6.2.1. GTSI-USM

When the developed General Discrete Force Model (GDFM) described in the previous chapter is implemented within the existing TSI-USM propagator, the generalized TSI-USM or briefly GTSI-USM is formed. As the name suggests, the GTSI-USM propagator is a generalized propagator based on the variable order Taylor Series integration and of which the spacecraft state is expressed in USM elements. As explained in Chapter 5, the propagator is generalized in the sense that it allows the propagation of any predetermined thrust profile, defined with respect to time, without the need to change the internal relations of the TSI-USM.

The GTSI-USM was initially programmed in Matlab since the existing TSI-USM code, provided by V. Vittaldev [Vittaldev, 2010], was coded in Matlab. After the Matlab code was finished, the GTSI-USM was converted to C++ which serves two purposes: making the code faster with the use of classes and passing variables by reference, and being able to easily share the code with the TU Delft Astrodynamics Toolbox (TUDAT) support team so that it can be used by future students.

IMPLEMENTATION

The implementation of the GDFM in the TSI-USM to form the GTSI-USM involves two main tasks. The first one is adding the cubic spline interpolation of the thrust profile and the storage of the coefficients of the cubic spline. The second one is the calculation of the derivatives of the thrust force magnitude and the computation of the derivatives of the thrust accelerations in the RTN frame.

First, before the integration loop of the TSI-USM is started, the entire pre-set thrust profile in the velocity frame is interpolated using cubic spline interpolation and the coefficients of the cubic spline are stored. This procedure is done four times: once for the three components of the thrust force and a fourth time for the magnitude of the thrust force. Subsequently, the integration loop over the specified integration time window starts. In every iteration of the loop, the following tasks are performed:

1 For the current time, the values of the three components of the thrust force, and the thrust magnitude as well as their derivatives are computed, in the velocity frame, from the stored coefficients of the interpolating cubic spline using Eq. (5.21).

2 The thrust accelerations in the VF as well as their derivatives in the VF are computed from the thrust force and its derivatives using Eq. (5.22) and Eq. (5.23).

3 The thrust accelerations in the velocity frame and their derivatives are transformed to the RTN frame using Eq. (5.26) to Eq. (5.33).

4 The thrust accelerations in the RTN frame and their derivatives can now be substituted into the recurrence relations of the TSI (e.g. in Eq. (4.49l), Eq. (4.49m), Eq. (4.49n) and Eq. (4.51))

5 The normalized derivatives are calculated using Eq. (4.54).

6 The current state, the current step-size, the normalized derivatives $U_n(k)$ and the order of the Taylor series K are substituted into Eq. (4.20c). This can be done for each state variable x_n thus forming the next state vector.

7 The current state and current time are updated.

8 In case of a fixed step-size the body of the loop ends here. In case of variable step-size, the body of the loop continues with step **9**.

9 The error control method found in [Scott and Martini, 2008] has been used to compute the next step-size in an iterative way. The step-size loop is described below:

9-1 A step-size is computed using Eq. (6.1) for each of the seven USM state variables x_n using the current step-size h_i, the absolute tolerance τ_n that corresponds to the n^{th} state variable, the TSI order K, the safety factor η, which is chosen to be 0.9, and the earlier computed normalized derivatives X_n which relation to U_n has been given in Eq. (4.16).

$$h_{i+1} = \eta \exp\left(\frac{1}{K-1}\ln\left(\frac{\tau_n}{|X_n(K-1)| + h_i|X_n(K)|}\right)\right) \tag{6.1}$$

9-2 The minimum demanded step-size of all state variables is determined.

9-3 The relative difference between the computed minimum demanded step-size and the previous step-size is computed. When this relative difference is below a certain specified value (here: 1e-6), the program goes to step 10. If the step-size has not converged, then the previous step-size is updated using the computed step-size and the program goes back to **9-1**.

10 It is checked whether the step-size lies between the predetermined minimum and maximum allowed step-sizes.

11 Optionally, the state history vector can be refined by calculating intermediate states with the same accuracy as the states at the original nodes. This is possible because the stored normalized derivatives X_n can be used directly in the Taylor series expansion of the state variables (see Eq. (4.20b)).

The output of the program is the state history of the spacecraft (i.e. the seven USM elements and the mass) as well as the interpolated thrust force in the RTN frame at every step the propagation has taken.

The post-processing of the output data includes the transformation of the USM state to Cartesian coordinates and the transformation of the thrust force from the RTN frame to the velocity frame which simplifies the interpretation of the output. For a flow diagram of the implemented TSI-USM the reader is referred to Appendix C.3.

6.2.2. RK-CART

The RK-Cart propagator which combines RK8(7)13M with the Cartesian state derivative model is already available in Tudat. The RK8(7)13M scheme can be found in [Prince and Dormand, 1981] and was explained in Section 1.4.2. However, it has to be combined with a limited version of the discrete force model. The author

of this thesis provided Tudat with the functionality of propagating a discrete thrust profile with the use of the Tudat module for cubic spline interpolation. The Cartesian state derivative model is the following:

$$\frac{dx}{dt} = v_x \tag{6.2a}$$
$$\frac{dy}{dt} = v_y \tag{6.2b}$$
$$\frac{dz}{dt} = v_z \tag{6.2c}$$
$$\frac{dv_x}{dt} = -\mu\frac{x}{r^3} + \frac{F_x}{m} \tag{6.2d}$$
$$\frac{dv_y}{dt} = -\mu\frac{y}{r^3} + \frac{F_y}{m} \tag{6.2e}$$
$$\frac{dv_z}{dt} = -\mu\frac{z}{r^3} + \frac{F_z}{m} \tag{6.2f}$$
$$\frac{dm}{dt} = \frac{-F}{g_0 I_{sp}} \tag{6.2g}$$

The forces F_x, F_y, F_z and the magnitude of the thrust force F are determined by cubic spline interpolation of the pre-set thrust profile with respect to time. In contrast to the GTSI-USM, the derivatives of the thrust force do not need to computed.

Like the GTSI-USM, RK-based propagators can be used with either fixed or variable step-size control. The default step-size control method in Tudat has been used. This method is based on limiting the truncation error estimated from the difference between the higher order and the lower order solution. For RK8(7), these orders are eight and seven, respectively. As described by Montenbruck and Gill in Satellite Orbits [Montenbruck and Gill, 2001], the estimate of the absolute local truncation is:

$$\varepsilon \approx |\boldsymbol{x} - \boldsymbol{x}^*| \tag{6.3}$$

where $\boldsymbol{x}$ is the higher order solution and $\boldsymbol{x}^*$ is the lower order solution of a state vector $\boldsymbol{x}$ at a step in the integration. Subsequently, the error tolerance τ is determined from the pre-set relative and absolute tolerance requirements as follows:

$$\tau = |\boldsymbol{x}| \cdot \tau_{rel} + \tau_{abs} \tag{6.4}$$

Next, the following relation is used to compute the maximum allowed next step-size:

$$h_{i+1} = \eta \cdot \sqrt[k+1]{\frac{\tau}{\varepsilon}} \cdot h_i \tag{6.5}$$

where k is 8 for RK8(7), h_i is the previous step-size and η is a safety factor chosen to equal 0.9. Similar to the step-size loop of the TSI-USM, an acceptance test for the computed step-size is executed. This test is the following:

$$\frac{\varepsilon}{\tau} \leq 1 \tag{6.6}$$

If the step-size is accepted, the step-size is stored and will be used for the next step of the integrator. If the step-size is not accepted, h_i is updated with the computed unaccepted step-size and Eq. (6.5) is run again. This iteration continues until the step-size is accepted. In case of a fixed step-size the lower step-size limit is simply set equal to the higher step-size limit.

6.2.3. RK-USM

The RK-USM propagator requires the combination of the selected RK8(7)13M integrator with the USM state derivative model. The USM state derivative model consists of the following set of differential equations. Note

that unlike the eleven state variables used in GTSI-USM, here only eight state variables are used: seven USM elements and the mass of the spacecraft m.

$$\frac{dC}{dt} = -pf_{e2} \tag{6.7a}$$
$$\frac{dR_{f1}}{dt} = f_{e1}\cos\lambda - f_{e2}(1+p)\sin\lambda - f_{e3}l\frac{R_{f2}}{v_{e2}} \tag{6.7b}$$
$$\frac{dR_{f2}}{dt} = f_{e1}\sin\lambda + f_{e2}(1+p)\cos\lambda + f_{e3}l\frac{R_{f1}}{v_{e2}} \tag{6.7c}$$
$$\frac{d\epsilon_{O1}}{dt} = \frac{1}{2}\left(\omega_3\epsilon_{O2} + \omega_1\eta_O\right) \tag{6.7d}$$
$$\frac{d\epsilon_{O2}}{dt} = \frac{1}{2}\left(-\omega_3\epsilon_{O1} + \omega_1\epsilon_{O3}\right) \tag{6.7e}$$
$$\frac{d\epsilon_{O3}}{dt} = \frac{1}{2}\left(-\omega_1\epsilon_{O2} + \omega_3\eta_O\right) \tag{6.7f}$$
$$\frac{d\eta_O}{dt} = \frac{1}{2}\left(-\omega_1\epsilon_{O1} - \omega_3\epsilon_{O3}\right) \tag{6.7g}$$
$$\frac{dm}{dt} = \frac{-F}{g_0 I_{sp}} \tag{6.7h}$$

The variables C, R_{f1}, R_{f2}, ϵ_{O1}, ϵ_{O2}, ϵ_{O3} and η_O are known, because they are part of the USM7 state vector. Furthermore the elements $\cos\lambda$, $\sin\lambda$, v_{e2}, p, k, ω_1, ω_3 can be calculated in advance using the USM7 state vector:

$$\cos\lambda = \frac{\eta^2 - \epsilon_{O3}^2}{\epsilon_{O3}^2 + \eta^2} \tag{6.8a}$$
$$\sin\lambda = \frac{2\epsilon_{O3}\eta_O}{\epsilon_{O3}^2 + \eta_O^2} \tag{6.8b}$$
$$v_{e2} = C - R_{f1}\sin\lambda + R_{f2}\cos\lambda \tag{6.8c}$$
$$p = \frac{C}{v_{e2}} \tag{6.8d}$$
$$l = \frac{\epsilon_{O1}\epsilon_{O3} - \epsilon_{O2}\eta_O}{\epsilon_{O3}^2 + \eta_O^2} \tag{6.8e}$$
$$\omega_1 = \frac{f_{e3}}{v_{e2}} \tag{6.8f}$$
$$\omega_3 = \frac{C v_{e2}^2}{\mu} \tag{6.8g}$$

This leaves the thrust accelerations f_{e1}, f_{e2} and f_{e3} which are determined through cubic spline interpolation of the thrust profile. In contrast to the cubic spline interpolation for the RK-Cart propagator, the method used in RK-USM7 does not make use of the Tudat propagation setup. Instead, the interpolation of the thrust profile is executed by a separate function similar to the one used for the TSI-USM7 propagator. The reason for this is that due to time constraints it was not possible to adapt the Tudat propagator setup in a way that would allow propagation of state vectors of more than six elements (excluding the spacecraft's mass).

The variable step-size control method used for the RK-USM is identical to the one used for the RK-Cart propagator described in the previous section.

6.2.4. RK-CART-TPS

The RK-Cart-TPS propagator is similar to the discussed RK-Cart propagator, the only difference is that the RK-Cart-TPS makes full use of the Tudat propagation setup (TPS) whereas the RK-Cart's integration was performed using a Cartesian state derivative outside of Tudat. The purpose of having these two propagators is twofold. First, it allows to test the influence of the Tudat propagation setup on the CPU time and, secondly, it

allows a more correct comparison between the computational efficiency of RK-Cart, GTSI-USM and RK-USM since GTSI-USM and RK-USM do not make full use of the tudat propagation setup either.
As will be seen in the next section, the RK-Cart-TPS will completely verified for all thrust cases. Therefore, it will also be used to produce assumed 'exact' trajectories that can be used to determine the error of the trajectories propagated by the other propagators.

6.3. VERIFICATION OF RK-CART-TPS

Before the developed RK-Cart-TPS propagator can be used to validate the TSI (which will be done in Chapter 7), it in itself needs to be verified. Fortunately, the RK-Cart-TPS is built inside Tudat with the use of existing modules. The existing modules have all been individually validated by previous Tudat developers through the following unit tests [Dirkx, 2017]:

- unitTestRungeKutta87DormandPrinceIntegrator.cpp
- unitTestRungeKuttaVariableStepSizeIntegrator.cpp
- unitTestCubicSplineInterpolator.cpp
- unitTestNearestNeigbourSearch.cpp
- unitTestReferenceFrameTransformations.cpp
- unitTestAccelerationModel.cpp
- unitTestUnitConversions.cpp

Besides these unit tests, the entire RK-Cart-TPS is verified using the following tests:

- A test is performed by the Tudat support team in which the thrust force is equated with the opposite of gravity. This indeed shows a non-moving spacecraft.
- A test case is performed in which the spacecraft does not experience gravitational acceleration. This allows the thrust force to be checked using the distance traveled by the spacecraft. This test case was performed by the author of this thesis and is described below.
- Moreover, the RK propagation setup has already been successfully used in many different projects and theses so that it can be declared to be verified.

DESCRIPTION OF TEST CASE WITH THRUST AND WITHOUT GRAVITY

The test involves the propagation of the trajectory of a spacecraft with constant mass subject to a constant thrust force in a fixed inertial direction without the influence of gravity. It is decided to apply a constant thrust force of 0.15 N to a 2000 kg spacecraft. This results in a constant acceleration of $a = 0.000075$ m/s^2.
The velocity and traveled distance are computed as follows:

$$v = v_0 + at \tag{6.9}$$

$$d = d_0 + v_0 t + \frac{1}{2} a t^2 \tag{6.10}$$

in which v_0 is the initial velocity, d_0 is the initial distance and $t = 365.25 \cdot 86400 \cdot 10$ s ≈ 10 years. Table 6.1 compares the outcome of Eq. (6.10) with the propagation results.

Table 6.1: Comparison of the analytic and propagated solution

	Analytic result	**Propagation**	**Number of correct digits**
Velocity [m/s]	24668.2	24668.200000099	12
Distance [m]	4050133941600	4050133941604.92	12

From the table, it can be concluded that the propagation is accurate up to at least 5e-12 level. The values in the table indicate that numerical errors (truncation and round-off errors) are an inherent part of numerical propagation.

To conclude, as the RK-Cart-TPS is validated for the thrust case, it can be used to produce a reference solution that can in turn be used to determine the accuracy of the GTSI-USM. At this point, there is no need to validate the RK-Cart and the RK-USM for the non-zero thrust case as a reference solution is already available through the validated RK-Cart-TPS propagator.

6.4. ORDER OF ACCURACY DETERMINATION OF RK-CART-TPS

Since the version of RK-Cart propagator programmed completely within the Tudat propagation setup (RK-Cart-TPS) will be used to produce the reference solutions for the non-zero thrust cases, it is important to determine the level of accuracy up to which the RK-Cart-TPS solution is correct. For this reason the order of accuracy of the the propagator is evaluated after 5 days of integration in an elliptical orbit around the Earth (see Table 6.2 and the thrust profile in Table 6.3). This propagation is executed several times while halving the step-size until the difference between one solution and the next reaches the maximum attainable accuracy. Note that the difference between the solutions is defined as the difference in position, velocity and mass of the spacecraft after 5 days of integration. Figs. 6.1 to 6.3 show a log-log plot of the step size as a function of the relative difference between one solution and the next.

Table 6.2: Propagation settings for accuracy determination of RK8(7)

Initial orbit	
semi-major axis	72130e3 [m]
eccentricity	0.6
inclination	169 [deg]
longitude of ascending node	80 [deg]
argument of pericenter	45 [deg]
true anomaly	15 [deg]
Integration settings	
duration	5 [days]
step-size	fixed
thrust profile	non-constant thrust

Table 6.3: Non-constant thrust profile expressing the thrust in the three directions of the velocity frame as a function of time.

Time [s]	F_x **[N]**	F_y **[N]**	F_z **[N]**
0	0	0.5	1
86400	0.5	1	0.5
172800	1	0.7	0.3
259200	0.8	0.2	0.6
345600	0.5	0.5	0.5
432000	0	0	0

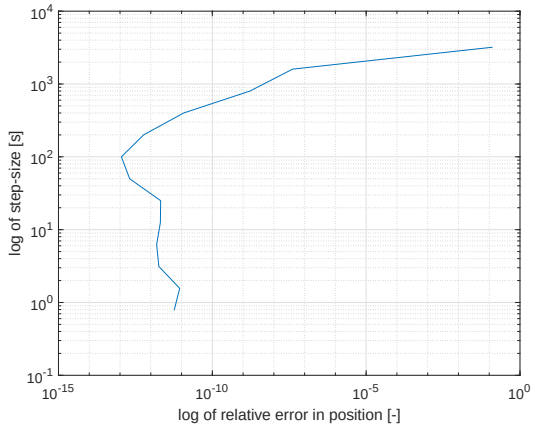


Figure 6.1: Accuracy determination of RK-Cart-TPS in the position after 5 days of integration with non-constant thrust force

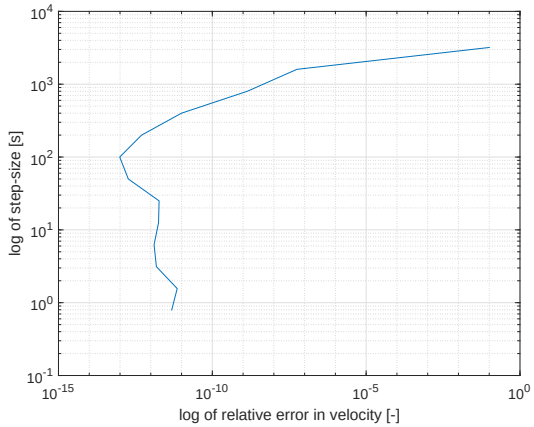


Figure 6.2: Accuracy determination of RK-Cart-TPS in the velocity after 5 days of integration with non-constant thrust force

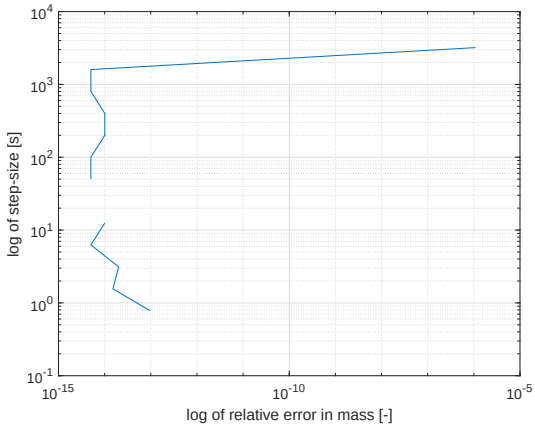


Figure 6.3: Accuracy determination of the RK-Cart-TPS in the mass after 5 days of integration with non-constant thrust force

From the figures it can be seen that the maximum achievable accuracy of the RK-Cart-TPS is $1 \cdot 10^{-11}$. Below this value, the difference between one solution and the next does not decrease monotonically with decreasing step-sizes. Instead, for step-sizes below 100 s the function has a seemingly random shape. A second run of the comparison confirmed that the shape of the lower left part of the curve is random and originates from

machine error. The reason for a similar random shape to occur in both Figs. 6.1 and 6.2 is that the position and velocity are calculated from the three hodograph elements of the USM state vector. Therefore both the Cartesian position and velocity will mimic the random machine error pattern induced by the numerical integration of the USM state vector. It can be concluded that the RK-Cart-TPS solution can be used as a highly accurate solution to validate the GTSI-USM solution up to a relative accuracy level of $1 \cdot 10^{-11}$ in the position and velocity and $1 \cdot 10^{-13}$ for the mass. To envision these accuracy values, remember that the values result from a comparison of the position and velocity after 5 days of propagation under the influence of a non-constant thrust force. If this accuracy level would be scaled to GTOC3 proportions, we would arrive at an absolute accuracy of $10^{-11} \cdot 1$ AU ≈ 1.5 m in position, $10^{-11} \cdot 40$ km/s $= 0.0004$ mm/s in velocity and $10^{-13} \cdot 2$ kg ≈ 0.2 ng after 5 days of propagation, which is definitely sufficient.

6.5. VERIFICATION OF GTSI-USM

To verify the GTSI-USM, a quantitative comparison of the output of the GTSI-USM with output from Matlab was performed and is described in Section 6.5.1. This is followed by a qualitative test of the thrust direction in Section 6.5.2. Note that the full tests of the accuracy and computational efficiency of the GTSI-USM will be provided in the next chapter as these tests are part of the validation of the GTSI-USM and are referred to as the results of thesis.

6.5.1. UNIT TESTS

The GTSI-USM was first implemented in Matlab and then converted to C++. The advantage of C++ over Matlab is twofold. First, the C++ language allows faster propagation, especially due to the 'pass-by-reference' functionality which allows the input of C++ functions to be passed by providing the location in the computer memory where the input is stored,k instead of copying the input. Second, the C++ code is more universal, which allows the implementation of the code in TUDAT, where future students can benefit from. The Matlab GTSI-USM code was built upon Vittaldev's TSI-USM code which he developed in his master's thesis and which was made available to the author [Vittaldev, 2010]. Of course, Vittaldev's code has been extensively verified in his thesis. It was decided to convert the GTSI-USM Matlab code to C++ before starting to verify it. This is because the change from Matlab to C++ would drastically change the structure of the code and therefore it would not be advantageous to verify the code in Matlab first, as this would not guarantee the verification of the C++ code. After the code is fully implemented in C++, the output of the code was compared to the output of the Matlab program in order to check whether the code was converted correctly. In order to perform this comparison, the following unit test functions were created:

- unitTestFrameTransformation.cpp
- unitTestLagrangeInterpolator.cpp
- unitTestProblemRecurrenceRelations.cpp
- unitTestStepSizeControlTSI.cpp
- unitTestTaylorSeriesIntegrator.cpp

The verification was successful. All tested cases (non-zero thrust, constant tangential thrust, constant normal thrust, constant thrust normal to the orbital plane and non-constant thrust in varying direction) were accurate to a relative error of at least 10^{-12}. To judge the relative accuracy, imagine an absolute accuracy limit of 1 km in position and 1 m/s in velocity for orbits that stay within a 2 AU radius from the Sun. This sets a corresponding relative accuracy limit of 3.3e-9 for the position. The relative accuracy constraint for the velocity (in order to meet a absolute accuracy of 1 m/s) is less imposing. Since the detected relative accuracy of 10^{-12} is much lower than this value, the code was hereby declared to be verified. This leaves the validation of the GTSI-USM in which the output of the GTSI-USM will be compared to a reference solution.

Ideally, one has an exact reference trajectory available. Unfortunately, although shape-based methods and the method of manufactured solutions (which will be discussed in Section 6.7) can produced exact reference solutions, they are not suited for the propagation of a spacecraft subject to a low-thrust force of choice. What does exist is an analytical solution for the unperturbed case, i.e. a zero thrust force. Therefore the validation process is divided into two steps. First is the comparison of the TSI against the analytical solution for the zero

thrust case and the results of this test will be given in the next section. The test will only validate the part of the GTSI-USM propagator not involving the implementation of the thrust force. To validate the implementation of the thrust force, a qualitative test of the thrust direction is provided in the section below. A detailed validation of the GTSI-USM will be presented in the Chapter 7. There, the output of the GTSI-USM will be compared against the output of the validated RK-Cart-TPS. As this validation will determine the performance of the GTSI-USM, it will interpreted as the result of this thesis.

6.5.2. THRUST DIRECTION TEST

This section shows a qualitative test of the thrust direction for the GTSI-USM. The tested thrust profiles are given in Tables 6.4 to 6.7 and correspond to the trajectories depicted in Figs. 6.4(a) to 6.4(d), respectively.

