TU Delft Delft University of Technology

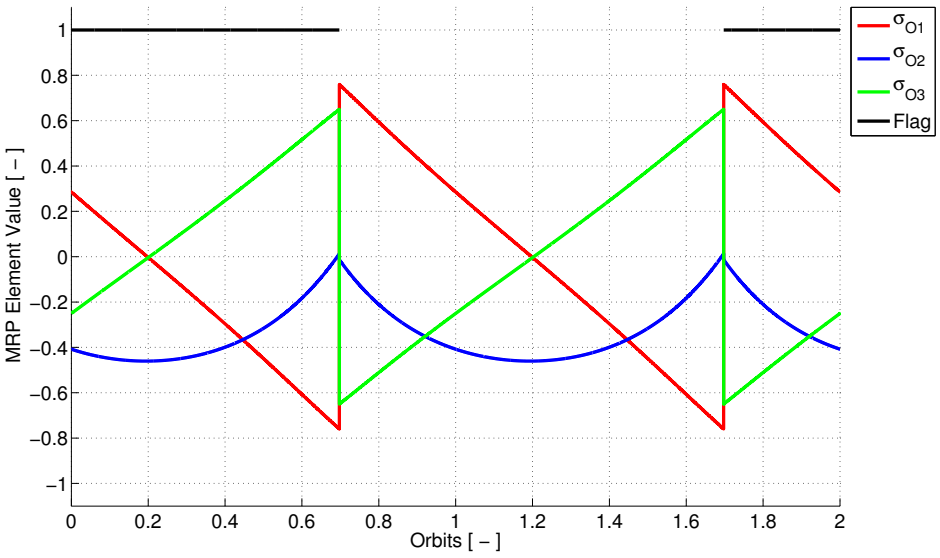
Master of Science Thesis

The Unified State Model

Derivation and Applications in Astrodynamics and Navigation

Vivek Vittaldev

May 31, 2010



The Unified State Model

Derivation and Applications in Astrodynamics and Navigation

A Thesis
Presented to
The Chair of Astrodynamics and Space Missions

by

Vivek Vittaldev

In Partial Fulfillment
of the Requirements for the Degree of
Master of Science in Aerospace Engineering

Delft University of Technology

May 31, 2010

Delft University of Technology
Chair of
Astrodynamics and Space Missions

This thesis is approved by a graduation committee consisting of:

Head of Department:

Prof. ir. B. A. C. Ambrosius
Delft University of Technology

Supervisor:

Dr. ir. E. Mooij
Delft University of Technology

Supervisor:

ir. M. C. Naeije
Delft University of Technology

Reader:

Dr. D. Izzo
European Space Agency ESA/ESTEC

Reader:

ir. R. Noomen
Delft University of Technology

Preface

This thesis report is the culmination of research carried out in order to graduate with a Master of Science degree in Aerospace Engineering from Delft University of Technology. I have immensely enjoyed working on this challenging and extremely interesting topic and I hope you, the reader, will enjoy reading it just as much. I would like to thank Prof. ir. B. A. C. Ambrosius, Dr. D. Izzo, Dr. ir. E. Mooij, ir. M. Naeije, and ir. R. Noomen for taking the time to read this report and for being on my graduation committee.

I am extremely grateful to my thesis supervisors Dr. E. Mooij and ir. M. Naeije for allowing me to work on this stimulating topic. They have always had their doors open for my many questions and have provided me with expert guidance during the whole thesis. Apart from clarifying all of my doubts, they have taught me to be critical whilst performing research. They have also given me the hope that meetings can indeed be enjoyable.

My fellow students, and now friends, on the *ninth* floor have made the months spent there fly by in the blink of an eye. It has been extremely fun sharing our numerous small frustrations and achievements, and discussing topics ranging from optimizers to our very own theme songs. I would also like to thank all of my friends for providing me with distractions and good company not only when I wanted them, but also when I needed them.

I would especially like to thank my parents, whose support has been invaluable for any and all of my endeavors. They have, and always will be, my biggest source of motivation and inspiration.