Table 6.4: Constant tangential thrust profile defined in the velocity frame

Time [s]	F_x **[N]**	F_y **[N]**	F_z **[N]**
0.0000000e+00	1.5000000e-01	0.0000000e+00	0.0000000e+00
3.1557600e+07	1.5000000e-01	0.0000000e+00	0.0000000e+00
6.3115200e+07	1.5000000e-01	0.0000000e+00	0.0000000e+00
9.4672800e+07	1.5000000e-01	0.0000000e+00	0.0000000e+00
1.2623040e+08	1.5000000e-01	0.0000000e+00	0.0000000e+00
1.5778800e+08	1.5000000e-01	0.0000000e+00	0.0000000e+00
1.8934560e+08	1.5000000e-01	0.0000000e+00	0.0000000e+00
2.2090320e+08	1.5000000e-01	0.0000000e+00	0.0000000e+00
2.5246080e+08	1.5000000e-01	0.0000000e+00	0.0000000e+00
2.8401840e+08	1.5000000e-01	0.0000000e+00	0.0000000e+00
3.1557600e+08	1.5000000e-01	0.0000000e+00	0.0000000e+00

Table 6.5: Constant normal thrust profile defined in the velocity frame

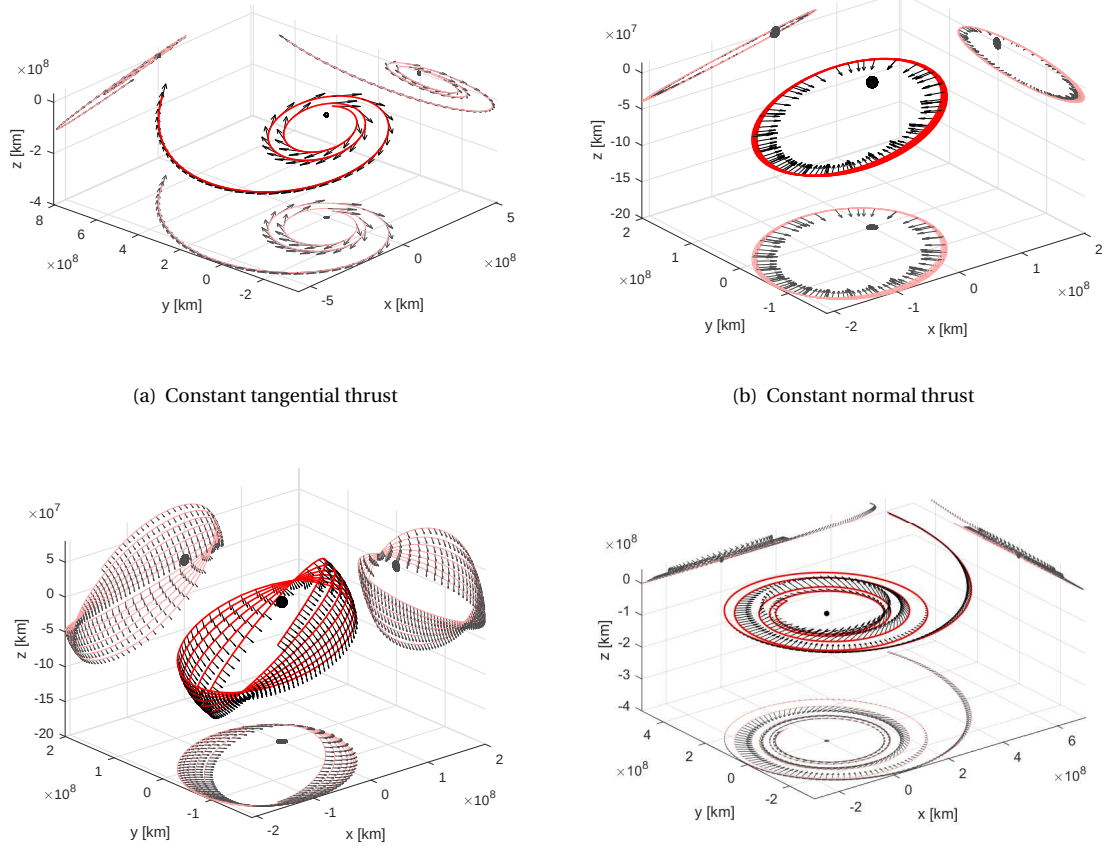
Time [s]	F_x **[N]**	F_y **[N]**	F_z **[N]**
0.0000000e+00	0.0000000e+00	1.5000000e-01	0.0000000e+00
3.1557600e+07	0.0000000e+00	1.5000000e-01	0.0000000e+00
6.3115200e+07	0.0000000e+00	1.5000000e-01	0.0000000e+00
9.4672800e+07	0.0000000e+00	1.5000000e-01	0.0000000e+00
1.2623040e+08	0.0000000e+00	1.5000000e-01	0.0000000e+00
1.5778800e+08	0.0000000e+00	1.5000000e-01	0.0000000e+00
1.8934560e+08	0.0000000e+00	1.5000000e-01	0.0000000e+00
2.2090320e+08	0.0000000e+00	1.5000000e-01	0.0000000e+00
2.5246080e+08	0.0000000e+00	1.5000000e-01	0.0000000e+00
2.8401840e+08	0.0000000e+00	1.5000000e-01	0.0000000e+00
3.1557600e+08	0.0000000e+00	1.5000000e-01	0.0000000e+00

Table 6.6: Constant binormal thrust profile defined in the velocity frame

Time [s]	F_x **[N]**	F_y **[N]**	F_z **[N]**
0.0000000e+00	0.0000000e+00	0.0000000e+00	1.5000000e-01
3.1557600e+07	0.0000000e+00	0.0000000e+00	1.5000000e-01
6.3115200e+07	0.0000000e+00	0.0000000e+00	1.5000000e-01
9.4672800e+07	0.0000000e+00	0.0000000e+00	1.5000000e-01
1.2623040e+08	0.0000000e+00	0.0000000e+00	1.5000000e-01
1.5778800e+08	0.0000000e+00	0.0000000e+00	1.5000000e-01
1.8934560e+08	0.0000000e+00	0.0000000e+00	1.5000000e-01
2.2090320e+08	0.0000000e+00	0.0000000e+00	1.5000000e-01
2.5246080e+08	0.0000000e+00	0.0000000e+00	1.5000000e-01
2.8401840e+08	0.0000000e+00	0.0000000e+00	1.5000000e-01
3.1557600e+08	0.0000000e+00	0.0000000e+00	1.5000000e-01

Table 6.7: Non-constant thrust profile defined in the velocity frame

Time [s]	F_x **[N]**	F_y **[N]**	F_z **[N]**
0.0000000e+00	1.3416408e-01	0.0000000e+00	6.7082039e-02
3.1557600e+07	7.1754731e-02	3.5877365e-03	2.1526419e-02
6.3115200e+07	1.0502101e-01	2.1004201e-02	1.0502101e-01
9.4672800e+07	1.2247449e-01	6.1237244e-02	6.1237244e-02
1.2623040e+08	7.5761441e-02	1.2121831e-01	4.5456865e-02
1.5778800e+08	1.5722923e-02	1.1792192e-01	1.5722923e-02
1.8934560e+08	0.0000000e+00	0.0000000e+00	0.0000000e+00
2.2090320e+08	0.0000000e+00	0.0000000e+00	0.0000000e+00
2.5246080e+08	1.5000000e-01	0.0000000e+00	0.0000000e+00
2.8401840e+08	4.3301270e-02	4.3301270e-02	4.3301270e-02
3.1557600e+08	1.7320508e-02	1.7320508e-02	1.7320508e-02
3.4713360e+08	1.7320508e-02	1.7320508e-02	1.7320508e-02



(a) Constant tangential thrust

(b) Constant normal thrust

(c) Constant binormal thrust

(d) Non-constant thrust in varying direction

Figure 6.4: Three-dimensional representation and projections onto the principal planes of the trajectory of a spacecraft propagated by GTSI-USM in a central gravitational field subject to four different thrust profiles. The arrows indicate the thrust acceleration. The thrust profiles are defined in Tables 6.4 to 6.7. Note that the black sphere, indicating the Sun, is magnified by a factor 10.

From the black arrows in the figure that indicate the magnitude and orientation of the thrust profile, it can be concluded that the thrust profiles are interpreted correctly by the GTSI-USM.

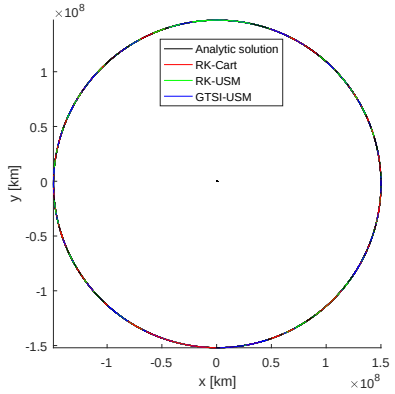
6.6. VERIFICATION FOR ZERO THRUST

The Kepler case addresses the case in which there is no perturbing thrust force or a zero perturbing thrust force. For this case, the trajectories of the GTSI-USM, RK-Cart and RK-USM propagators can be compared with the analytical solution. The comparison is done for two different orbits which are referred to as the Earth orbit and the Asteroid orbit of which the Kepler elements are listed in Table 6.8.

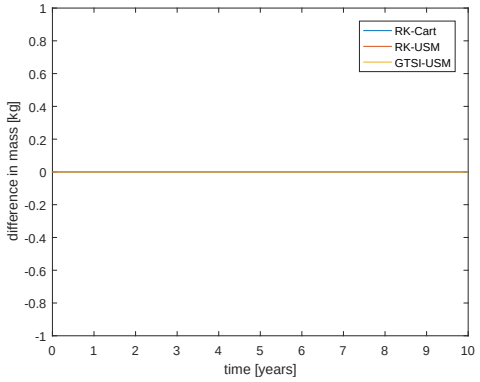
Table 6.8: Definition of the two orbits used to generate the results

	Earth Orbit	**Asteroid Orbit**
a [m]	1.495960829265199e+11	1.495960829265199e+11
e [-]	1.671681163160e-2	0.6
i [deg]	0.8854353079654e-3	169
Ω [deg]	175.40647696473	80
ω [deg]	287.61577546182	45
θ [deg]	0	15

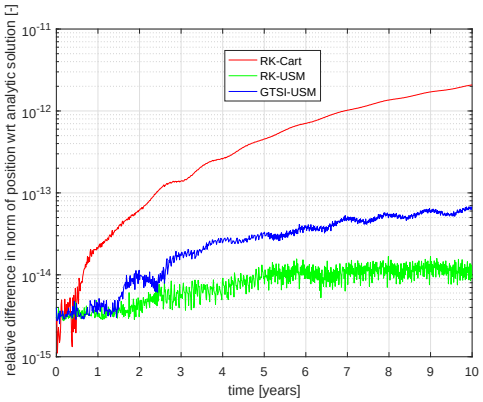
Fig. 6.5 shows the difference between the trajectories propagated with GTSI-USM, RK-Cart and RK-USM, and the analytical solution for the Earth orbit. The difference can be interpreted as the error of the propagator. The graphs for the Asteroid orbit are shown in Fig. 6.6.



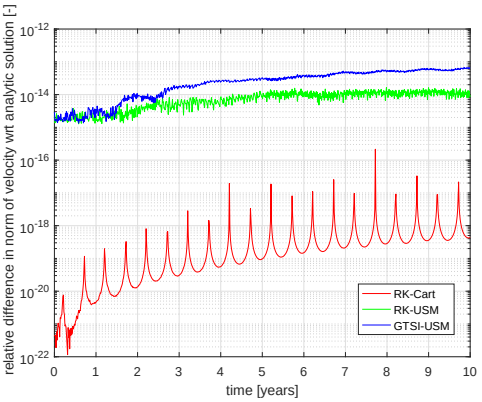
(a) The propagated trajectory resembles Earth's orbit around the Sun projected onto the ecliptic. The difference between the propagated trajectories is so small that they overlap each other.



(b) Relative difference between the propagated spacecraft mass and the exact solution. The mass is constant due to zero thrust.

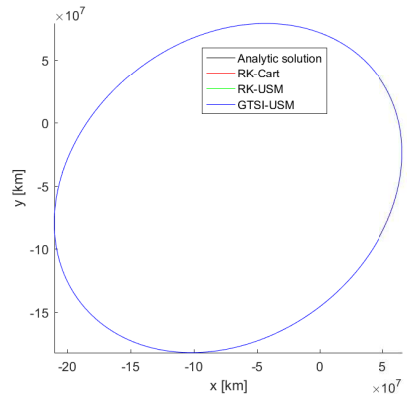


(c) Relative difference in the norm of the position between the propagated trajectory and the analytic solution

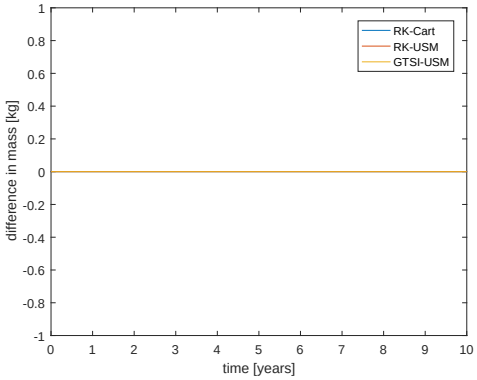


(d) Relative difference in the norm of the velocity between the propagated trajectory and analytic solution

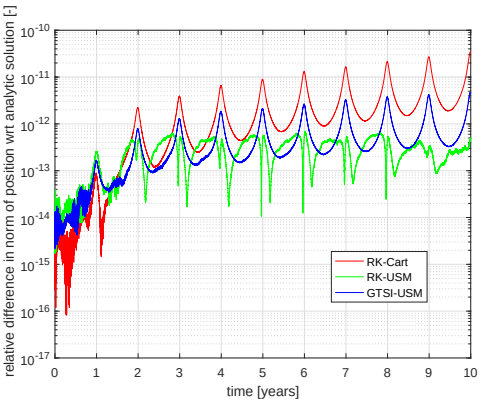
Figure 6.5: Relative difference between the GTSI-USM, RK-Cart and RK-USM propagated trajectories and analytic solution with a fixed step of 4000s for 10 years. The initial conditions are the one of the Earth orbit.



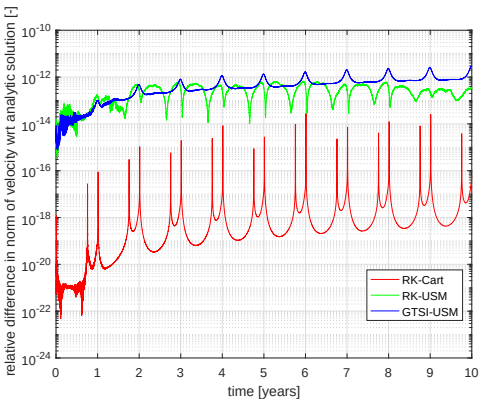
(a) The propagated trajectory using the Asteroid's initial condition projected onto the ecliptic. The difference between the propagated trajectories is so small that they overlap each other.



(b) Relative difference between the propagated spacecraft mass and the exact solution. The mass is constant due to zero thrust.



(c) Relative difference in the norm of the position between the propagated trajectory and the analytic solution



(d) Relative difference in the norm of the velocity between the propagated trajectory and analytic solution

Figure 6.6: Relative difference between the GTSI-USM, RK-Cart and RK-USM propagated trajectories and analytic solution with a fixed step of 4000 s for 10 years. The initial conditions are the one of the Asteroid orbit.

From the figures it can be seen that for both the Earth and Asteroid orbits the relative difference in position and velocity between the propagated trajectories and the analytical solution is below 3.45e-11 which corresponds to a sufficient absolute error of well under 1 km and 1 m/s. Furthermore, the mass is propagated exactly. Therefore, the GTSI-USM, RK-Cart and RK-USM are declared to be validated for the Kepler case. Note that the RK-USM and RK-Cart will not be used to propagate the trajectory of the spacecraft subject to perturbing forces, so to save time, they will not be verified for the thrust case. Before moving to the applied changes to the GTSI-USM after a first design iteration of the GDFM, the next section will discuss a useful method to produce exact analytic solutions for the thrust case.

6.7. METHOD OF MANUFACTURED SOLUTIONS

The Method of Manufactured Solutions (MMS) can be used to generate exact reference solutions for any type of partial differential equations, on any type of problem domain. [Roache, 2002; Roy, 2005]. In this method an exact solution is manufactured in the sense that the one assumes a solution and finds the corresponding problem. In this thesis the manufactured solution represents a trajectory. The method will become clear in the following text in which a manufactured trajectory is generated that can later be used for the validation of the developed propagators.

First, a trajectory is assumed. In order to use the trajectory to validate the propagators for the non-Kepler case, a non-Kepler trajectory (i.e. a perturbed orbit) must be chosen. This assures that in order to follow the assumed trajectory, some non-zero thrust is required. The following solution is assumed in Cartesian coordinates:

$$x = a\cos\left(n_f t\right) \tag{6.11}$$
$$y = a\sin\left(n_f t\right) \tag{6.12}$$
$$z = 0 \tag{6.13}$$
$$\tag{6.14}$$

This represents a circular-shaped orbit in the x-y plane. Now, the mean motion factor n_f is determined such that it does not quite equal the mean motion of a circular orbit, this will force the problem to demand a non-zero thrust. Hence a factor of 0.9 in the following equation:

$$n_f = n \cdot 0.9 = \frac{2\pi}{T} \cdot 0.9 \tag{6.15}$$

Using

$$T = 2\pi\sqrt{\frac{a^3}{\mu_S}} \tag{6.16}$$

results in

$$n_f = 0.9\sqrt{\frac{\mu_S}{a^3}} \tag{6.17}$$

At this point, the equations for the velocity can be obtained by derivation of Eq. (6.14):

$$v_x = \dot{x} = -an_f\sin\left(n_f t\right) \tag{6.18}$$
$$v_y = \dot{y} = an_f\cos\left(n_f t\right) \tag{6.19}$$
$$v_z = \dot{z} = 0 \tag{6.20}$$

The initial Cartesian state ($t = 0$) becomes:

$$x(0) = a \tag{6.21}$$
$$y(0) = 0 \tag{6.22}$$
$$z(0) = 0 \tag{6.23}$$
$$v_x(0) = 0 \tag{6.24}$$
$$v_y(0) = an_f \tag{6.25}$$
$$v_z(0) = 0 \tag{6.26}$$

The equations of motion for the accelerations are:

$$\ddot{x} = \mu_S\frac{x}{r^3} + f_x \tag{6.27}$$
$$\ddot{y} = \mu_S\frac{y}{r^3} + f_y \tag{6.28}$$
$$\ddot{z} = \mu_S\frac{z}{r^3} + f_z \tag{6.29}$$
$$\tag{6.30}$$

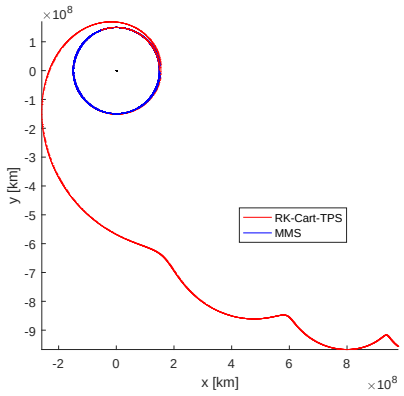
Substituting Eq. (6.14) inside the equations of motions and solving for the thrust accelerations f yields:

$$f_x = x\left(\frac{\mu_S}{r^3} - n_f^2\right) \tag{6.31}$$
$$f_y = y\left(\frac{\mu_S}{r^3} - n_f^2\right) \tag{6.32}$$
$$f_z = y\left(\frac{\mu_S}{r^3} - n_f^2\right) \tag{6.33}$$

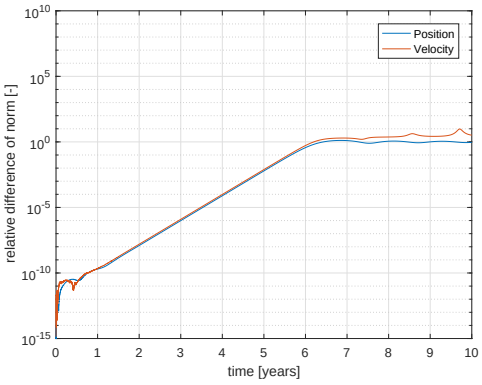
Subsequently the thrust accelerations are calculated for a series of time points and multiplied by the initial mass of the spacecraft to yield the thrust force. These values are then used to construct the thrust profile. The thrust profile together with the initial Cartesian state can now be used as input for the propagator. If the propagator works correctly, the output trajectory should be equal to the assumed trajectory in Eq. (6.14).

It is important to note that the mass is assumed to be constant in this method. This simplifies the relation between the thrust acceleration and the thrust force that is required to compute the thrust profile. This assumption is responsible because the mass propagation has already been validated on its own.

Although this method provides a beautiful way to produce exact reference solutions for perturbed trajectories, it was not used for the verification of the GTSI-USM. The reason for this is that after having applied the MMS to the RK-Cart-TPS it was observed that the difference between the manufactured 'exact' trajectory and the trajectory of the RK-Cart-TPS becomes very large after two or three orbital periods, as can be seen in Fig. 6.7. After a reflection on this method, it became clear that the large difference is due to the amplification of numerical error in the trajectory of RK-Cart-TPS. Since the thrust profile, computed with MMS, only results in an circular orbit if the spacecraft *remains* in a circular orbit, any small deviation from a circular orbit (due to the numerical errors) will lead to larger deviations from the circular orbit. To solve this problem, the RK-Cart-TPS should be adapted to compute the input thrust profile during the integration. Since the MMS became available only at the final stage of the thesis, insufficient time was available to make the required changes to the RK-Cart-TPS.



(a) Trajectory of the spacecraft around the Sun projected onto the ecliptic



(b) Relative difference in the norm of the position and velocity over time

Figure 6.7: Difference between the propagated trajectory of the manufactured solution and the RK-Cart-TPS for the thrust force as specified in Eq. (6.33) for a fixed step-size of 100 s for an integration time of 10 years.

6.8. CORRECTIONS AFTER FIRST DESIGN OF GTSI-USM

Before moving to the validation of the GTSI-USM in case of perturbed orbits in the next chapter, a first design iteration of the GTSI-USM was performed that was based on the provisional output of the GTSI-USM. In this iteration the following corrections or ideas were tested.

6.8.1. SEPARATED SUMMATION OF POSITIVE AND NEGATIVE TERMS

In this section the separate summation of the positive and negative terms in the Taylor series will be tested. In [Scott and Martini, 2008] it is described that this can avoid the cancellation of significant digits and hence increase the accuracy of the propagation. For this purpose a version of GTSI-USM was developed for which the positive and negatives terms in the Taylor series expansion are first summed separately before being added up. The GTSI-USM version is used to propagate a trajectory under the influence of a non-constant thrust of varying direction for 10 years with a fixed step-size of 400 s and order 20.
Comparison of the accuracy of the propagator before and after the implementation of the separate summation of the positive and negative terms of the Taylor series showed that this does not improve the accuracy of the GTSI-USM propagator.

6.8.2. CORRECTION OF INITIAL CONDITIONS

In GTSI-USM the initial conditions were input in USM elements, whereas in the RK-based propagators the initial state was given in Kepler elements and immediately transformed to either USM elements (i.e. in case of RK-USM) or Cartesian coordinates (i.e. in case of RK-Cart and RK-Cart-TPS). Although the state in initial USM and Kepler elements were referring to the exact same state, the transformation between them was performed using $\mu_S = 1.3271244 \cdot 10^{20}$, whereas the central gravitational field of the propagators was set to $\mu_S = 1.32712440018 \cdot 10^{20}$ [Casalino et al., 2007]. The difference in those two values for μ_S caused a large difference between the propagated states after ten years. It was corrected by recalculation of the initial USM elements from the Kepler elements with $\mu_S = 1.32712440018 \cdot 10^{20}$.

6.8.3. ALTERNATIVE TIME-SCALE FOR THE MASS PROPAGATION

In this section it will be tested if the use of an alternative time-scale would allow the reduction of the effect of machine error on the results. The alternative time-scale will be tested for the propagation of the mass.

The Taylor series expansion for the mass around time point t_0 is:

$$m(t) = m(t_0) + m'(t_0)(t-t_0) + \frac{1}{2}m''(t_0)(t-t_0)^2 + \frac{1}{6}m'''(t_0)(t-t_0)^3 + \dots \tag{6.34}$$

in which

$$m'(t_0) = \frac{-F(t_0)}{g_0 I_{sp}} \tag{6.35}$$

$$m''(t_0) = \frac{-F'(t_0)}{g_0 I_{sp}} \tag{6.36}$$

$$\tag{6.37}$$

Since the mass of the spacecraft should always be decreasing while thrusting, $m(t)$ should always be smaller than $m(t_0)$ for $t > t_0$ in case of non-zero thrust. This results in the following condition:

$$m'(t_0)(t_0 - t) + \frac{1}{2}m''(t_0)(t-t_0)^2 + \dots < 0 \tag{6.38}$$

Since F is the magnitude of the thrust force, $m'(t_0)$ will always be negative. But $m''(t_0)$ can be positive or negative because $F'(t_0)$ can be positive or negative. The value of the higher order derivatives of the mass become smaller for higher order terms and are eventually zero because of Eq. (5.21e). In each term, this small value is multiplied by $(t-t_0)^n$ where n indicates term of the TSI. When the step-size is larger than one, this value becomes very large for higher order terms. It is known that the multiplication of a very small value with a very large value can amplify the effect of machine error [Press et al., 2007]. Due to this machine error, the value of a term in the TS can become larger in magnitude than the second term, which is negative. If at the same time, the sign of the higher order derivative of m is positive, this results in the violation of the condition expressed in Eq. (6.38).

To avoid this problem, the use of an alternative time-scale for the Taylor Series of the mass is tested. Let us define the alternative time τ as

$$\tau = \frac{t}{h} \tag{6.39}$$

Next τ is used for the TS expansion of the mass:

$$m(\tau) = m(\tau_0) + m'(\tau_0)(\tau - \tau_0) + \frac{1}{2}m''(\tau_0)(\tau-\tau_0)^2 + \dots \tag{6.40}$$

Now with the use of Eq. (6.39) $\tau - \tau_0$ equates to:

$$\tau - \tau_0 = \frac{t}{h} - \frac{t_0}{h} = \frac{t-t_0}{h} = \frac{h}{h} = 1 \tag{6.41}$$

As can be seen, due to the smart definition of τ, the factor $(\tau-\tau_0)^n$ becomes 1 for every term of Eq. (6.40), hence simplifying it to:

$$m(\tau) = m(\tau_0) + m'(\tau_0) + \frac{1}{2}m''(\tau_0) + ... \tag{6.42}$$

In Eq. (6.42) there is no multiplication of very small values by very large values anymore, which reduces the effect of machine error. This was of course the purpose of introducing the alternative time τ. Now one problem remains: in Eq. (6.42) the derivatives of m are with respect to τ are unknown.

The derivatives of m with respect to τ can be derived as follows:

$$\frac{dm}{d\tau} = \frac{dm}{dt}\frac{dt}{d\tau} \tag{6.43}$$

Reforming Eq. (6.39) yields:

$$t = \tau h \tag{6.44}$$

Now the derivative of t with respect to τ becomes

$$\frac{dt}{d\tau} = \frac{d(\tau h)}{\tau} = h \tag{6.45}$$

Substituting this result back into Eq. (6.43) yields

$$\frac{dm}{d\tau} = \frac{-F}{g_0 I_{sp}}h \tag{6.46}$$

Applying the same method for the second derivative of m with respect to τ results in

$$\frac{d^2m}{d\tau^2} = \frac{-F'}{g_0 I_{sp}}h^2 \tag{6.47}$$

This leads to the following general equation:

$$\frac{d^n m}{d\tau^n} = \frac{-F^{(n-1)}}{g_0 I_{sp}}h^n \tag{6.48}$$

in which $F^{(n)}$ represents the n^{th} derivative of F with respect to t. However, the problem that has been avoided by introducing τ is reintroduced in Eq. (6.48) through the multiplication of $-F^{(n-1)}$ by h^n. Indeed, $F^{(n-1)}$ will be small for large n and h will be large for large n. Apparently, using a different time-scale for the propagation of the mass and by extension every other state variable does not help to limit the amplification of the machine error. No confirmation of this conclusion could be found in literature.

7

VALIDATION & RESULTS

In Chapter 1 it has been identified that the combination of Taylor Series Integration (TSI) and Unified State Model elements (USM) is a fast method for the propagation of spacecraft trajectories. Furthermore, it has been shown that the generalized TSI-USM (GTSI-USM) is a problem-specific method and that this is detrimental for its use in evolutionary algorithms for the purpose of optimizing low-thrust interplanetary trajectories. Chapter 4 clarified the cause of this problem-specificity. In Chapter 5 a method was developed to generalize TSI-USM for this purpose, that led to the generalized TSI-USM (GTSI-USM). After the implementation of the GTSI-USM propagator in C++, it has been verified for the zero thrust case in the previous chapter. Also in the previous chapter, three RK8(7)13M-based propagators were introduced and verified. In this chapter, these reference propagators will be used to determine the accuracy and computational efficiency of the developed GTSI-USM. The results of the accuracy tests also serve as the validation of the GTSI-USM for the non-zero thrust case. The conclusions based on the results presented in this chapter, will be provided in the Chapter 8. The final chapter of this thesis contains the recommendations for future work as well as some tips for future master students.

Contents

7.1. INTRODUCTION

As explained in the previous chapter, three RK8(7)13M-based propagators have been implemented to determine the accuracy and the computational efficiency of the developed generalized TSI-USM (GTSI-USM) propagator: RK8(7)13M combined with Cartesian coordinates (RK-Cart), RK8(7)13M combined with USM elements (RK-USM) and a version of the RK-Cart in the Tudat propagation setup (RK-Cart-TPS). In Section 7.4, the RK-Cart-TPS, which has been completely verified for all thrust cases, is used to determine the accuracy of the GTSI-USM for a range of thrust cases. A discussion of these results is provided in Section 7.4.8. Besides the accuracy of the GTSI-USM, the computational efficiency is an important parameter that will determine the usefulness of GTSI-USM compared to more common RK-based propagators. The computational efficiency depends on the combination of accuracy and CPU time. A distinction is made between the unperturbed and perturbed orbit cases. It will be seen that the main result is that the GTSI-USM propagator requires more CPU time for a given accuracy than the RK8(7) based propagators. Besides the main results of the GTSI-USM, several additional results are presented in Sections 7.6 to 7.10, including an analysis of the optimal order of the TSI and a CPU time per integration step comparison between the implemented propagators. Prior to presenting the results of this thesis, the method used to measure the performance of the GTSI-USM as well as an overview of the propagation settings are provided in Sections 7.2 and 7.3, respectively.

7.2. PERFORMANCE MEASUREMENT

The performance of the GTSI-USM will be determined by its accuracy and its computational efficiency, of which the latter in turn depends on the combination of accuracy and CPU time. The following two sections explain how the accuracy and CPU time will be determined.

7.2.1. ACCURACY MEASUREMENT

The accuracy of a propagator is determined by the difference between the computed trajectory and a reference trajectory. If the method used to produce the reference trajectory is validated, the difference between the two trajectories can be regarded as the error of the propagator. The relative and absolute error of state variable x_n can be computed at every point on the computed trajectory using the reference state variable $x_{n,\text{ref}}$:

$$\varepsilon_{\text{abs}}(t) = \left| x_n(t) - x_{n,\text{ref}}(t) \right| \tag{7.1}$$

$$\varepsilon_{\text{rel}}(t) = \frac{\epsilon_{abs}}{\left| x_{n,\text{ref}}(t) \right|} \tag{7.2}$$

Where the error is used to quantize the accuracy of the propagator at any point on the trajectory, the Root Mean Square (RMS) error is used to express the error of a entire trajectory in one number. The RMS error in position for a trajectory with radius $\boldsymbol{r}(t_i) = [x_i\ y_i\ z_i]$ where $i = 1, \ldots, k$ with k being the number of nodes on the trajectory, is defined as follows:

$$\varepsilon_{\text{RMS, pos}} = \sqrt{\frac{1}{k}\sum_{i=1}^{k}\left(\left(x_i - x_{i,\text{ref}}\right)^2 + \left(y_i - y_{i,\text{ref}}\right)^2 + \left(z_i - z_{i,\text{ref}}\right)^2\right)} \tag{7.3}$$

Indeed, in order to calculate the RMS error in position of a trajectory, the USM state needs to be transformed to Cartesian coordinates in case of GTSI-USM and RK-USM. A trajectory with a sufficient accuracy in position will be defined as a trajectory of which the error in the norm of the position at every point on the trajectory is below 1 km. For a sufficient accuracy in velocity, the error in the norm of the velocity should stay below 1 m/s at all points of the trajectory. These accuracy limits are valid for orbits with a semi-major axis of 1 Astronomical Unit (149 597 871 km) (AU) or higher.

7.2.2. CPU TIME MEASUREMENT

As mentioned before, the computational efficiency of a propagator depends on the variation of the CPU time for different accuracy settings. In the previous section it was explained how the RMS error is used to quantize the accuracy of an entire trajectory. This leaves the measurement of the total accumulated CPU time during a run of the propagator. In C++, this CPU time is measured using the `cstdio` and `ctime` libraries and using `clock_t` type and the `clock()` function inside the `std` namespace. To start the CPU time registration, the following command is used:

```
std::clock_t startCPUtime = std::clock( );
```

The timing is stopped and the required CPU time is stored in the variable `CPUduration`:

```
double CPUduration = ( std::clock( ) - startCPUtime ) / (double)CLOCKS_PER_SEC;
```

This timing method ensures the correct measurement of the CPU time. The CPU time differs from the *wall time* because the latter is the elapsed time, which includes the time spent waiting for the propagator's turn on the CPU while other processes get to run. When multiple CPU cores are used simultaneously, the CPU time can be larger than the elapsed time. However, in case only one CPU core is used during the work, the CPU time usually does not differ much from the wall time. The amount of runs of the propagator that are used to determine the average run time vary from 10 to 10000, depending on the calculation time of a single run. The precision of the `std::clock()` function is one millisecond. Therefore, if a run takes in the order of a millisecond to execute, the relative precision of the `std::clock` function is insufficient and the CPU time is measured over multiple runs.
The CPU time measurements in this chapter are done with a version of the propagator for which the state history is not stored. This is purposely done to exclude the CPU time required for data storage, as this is expected to be similar for each propagator. An exception to this is Section 7.6 in which the effect of data storage on the CPU time will be revealed.

7.3. OVERVIEW OF PROPAGATION SETTINGS

This section accommodates the propagation settings that will be used to generate the results presented in this chapter. The settings will be accompanied by a rationale. For each section of this chapter, the most important settings will still be provided in a table in that section.

7.3.1. INITIAL STATES

The trajectory of a spacecraft always needs to be propagated from a starting point in space, i.e. the initial state of the spacecraft. Two initial states are considered: the Earth state and the Asteroid state. All results presented in this chapter are generated for either one of these initial states. The Kepler elements of these initial states are given in Table 7.1. The definition of the Earth state is taken from the official GTOC3 problem statement [Casalino et al., 2007]. The semi-major axis of the Asteroid state is taken from that of the Earth state. The other values are different to create a more challenging orbit to propagate. The names *Earth orbit* and *Asteroid orbit* will be used throughout this chapter to refer to the orbits that result from the corresponding Earth state and Asteroid state. For the conversion of the Kepler elements to Cartesian coordinates and USM elements the following value for the gravitational parameter is used: $1.32712440018 \cdot 10^{20}$. This is the Sun's gravitational parameter as listed in [Casalino et al., 2007].

Table 7.1: Definition of the two orbits used to generate the results

Kepler element	Earth state	Asteroid state
a [m]	1.495960829265199e+11	1.495960829265199e+11
e [-]	1.671681163160e-2	0.6
i [deg]	0.8854353079654e-3	169
Ω [deg]	175.40647696473	80
ω [deg]	287.61577546182	45
θ [deg]	0	15

7.3.2. SPACECRAFT CHARACTERISTICS

Besides the celestial coordinates of the spacecraft, the spacecraft's state vector also contains its mass. The initial mass is set to 2000 kg. Another spacecraft characteristic is the specific impulse of its low-thrust propulsion system. The specific impulse is set to 3000 s. Both the values of the mass and the specific impulse are adopted from [Casalino et al., 2007].

7.3.3. INTEGRATION SETTINGS

The CPU time measurements of the three different propagators can be done for two settings of the step-size. The step-size is either fixed or variable. The values for the fixes step-sizes are chosen such that they correspond to an integer amount of steps and increase more or less exponentially with base 2. In case the step-size is variable, the absolute tolerance setting is varied. Table 7.2 lists the values for the step-size and absolute tolerances that are used in the tests presented in this chapter. Note that for the GTSI-USM, the tolerances for the quaternion are three orders of magnitude lower than the ones shown in the table. This is done to account for the fact that the quaternion elements are much smaller than the other USM elements.

Table 7.2: List of the step-sizes and absolute tolerances used to produce the results presented in this chapter. (*)These tolerances are only used for the TSI order comparison in Section 7.9.

Step-size [s]	**Absolute tolerance [m]**
1000	1e-5
4000	1e-6*
8000	1e-7
15778.8	1e-8*
31557.6	1e-9
252460.8	1e-11
504921.6	1e-13
1008230.032	1e-15
2010038.217	
4045846.154	
8304631.579	
16609263.16	
31557600	

It is important to note that the orbits shown in these chapter are propagated over a period of 10 years ($10 \cdot 365.25 \cdot 86400 = 315576000$ s), unless stated otherwise.
In case of propagation with variable step-size control, three more integration settings are to be defined. First, the step-size safety factor is the safety factor that is multiplied with the computed next step-size to decrease the risk of violating the set tolerances. Furthermore, to avoid rapid changes between two consecutive step-sizes, a maximum increase and a minimum decrease factor are installed. The values of these factors are given in Table 7.3. Note that these factors are not used in case of fixed step-sizes.

Table 7.3: Integration settings for variable step-size control. (*) These factors are not used in the generalized TSI-USM.

Factor	**Value**
Step-size safety factor	0.9
Maximum step-size increase factor*	4.0
Minimum step-size decrease factor*	0.1

In case of the GTSI-USM propagator, it is possible to compute the state history at intermediate steps without the loss of accuracy. Although, this capability is implemented in the generalized TSI-USM, it is not used. This means that the refinement factor, i.e. the factor representing the amount of intermediate steps that have to be computed, is set to zero.
Lastly, in case of TSI it is possible to choose the order of Taylor series expansions. Unless explicitly mentioned otherwise, the order will be 20. In literature, this value is either found to be optimal for the propagation of

spacecraft trajectories, or close to the optimal order. The order is deliberately not optimized in order not to bias the comparison of the computational efficiency in Section 7.5. Indeed, the CPU time required to optimize the TSI order optimization would otherwise have to be accounted for in the overall CPU time.

7.4. VALIDATION OF GTSI-USM

In this section the accuracy of the generalized TSI-USM (GTSI-USM) will be tested. The validation of the GTSI-USM is an important result of this thesis.

The accuracy tests are distributed over the next seven subsections, each containing the accuracy test of a different thrust scenario. The seven scenarios are: the propagation of the trajectory of a spacecraft in a central gravitational field subject to

- a constant tangential thrust force,
- a constant normal thrust force,
- a constant binormal thrust force,
- a non-constant tangential thrust force,
- a non-constant normal thrust force,
- a non-constant binormal thrust force,
- and a non-constant thrust force in varying direction.

In all tests the accuracy of the GTSI-USM is calculated from the difference between the GTSI-USM trajectory and the RK-Cart-TPS trajectory. Since the latter one has been verified up to a relative accuracy of 1e-11 in position and velocity (see Section 6.4), the difference between the trajectories can be considered as the error of the GTSI-USM.

Note that the GTSI-USM has already been verified for the non-zero thrust scenario (Section 6.6).

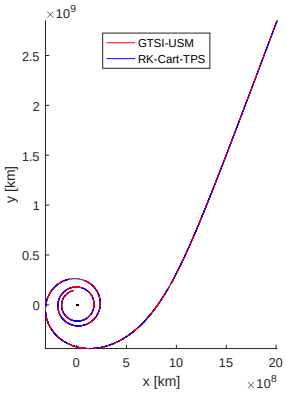
7.4.1. CONSTANT TANGENTIAL THRUST

The first considered thrust case is one where the thrust is constant and purely tangential (i.e. in x-direction of the velocity frame, see Section 2.3.2) at every instant of the trajectory. The norm of the thrust vector will equal 0.15 N, which is the maximum allowed thrust force in the reference GTOC3 problem. The propagation settings are the following:

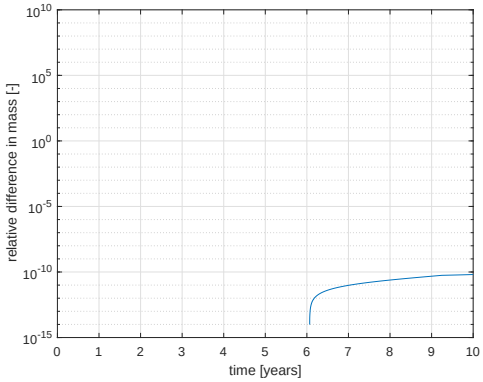
Table 7.4: Propagation settings for GTSI-USM and RK-Cart-TPS

Parameter	Value
initial state	Earth orbit
duration	10 [years]
step-size	400 [s] (fixed)
TSI order	20
refinement factor	0
thrust profile	constant tangential thrust

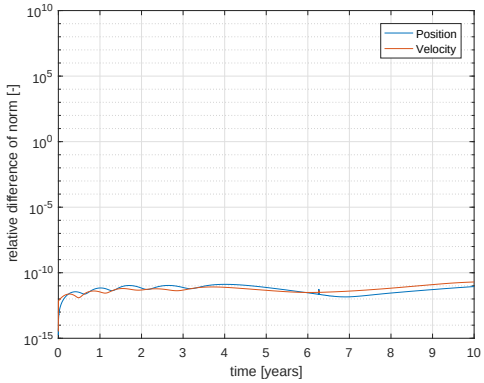
The figures below show the difference between the propagated trajectories of GTSI-USM and RK-Cart-TPS for a 10 year period under the influence of the gravitational attraction of the Sun, and a constant tangential thrust force of 0.15 N. Since the RK-Cart-TPS is validated (see Section 6.3), the difference may be interpreted as the error of the GTSI-USM.



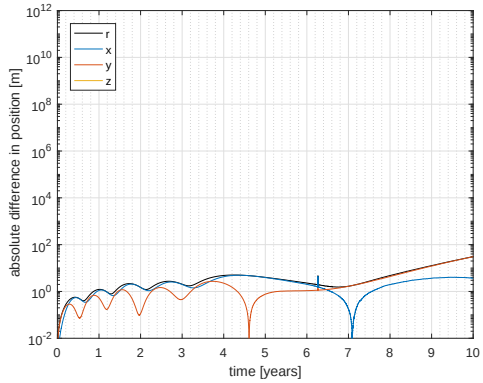
(a) Trajectory of the spacecraft around the Sun projected onto the ecliptic



(b) Relative difference in the spacecraft mass over time



(c) Relative difference in the norm of the position and velocity vectors over time



(d) Absolute difference in the position with $r = \sqrt{x^2 + y^2 + z^2}$

Figure 7.1: Difference between the propagated trajectory of the GTSI-USM and the RK-Cart-TPS for a constant tangential thrust force. The integration settings are given in Table 7.4

From the figures, it can be concluded that the relative accuracy of the GTSI-USM is at least 2e-11 in position and velocity, and at least 6.5e-11 in mass after 10 years of propagation. This meets the predetermined absolute limits of 1 km in position and 1 m/s in velocity (see Section 7.2). Therefore, the GTSI-USM is declared to be validated for the constant tangential thrust case.

7.4.2. CONSTANT NORMAL THRUST

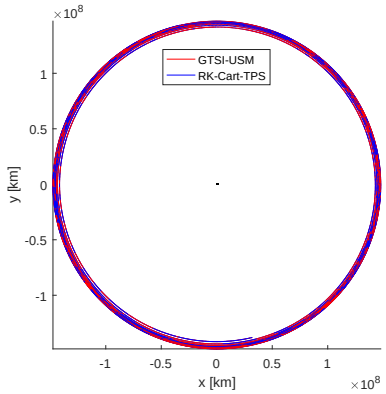
Consider the case where the thrust is constant and purely in normal direction (i.e. in the y-direction of the velocity frame) at every instant of the trajectory. The norm of the thrust vector will again equal 0.15 N. The propagation settings are the following:

Table 7.5: Propagation settings for GTSI-USM and RK-Cart-TPS propagators

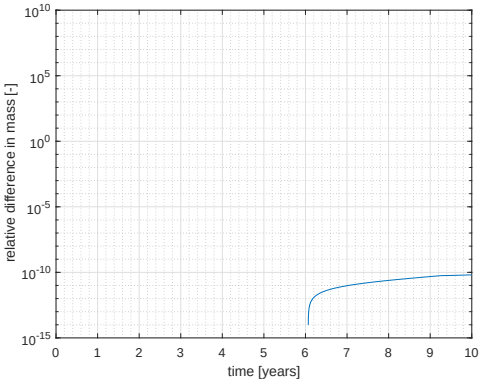
Parameter	Value
initial state	Earth orbit
duration	10 [years]
step-size	400 [s] (fixed)
TSI order	20
refinement factor	0
thrust profile	constant normal thrust

The figures below show the difference between the propagated trajectories of GTSI-USM and RK-Cart-TPS for a 10 year period under the influence of the gravitational attraction of the Sun, and a constant normal

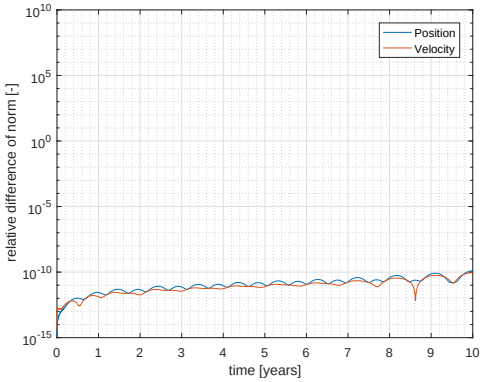
thrust force of 0.15 N. And again, since the RK-Cart-TPS is validated (see Section 6.3), the difference may be interpreted as the error of the GTSI-USM.



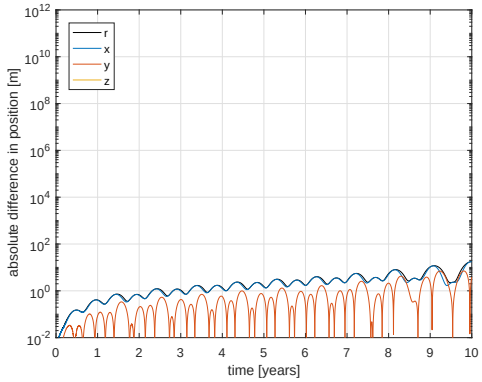
(a) Trajectory of the spacecraft around the Sun projected onto the ecliptic



(b) Relative difference in the spacecraft mass over time



(c) Relative difference in the norm of the position and velocity vectors over time



(d) Absolute difference in the position with $r = \sqrt{x^2 + y^2 + z^2}$

Figure 7.2: Difference between the propagated trajectory of the GTSI-USM and the RK-Cart-TPS for a constant normal thrust force. The integration settings are given in Table 7.5

From the figures, it can be concluded that the relative accuracy of the GTSI-USM is at least 1.3e-10 in position and velocity, and at least 6.5e-11 in mass after 10 years of propagation. When converted to absolute accuracies, this meets the predetermined absolute limits of 1 km in position and 1 m/s in velocity. Therefore, the GTSI-USM is declared to be validated for the constant normal thrust case.

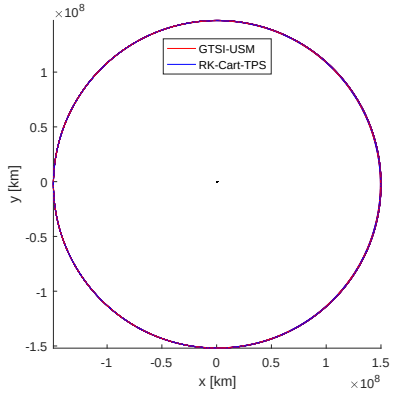
7.4.3. CONSTANT BINORMAL THRUST

Consider the case where the thrust is constant and purely in binormal direction (i.e. in the y-direction of the velocity frame) at every instant of the trajectory. The norm of the thrust vector will again equal 0.15 N. The propagation settings are the following:

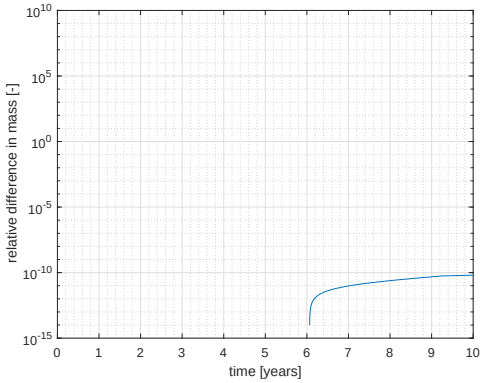
Table 7.6: Propagation settings for GTSI-USM and RK-Cart-TPS propagators

Parameter	Value
initial state	Earth orbit
duration	10 [years]
step-size	400 [s] (fixed)
TSI order	20
refinement factor	0
thrust profile	constant normal thrust

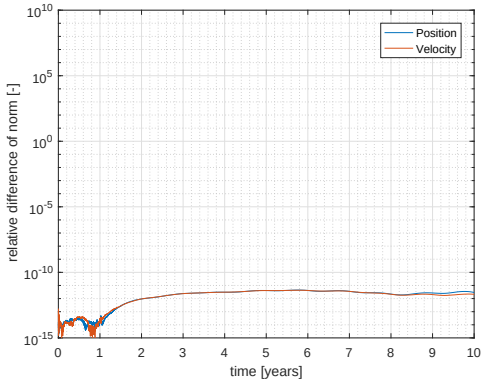
The figures below show the difference between the propagated trajectories of GTSI-USM and RK-Cart-TPS for a 10 year period under the influence of the gravitational attraction of the Sun, and a constant binormal thrust force of 0.15 N. Once again, the difference may be interpreted as the error of the GTSI-USM.



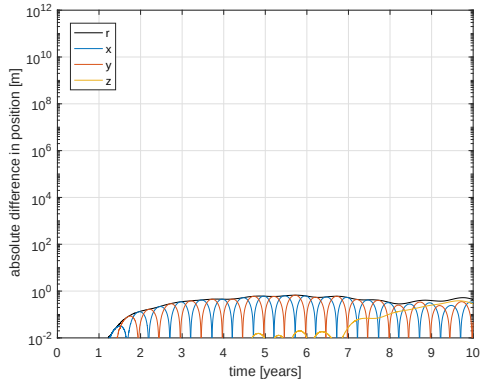
(a) Trajectory of the spacecraft around the Sun projected onto the ecliptic



(b) Relative difference in the spacecraft mass over time



(c) Relative difference in the norm of the position and velocity vectors over time



(d) Absolute difference in the position with $r = \sqrt{x^2 + y^2 + z^2}$

Figure 7.3: Difference between the propagated trajectory of the GTSI-USM and the RK-Cart-TPS for a constant binormal thrust force. The integration settings are given in Table 7.6

From the figures, it can be concluded that the relative accuracy of the GTSI-USM is at least 1e-11 in position and velocity, and at least 6.5e-11 in mass after 10 years of propagation. This meets the predetermined absolute limits of 1 km in position and 1 m/s in velocity. Therefore, the GTSI-USM is declared to be validated for the constant binormal thrust case.

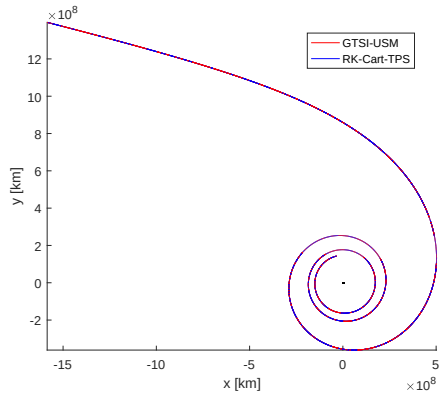
7.4.4. NON-CONSTANT TANGENTIAL THRUST

In this section the non-constant tangential thrust case is considered (i.e. in the x-direction of the velocity frame). The norm of the thrust vector will never exceed 0.15 N. The propagation settings are the following:

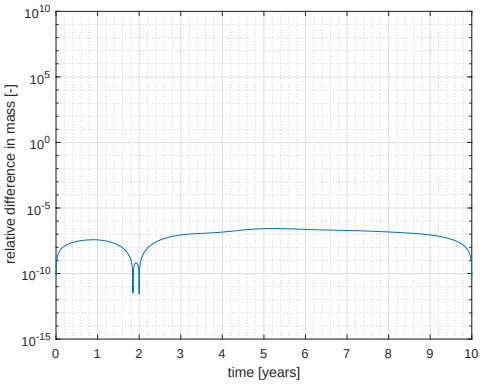
Table 7.7: Propagation settings for GTSI-USM and RK-Cart-TPS propagators

Parameter	Value
initial state	Earth orbit
duration	10 [years]
step-size	400 [s] (fixed)
TSI order	20
refinement factor	0
thrust profile	non-constant tangential thrust

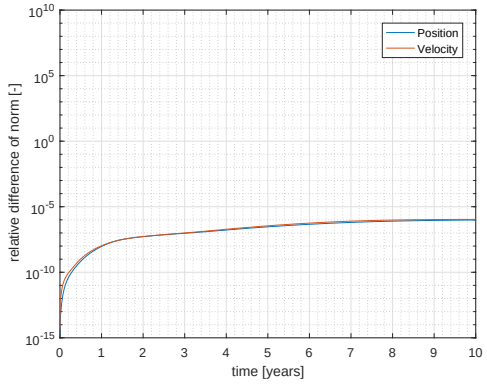
The figures below show the difference between the propagated trajectories of GTSI-USM and RK-Cart-TPS for a 10 year period under the influence of the gravitational attraction of the Sun, and a non-constant tangential thrust force of 0.15 N. The difference may be interpreted as the error of the GTSI-USM.