Abstract

The Unified State Model (USM) is a method for expressing orbits using a set of seven elements. The elements consist of a quaternion and three parameters based on the velocity hodograph. This elegant theory was proposed by Dr. Samuel Altman in 1971. Unfortunately, there have only been two other works in literature that use this theory with The set of equations for the USM in the more recent works being different to the original set. Thus, the theory was available with a literature pool of three works with two conflicting sets of equations.

Before the USM could be implemented, a choice had to be made regarding the set of equations to be used. Instead of finding the correct set in a trial and error way by implementing both sets, the complete set of equations for the USM was derived. The derivation provides concrete evidence that it is the more recent set of equations that is correct and not the original set. There is no complete step-by-step derivation of the USM available in literature and therefore, the derivation carried out in this thesis can be helpful to anybody interested in implementing the USM. In addition to the original USM, an alternative that uses Modified Rodrigues Parameters (MRP) instead of a quaternion has been proposed. This modified USM using MRP has only six state elements (USM6) instead of the seven state elements of the classical USM (USM7). The full equation set for the USM6 has been presented in addition to that of the USM7.

Numerical simulations using Runge Kutta (RK) integrators for orbit propagation comparing the USM6 and the USM7 with the traditional Cartesian coordinates have been carried out. 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 propulsion with CPU simulation time being an order of magnitude lower than for the simulation using Cartesian coordinates. The performance difference between the USM6 and the USM7 is minimal. Apart from orbit propagation, some numerical simulations have been carried out for atmospheric re-entry. For this case, the performances of both USM6 and USM7 were inferior to Cartesian coordinates.

For a realistic space mission, the ability to navigate the satellite is crucial. Some simulations of ground-station tracking of a satellite in orbit, to determine the accuracy of its position estimate, have been made. An essential part of navigation is filtering and therefore, four different non linear filters were implemented, i.e., the Extended Kalman Filter (EKF), the Unscented Kalman Filter (UKF), the Divided Difference Filter 1^{st} order (DD1), and the Divided Difference Filter 2^{nd} order (DD2). A Particle Swarm Optimizer (PSO) was used to tune these filters in an automated manner for the various models and testing scenarios. The PSO was able to successfully tune the various filters and it was found that the most accurate estimate occurs when the UKF with the USM7 is used. However, the differences in estimation accuracy between the different models with the UKF was minimal.

The final part of the thesis is Taylor Series Integration (TSI). This is another method for the numerical integration of functions. TSI uses recursive relations between the variables to compute the coefficients of a Taylor series that can approximate the function. The order of this approximation can be set to very high values and thus, large step-sizes can be taken whilst still producing very accurate results. There are some software packages available that can automatically generate the coefficients. However, no external software packages were used in this thesis.

The TSI is more challenging than the RK integration methods to implement. However, TSI can produce very accurate integration results with very little CPU time. 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 orders 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 propulsion.

Contents

List of Figures

List of Tables

Nomenclature

Notation

$\otimes$	Modified Rodrigues Parameters multiplication
$\mathbf{a}$	Vector
$\hat{\mathbf{a}}$	Unit vector
$[\mathbf{a}\times]$	Skewsymmetric form of vector $\mathbf{a}$
$\dot{\mathbf{a}}$	Time derivative of vector $\mathbf{a}$
$\ddot{\mathbf{a}}$	Double time derivative of vector $\mathbf{a}$
$\dot{a}$	Time derivative of scalar a
$\mathbf{a}^H$	Hermitian conjuagate of $\mathbf{a}$
a_i	Component i of vector $\mathbf{a}$
$\mathbf{a}^T$	Transpose of vector $\mathbf{a}$
c	Cosine
$\mathbf{C}$	Rotation matrix
$\mathbf{C}_{b,a}$	Rotation matrix from reference frame a to reference frame b
$\mathbf{C}_i(\theta)$	Euler rotation about principal axis i with angle θ
$\tilde{\mathbf{D}}_{\Delta\mathbf{x}}\mathbf{f}$	First order divided difference operator
$\tilde{\mathbf{D}}^2_{\Delta\mathbf{x}}\mathbf{f}$	Second order divided difference operator
δ	Partial difference operator
$\mathcal{F}_i$	Reference frame i
μ	Average operator
s	Sine
Tr	Trace

Latin Symbols

In case the units are not specified, it means that the units can vary depending on the implementation.