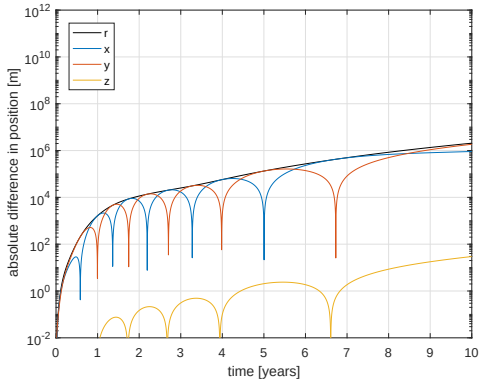
(a) Trajectory of the spacecraft around the Sun projected onto the ecliptic



(b) Relative difference in the spacecraft mass over time



(c) Relative difference in the norm of the position and velocity vectors over time



(d) Absolute difference in the position with $r = \sqrt{x^2 + y^2 + z^2}$

Figure 7.4: Difference between the propagated trajectory of the GTSI-USM and the RK-Cart-TPS for a non-constant tangential thrust force. The integration settings are given in Table 7.7

From the figures, it can be concluded that the relative accuracy is much worse than in the previous cases. The GTSI-USM is only accurate to 9.8e-7 in position and velocity, and at least 2.7e-7 in mass after 10 years of propagation. The difference in the thrust force (not shown in the figure) has a relative accuracy of 8.3e-14. After conversion to absolute accuracy, this does not meet the predetermined limit of 1 km in position (2000 km), but does meet the limit of 1 m/s in velocity (2.8 cm/s). Therefore, the GTSI-USM is not validated for the non-constant tangential thrust case.

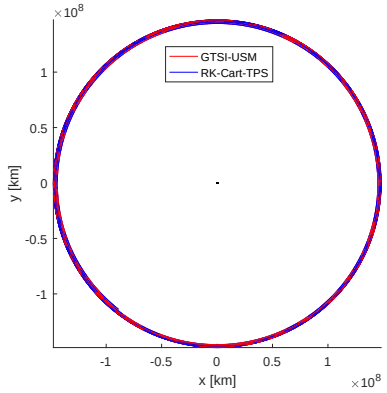
7.4.5. NON-CONSTANT NORMAL THRUST

In this section the non-constant normal thrust case is considered (i.e. in the y-direction of the velocity frame). The norm of the thrust vector will never exceed 0.15 N. The propagation settings are the following:

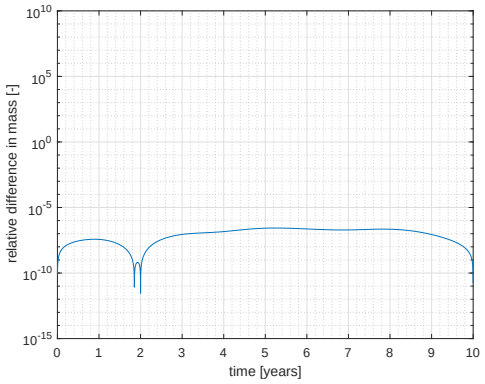
Table 7.8: Propagation settings for GTSI-USM and RK-Cart-TPS propagators

Parameter	Value
initial state	Earth orbit
duration	10 [years]
step-size	400 [s] (fixed)
TSI order	20
refinement factor	0
thrust profile	non-constant normal thrust

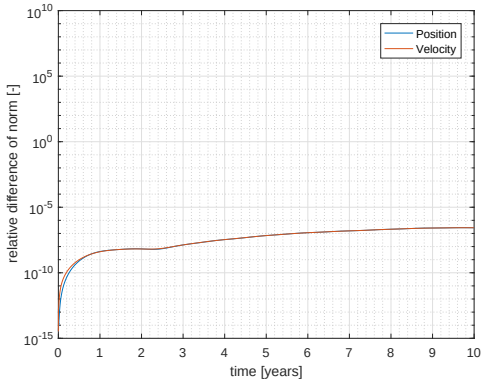
The figures below show the difference between the propagated trajectories of GTSI-USM and RK-Cart-TPS for a 10 year period under the influence of the gravitational attraction of the Sun, and a non-constant normal thrust force of 0.15 N. The difference may be interpreted as the error of the GTSI-USM.



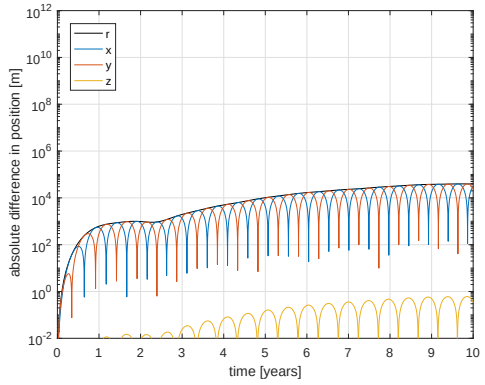
(a) Trajectory of the spacecraft around the Sun projected onto the ecliptic



(b) Relative difference in the spacecraft mass over time



(c) Relative difference in the norm of the position and velocity vectors over time



(d) Absolute difference in the position with $r = \sqrt{x^2 + y^2 + z^2}$

Figure 7.5: Difference between the propagated trajectory of the GTSI-USM and the RK-Cart-TPS for a non-constant normal thrust force. The integration settings are given in Table 7.8

From the figures, it can be concluded that the relative accuracy is rather low. The GTSI-USM is only accurate to 2.8e-6 in position and velocity, and at least 2.7e-7 in mass after 10 years of propagation. The difference in the thrust force (not shown in the figure) has a relative accuracy of 7.0e-14. After conversion to absolute accuracy, this does not meet the predetermined limit of 1 km in position, but does meet the limit of 1 m/s in velocity. Therefore, the GTSI-USM is not validated for the non-constant normal thrust case.

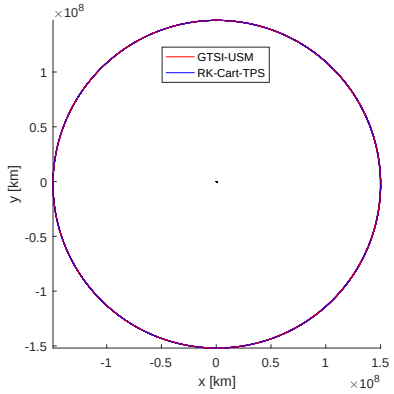
7.4.6. NON-CONSTANT BINORMAL THRUST

In this section the non-constant binormal thrust case is considered (i.e. in the z-direction of the velocity frame). The norm of the thrust vector will never exceed 0.15 N. The propagation settings are the following:

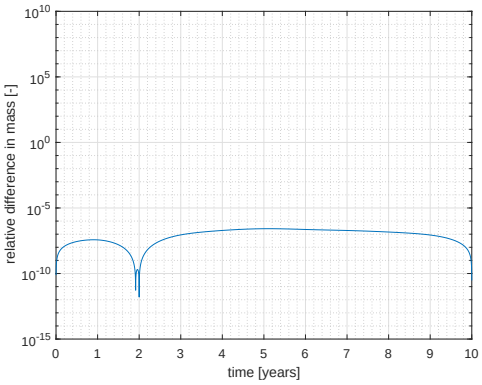
Table 7.9: Propagation settings for GTSI-USM and RK-Cart-TPS propagators

Parameter	Value
initial state	Earth orbit
duration	10 [years]
step-size	400 [s] (fixed)
TSI order	20
refinement factor	0
thrust profile	non-constant binormal thrust

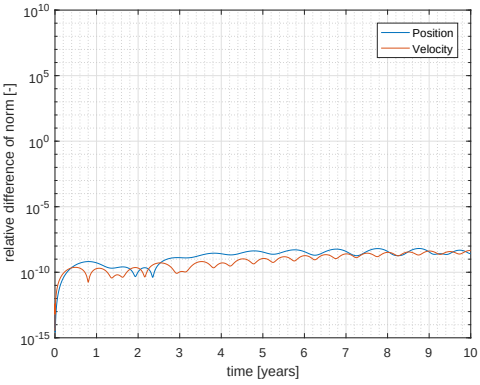
The figures below show the difference between the propagated trajectories of GTSI-USM and RK-Cart-TPS for a 10 year period under the influence of the gravitational attraction of the Sun, and a non-constant binormal thrust force of 0.15 N. The difference may be interpreted as the error of the GTSI-USM.



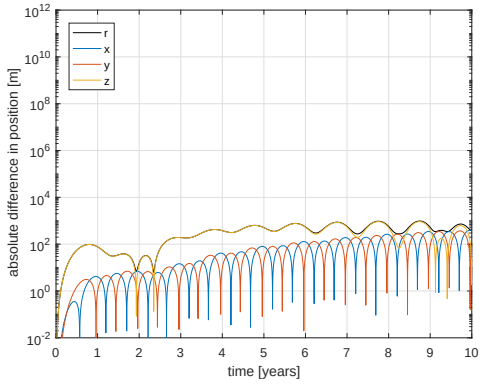
(a) Trajectory of the spacecraft around the Sun projected onto the ecliptic



(b) Relative difference in the spacecraft mass over time



(c) Relative difference in the norm of the position and velocity vectors over time



(d) Absolute difference in the position with $r = \sqrt{x^2 + y^2 + z^2}$

Figure 7.6: Difference between the propagated trajectory of the GTSI-USM and the RK-Cart-TPS for a non-constant binormal thrust force. The integration settings are given in Table 7.9

From the figures, it can be concluded that the relative accuracy is rather low. The GTSI-USM is only accurate to 4.7e-9 in position and velocity, and at least 2.7e-7 in mass after 10 years of propagation. The thrust force (not shown in the figure) has a relative accuracy of 1.6e-14. After conversion to absolute accuracy, this does not meet the predetermined limit of 1 km in position, but does meet the limit of 1 m/s in velocity. Therefore,

the GTSI-USM is not validated for the non-constant binormal thrust case.

7.4.7. NON-CONSTANT THRUST IN VARYING DIRECTION

In this section, the most challenging propagation conditions will be tested. Consider the case where the thrust force is non-constant and of varying direction, but of which the magnitude never surpasses 0.15 N. A thrust profile that meets these requirements is given in Table 7.10. Note that the thrust profile contains an extra beyond 10 years (3.1557600e+08 s). It was found that providing an extra row holding the same thrust values as the row above improves the accuracy of the propagation.

Table 7.10: Non-constant thrust profile along the three axes of the velocity frame

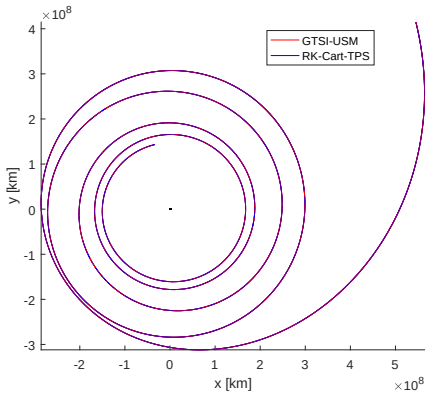
Time [s]	F_x **[N]**	F_y **[N]**	F_z **[N]**
0.0000000e+00	1.3416408e-01	0.0000000e+00	6.7082039e-02
3.1557600e+07	7.1754731e-02	3.5877365e-03	2.1526419e-02
6.3115200e+07	1.0502101e-01	2.1004201e-02	1.0502101e-01
9.4672800e+07	1.2247449e-01	6.1237244e-02	6.1237244e-02
1.2623040e+08	7.5761441e-02	1.2121831e-01	4.5456865e-02
1.5778800e+08	1.5722923e-02	1.1792192e-01	1.5722923e-02
1.8934560e+08	0.0	0.0	0.0
2.2090320e+08	0.0	0.0	0.0
2.5246080e+08	1.5000000e-01	0.0000000e+00	0.0000000e+00
2.8401840e+08	4.3301270e-02	4.3301270e-02	4.3301270e-02
3.1557600e+08	1.7320508e-02	1.7320508e-02	1.7320508e-02
3.4713360e+08	1.7320508e-02	1.7320508e-02	1.7320508e-02

The propagation settings are the following:

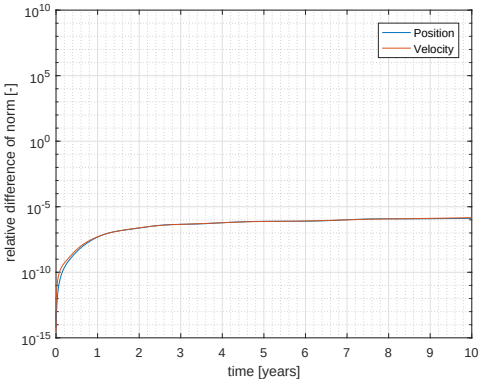
Table 7.11: Propagation settings for TSI and RK8(7)-based propagators

Parameter	**Value**
initial state	Earth orbit
duration	10 [years]
step-size	400 [s] (fixed)
TSI order	20
refinement factor	0
thrust profile	non-constant thrust

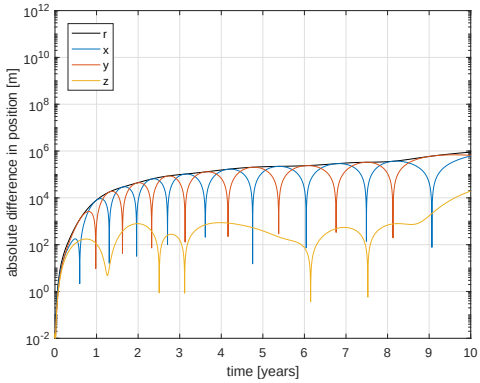
The figures below show the difference between the trajectories propagated with GTSI-USM (order 20) and the validated RK-Cart-TPS for the above thrust profile and thrust settings. The trajectories can be interpreted as the error of the GTSI-USM.



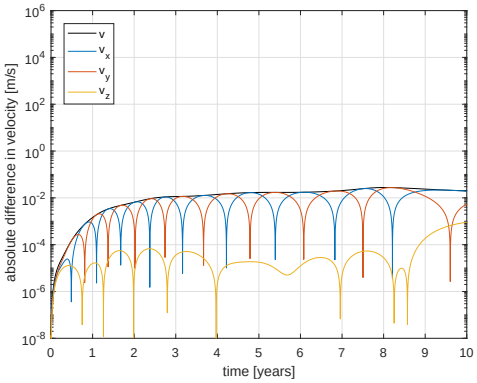
(a) Trajectory projected onto the ecliptic



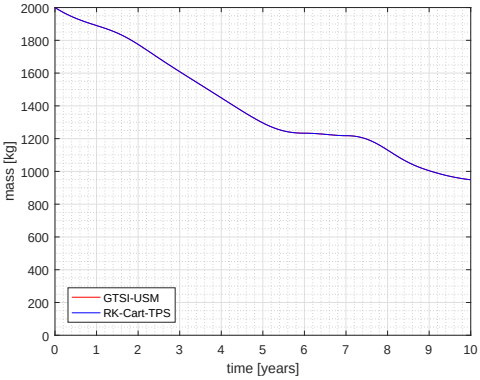
(b) Relative difference in position and velocity between the two trajectories



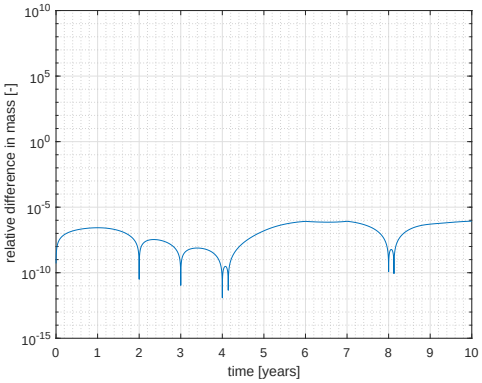
(c) Absolute difference in position between the two trajectories



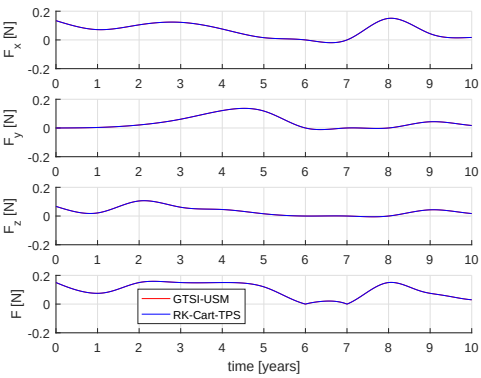
(d) Absolute difference in velocity between the two trajectories



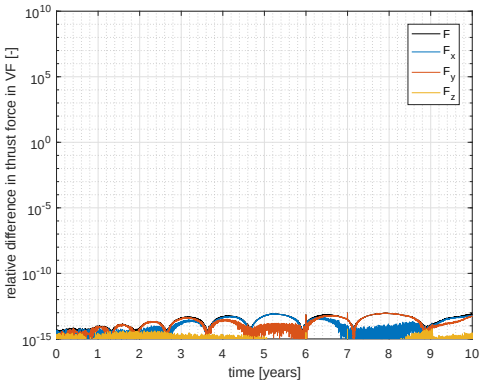
(e) Variation of the spacecraft mass over time



(f) Relative difference in spacecraft mass between the two trajectories



(g) Variation of the interpolated thrust force components over time



(h) Relative difference in the interpolated thrust force between the two trajectories

Figure 7.7: Difference between the trajectory of GTSI-USM (order 20) and the reference trajectory obtained with RK-Cart-TPS for a non-constant thrust in varying direction with a fixed step of 400 s for 10 years.

In the figures, the increase of the error with increasing time indicates the built up of numerical error. This error is initiated by truncation and round-off errors and gets amplified due to the large number of computations that succeed one another. more importantly, the figures show that the relative accuracy of the TSI is 1.5e-6 in position (9.5e5 km), 1.5e-6 in velocity (2.8 cm/s) and 8.6e-7 in mass. The thrust force has a relative accuracy of 5.4e-14. After conversion to absolute accuracy, this does not meet the predetermined limit of 1 km in position, but does meet the limit of 1 m/s in velocity. Therefore, the GTSI-USM is not validated for a non-constant thrust in varying direction.

7.4.8. DISCUSSION OF VALIDATION RESULTS

From the above results of the validation tests, it follows that the GTSI-USM is sufficiently accurate for constant thrust force scenarios. When the thrust force is not constant, the error in the position violates the predetermined limit of 1 km. However, the absolute accuracy in velocity is well below the predetermined limit of 1 m/s. In fact, the position and velocity are derived from the same USM elements which means that the an accuracy of 1 m/s in velocity corresponds to a lower relative accuracy compared to 1 km in position. The observations, interpretation of the observations, and consequences of the validation tests are summarized in Table 7.12.

Table 7.12: Observations, interpretation and consequences of the results of the validation of the GTSI-USM

Observation	Interpretation	Consequences
For a constant thrust force in a fixed-direction in the VF, the GTSI-USM is accurate to within 1 km and 1 m/s.	When the coefficients D_i, C_i and B_i of the interpolating cubic splines of all three thrust components in the VF in Eq. (5.21) are zero, then the GTSI-USM is sufficiently accurate.	The derivatives of the thrust force in the VF will be zero. Also, the second and higher order derivatives of the mass will be zero. In general, the derivatives of the thrust acceleration in VF can still be non-zero (see Eq. (5.22) and Eq. (5.23)) as well as those in the RTN frame.
For a non-constant thrust force in at least one direction of the VF, the GTSI-USM is accurate to within 1 m/s but not to within 1 km.	When the coefficients D_i, C_i and B_i of the interpolating cubic splines of at least one thrust component in the VF in Eq. (5.21) are non-zero, then the GTSI-USM is not sufficiently accurate in position. Furthermore, as the position and velocity are computed from the same USM elements, this shows that the absolute accuracy limit of the velocity is less strict than the one of the position.	The derivatives of the thrust force in the VF will be non-zero for at least one thrust component. In general, the corresponding derivatives of the thrust acceleration in VF are non-zero as well as those in the RTN frame, because of the non-zero derivatives of the mass.

From the consequences' column in the Table 7.12, the following causes for the insufficient accuracy for the non-constant can be identified:

- Incorrect implementation of Eq. (5.21)
- Incorrect implementation of Eq. (5.22)
- Incorrect implementation of Eq. (5.23)

Unfortunately, checking the implementation of the implementation of these equation did not expose any faults.

Contrary to what one might expect, Eq. (5.26) to Eq. (5.28) are correct, since all terms in that equation are computed for the validated constant thrust case too. Indeed, the derivatives of the thrust acceleration are not zero in general when the thrust force is constant as can be seen from Eq. (5.23).

Other possible causes for the error still exist, such as round-off errors and truncation errors. Although those errors are present in the constant thrust case already, they might be amplified for the non-constant thrust case, leading to an insufficient accuracy in position. Indeed, a small numerical error in the USM state vector can cause a small error in the transformation of the thrust accelerations from the VF to the RTN frame through v_r and v_{tr} (see Eqs. (5.30) to (5.33)). As the thrust acceleration in the RTN frame is used for the propagation of the USM state vector an additional error is introduced in the next USM state vector which is in turn propagated via the transformation of the thrust and so on. No solution for this inaccuracy problem has been found in the available project time. Therefore, correcting the GTSI-USM for the non-constant thrust case will be the most important recommendation of this work.

7.5. COMPARISON OF THE COMPUTATIONAL EFFICIENCY

In this section the computational efficiency of the generalized TSI-USM, the RK-Cart, the RK-USM and the RK-Cart-TPS will be compared. As mentioned before, the computational efficiency of a propagator is determined by the combined performance on accuracy and CPU time. Since the GTSI-USM has only been validated for the constant thrust case, the performance of this propagator will only be investigated for this thrust case.
First, in Section 7.5.1 the comparison of the computational efficiency will be given for unperturbed orbits. This section will be followed by the comparison for perturbed orbits.

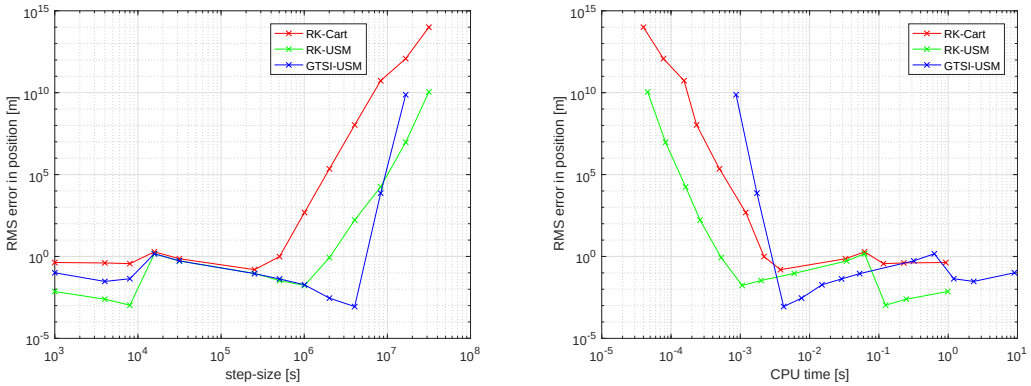
7.5.1. COMPARISON BETWEEN PROPAGATORS FOR UNPERTURBED ORBITS

In this section, the comparison of the computational efficiency of the propagators for unperturbed orbits will first be given for fixed step-sizes. Subsequently the comparison for variable step-size control will be provided.

FIXED STEP-SIZE

Fig. 7.8(a) shows the RMS error in the norm of the position for the trajectories of the three propagators as a function of the step-size. The RMS error has been calculated through comparison with the analytic solution. The CPU times were measured for the fixed step-size propagation of a Kepler orbit for 10 years without the storage of the state history. From the figure it can be concluded that:

- the Runge-Kutta 8(7)-13M propagator combined with USM elements (RK-USM) combination is the most accurate propagation method for small step-sizes (< 15778 s)
- the three propagation methods have a similar RMS error in position for medium step sizes ($10^{4.1}$ s to $10^{5.1}$ s)
- when the step-size is larger than $10^{6.3}$ s, the RMS error in position of the GTSI-USM suddenly increases faster than the RK8(7)-based propagators for increasing step-sizes. This is probably due to the high order of the TSI. The combination of a high order, with a large step-size can be lead to large errors in the position. This has been explained already in Section 4.5.1.



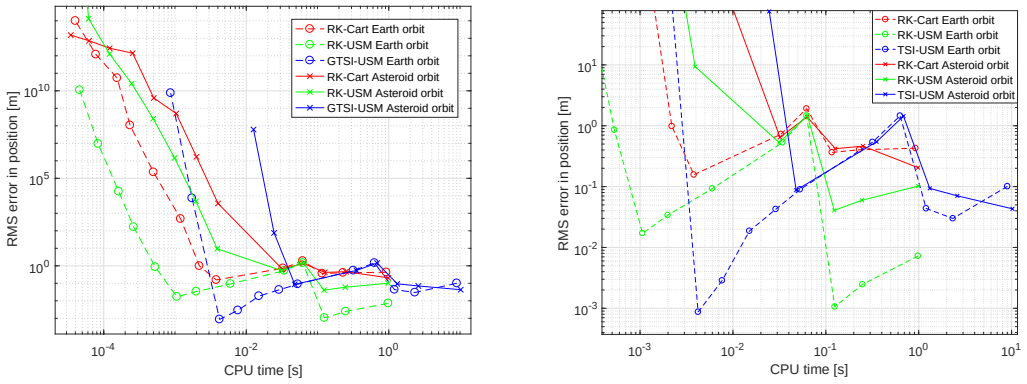
(a) RMS error in position vs step-size

(b) RMS error in the position vs CPU time

Figure 7.8: Comparison of the three propagators for the Earth orbit for fixed step-sizes. Order 20 is used for the TSI.

Next, in Fig. 7.8(b) RMS error in the position is plotted against the CPU time for the Earth orbit. The step-sizes are varied over the values in table Table 7.2 to end up with different CPU times. When looking at the Pareto front, it can be observed that the RK-USM performs better than the RK-Cart without exceptions. The GTSI-USM has only one point that lies on the Pareto front. Again, for the reason explained in Section 4.5.1, for the GTSI-USM the RMS error in the position increases rapidly for shorter CPU times (i.e. larger step-sizes).
Adding the data points for the Asteroid orbit to Fig. 7.8(b) results in Fig. 7.9(a). The most interesting part of the graph is the part for which the RMS error in the position is below 1 m. Zooming in at this part of the graph results in Fig. 7.9(b). From the graphs the following conclusions can be drawn:

- The trajectory of the Asteroid orbit has a larger RMS error in the position than the Earth orbit for similar CPU times. This is due to the higher eccentricity of the Asteroid orbit, which causes the orbital elements to verify faster .
- For an RMS error in position of below 1 mm for the Earth orbit or 10 cm for the Asteroid orbit, the GTSI-USM is the fastest propagator.
- The RK-Cart has the worst performance for both the Earth orbit and the Asteroid orbit.
- Looking at the GTSI-USM curves only (blue lines), it can be seen that the RMS error initially remains low. When a certain limit in the step-size is passed, the RMS error explodes. This phenomenon is a known problem for high order propagators and is explained in Section 4.5.1.
- Below the one meter RMS error level, both the RK-USM and GTSI-USM propagators see their RMS error in the position decrease while the CPU time is also decreasing (i.e. smaller step-sizes). Also the RK-Cart shows this trend to a lesser extend for a CPU time between $10^{-1.5}$ and $10^{-2.6}$. At first sight, this seems counter-intuitive. However, this behavior can be explained by the high order of both propagators, i.e. 20 for the TSI propagator and 8 for the RK8(7) propagator. The high order of the propagators leads to a very high accuracy (sub 1 m level $\sim 6.7 \times 10^{-12}$) for an integration step, even for a large step. In fact, the accuracy is so high that the error induced by taking a new step is higher than the one induced by lengthening the step, hence postponing taking a new step. When taking a new step, the spacecraft state at the previous step needs to be stored and recalled again at the beginning of the new step. This induces round-off errors that could have been avoided by skipping the step. Of course taking longer steps also creates errors (truncation errors), but these errors are in this case less than the induced round-off errors. (see Fig. 7.13).
- Note that for step-sizes larger than 1024000 s, the GTSI-USM RMS error becomes so large that is has been removed from the plot.



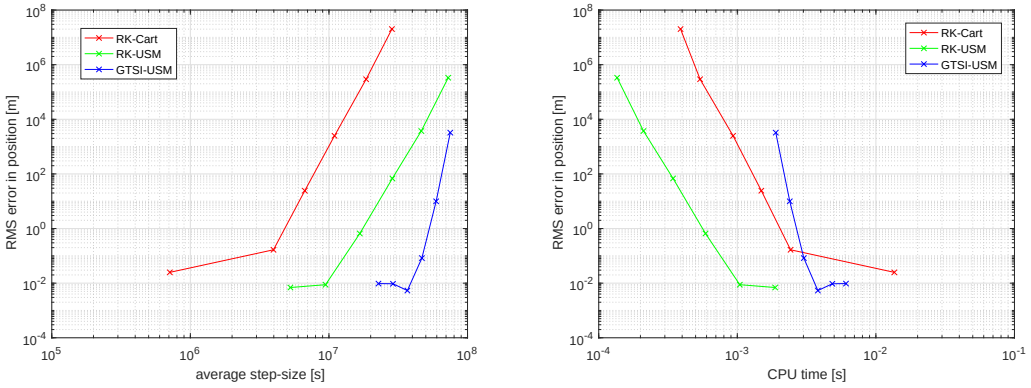
(a) RMS error in position vs CPU time

(b) Zoom of left graph for sub-meter RMS error level

Figure 7.9: RMS error in position vs CPU time comparison of the propagation of two different Kepler orbits with RK8(7)-Cart for fixed step-sizes. Order 20 is used for the TSI.

VARIABLE STEP-SIZE

Fig. 7.20 shows the RMS error in the position of the trajectory propagated using a variable step-size. The average step-size is varied by varying the tolerance settings as specified in Table 7.2. For the TSI, these values are used as the absolute tolerance. However, for RK-Cart and RK-USM, these values are used as relative tolerances. Only in case the state variables are small, they are taken as absolute tolerances. The usage of the tolerance values does not influence the plots, because the average step-size and total CPU time are independent of the tolerance values.



(a) RMS error in position vs step-size

(b) RMS error in position vs CPU time

Figure 7.10: Comparison between RK-Cart, RK-USM and GTSI-USM for variable step-size integration of the Earth orbit. Order 20 is used for the TSI.

From the two graphs, the following observations can be made:

- For a certain accuracy (i.e. a certain RMS error in position), the RK-Cart requires more steps (i.e. smaller average step-size) than RK-USM which, in turn, requires more steps than GTSI-USM.

- For the same accuracy, RK-USM is faster than RK-Cart and GTSI-USM. At the 1 m RMS level, this corresponds to 1.7 times faster than RK-Cart and 2.4 times faster than GTSI-USM. Although, GTSI-USM requires less integration steps than the other two propagators for a similar accuracy, it has the slowest performance of the three. Clearly, the CPU time per step is much higher than that of the other two propagators.

- In case the CPU time does not play a role, GTSI-USM performs best as it has the lowest minimum RMS error in the position. The RK-Cart propagator cannot even reach an RMS error of under cm level.

7.5.2. COMPARISON BETWEEN PROPAGATORS FOR PERTURBED ORBIT

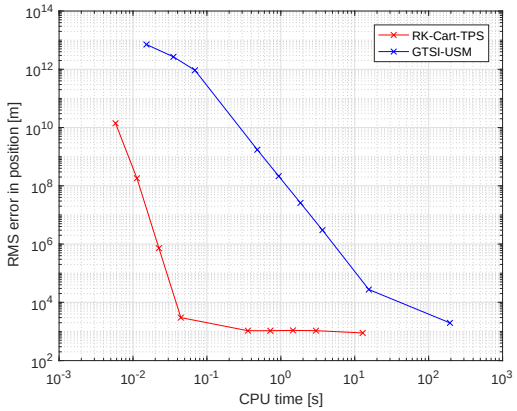
First, the comparison of the computational efficiency of the GTSI-USM and the RK-Cart-TPS will be given for fixed-step sizes. This will be followed by a section on the comparison for variable step-size control.

FIXED STEP-SIZE

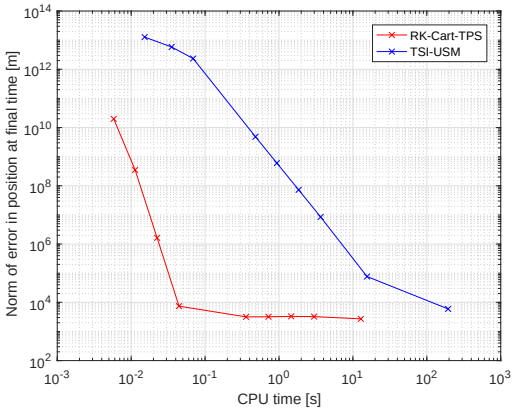
Fig. 7.11 displays the comparison of the computational efficiency (i.e. the combination of accuracy and CPU time) as well as the variation of the accuracy with the integration step-size for fixed step-sizes. The trajectories produced by the RK-Cart-TPS and GTSI-USM propagators are compared for the following propagation settings:

Table 7.13: Propagation settings

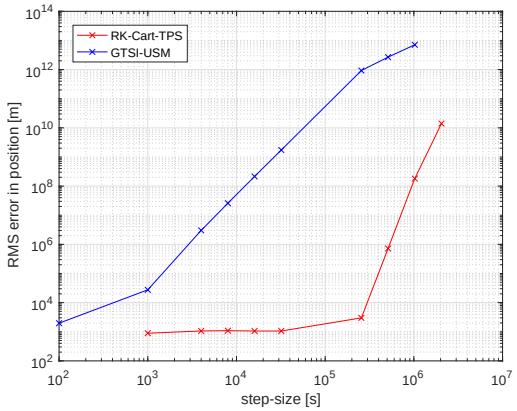
Parameter	Value
Initial state	Asteroid orbit
Duration	10 [years]
Step-size	fixed
TSI order	20
Refine factor	0
Thrust profile	constant tangential thrust of 0.15 N



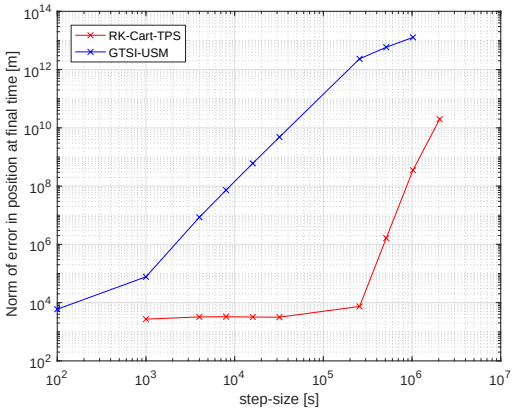
(a) RMS error in position vs CPU time



(b) Absolute error in norm of position at the final time vs CPU time



(c) RMS error in position vs step-size



(d) Absolute error in norm of position at the final time vs step-size

Figure 7.11: Comparison of the computational efficiency (combination of accuracy and CPU time) between the GTSI-USM and the RK-Cart-TPS for constant tangential thrust over an integration period of 10 years with a fixed step-size in an elliptic interplanetary orbit. In the left figures, the RMS error in the position is used as a measure for the accuracy, whereas in the right figures the absolute error in the norm of the position at the final time is used.

From the figures, the following observations can be made:

- The difference in accuracy between the GTSI-USM and RK-Cart-TPS is large. At a fixed step-size of 256000 s, the RK-Cart-TPS propagator has a RMS error in position of 2.56e5 m (7.6e-5 relative accuracy) whereas for the GTSI-USM this amounts to 9.26e11 m (1e0 relative accuracy). Clearly, the trajectory of the GTSI-USM is way off course and the integration has exploded.

- The RK-Cart-TPS stays very close (1.09 km, 3.1e-7 relative accuracy) to the predetermined accuracy limit of 1 km in position for step-sizes up to 3.2e4 s.

- The predetermined accuracy limit of 1 km in position (2.9e-7 relative accuracy) cannot be reached by the GTSI-USM. To achieve an RMS error in position of below 2 km, the RK-Cart-TPS and GTSI-USM require an average CPU time of 0.36 s and 194.1 s, respectively. Hence, for this accuracy, the RK-Cart-TPS is faster than the GTSI-USM by a factor of approximately 540.
- The left and right figures display the same shape although in the figures on the right hand side the absolute error of the norm in the position at the final time is used instead of the RMS error in position. The latter one can be regarded as the average error of the entire trajectory whereas the former one represents the error of the final position only.

From the above observations, the following conclusions can be drawn:

- For an acceptable accuracy in position, the RK-Cart-TPS is at least two orders of magnitude faster than the GTSI-USM propagator in case of constant tangential thrust.
- The fact that the GTSI-USM is accurate for a small step-size only (100 s) indicates that the GTSI-USM is not taking advantage of its higher order. Instead, it can be concluded that there is a major inaccuracy in the GTSI-USM method.
- The fact that the RMS error in position is very similar to the absolute error in position at the final time, means that the error in position increases smoothly with integration time.

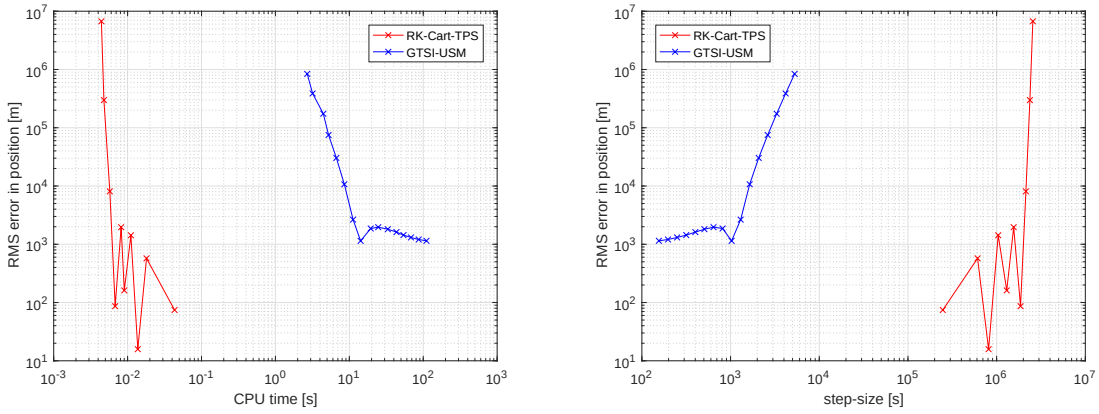
VARIABLE STEP-SIZE

This section presents the results for the comparison of the computational efficiency between the GTSI-USM and the RK-Cart-TPS for variable step-size control. The following propagation settings are used:

Table 7.14: Propagation settings

Parameter	Value
Initial state	Asteroid orbit
Duration	10 [years]
Step-size	variable
TSI order	20
Refine factor	0
Thrust profile	constant tangential thrust of 0.15 N)

Fig. 7.12 depicts the comparison of the computational efficiency (i.e. the combination of accuracy and CPU time) as well as the variation of the accuracy with the average step-size. The fact that the left and right figures look mirrored, indicates that the CPU time is inversely proportional with the step-size, as is expected.



(a) RMS error in position vs CPU time

(b) RMS error in position vs average step-size

Figure 7.12: Comparison of the computational efficiency (combination of accuracy and CPU time) between the GTSI-USM and the RK-Cart-TPS for constant tangential thrust over an integration period of 10 years with variable step-size control in an elliptic interplanetary orbit. The RMS error in the position is used as a measure for the accuracy.

From the figures, the following observations can be made:

- It can be observed that the RK-Cart-TPS has a minimum RMS error in position (i.e. maximum accuracy) of 16 m (4.6e-9 relative accuracy) which occurs at an average step-size of approximately 811000 s (9.4 days).
- The maximum accuracy of the GTSI-USM is approximately 1.15 km (3.3e-7 relative accuracy), which occurs at an average step-size of approximately 1028 s.
- The predetermined accuracy limit of 1 km in position cannot be reached. For an accuracy limit of 2 km in position (5.8e-7 relative accuracy), the GTSI-USM requires 14.21 s and the RK-Cart-TPS requires 0.00679 s. Hence, for this accuracy, the RK-Cart-TPS is faster than GTSI-USM by a factor of approximately 2093.
- From the comparison of the results in Fig. 7.11 and Fig. 7.12, it can be concluded that for an acceptable accuracy of 2 km in position, the variable step-size is 13.6 and 53 times faster than fixed step-size control, respectively, for GTSI-USM and RK-Cart-TPS.

From the above observations, the following conclusions can be drawn:

- The maximum achievable accuracy of RK-Cart-TPS is at least one order of accuracy better than that of RK-Cart-TPS.
- At a respectable accuracy of 2 km in position, the RK-Cart-TPS is at least three orders of magnitude faster than the GTSI-USM.
- From the jumpy pattern of the RK-Cart-TPS it can be concluded that the RK-Cart-TPS reaches its maximum accuracy for step-sizes below 2 million seconds (approximately 23.1 days), while for the GTSI-USM this is for step-sizes below 1028 s.
- The decrease in CPU time caused by shifting from variable step-size control to fixed step-sizes is 3.9 times larger for RK-Cart-TPS than for GTSI-USM.
- Overall, it can be concluded that there is a major inaccuracy in the GTSI-USM method.

7.5.3. POSSIBLE CAUSES FOR THE INSUFFICIENT PERFORMANCE OF GTSI-USM

As can be observed from the graphs in Section 7.5.2, the RK-Cart-TPS propagator for a given accuracy of 2 km in position, the RK-Cart-TPS is at least 540 times faster than the GTSI-USM. This is an unexpected result, as in previous research in literature the TSI-USM has been shown to be two orders of magnitude faster than an RK8(7)13M-based propagator with Cartesian coordinates and one order of magnitude faster than with USM elements [Vittaldev et al., 2010]. The GTSI-USM can either be much slower than the RK-Cart-TPS, much less accurate than the GTSI-USM, or a combination of both.

Indeed, it could be expected that due to the additional computations that are part of the implementation of the General Discrete Force Model, the GTSI-USM requires more CPU time than the TSI-USM, which can in fact be observed from the comparison of Fig. 7.20 and the earlier stated results in [Vittaldev, 2010] since even without thrust (so no possible error caused by the GDFM), the GTSI-USM is slower than the RK-USM. Nevertheless, it looks like the cause for the poor performance of the GTSI-USM is its low accuracy rather than its long CPU time. Even when allowing the GTSI-USM to use a long CPU-time, the accuracy does not get lower than 1.15 km (3.3e-7 relative accuracy). The long CPU time is caused by the very small step-sizes that are required to reach such an accuracy. It was expected that for a given step-size, the GTSI-USM (order 20) would have a higher accuracy than the RK-Cart-TPS (order 8), but Figs. 7.11(c) and 7.12(b) show that this is not the case. However, in case of zero thrust, i.e. when the GDFM does not influence the accuracy, it was observed from Fig. 7.10(a) that the GTSI-USM indeed displays a higher accuracy than the RK8(7)13M-based propagators.

From the above findings, it can be concluded that the GDFM itself causes the inaccuracy of the GTSI-USM. As was discussed in Section 7.4.8 it is possible that an amplification of numerical errors lead to the lack of accuracy. There are two main types of errors for numerical integration namely truncation errors and rounding-off

errors. Truncation errors occur when a function is approximated by a simpler function. In case of TSI, this is when an infinite series of terms is approximated by a finite number of terms. The rounding-off error, also known as the floating point error, is caused by the way computers store numbers, as only a limited number of bits is dedicated to each number [Press et al., 2007; Bergsma, 2015]. As can be seen from Fig. 7.13, for small step-sizes, the total numerical error mainly consists of round-off error. As the step-size increases, the contribution of the round-off error to the total numerical error decreases as there are fewer steps and therefore fewer computations that are exposed to round-off errors. In contrast to the round-off error, the truncation error increases for increasing step-size [Press et al., 2007]. In case of TSI this is because the truncation error of the approximating Taylor series is larger when moving away from the point around which the Taylor series was expanded. The larger the step-size, the wider the region in the time-window that will be span by one Taylor series approximation.

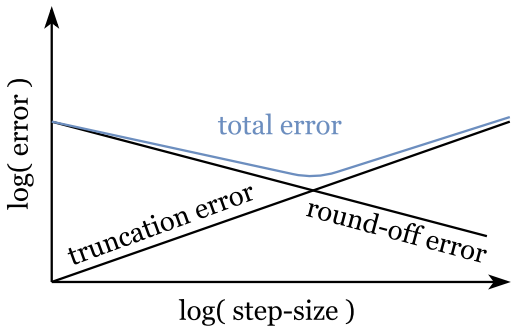


Figure 7.13: Illustration of the two components of the numerical error: truncation error and round-off error

It is plausible that the initial round-off errors in the calculation of the first USM state are amplified through the transformation of the thrust force from the velocity frame to the RTN frame as explained in Section 7.4.8. This hypothesis is also supported by the results of the application of the MMS in Section 6.7. Indeed, in that section it was shown that a small error in the state vector can lead to large errors in the trajectory of a validated propagator when the exact same thrust profile is used in both the validated propagator and the reference trajectory. The only solution to this would be to express the thrust acceleration as a function of the state itself. This would allow the correction of the induced numerical error in the state vector by constantly recomputing the thrust profile. Unfortunately, this solution cannot be implemented in the GDFM as it does not allow the a-priori interpolation of the thrust profile (i.e. the thrust profile would change during the integration), which is required to be able to compute the derivatives of the thrust accelerations and thrust magnitude.

Another possibility is that the only assumption present in the GDFM in Eq. (5.29) (the fourth and higher order derivatives of the thrust accelerations in the RTN frame are zero) poses a significant limit on the accuracy of the GDFM. It can be argued that this drops the order of the GTSI-USM from 20 to 3 when using a non-zero thrust force. It is possible to correct this assumption by deriving the equations for the fourth and higher order derivatives of the thrust accelerations in the RTN frame. However, this will add a significant number of additional computations which will increase the CPU-time even more. This approach is not recommended by the author because it will decrease the generality of the GTSI-USM, since the equations will be different for different TSI orders.

7.6. EFFECT OF DATA STORAGE

In low-thrust trajectory optimization the value of the objective function sometimes only depends on the final state, which does not require the entire state history to be stored. Therefore, it is interesting to investigate the effect on the CPU time of storing the state history of the trajectory during propagation. The effect of data storage is analyzed by comparing the calculation time of two versions of the RK-Cart propagator (one that stores the state history, and one that doesn't) for an Earth-like Kepler orbit around the Sun. The result is shown in Fig. 7.14.

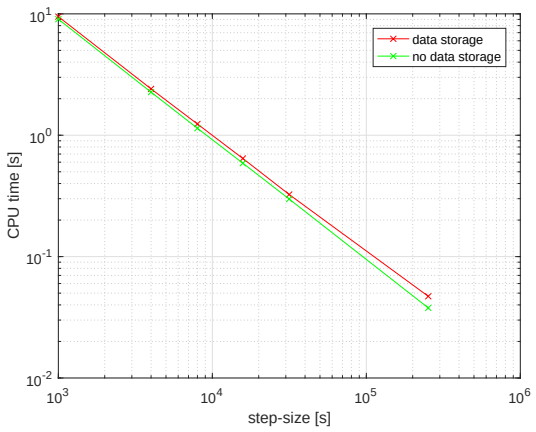


Figure 7.14: The effect of storing the state history on the CPU time for RK8(7)-Cart

It can be seen from the graph that the data storage only has a small impact on the CPU time. For low step-sizes, the data storage increases the CPU time with approximately 5%. This value increases gradually to 9% for step-sizes of 3.16e4 s. For the largest step-size there is an increase of approximately 25% in the CPU time. The higher percentage number for large step-sizes (i.e. low CPU time) is probably due to the computation time of storage commands that are independent of the amount of data to store, e.g opening the folder, creating the text file, declaration of variables related to storing the state history. It is important to note that all other CPU time measurements in this chapter are performed using the version of the propagator that does not store the state history. This is done to increase the precision of the CPU measurements of the integration routine itself.

7.7. EFFECT OF PROPAGATION SETUP

Fig. 7.15 shows the effect of the difference in programming style on the CPU time for an Earth-like Kepler orbit around the Sun. Two programming styles were implemented, the first one is the Tudat propagation setup. In this setup, a zero thrust profile is fed to the RK8(7) integrator and interpolated by the Tudat cubic spline interpolator. In the problem-specific propagator setup, a Cartesian state derivative model outside of the Tudat propagation setup. In this model there is not accounted for a perturbing thrust force. As a result, there is no cubic spline interpolation to be done.

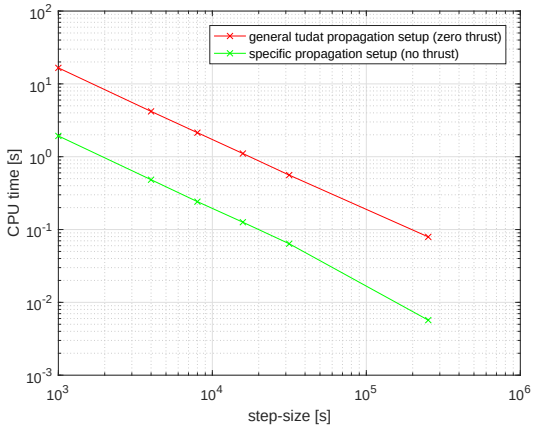


Figure 7.15: The effect of the propagation setup on the CPU time for RK8(7)-Cart.

From the graph it can be seen that the propagation setup has a significant influence on the CPU time. Clearly, the specific propagator setup requires less CPU time because of the absence of the cubic spline interpolation. It is important to note that whenever the perturbing thrust force is non-zero, the Tudat propagation setup has to be used as the cubic spline interpolation is required in this case.

7.8. CPU TIME PER INTEGRATION STEP

Fig. 7.16 shows the CPU time per integration step as a function of the step-size. The measurements are done for a fixed step-size integration of the Earth orbit without storing the state history. From the figure it can be concluded that the GTSI-USM combination requires between 8.8 and 18.0 times more CPU time per step than Runge-Kutta 8(7)-13M propagator combined with Cartesian coordinates (RK-Cart) and RK-USM, depending on the step-size. This is due to the fact that during one step of the GTSI-USM, much more variables have to be computed. There is only a small difference in the CPU time between RK-Cart and RK-USM: RK-USM requires approximately 7.8% more CPU time per step on average than RK-Cart. Moreover, it can be seen from the graph that there is a dependency of the CPU time per integration step on the step-size. The former increases slightly with increasing step-sizes. This can be explained by pointing out that an increase in step-size means a decrease in the total number of integration steps. When the number of steps is small, the processes that are independent of the number of steps have a larger relative share in the total CPU time. This leads to a higher CPU time per integration step.