$\mathbf{a}$	Euler axis	[-]
$\mathbf{a}$	Acceleration	[m s^{-2}]
a	Semi-major axis	[m]
A	Cross-sectional area	[m^2]
A	Azimuth	[rad]
b	Semi-minor axis	[m]
b	Bias	[m s^{-2}]
$\mathbf{b}$	Baseline vector	[m]
b_{ref}	Reference length for roll and yaw	[m]
$\mathbf{C}$	Constant velocity in normal direction	[m s^{-1}]
c	Speed of light	$299,792,458$ [m s^{-1}]
c_i	Cosine of angle between a vector and a reference unit vector $\hat{\mathbf{a}}_i$	[-]
$c_{i,k}$	Element (i,k) of rotation matrix	[-]
C_D	Drag coefficient	[-]

C_L	Lift coefficient	[-]
C_l	Roll moment coefficient	[-]
C_m	Pitch moment coefficient	[-]
C_n	Yaw moment coefficient	[-]
$C_{n,m}$	Geopotential coefficient	[-]
C_R	Solar radiation pressure coefficient	[-]
c_{ref}	Reference length for pitch	[m]
C_S	Side force coefficient	[-]
$\mathbf{D}$	Diagonal matrix	
D	Drag	[N]
d	Drift	[rad s^{-1}]
E	Elevation	[rad]
$\mathcal{E}$	Orbital energy per unit mass	[J kg^{-1}]
e	Eccentricity	[-]
$\mathbf{e}$	Eccentricity vector	[-]
$\mathbf{e}$	Eigenvector	[-]
$\mathbf{e}$	*a posteriori* error	
$\mathbf{e}^-$	*a priori* error	
$\mathbf{F}$	Force	[N]
f	Ellipsoid flatness	[-]
f	Frequency of signal	[Hz]
G	Universal gravity constant	6.67428×10^{-11} [N m^2 kg^{-2}]
$\mathbf{g}$	Acceleration due to gravity	[m s^{-2}]
$\mathbf{h}$	Angular momentum	[m^2 s^{-1}]
h	Height	[m]
h	Interval length	
H_0	Density scale height	[m]
$\mathbf{I}$	Identity matrix	[-]
i	Inclination	[rad]
i	Complex number	[-]
$\mathbf{I}_n$	(n×n) identity matrix	[-]
$\mathbf{J}$	Inertia tensor	[kg m^2]
j	Complex number	[-]
$\mathbf{j}$	Jerk	[m s^{-3}]
J_n	Jeffreys constant	[-]
k	Complex number	[-]
$\mathbf{K}$	Gain	[-]
L	Dimension of system	[-]
$\mathbf{L}$	Lower triangle matrix	
L	Lift	[N]
L	Roll moment	[N m]
L_Q	Weight for adaptive filtering	[-]
$\mathbf{M}$	Torque	[N m]
M	Pitch moment	[N m]
m	Mass	[kg]
m	Misalignment	
N	Prime vertical	[m]
N	Yaw moment	[N m]
ΔN	Integer difference	[-]
p	Magnitude of the Modified Rodrigues Parameters vector	[-]
p	Semi-latus rectum	[m]
P	Solar radiation pressure	[N m^{-2}]
p	Parameter used in the dynamics of the Unified State Model	[-]
$\mathbf{P}$	*a posteriori* error covariance	
$\mathbf{P}^-$	*a priori* error covariance	
$P_{n,m}$	Associate Legendre polynomial of degree n and order m	[-]

P_n	Legendre polynomial of degree n	[-]
$\mathbf{Q}$	Process noise covariance	
$\mathbf{Q}$	Householder matrix	
$\mathbf{Q}^*$	Process covariance residual	
$\hat{\mathbf{Q}}$	Process covariance estimate	
q_{dyn}	Dynamic pressure	[N m^{-2}]
$\mathbf{r}$	Distance	[m]
$\mathbf{R}$	Constant hodographic velocity	[m s^{-1}]
$\mathcal{R}$	Universal gas constant	8.314472 [J K^{-1} mol^{-1}]
$\mathbf{R}$	Measurement noise covariance	
r_a	Apocenter radius	[m]
r_p	Pericenter radius	[m]
$\mathbf{S}$	Lower triangle square root matrix	
S	Side force	[N]
s_i	Scale factors	[-]
$S_{n,m}$	Geopotential coefficient	[-]
S_{ref}	Aerodynamic reference area	[m^2]
T	Orbital period	[s]
T	Absolute temperature	[K]
t	Time	[s]
T_p	Time between 2 consecutive crossings of the semi-latus rectum	[s]
U	Potential	[N m kg^{-1}]
u	Argument of latitude	[rad]
$\mathbf{u}$	Input vector	
$\mathbf{U}$	Upper triangular matrix	
$\mathbf{v}$	Velocity	[m s^{-1}]
$\mathbf{v}$	Measurement noise	
v_c	Circular velocity	[m s^{-1}]
v_{esc}	Escape velocity	[m s^{-1}]
V_G	Groundspeed	[m s^{-1}]
v_∞	Velocity at infinity	[m s^{-1}]
$\mathbf{w}$	System noise	
$\mathbf{x}$	State vector	
$\hat{\mathbf{x}}$	*a posteriori* state estimate	
$\hat{\mathbf{x}}^-$	*a priori* state estimate	
$\mathbf{z}$	Measurement vector	
$\mathcal{Z}$	Estimated measurement sigma point	
$\hat{\mathbf{z}}$	predicted measurement	

Greek Symbols

In case the units are not specified, it means that the units can vary depending on the implementation.

α	Angular acceleration	[rad / s^2]
α	Scaling parameter	[-]
α	Angle of attack	[rad]
β	Angle of sideslip	[rad]
β	Electrical line bias	[-]
γ	Angle between 2 Euler axes	[rad]
γ	Parameter used in the dynamics of the Unified State Model	[-]
γ	Flight path angle	[rad]
$\boldsymbol{\epsilon}$	Vector part of Euler parameters	[-]
ε	Reflectivity coefficient	[-]
η	Scalar part of Euler parameters	[-]

θ	Pitch angle	[rad]
λ	Eigenvalue	[-]
λ	Longitude	[rad]
λ	Sum of the right ascension of ascending node and the argument of latitude	[-]
λ	Scaling parameter	[-]
κ	Scaling parameter	[-]
μ	Gravitational parameter	[N m^2 kg^{-1}]
ν	True anomaly	[rad]
$\Delta\Phi$	Phase difference	[-]
ρ	Density	[kg m^{-3}]
ρ	Range	[m]
ρ_0	Reference height density	[kg m^{-3}]
σ	Trace of rotation matrix	[-]
σ	Bank angle	[rad]
$\boldsymbol{\sigma}$	Modified Rodrigues Parameters vector	[-]
$\boldsymbol{\sigma}^S$	Shadow Modified Rodrigues Parameters vector	[-]
Φ	Euler angle	[rad]
Φ	Solar radiation flux	[W m^{-2}]
ϕ	Roll angle	[rad]
ϕ_{gc}	Geocentric latitude	[rad]
ϕ_{gd}	Geodetic latitude	[rad]
χ	Sigma point	
χ	Heading angle	[rad]
ψ	Yaw angle	[rad]
ω	Angular velocity	[rad / s]
Ω	Right ascension of ascending node	[rad]
ω	Argument of pericenter	[rad]

Acronyms and Abbreviations

ARTEMIS	Autonomous Rendezvous and rapid Turnaround Experiment Maneuverable Inspection Satellite
AB	Adams-Bashforth
ABM	Adams-Bashforth-Moulton
AM	Adams-Moulton
COESA	Committee on Extension to the Standard Atmosphere
DD1	First Order Divided Difference Filter
DD2	Second Order Divided Difference Filter
DDF	Divided Difference Filter
DOF	Degrees Of Freedom
DOP	Dilution of Precision
DOP	Dormand & Prince
DTM	Drag Temperature Models
ECEF	Earth Centered Earth Fixed
ECI	Earth Centered Inertial
EKF	Extended Kalman Filter
EnKF	Ensemble Kalman Filter
ESA	European Space Agency
ESRO	European Space Research Organisation
FILG	Filipi and Gräf
GDOP	Geometric Dilution of Precision
GEO	Geosynchronous Earth Orbit
GPS	Global Positioning System