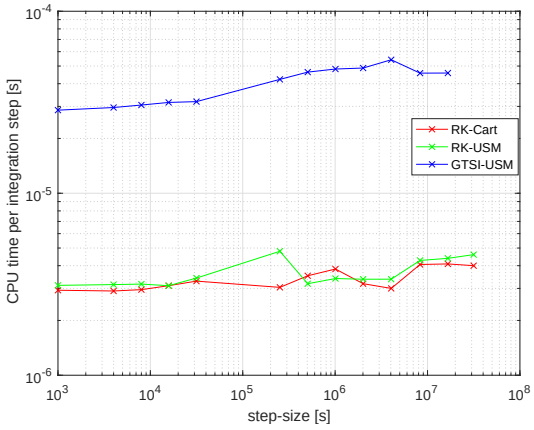


Figure 7.16: The CPU per integration step of TSI is significantly larger than that of RK8(7). Furthermore the CPU time per integration step increases slightly for increasing step-sizes. The measurements are done for fixed step-size integration of a quasi-circular Kepler orbit without storing the state history.

7.9. OPTIMAL ORDER OF TSI

Selecting the optimal order of the TSI is not straightforward. For a given tolerance, a low order generally leads to smaller steps. I.e. due to the lower tolerance per step, more steps are required to reach the tolerance. More steps results in more CPU time. For a high order, less steps are needed to meet the specified tolerance. However, the high order induces the calculation of more derivatives. This results in a larger CPU time per step. The question now is: 'Which effect will be larger: more steps, but less CPU time per step; or, less steps, but more CPU time per step?' Fig. 7.18(a) reveals that the optimal order is something in between.

The following observations can be made:

- From Figs. 7.18(a) and 7.18(b) it can be seen that is difficult to select the optimal order (i.e. lines that are closer to the bottom left corner of the graph, combining a low RMS error with a low CPU time), because the Pareto Front contains lines corresponding to different orders. Focusing on the sub-meter RMS error level, order 16 is selected as the most optimal order for Taylor Series integration of the Earth orbit.

- Fig. 7.17 shows that a higher order TSI can take larger step-sizes for the same RMS error in the position.

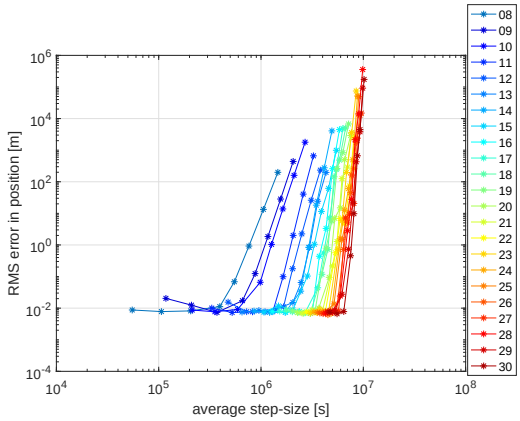
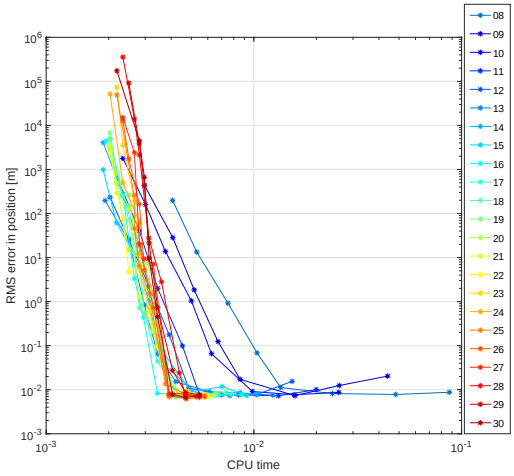
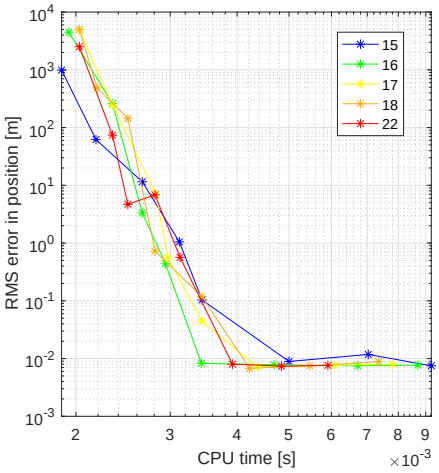


Figure 7.17: TSI order comparison for the Earth orbit. The RMS error in the position, measured w.r.t. an analytic solution is plotted against the average step size for a variable step-size control.



(a) Full plot displaying all orders



(b) Reduced plot focusing on orders that have data points that are either members of the Pareto front or are very close to being members.

Figure 7.18: TSI order comparison for the Earth orbit. The RMS error in the position, measured w.r.t an analytic solution is plotted against the CPU time for a variable step-size control.

The same analysis of the TSI order has been performed for propagation of the Asteroid orbit. The results are shown in Figs. 7.19, 7.20(a) and 7.20(b). It is important to note that 35 out of the 184 data points have been removed from the plots. This is because for the combination of a high order (≥ 20) and a high tolerance (≥ 1e-9) the propagated trajectory explodes because the step-size surpasses the TSI step-size limit, as explained in Section 4.5.1. The following observations can be made from the figures:

- Fig. 7.19 shows an explainable relation. Surely, a higher order TSI can take larger step-sizes for the same RMS error in the position.
- Again, it is difficult to pick an optimal order from Fig. 7.20(a). Lines closer to the bottom left corner of the graph are better, because of the combination of a low RMS error in the position and a low CPU time. Zooming in on a selection of the plot, focusing on the sub-meter RMS error level, and omitting the curves that are not part of the Pareto front reveals that the optimal order is 15. This is close to the optimal order that was found for the Earth orbit case. It can be concluded that although the optimal orders are close, they are not exactly the same. Thus, the optimal optimal order depends on the trajectory that is being propagated, and is therefore problem dependent. That means that for every new

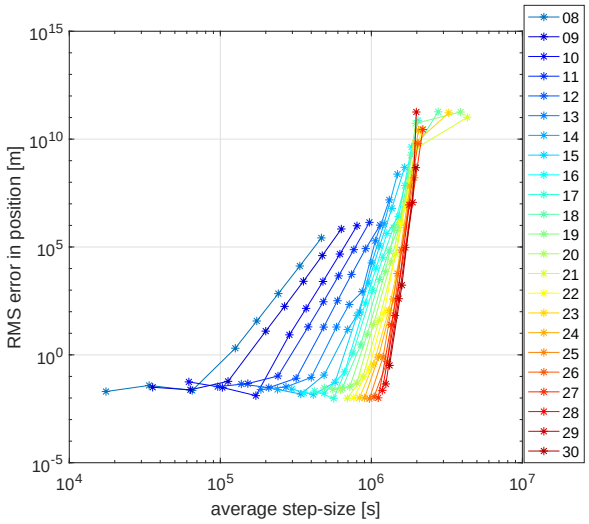
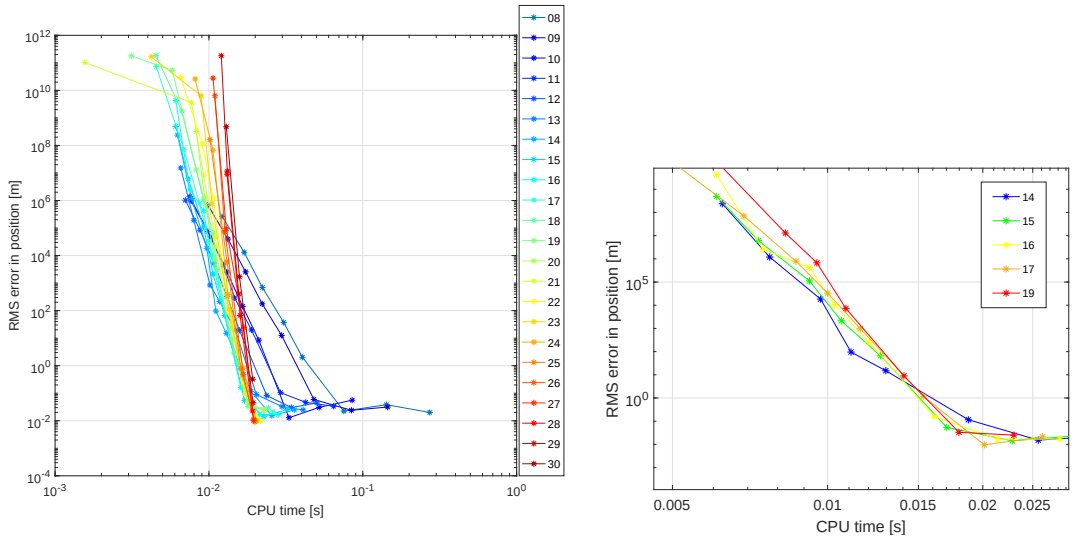


Figure 7.19: TSI order comparison for the Asteroid orbit. The RMS error in the position, measured w.r.t. an analytic solution is plotted against the average step size for a variable step-size control.

problem (i.e. changing the thrust settings would also count as a new problem) would require a new determination of the optimal order to guarantee optimal performance.

Comparison of Figs. 7.17 and 7.19 shows that

- the integrated trajectory of the Asteroid orbit has higher RMS errors for high tolerances (1e-5 to 1e-6) than the Earth orbit.
- there is very little difference in the lowest reachable RMS error. Both the Asteroid and Earth orbits can be propagated to cm level accuracy.
- propagation of the Asteroid orbit requires up to 10 times more steps (i.e. 10 times smaller step-sizes) than the Earth orbit given a certain RMS error.



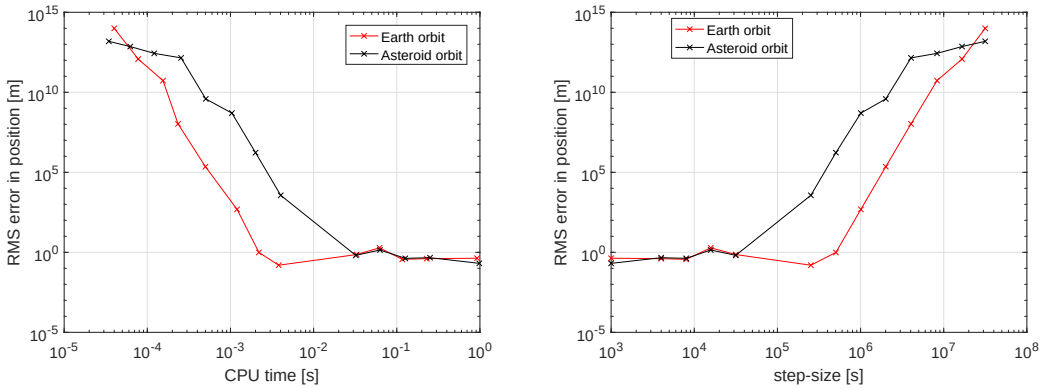
(a) Full plot displaying all orders

(b) Reduced plot focusing on orders that have data points that are either members of the Pareto front or are very close to being members.

Figure 7.20: TSI order comparison for the Asteroid orbit. The RMS error in the position, measured w.r.t an analytic solution is plotted against the CPU time for a variable step-size control.

7.10. EFFECT OF ORBIT TYPE

Figs. 7.21(a) and 7.21(b) show the effect of propagating different orbits on the RMS error in the position as a function of CPU time and step-size, respectively. The RMS error has been calculated through comparison with the analytical solution. The properties of the Earth and Asteroid orbits have been given in Table 7.1. The graphs reveal that the Asteroid orbit is propagated equally accurate w.r.t. the Earth orbit for small step-sizes (i.e. large CPU times). For increasing step-sizes (decreasing CPU times) the Earth orbit initially stays accurately propagated, while the inaccuracy in the Earth orbit starts decreasing already. This is due to the higher eccentricity of the Asteroid orbit, which causes the Cartesian elements to vary faster, hence decreasing the accuracy already at smaller step-sizes. For very large step-sizes (i.e. very small CPU times), the RMS error in position is so large that the orbits are completely off. Therefore, it is difficult to say which one is the most accurate.



(a) RMS error in position vs CPU time

(b) RMS error in position vs step-size

Figure 7.21: Comparison of the propagation of two different Kepler orbits with RK8(7)-Cart for fixed step-sizes.

8

CONCLUSION

This thesis answered the following research question:

"Can the Taylor Series Integration method be generalized to allow fast evolutionary optimization of low-thrust spacecraft trajectories?"

In order to answer the research question, the following tasks have been carried out:

- A literature study was conducted indicating that the combination of Taylor Series Integration (TSI) with Unified State Model (USM) elements results in a promisingly fast propagator (TSI-USM) for low-thrust trajectories. However, its problem-specificity was found to be detrimental for its use in evolutionary algorithms for the optimization of low-thrust trajectories. This resulted in the above stated research question.
- An approach was taken toward generalizing the TSI-USM propagation method for low-thrust trajectories. The goal was to adapt the TSI-USM so that it can be used to propagate the trajectory of a spacecraft in a central gravitational field subject to a perturbing force of any kind, as long as the force is specified in advance as a function of time. This would allow the TSI-USM to be used to optimize low-thrust trajectories with an evolutionary algorithm.
- A method was developed, called the General Discrete Force Model (GDFM), that allows the propagation of any perturbing force as long as it is specified in advance as a function of time.
- The GDFM was implemented in the existing TSI-USM, as programmed by V. Vittaldev [Vittaldev, 2010] in MATLAB to form a generalized TSI-USM which will be referred to as GTSI-USM.
- The GTSI-USM was converted to C++ and was implemented in the TU Delft Astrodynamics Toolbox (TUDAT) environment to support future projects as well as to increase its computational efficiency.
- In addition to the GTSI-USM, a diminished version of the GDFM was implemented in an eighth order Runge-Kutta (RK) propagator. The RK8(7)13M was combined with both Cartesian coordinates (RK-Cart) as well as the USM (RK-USM).
- The accuracy of the GTSI-USM was determined by comparing its output trajectory for several thrust cases to a reference trajectory produced with the validated RK-Cart propagator.
- Furthermore, the computational efficiency, i.e. the combination of accuracy and CPU time, of the GTSI-USM was compared to the RK-Cart and RK-USM propagators.
- Finally, several other tests were conducted including checking whether the orbit type changes the optimum order of the TSI-USM and investigating the effect on the CPU time of storing only the final state instead of the entire trajectory.

The main conclusion is that, indeed, the TSI can be generalized for low-thrust trajectories. Doing this with the GDFM, results in the sufficiently accurate propagation of low-thrust trajectories with constant thrust. However, for low-thrust trajectories of non-constant thrust in varying direction, the GTSI-USM is not sufficiently accurate in the position. However, it was found that, given a certain accuracy, the GTSI-USM is not faster than RK8(7)-based propagators. A single exception to this is the non-zero thrust case, for which the GTSI-USM can reach a higher maximum accuracy than the RK8(7)-based propagators and, as a consequence, is the fastest method for this accuracy. A sufficient accuracy is defined as an absolute accuracy in position and velocity of below 1 km and 1 m/s respectively for interplanetary low-thrust trajectories ($\geq$ 1 AU) after an integration period of 10 years.

The following additional conclusions can be drawn from this thesis:

- The TSI is a complex method to implement. The combination of TSI with USM elements increases the complexity even more. Therefore, it is expected that the time required to implement the TSI-USM is longer than the time required to implement an RK8(7)13M-based propagator. An actual comparison of this duration cannot be made in this thesis as existing Tudat modules were used in the implementation of the RK8(7)13M-based propagator.
- The fact that USM elements are difficult to interpret by humans, significantly complicates the debugging of the TSI-USM implementation.
- For the zero thrust case, the GTSI-USM reaches a higher maximum accuracy than the RK-USM and RK-Cart. Between the two RK8(7)13M-based propagators, for a given accuracy, the version with USM elements has a CPU time that is five times lower than that of the version with Cartesian coordinates for fixed step-sizes, and 3.2 times lower in case of variable step-size control.
- Furthermore, for the Kepler case, the GTSI-USM requires more CPU time than a general RK-USM and a general RK-Cart, for the same accuracy.
- In case of constant non-zero thrust, given a respectable accuracy of 2 km in position for interplanetary orbits, the following conclusions can be drawn:
  - The GTSI-USM requires at least three orders of magnitude more CPU time than RK-Cart.
  - The maximum achievable accuracy of the RK-Cart is at least one order of magnitude better than the GTSI-USM.
  - The variable step-size is 13.6 and 53 times faster than fixed step-size control, respectively, for GTSI-USM and RK-Cart-TPS.
  - The decrease in CPU time caused by shifting from fixed step-sizes to variable step-size control is 3.9 times larger for RK-Cart-TPS than for GTSI-USM.
- For non-zero thrust, there is a major inaccuracy in the GTSI-USM method which causes the method to require more CPU time, for any given accuracy, than an RK8(7)13M propagator with Cartesian coordinates.
- Since a specific TSI-USM has been shown to be superior to RK-USM and RK-Cart [Vittaldev et al., 2010], it seems that the improvements in generality of the TSI come with an increase in CPU time. The reader is referred to Section 7.5.3 for a discussion of possible causes for this.
- The fact that the optimum TSI order depends on the type of orbit that is propagated is disadvantageous for the use of the GTSI-USM in evolutionary algorithms. Indeed, the CPU time required to optimize the order of the TSI has to be accounted for in the overall CPU time. Hence, making the method slower with respect to RK methods for which the order can not and should not be optimized.
- The effect on the CPU time of storing the entire state history of a trajectory instead of only storing the final state is negligible. Clearly, the CPU time spent on the actual calculation of the state vector at every step of the propagator is much larger than the time required to store the value for later use.
- Due to the generality of the TUDAT propagation setup, a propagator set-up within TUDAT has a significantly longer (10 times longer) CPU time for zero thrust scenarios compared to an identical propagator outside of TUDAT.

- For the same step-size, the GTSI-USM requires respectively 8.8 and 18.0 times more CPU time per integration step than the RK-USM and RK-Cart propagators. There is only a small difference between the latter two propagators with the RK-USM requiring 7.8% more CPU time per integration step.
- For the computation of the Taylor series inside the TSI it was found that the use a different time-scale for one or more state variable does not help to limit the influence of machine error in the TSI (see Section 6.8.3).

The conclusions of this thesis are important for future research on generalizing the Taylor Series Integration. Using the general discrete force model to generalize the TSI is not preferred as the results have shown that there is a major inaccuracy in the method. Only for the zero thrust case and when used with the right order, the generalized TSI-USM can deliver a higher accuracy then RK8(7)13M-based propagators. It can be expected that future attempts to generalize the TSI-USM propagator to allow its unlimited use in evolutionary algorithms, without sacrificing its accuracy and computational speed, will remain challenging.

9

RECOMMENDATIONS

Although many questions were answered during this thesis study, several new question have been raised. These questions can be useful to lay the foundation of future research and theses.

First, a series of recommendations will be given that relate to the scientific content of this thesis. These will be followed by a set of more general recommendations that are based on the author's personal experience of conducting a thesis study.

- An important recommendation would be to improve the accuracy of the GTSI-USM for the propagation of a spacecraft's trajectory subject to non-zero thrust. The developed GDFM that was implemented to form the generalized TSI-USM propagator was found to be sufficiently accurate for the propagation of constant low-thrust trajectories. However, for non-constant thrust the predetermined absolute accuracy limits of 1 km in position and 1 m/s in velocity for interplanetary trajectories could not be reached. Furthermore, in case of non-zero thrust the GTSI-USM was found to require more CPU time, for any given accuracy, than an RK8(7)13M-based propagator. Sections 7.4.8 and 7.5.3 can be useful to explore the deficiencies of the GTSI-USM. The improvement in accuracy could also enhance the computational efficiency of the GTSI-USM due to the fact that a more accurate propagator allows larger step-sizes, hence decreasing the total number of steps and the total CPU time. The entire C++ code for this project has been shared with the TUDAT support team.
- Due to the conclusion that for non-zero thrust, the GTSI-USM requires at least three orders of magnitude more CPU time than RK-Cart-TPS, it is likely that, even after improving its accuracy, the GTSI-USM will still be slower than RK-Cart-TPS and RK-USM. Therefore, it is recommended to use one of the latter propagators for fast propagation of low-thrust interplanetary trajectories.
- As can be read in Chapter 5, the thrust profile should be specified a priori, using time as the independent variable. A recommendation would be to use the mean anomaly as the independent variable of the thrust profile. This could improve the computational efficiency of the generalized TSI-USM when used inside an evolutionary algorithm. When instead of time, mean anomaly is used to specify the thrust profile, the thrust profile of different, but similar, trajectories is expected to vary less. It is known that smaller variations in the trajectories that form the population in an evolutionary algorithm, lead to faster convergence of the optimizer. Although using mean anomaly as the independent variable for the definition of the thrust profiles will generally not make the TSI-USM faster, it will make the overall low-thrust trajectory optimization process faster.
- Thirdly, it would be interesting to use the general TSI-USM to propagate the trajectory of a spacecraft subject to other non-conservative perturbing forces, such as solar radiation pressure force, and aerodynamic forces. As those forces cannot be readily specified with respect to time, an iterative procedure should be used. The idea is to assume a thrust profile, compute the trajectory, compute the perturbing force at every location along the trajectory and feed this back into the thrust profile and iterate until convergence of the thrust profile is reached. Although this approach will probably be slower than existing methods, it might be preferred because it could result in a higher accuracy of the trajectory, given that the inaccuracy of the GTSI-USM is improved.

- In the comparison of the computational efficiency of the different propagators in Section 7.5.1, it would be interesting to see how the combination of the generalized TSI with Cartesian coordinates would perform. Consequently, the development of a general TSI-Cart and the exploration of its computational efficiency would complete the comparison.

- Another interesting research topic would be to compare the computational efficiency of the generalized TSI-USM, as developed in this thesis, to a TSI that is generalized using an automatic differentiation toolbox. The automatic differentiation toolbox would replace the developed GDFM. The major difference between the two would be that the GDFM is a numerical method that is based on cubic spline interpolation, whereas an automatic differentiation toolbox would analytically differentiate a given thrust function.

- An interesting application would be to use the generalized TSI-USM in an evolutionary optimizer (e.g. PaGMO's adaptive differential evolution algorithm) would be an interesting application.

- As was explained in Section 6.7, the Method of Manufactured Solutions [Hulshoff, 2013] is an easy method that allows the creation of exact reference solutions that can be used for the validation of numerical integration of ODE's. Preferably, this method should be used to determine and judge the accuracy of a numerical integrator.

What follows is a set of recommendations and tips for future master students. These tips are classified in the the following categories: literature study, planning, report and software.

LITERATURE STUDY

- It is best to plan the literature study so that the internship does not fall in between, if possible.

- Literature studies of former and fellow master students are often not readily accessible, therefore it is recommended to mutually share the topic one is working on. This information can preferably be summarized on a list that is accessible across the university faculties.

- It is recommended to attend thesis presentations of master students on a broad range of topics. Besides inspiring thesis students, this can help first year master students to choose a topic for their literature study. It can also provide bachelor students with a good view on what a specific master profile has to offer.

- Probably the most important aspect of the literature study is to bound your planned thesis work as much as possible. This can be achieved by posing clear constraints on and/or applying necessary simplifications to the subsequent thesis study.

PLANNING

- To make a good plan and adjust the plan every day, if needed, is important. This way there will always be an up-to-date plan, even though one is deviating from the initial schedule.

- It is good practice to plan meetings with your supervisor at fixed times of the week so that you have an extra stimulant to finish a specific part and to assure that your supervisor can keep track of your work.

REPORT

- It is recommended to start writing the thesis report already from the first month of the thesis project. Of course, this will sometimes not be possible as one first needs to produce something before it can be documented. Nevertheless, keeping track of your work daily with the use of a log book is highly recommended.

- When writing the thesis, it is best to write every part as if it would be the final version of that part. This will save you a lot of time during proof reading.

SOFTWARE

- When using QtCreator, it is recommended to use TUDAT on Linux. Although most aspects of TUDAT will work fine on Windows, one of the most useful and essential tools in modern code development, the debugger, might not work well.
- The free online tool Draw.io[1] is an easy and powerful tool for all kinds of vectorized and non-vectorized drawings, including reference frames, adapted graphs and flow diagrams.
- Git is a version control system that can be very useful for tracking changes in the code as well as in the thesis report. Although several Git interfaces exist, of which Smartgit is one, using Git from the command window in Windows or the terminal in Linux is probably better, both in terms of simplicity and online support.

As a last recommendation, a master course on the optimization of spacecraft trajectories would be helpful for many students of the space exploration track. I think it would be advantageous for the Astrodynamics and Space Missions department of the TU Delft to install such a course.

[1] `https://www.draw.io/`

BIBLIOGRAPHY

Altman, S. P. (1972). A Unified State Model of orbit trajectory and attitude dynamics. *Celestial Mechanics*, 6(4):425–446.

Altman, S. P. (1975). Velocity-space maps and transforms of tracking observations for orbital trajectory state analysis. *Celestial Mechanics*, 11(4):405–428.

Belló, M. and Cano, J. L. (2008). Deimos space solution to GTOC3. Presentation.

Bergsma, M. C. W. (2015). Application of taylor series integration to reentry problems. Master's thesis, Delft University of Technology.

Bertrand, R. (2008). Results found by the CNES team. `http://areeweb.polito.it/gtoc/ws3-team4.pdf`. Accessed: 11MAY2015.

Broucke, R. A. and Cefola, P. J. (1972). On the equinoctial orbit elements. *Celestial Mechanics*, 5(3):303–310.