HDOP	Horizontal Dilution of Precision
HEOS	High Eccentric Orbiting Satellite
IAU	International Astronomical Union
IEKF	Iterated Extended Kalman Filter
IERS	International Earth Rotation and Reference Systems Service
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
ISRO	Indian Space Research Organisation
JPL	Jet Propulsion Laboratories
LEO	Low Earth Orbit
LKF	Linear Kalman Filter
MRP	Modified Rodrigues Parameters
MSIS	Mass Spectrometer and Incoherent Scatter
NASA	National Aeronautics and Space Administration
NRLMSISE	United States Naval Research Laboratory Mass Spectrometer and Incoherent Scatter Radar Exosphere
NSSDC	National Space Science Data Center
ODAI	Orbit Determination and Improvement
RK	Runge-Kutta
RKF	Runge-Kutta-Fehlberg
RKN	Runge-Kutta-Nyström
SARSAT	Search And Rescue Satellite
SLR	Satellite Laser Ranging
SMRP	Shadow Modified Rodrigues Parameters
TD	Total Density
TDOP	Time Dilution of Precision
TEAM	Terminal Area Energy Management Phase
TS	Taylor Series
TSI	Taylor Series Integration
USM6	6 element Unified State Model
USM7	7 element Unified State Model
UD	Upper Diagonal
UKF	Unscented Kalman Filter
USAF	United States Air Force
USM	Unified State Model
USMRP	Unified State Model using modified Rodrigues Parameters
UT	Universal Time
UTC	Coordinated Universal Time
UTIAS	Institute for Aerospace Studies University of Toronto
VDOP	Vertical Dilution of Precision
WGS	World Geodetic System

Chapter 1

Introduction

The space industry has now become a major economic player. Almost every form of communication in the modern world utilizes satellites in one form or another. Thus, space flight has gone from just a simple orbiting radio beacon to complex instruments in very precise orbits. Space is not just the domain of economically developed nations anymore. Even economically developing countries such as India and China are taking part in space flight. Good examples of this are the spacewalk mission by China in September 2008 and the Chandrayaan-1 Moon mapping mission launched by India in October 2008. Not only are an increasing number of countries taking part in space flight, but the missions are getting increasingly complex. Space flight is now also in the reach of academic institutions. An example of a university built satellite is the Delfi-C^3, which was designed and constructed by students of Delft University of Technology, and was launched in 2007[1]. This satellite was built using a CubeSat Kit that facilitates the ease of constructing satellites. An increasing number of satellite missions combined with an increased complexity of the average mission put stringent demands on precise orbit simulation and determination of satellites. Thus, any method theory that could provide an increase in computation speed, should be investigated further.

Cartesian coordinates and classic Keplerian elements are well-established for the propagation and visualization of orbits. However, there are many other ways in which satellite orbits can be described. One of these methods is a novel concept known as the Unified State Model (USM). The USM, first proposed by Samuel P. Altman [Altman, 1972], uses quaternions and velocity hodograph parameters to express orbits. The 4 quaternion elements express the orientation of a reference frame fixed to the orbiting body, with respect to an inertial frame fixed to the central body. The velocity hodograph parameters give information about the size and shape of the orbit. During an unperturbed orbit, the velocity hodograph parameters remain constant and only the quaternion varies with time. This is because the orbital shape remains unchanged for such an orbit, and only the location of the orbiting body within the orbit changes with time. When perturbations are present, the variation of the hodographic parameters will be small compared to the variation of Cartesian coordinates. Thus, the USM is expected to have better numerical stability than Cartesian coordinates as there are 4 rapidly varying and 3 slowly varying elements, compared to the 6 rapidly varying elements. The classic Keplerian elements have only one rapidly varying element, which is the true anomaly. However, singularities exist and the equations for the influence of perturbations are rather complex.