Buhmann, M. D. (2003). *Radial Basis Functions*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press.

Butcher, J. C. (1963). Coefficients for the study of Runge-Kutta integration processes. *Journal of the Australian Mathematical Society*, 3(2):185–201.

Butcher, J. C. (2008). *Numerical Methods for Ordinary Differential Equations*. John Wiley & Sons, New York.

Casalino, L., Colasurdo, G., and Sentinella, M. R. (2007). Problem description for the 3rd Global Trajectory Optimisation Competition. `http://www.esa.int/gsp/ACT/doc/MAD/ACT-RPT-MAD-GTOC3-problem_stmt.pdf`. Accessed: 07MAY2015.

Chodas, P. (1981). Application of the extended Kalman filter to several formulations or orbit determination. Technical Report UTIAS Technical Note 224, University of Toronto Institute for Aerospace Studies.

de Boor, C. (1978). *A Practical Guide to Splines*, volume 27 of *Applied Mathematical Sciences*. Springer-Verlag New York.

Dirkx, D. (2017). Personal conversation. Delft University of Technology.

Dwight, R. (2011). Applied numerical analysis. Delft University of Technology Lecture Notes.

ESA (2015). GTOC3 Multiple Sample Return. `http://sophia.estec.esa.int/gtoc_portal/?page_id=17`. Accessed: 07MAY2015.

Evertsz, C. (2007). GTOC3 Solution of STARS Team. Technical report, Delft University of Technology.

Frenet, F. (1852). Sur les courbes à double courbure. *Journal de mathématiques pures et appliqueées*, 17:437–447.

Ghosh, A. R. M. (2013). *Multi-cubesat mission planning enabled through parallel computing*. PhD thesis.

Goldstein, H., Poole, C. P., and Safko, J. L. (2002). *Classical Mechanics*. Addison Wesley, San Francisco, 3rd edition.

Haas, P. (2013). Implicit Runge Kutta methods for orbit propagation. `http://ccar.colorado.edu/asen5050/projects/projects_2013/Haas_Patrick/`. Accessed: 07MAY2015.

Hamilton, W. R. (1844). On quaternions, or on a new system of imaginaries in algebra. *Philosophical Magazine*, 25(3):489–495.

Hazewinkel, M. (2001). *Encyclopaedia of Mathematics: Lagrange Interpolation Formula*. Springer.

Hulshoff, S. (2013). Computational modelling. Lecture notes of Delft University of Technology.

IAU (1991). Terrestrial time. In *Resolution A4, recommendation IV, note 9*.

Kéchichian, J. A. (2008). Inclusion of higher order harmonics in the modeling of optimal low-thrust orbit transfer. *The Journal of the Astronautical Sciences*, 56(1):41–70.

Kutta, M. W. (1901). Beitrag zur näherungsweisen integration totaler differentialgleichungen. *Zeitschrift für Mathematik und Physik*, 46:435–453.

L. Casalino, G. C. and Sentinella, M. R. (2007). Results of the 3rd global trajectory optimisation competition. `http://areeweb.polito.it/gtoc/gtoc3_results.pdf`. Accessed: 11MAY2015.

Montenbruck, O. and Gill, E. (2001). *Satellite Orbits*. Springer.

Mooij, E. (2012). Orbit-state model selection for solar-sailing mission optimization. In *AIAA 2012-4588*, Minneapolis, MN, United States. AIAA/AAS Astrodynamics Specialist Conference.

Mulder, J., van Staveren, W., van der Vaart, J., de Weerdt, E., Visser, C., de Visser, A., in 't Veld, A., and Mooij, E. (2013). *Flight Dynamics*. Delft University of Technology.

NASA (2015). There is no asteroid threatening Earth. `http://www.jpl.nasa.gov/news/news.php?feature=4692`. Accessed: 18JAN2016.

Noomen, R. (2014a). Space mission design: Basics v4.17. Lecture at Delft University of Technology.

Noomen, R. (2014b). Space mission design: Integrators v4.4. Lecture at Delft University of Technology.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (2007). *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, New York, third edition.

Prince, P. J. and Dormand, J. R. (1981). High order embedded Runge-Kutta formulae. *Journal of Computational and Applied Mathematics*, 7(1):67–75.

Raol, J. R. and Sinha, N. K. (1985). On the orbit determination problem. *IEEE Transactions on Aerospace and Electronic Systems*, AES-21(3):274–291.

Roache, P. J. (2002). Code verification by the method of manufactued solutions. *Journal of Fluids Engineering*, 124(1):4–10.

Roy, C. J. (2005). Review of code and solution verification procedures for computational simulation. *Journal of Computational Physics*, 205(1):131–156.

Sarra, S. A. (2006). Chebyshev interpolation. *Journal of Online Mathematics and Its Applications*, 6(1297).

Schaub, H. and Junkins, J. (2002). *Analytical Mechanics of Aerospace Systems*. AIAA Education Series. AIAA.

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

Serret, J. A. (1851). Sur quelgues formules relatives à la théorie des courbes à double courbure. *Journal de mathématiques pures et appliqueées*, 16:193–207.

Tudat (2017). Tudat contributors. `https://github.com/Tudat/tudat/graphs/contributors`. Accessed: 4JUN2017.

Vallado, D. A. (2007). *Fundamentals of Astrodynamics and Applications*. Space Technology Library. Springer New York, third edition.

Van den Broeck, M. (2016). GTOC3 literature study. Technical report, Delft University of Technology.

Van Verth, J. (2013). Understanding quaternions. Presentation at Game Developers Conference 2013.

Vittaldev, V. (2010). The Unified State Model: Derivation and application in astrodynamics and navigation. Master's thesis, Delft University of Technology.

Vittaldev, V., Mooij, E., and Naeije, M. (2012). Unified state model theory and application in astrodynamics. *Celestial Mechanics and Dynamical Astronomy*, 112(3):253–282.

Vittaldev, V., Mooij, E., and Naeije, M. C. (2010). Performance aspects of orbit propagation using the Unified State Model. In *AIAA-2010-7658*, Portland, OR, United States. AIAA Guidance, Navigation, and Control Conference.

Walker, M., Ireland, B., and Owens, J. (1985). A set of modified equinoctial orbit elements. *Celestial Mechanics*, 36(4):409–419.

Weisstein, E. W. (2017). Step function. `http://mathworld.wolfram.com/StepFunction.html`. Accessed: 4JUL2017.

Wikipedia (2012). Heliocentric ecliptic coordinates. `https://en.wikipedia.org/wiki/Ecliptic_coordinate_system`. Accessed: 29JUN2015.

A

COORDINATE TRANSFORMATIONS

This appendix contains all of the transformations from one set of coordinates to another. In Appendix A.1 and Appendix A.2 the transformation from Cartesian coordinates to Kepler elements and vice versa will be explained. The transformations from Kepler to USM elements and vice versa are explained in Appendix A.3 and Appendix A.4, respectively.

A.1. CARTESIAN COORDINATES TO KEPLER ELEMENTS

First the radius vector is constructed from the input data:

$$\boldsymbol{r} = [x\ y\ z] \tag{A.1}$$

Next, the magnitude of the radius vector is calculated:

$$r = \|\boldsymbol{r}\| \tag{A.2}$$

The same is done for the velocity vector:

$$\boldsymbol{v} = [\dot{x}\ \dot{y}\ \dot{z}] \tag{A.3}$$

$$v = \|\boldsymbol{v}\| \tag{A.4}$$

The vector of the angular momentum per unit mass is calculated

$$\boldsymbol{h} = \boldsymbol{r} \times \boldsymbol{v} \tag{A.5}$$

as well as the magnitude of this vector:

$$h = \|\boldsymbol{h}\| \tag{A.6}$$

Next, the N vector as well as its magnitude are calculated and they will be used for the determination of the right ascension of ascending node (RAAN) Ω, argument of pericenter ω and true anomaly θ:

$$\boldsymbol{N} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times \boldsymbol{h} \tag{A.7}$$

$$N = \|\boldsymbol{N}\| \tag{A.8}$$

Together with the Sun's gravitational constant μ_S, all the parameters are known to calculate the eccentricity vector and its magnitude:

$$\boldsymbol{e} = \frac{\boldsymbol{v} \times \boldsymbol{h}}{\mu} - \frac{\boldsymbol{r}}{r} \tag{A.9}$$

$$e = \|\boldsymbol{e}\| \tag{A.10}$$

Next the semi-major axis is calculated:

$$a = \frac{1}{\frac{2}{r} - \frac{v^2}{\mu_S}} \tag{A.11}$$

The inclination is determined using the unified angular momentum per unit mass in the z-direction:

$$i = \text{acos}\left(\frac{h_z}{h}\right) \tag{A.12}$$

The RAAN is calculated using the unified N vector in both x and y directions. These components are needed to end up with an angle in the correct quadrant.

$$\Omega = \text{atan2}\left(\frac{N_y}{N}, \frac{N_x}{N}\right) \tag{A.13}$$

The argument of pericenter can be determined using the following equation. The 'sign' parameter is required to appoint the correct sign to the argument of pericenter:

$$\omega = \text{sign} \cdot \text{acos}\left(\frac{\boldsymbol{e}\cdot\boldsymbol{N}}{e\cdot N}\right), \quad \text{sign} = 1 \text{ if } \left(\frac{\boldsymbol{N}}{N} \times \boldsymbol{e}\right)\cdot\boldsymbol{h} > 0, \quad \text{sign} = -1 \text{ otherwise} \tag{A.14}$$

The same sort of calculation is performed for the true anomaly:

$$\theta = \text{sign} \cdot \text{acos}\left(\frac{\boldsymbol{r}\cdot\boldsymbol{e}}{r\cdot e}\right), \quad \text{sign} = 1 \text{ if } (\boldsymbol{e}\times\boldsymbol{r})\cdot\boldsymbol{h} > 0, \quad \text{sign} = -1 \text{ otherwise} \tag{A.15}$$

Next, the eccentric anomaly is calculated:

$$E = 2\cdot\text{atan}\left(\tan\left(\frac{\theta}{2}\right)\sqrt{\frac{1-e}{1+e}}\right) \tag{A.16}$$

Finally, the mean anomaly is obtained using the eccentricity and the eccentric anomaly:

$$M = E - e\sin E \tag{A.17}$$

A.2. KEPLER ELEMENTS TO CARTESIAN COORDINATES

In order to simplify the transformation from Kepler elements into Cartesian coordinates first several parameters have to be defined. These parameters will be used later on in the transformation process [Noomen, 2014a].

$$l_1 = \cos\Omega\cos\omega - \sin\Omega\sin\omega\cos i \tag{A.18}$$
$$l_2 = -\cos\Omega\sin\omega - \sin\Omega\cos\omega\cos i \tag{A.19}$$
$$m_1 = \sin\Omega\cos\omega + \cos\Omega\sin\omega\cos i \tag{A.20}$$
$$m_2 = -\sin\Omega\sin\omega + \cos\Omega\cos\omega\cos i \tag{A.21}$$
$$n_1 = \sin\omega\sin i \tag{A.22}$$
$$n_2 = \cos\omega\sin i \tag{A.23}$$

Next, two other parameters need to be calculated [Noomen, 2014a]:

$$\begin{bmatrix} \xi \\ \eta \end{bmatrix} = \begin{bmatrix} r\cos\theta \\ r\sin\theta \end{bmatrix} \tag{A.24}$$

However it is possible that θ is not given. In that case either E or M must be given. When E is given, the following formula is used to go from E to θ:

$$\theta = 2\cdot\text{atan}\left(\tan\left(\frac{E}{2}\right)\sqrt{\frac{1+e}{1-e}}\right) \tag{A.25}$$

In case M is given, it is slightly more complicated to obtain θ because E has to be computed from M which can be done by interpolation or iteratively solving Eq. (A.17) for E so that in the end (A.25) can be used. Next, the position vector is calculated using the following formula. At this point half of the Cartesian state vector is known.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} l_1 & l_2 \\ m_1 & m_2 \\ n_1 & n_2 \end{bmatrix} \begin{bmatrix} \xi \\ \eta \end{bmatrix} \tag{A.26}$$

Next, the angular momentum per unit mass has to be calculated:

$$h = \sqrt{\mu\, a\left(1-e^2\right)} \tag{A.27}$$

Finally, the velocities in x, y and z directions can be computed using the following equations:

$$\begin{aligned} \dot{x} &= \tfrac{\mu}{h}\left(-l_1 \sin\theta + l_2\left(e+\cos\theta\right)\right) \\ \dot{y} &= \tfrac{\mu}{h}\left(-m_1 \sin\theta + m_2\left(e+\cos\theta\right)\right) \\ \dot{z} &= \tfrac{\mu}{h}\left(-n_1 \sin\theta + n_2\left(e+\cos\theta\right)\right) \end{aligned} \tag{A.28}$$

A.3. KEPLER ELEMENTS TO USM

A Unified State Model state vector of a spacecraft has the following form:

$$\boldsymbol{x} = \begin{bmatrix} C & R_{f1} & R_{f2} & \epsilon_{O1} & \epsilon_{O2} & \epsilon_{O3} & \eta_0 \end{bmatrix}^T \tag{A.29}$$

In order to remove any ambiguity, the conversion from Kepler elements to USM elements is shown below in algorithm-style. The conversion is based on [Vittaldev, 2010] and is extended so that any given combination of Kepler elements can be converted to USM elements without ambiguity. In order to make this possible, the following input rules have been posed:

- If the is orbit parabolic, the semi-major axis a in the input vector is replaced by the semi-latus rectum p.
- If the eccentricity e is zero, the argument of pericenter ω must be zero by definition.
- If the inclination i is zero, the right ascension of ascending node Ω must be zero by definition.

1 Define gravitational parameter of central body, e.g. $\mu = \mu_S = 1.32712440018 \times 10^{20}\, \frac{m^3}{s^2}$

2 Check whether the input is consistent and within the expected range.

2 If $e = 1$ (parabolic orbit), then

$$C = \sqrt{\frac{\mu}{p}} \tag{A.30}$$

else:

$$C = \sqrt{\frac{\mu}{a(1-e^2)}} \tag{A.31}$$

3 Compute R

$$R = eC \tag{A.32}$$

4 Compute the velocity components R_{f1} and R_{f2}

$$R_{f1} = -R\sin(\Omega+\omega) \tag{A.33}$$
$$R_{f2} = R\cos(\Omega+\omega) \tag{A.34}$$

5 Compute $u = \omega + \theta$

6 Compute the elements of the quaternion:

$$\epsilon_{O1} = \sin\left(\frac{i}{2}\right)\cos\left(\frac{\Omega-u}{2}\right) \tag{A.35}$$
$$\epsilon_{O2} = \sin\left(\frac{i}{2}\right)\sin\left(\frac{\Omega-u}{2}\right) \tag{A.36}$$
$$\epsilon_{O3} = \cos\left(\frac{i}{2}\right)\sin\left(\frac{\Omega+u}{2}\right) \tag{A.37}$$
$$\eta_O = \cos\left(\frac{i}{2}\right)\cos\left(\frac{\Omega+u}{2}\right) \tag{A.38}$$

A.4. USM TO KEPLER ELEMENTS

A Kepler state vector of a spacecraft has the following form:

$$\boldsymbol{x} = \begin{bmatrix} a & e & i & \Omega & \omega & \theta \end{bmatrix}^T \tag{A.39}$$

In order to remove any ambiguity, the conversion from USM elements and Kepler elements is shown below in algorithm-style. The conversion is based on [Vittaldev, 2010] and is extended so that any given combination of USM elements can be converted to Kepler elements without ambiguity. In order to make this possible, the following output rules have been posed:

- If the is orbit parabolic, the semi-major axis a in the output vector is replaced by the semi-latus rectum p.
- If the eccentricity e is zero, the argument of pericenter ω must be zero by definition.
- If the inclination i is zero, the right ascension of ascending node Ω must be zero by definition.

1 If $\epsilon_{O3} = 0$ and $\eta_O = 0$ (pure-retrograde orbit), then

Singularity due to pure-retrograde orbit: Elements can not be converted (A.40)

else:

$$\cos\lambda = \frac{\eta_O^2 - \epsilon_{O3}^2}{\epsilon_{O3}^2 + \eta_O^2} \tag{A.41}$$
$$\sin\lambda = \frac{2\epsilon_{O3}\eta_O}{\epsilon_{O3}^2 + \eta_O^2} \tag{A.42}$$
$$\lambda = \text{atan2}(\sin\lambda, \cos\lambda) \tag{A.43}$$

2 Compute υ_{e1} and υ_{e2}

$$\upsilon_{e1} = R_{f1}\cos\lambda + R_{f2}\sin\lambda \tag{A.44}$$
$$\upsilon_{e2} = C - R_{f1}\sin\lambda + R_{f2}\cos\lambda \tag{A.45}$$

3 Compute R

$$R = \sqrt{R_{f1}^2 + R_{f2}^2} \tag{A.46}$$

4 Compute $e = \frac{R}{C}$

5 If $e = 1$ (parabolic orbit), then

$$p = \frac{\mu}{C^2} \tag{A.47}$$

else:

$$a = \frac{\mu}{2C v_{e2} - \left(v_{e1}^2 + v_{e2}^2\right)} \tag{A.48}$$

6 Compute i

$$i = \operatorname{acos}\left(1 - 2\left(\epsilon_{O1}^2 + \epsilon_{O2}^2\right)\right) \tag{A.49}$$

7 If ($\epsilon_{O1} = 0$ and $\epsilon_{O2} = 0$) or ($\epsilon_{O3} = 0$ and $\eta_O = 0$) (pure-prograde or pure-retrograde orbit), then

$$\Omega = 0 \tag{A.50}$$

else:

$$\Omega = \operatorname{atan2}\left(\frac{\epsilon_{O1}\epsilon_{O3} + \epsilon_{O2}\eta}{\sqrt{\left(\epsilon_{O1}^2 + \epsilon_{O2}^2\right)\left(\eta_O^2 + \epsilon_{O3}^2\right)}}, \frac{\epsilon_{O1}\eta_O - \epsilon_{O2}\epsilon_{O3}}{\sqrt{\left(\epsilon_{O1}^2 + \epsilon_{O2}^2\right)\left(\eta_O^2 + \epsilon_{O3}^2\right)}}\right) \tag{A.51}$$

while $\Omega < 0$:

$$\Omega = \Omega + 2\pi \tag{A.52}$$

end while

8 If $R = 0$ (circular orbit), then

$$\omega = 0 \tag{A.53}$$

$$\theta = \lambda - \Omega \tag{A.54}$$

while $\theta < 0$

$$\theta = \theta + 2\pi \tag{A.55}$$

end while

else:

$$\theta = \operatorname{atan2}\left(\frac{v_{e1}}{R}, \frac{v_{e2} - C}{R}\right) \tag{A.56}$$

while $\theta < 0$

$$\theta = \theta + 2\pi \tag{A.57}$$

end while

$$\omega = \lambda - \Omega - \theta \tag{A.58}$$

while $\omega < 0$

$$\omega = \omega + 2\pi \tag{A.59}$$

end while

B

Literature Quotes on TSI & USM

This appendix contains quotes on the TSI, USM and the combination of the two TSI propagator combined with USM elements (TSI-USM). The purpose of this appendix is twofold. First, it supports the respective selection of the TSI and the USM for the integration method and coordinate system used in this thesis. Second, it can help future students to find sources in literature on TSI and USM.

B.1. USM7

- "Numerical simulations comparing the original USM, the USM with modified Rodrigues parameters, and the USM with Exponential Map, with traditional Cartesian coordinates have been carried out. The USM and its derivatives outperform the Cartesian coordinates for all orbit cases in terms of accuracy and computational speed, except for highly eccentric perturbed orbits. The performance of thrust the Unified State Model is exceptionally better for the case of orbits with continuous low-thrust propulsion with CPU simulation time being an order of magnitude lower than for the simulation using Cartesian coordinates. This makes the USM an excellent state propagator for mission optimizations." [Vittaldev et al., 2012]

- "The main goal of this paper is to make the USM more accessible to researchers and engineers so that the USM can once again be used and hopefully be made generally known." [Vittaldev et al., 2012]

- "USM7 has a larger error for the smaller time step-sizes while USMEM (USM with Exponential Mapping) has a larger error for the larger step-sizes. [With fixes step-size RK5(4)]" [Vittaldev et al., 2012]

- "[In the case of variable-step size integration of a circular parking orbit subject to a continuous tangential acceleration of 0.004905 m/s^2 without other perturbations, then] of the USMs, the USM7 performs the best, followed by USMEM and USM6." [Vittaldev et al., 2012]

- "Between the different USM variations, we recommend the newly proposed USMEM to be used for the numerical integration of orbits. It is outperformed by USM7 only in the case of low-thrust propulsion with constant acceleration and no perturbations. [variable thrust magnitude in variable direction was not tested]" [Vittaldev et al., 2012]

- "[The results of the single-objective optimization with an evolutionary algorithm show that] On average the Cartesian model requires about 3.5 times more function evaluations than the other two, with the USM being superior to the MEE. This means that for a given accuracy, the USM is still better than the Cartesian model, since it will achieve this result quicker. In terms of CPU time, taking into account the difference in computational load per single run, the accumulated CPU time for the Cartesian model is 51432 s (14.3 hr). For the USM and MEE this is 14215 s (3.9 hr) and 16730 s (4.6 hr), respectively. Thus, the USM would allow for a more extensive tuning of the optimization algorithm than the use of Cartesian elements at the same computation cost." [Mooij, 2012]

- "The USM6 and USM7 outperform the Cartesian coordinates for all cases in terms of accuracy and computational speed, except for highly eccentric perturbed orbits. The performances of the USM6

and USM7 are exceptionally better for the case of orbits with continuous low-thrust using Cartesian coordinates. The performance difference between the USM6 and USM7 is minimal." [Vittaldev, 2010]

B.2. TSI

- "[...] As a result, TSI is best applied to problems, where the step sizes are not kept artificially small by guidance, navigation or control systems. For such problems, the RKF5(6) integrator was found to be the fastest integration method for reentry applications for a given level of accuracy." [Bergsma, 2015]
- "Discrete data tables of environment and aerodynamic models were fitted with regression lines to obtain analytical approximations, as TSI can only handle analytical expressions." [Bergsma, 2015]
- "The strategy for order control is to use a fixed order, rather than variable order, as the variable order strategy in most cases did not improve the CPU time. For the cases where it did improve the CPU time, the improvement was too little to justify the large number of variables that had to be optimized for the variable-order controller. The optimal fixed order was determined for each error tolerance by comparing CPU times." [Bergsma, 2015]
- "[For a reentry application] TSI is 3.28 times faster than RKF5(6) for integration with an error tolerance of 1e-8. When the tolerance is lowered, the ratio between CPU times of RKF5(6) and TSI increases. This shows that TSI is especially preferred for high accuracy." [Bergsma, 2015]
- "Heat to-head comparison at five different error tolerances shows that, on average, Taylor series is faster than Runge-Kutta Fehlberg by a factor of 15.8. Results further show that Taylor series has superior convergence properties." [Scott and Martini, 2008]

B.3. TSI-USM COMBINATION

- "Some scenarios for orbits using low-thrust propulsion were implemented with TSI for both Cartesian coordinates and the USM7. The performance of both models was similar, but the Cartesian coordinates are faster than the USM7 for very high order (>30) because there are fewer computations involved. However, both Cartesian coordinates and the USM7 with TSI are an order of magnitude faster than the USM7 RK, which is an order of magnitude faster than Cartesian coordinates RK for orbits using low-thrust." [Vittaldev, 2010]

C

Flow Diagrams of the Implemented Propagators

This appendix contains the detailed flow diagrams of the code of the three main propagators: RK-Cart, RK-USM, TSI-USM and a version that used the Tudat propagator setup RK-Cart-TPS.

The legend is used in the flow diagrams of this section.

Table C.1: Legend used in the flow diagrams of this section

Description	Explanation
Yellow	Starting and stopping points of the program
Green	Manual user input inside the code
Blue	Input form an output to text file
Red	Data storage inside containers within the code
Gey oval	C++ class
Grey rectangle with missing corner	C++ information container (e.g. map)
Black arrow	Flow of processes in the code
Grey arrow	Flow of data within the program

C.1. RK-CART-TPS

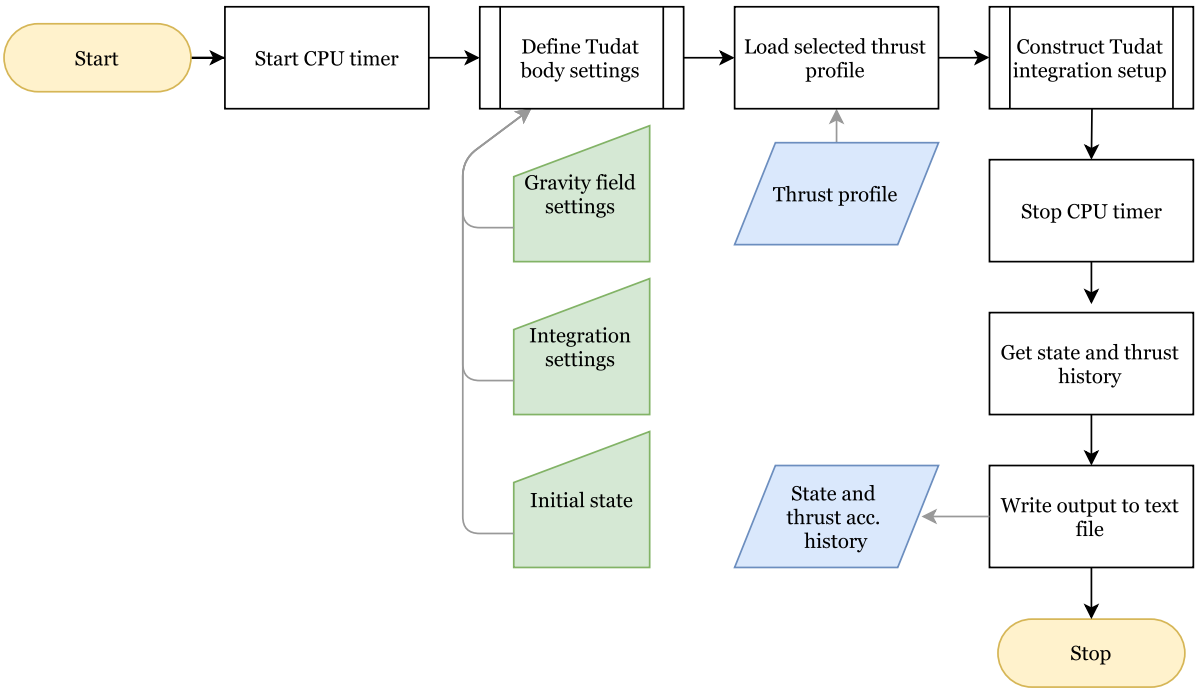


Figure C.1: Flow diagram of RK-Cart-TPS code (i.e. within Tudat propagation setup). A legend is provided in Table C.1.

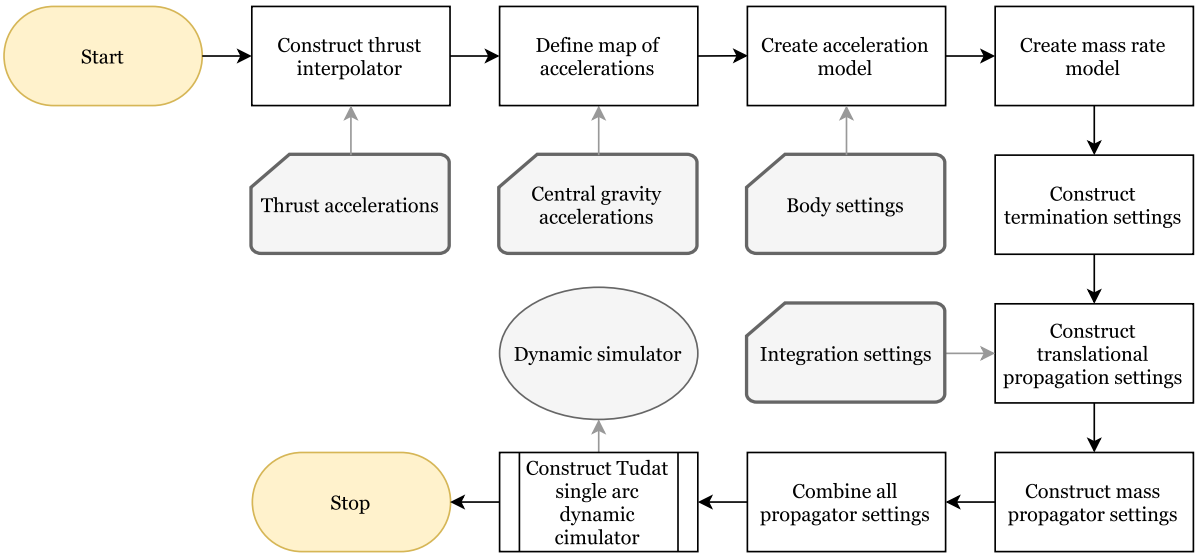


Figure C.2: Flow diagram of the function that performs the integration step in RK-Cart-TPS code. A legend is provided in Table C.1.

C.2. RK-CART AND RK-USM

The structure of the RK-Cart and RK-USM propagators are similar and therefore showing one diagram for both suffices.

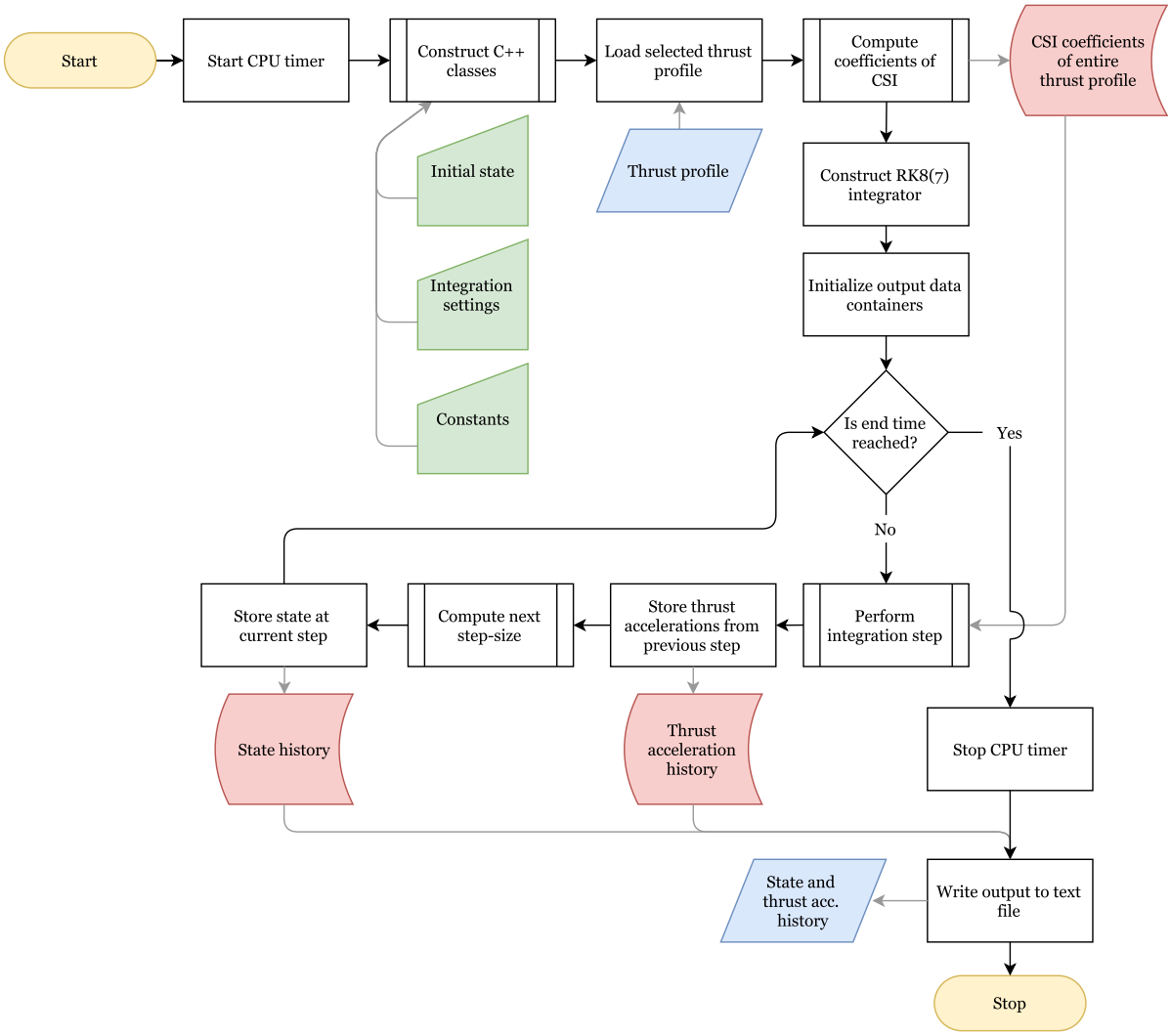


Figure C.3: Flow diagram of RK-Cart and RK-USM code. A legend is provided in Table C.1.

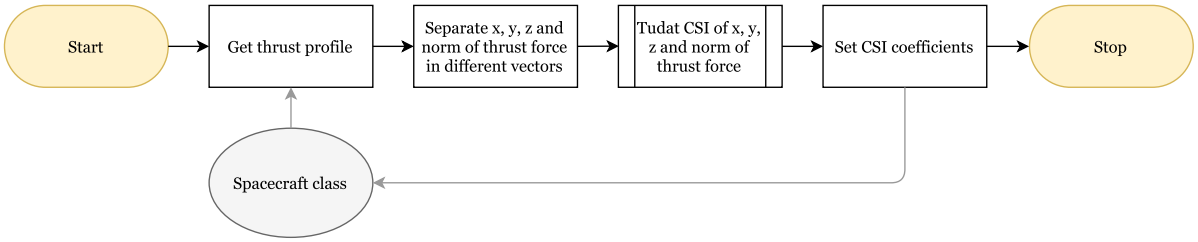


Figure C.4: Flow diagram of the function that computes and stores the CSI coefficients used in the RK-Cart and RK-USM code. A legend is provided in Table C.1.

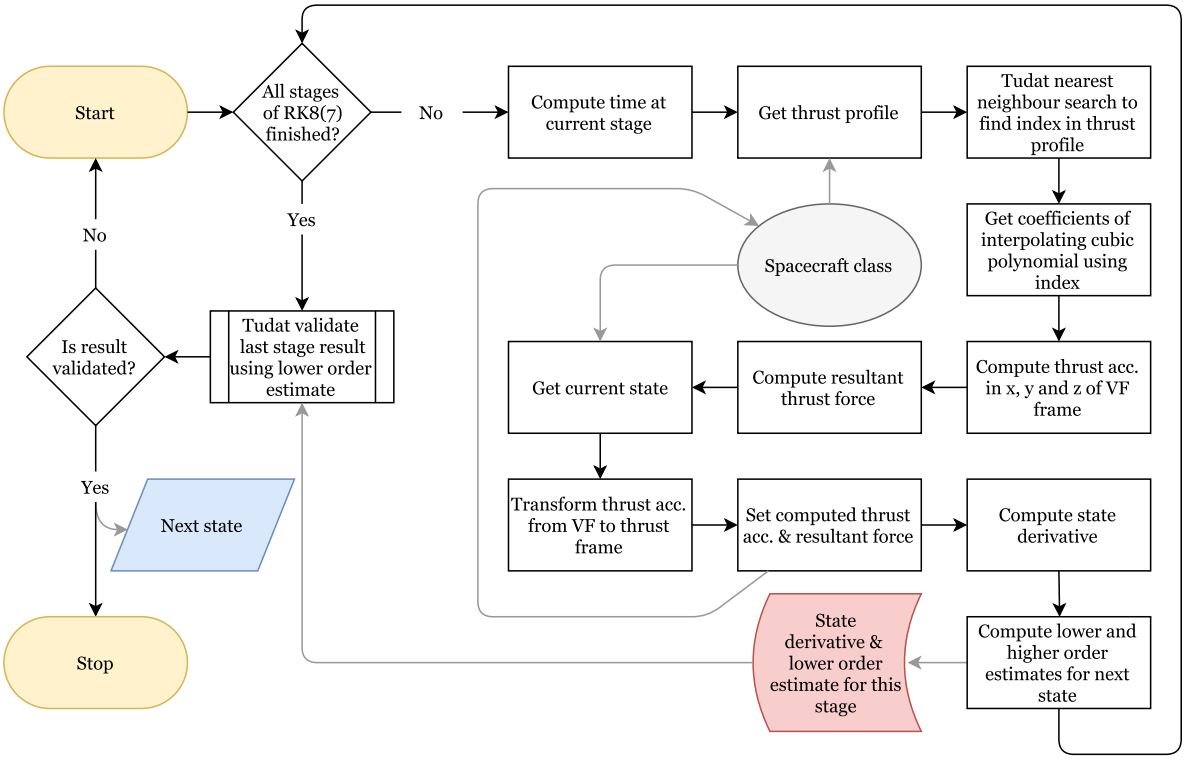


Figure C.5: Flow diagram of the function that performs the integration step in the RK-Cart and RK-USM code. A legend is provided in Table C.1.

C.3. GTSI-USM

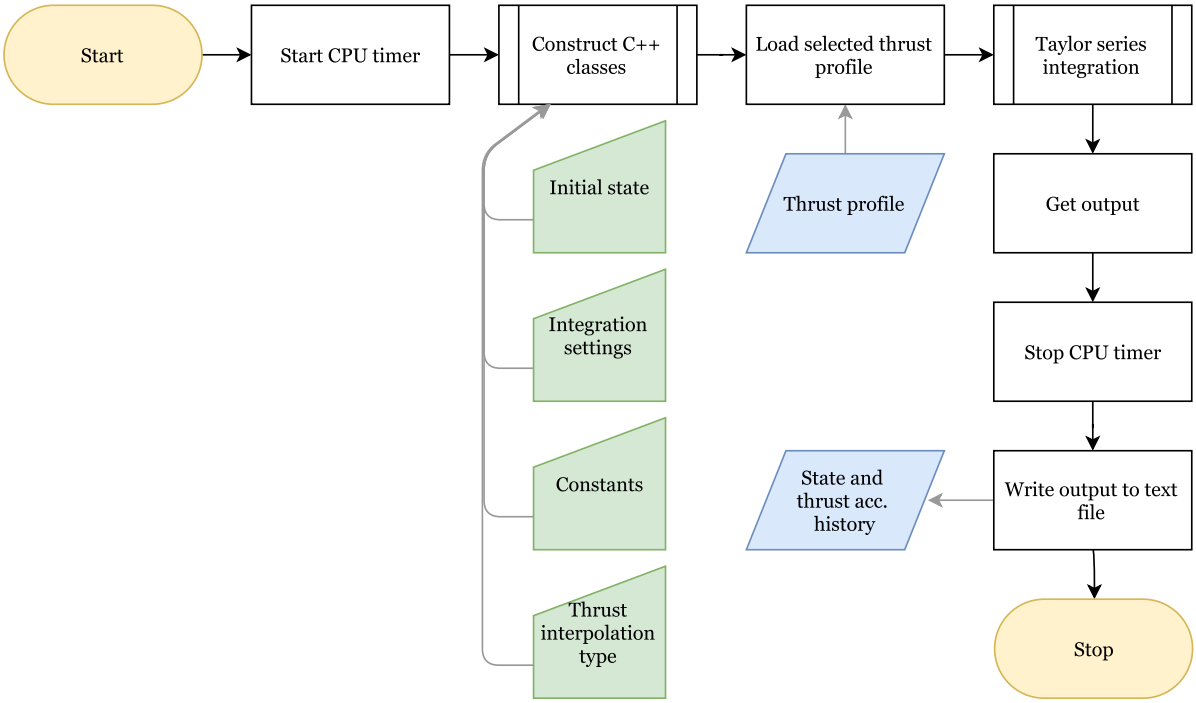


Figure C.6: Flow diagram of TSI-USM code. A legend is provided in Table C.1.

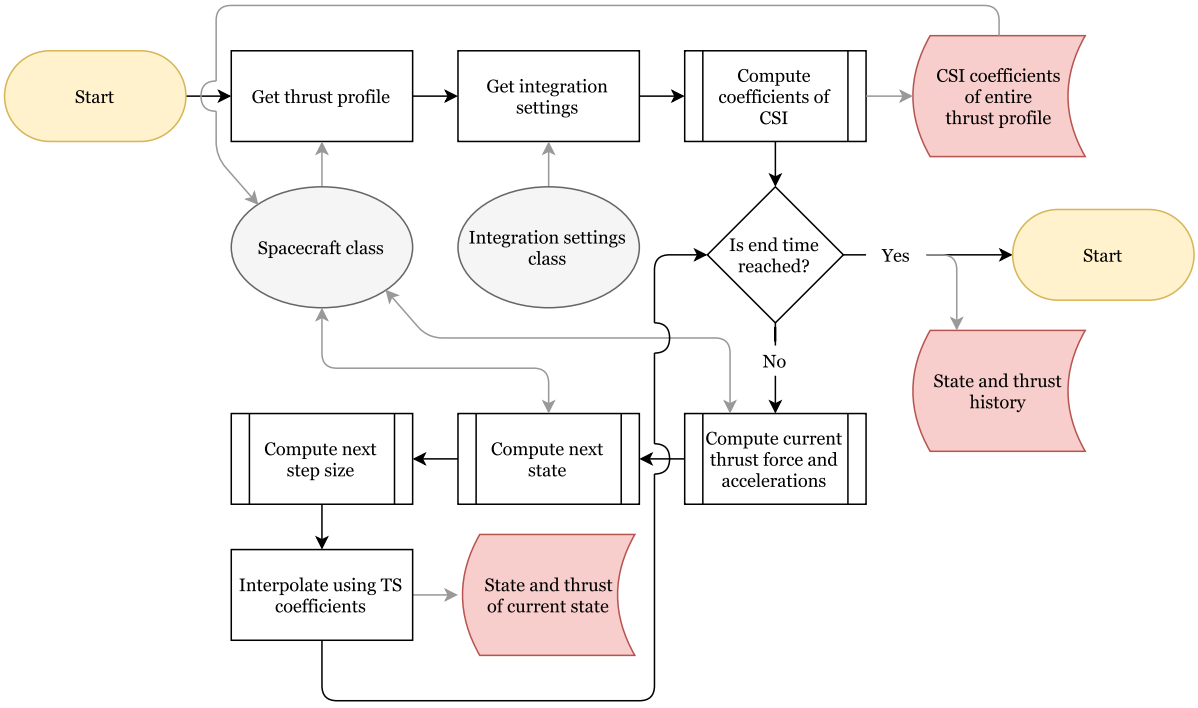


Figure C.7: Flow diagram of the TSI used in the TSI-USM code. A legend is provided in Table C.1.

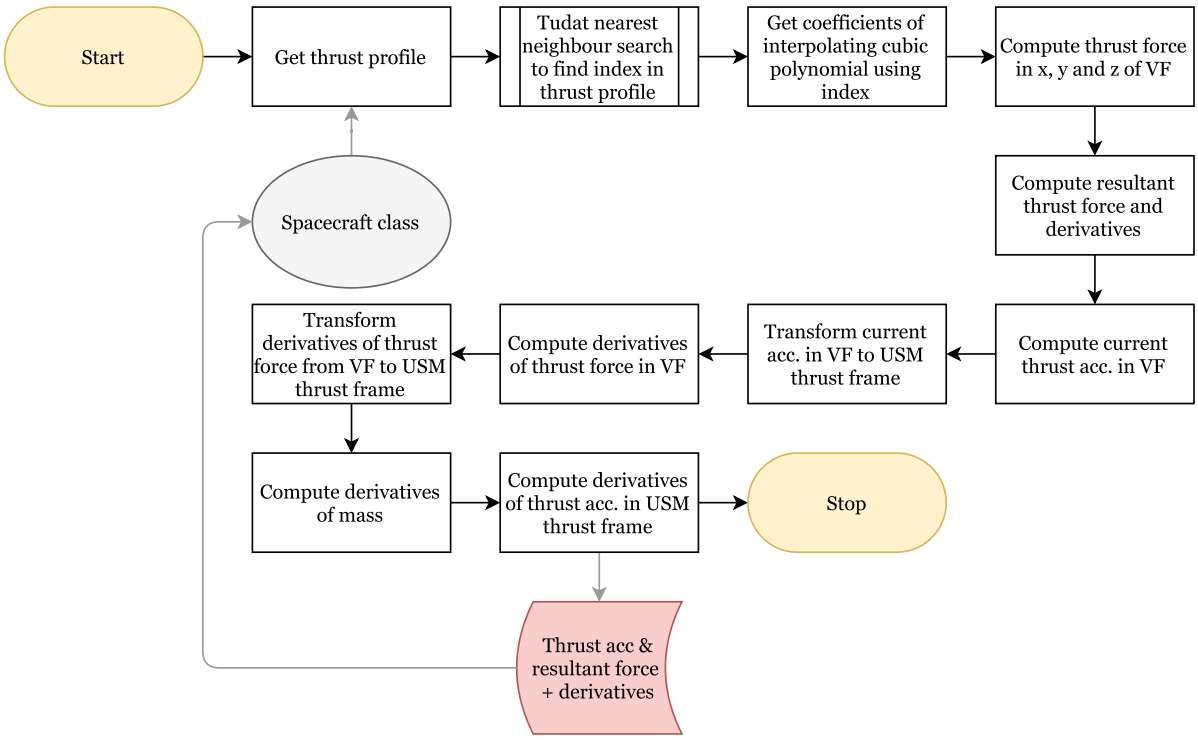


Figure C.8: Flow diagram of the function that computes the thrust accelerations and the resultant thrust force and its derivatives for use in the TSI-USM code. A legend is provided in Table C.1.

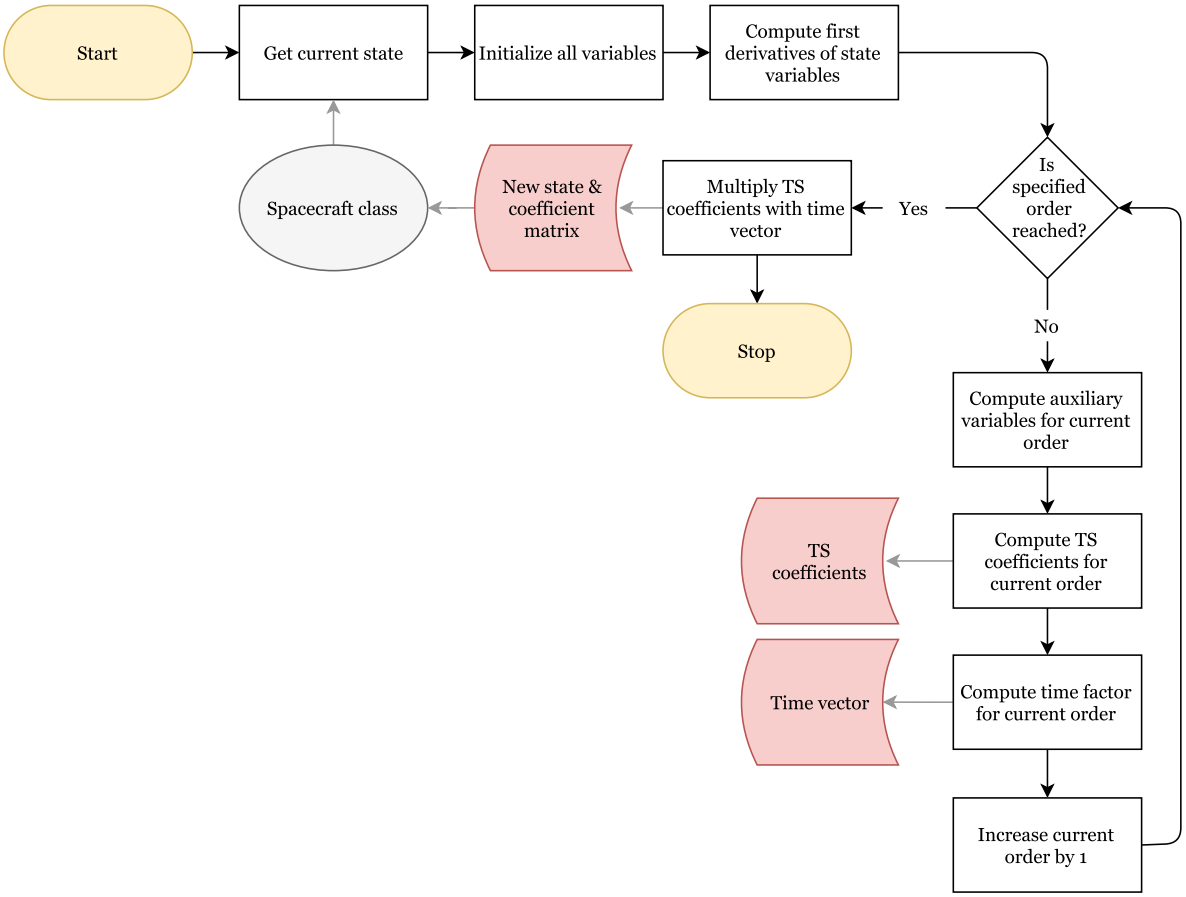


Figure C.9: Flow diagram of the function of the TSI-USM code that computes the next state. A legend is provided in Table C.1.

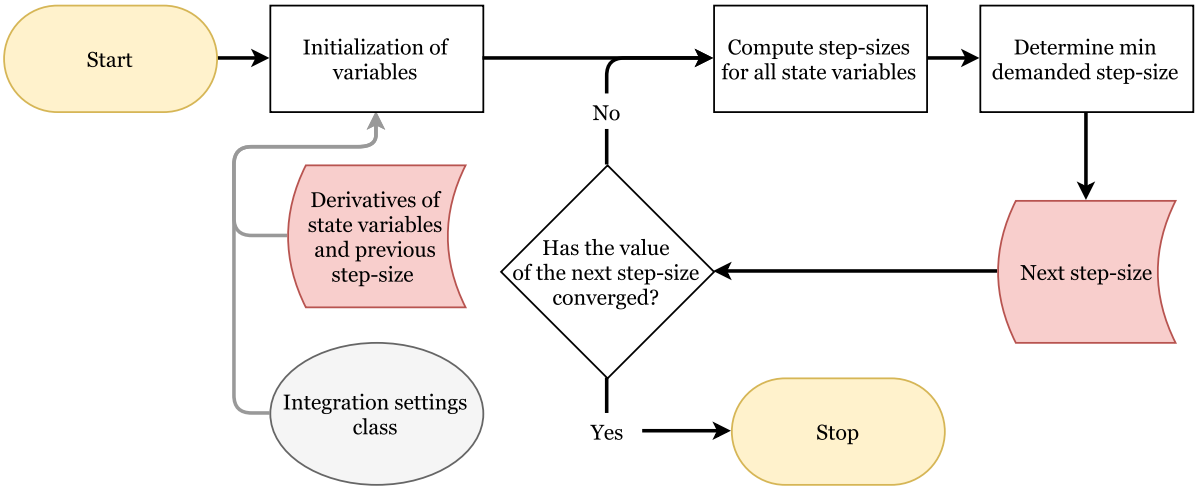


Figure C.10: Flow diagram of the function of the TSI-USM code that computes the next step-size. A legend is provided in Table C.1.