There has not been much research carried out on the USM since its inception. If the theory has been mentioned, it has always been in relation to navigation. Since the USM has not been used for any real life applications, a small historic overview of this theory and its 2 main proponents is provided here.[2] Even though Altman is and should solely be credited as the inventor of the USM, it is interesting to note that both the key ingredients, quaternions and Hodographs, were invented

[1] http://www.delfic3.nl

[2] The biographical information here is simply what could be found on the internet, and may contain some discrepancies.

by Sir William Rowan Hamilton. This situation is perfectly summed up by the quotation by Sir Isaac Newton

If I have seen further it is by standing on the shoulders of Giants.

The first proponent, and inventor, of the USM, Samuel P. Altman is now a senior member of AIAA. The chronologically earliest information that can be found online about him is that he was a Captain in the United States Air Force (USAF). In 1952 he wrote a memorandum report about the *Equations of Motion of the F-80 Aileron Boost* [Altman, 1952]. In 1953, he received a patent[3] for a servo loading stand. After his career at the Air Force, he went on to work in the *Spacecraft Department*, which was a part of the *Missile and Space Division*, of *General Electric*. Here, he started work on using Hodographs, found in Chapter 5, for orbits, which is one of the key pillars of the Unified State Model. The work on Hodographs was carried out under NASA contracts. The first publication on this topic by Altman was a book called *Orbital Hodograph Analysis* in 1965 [Altman, 1965]. From 1965 till 1968, Altman authored 3 technical reports under NASA contracts about Hodographs, [Altman, 1967a], [Altman, 1967b], and [Altman and Pistiner, 1968]. During this period, there seem to be many other NASA contracts given out for Hodographic analysis as seen in [Sun, 1965] and [Eades, 1968]. The work on Hodographs, combined with quaternions, led to the creation of the Unified State Model, which Altman published in 1972, [Altman, 1972]. After that point, there is no information available about Altman till 1975 when he wrote a paper about observation models for satellite state estimation that can be used with the USM, [Altman, 1975].

In 1981, the second proponent of this theory, Dr. Paul W. Chodas comes into the picture. Dr. Chodas is a research scientist at the Jet Propulsion Laboratories (JPL), has an asteroid named after him, and is also a senior member of AIAA. In 1981, Chodas wrote a technical note at the Institute for Aerospace Studies at the University of Toronto (UTIAS) about the usage of the Unified State Model for orbit determination. He received funding from the Department of Communications of Canada for this study. He still works on orbit determination, but this time on near-Earth asteroids. His technical note about the Unified State Model, [Chodas, 1981] is very clearly written and presents the USM in a manner, in which it can easily be implemented. However, it is claimed in [Chodas, 1981] that there is mistake in original USM dynamics equations from [Altman, 1975].

After the excellent technical note by Chodas, the USM has been mentioned only once more in a paper by J. R. Raol, and N. K. Sinha in 1985, [Raol and Sinha, 1985]. Both the authors were researchers at McMaster University, Canada. This is another interesting fact that all the work involving the USM seems to have been carried out in Canada. At one point in their paper, it is mentioned that the USM is widely used. If this was indeed the case, there do not seem to be any other publications than the ones already mentioned above. Again, orbit determination was carried out in [Raol and Sinha, 1985]. The set of equations for the USM found in [Raol and Sinha, 1985] is consistent with the ones found in [Chodas, 1981].

The explanation of the USM in [Chodas, 1981] contains more detail on implementation than [Altman, 1972]. It is, however, not easily available and it does not provide a rigorous derivation of the given equations. Since there are two sets of equations provided in three papers, a comprehensive derivation of the equations of the USM is shown, in order to discern the correct method. Thus, the derivation of the USM is one of the major components of this thesis. The motivation for this can perhaps best be summed up in the following quote about the USM by Hilderic Browne in an article in the newsletter of the Ottawa Centre of the Royal Astronomical Society of Canada (RASC)[4]

The USM parameter set was arcane to me then and is now a complete mystery...

Hilderic Browne worked with Chodas on the investigation of the USM. If this model is to be used during the thesis, and by others in the future, it is imperative to demystify it and make it accessible. Unfortunately, the original paper, [Altman, 1972], is not very easy to follow. [Chodas,

[3] United States Patent 2641925

[4] http://ottawa-rasc.ca/astronotes/1999/an9901p3.html

1981] is easier to read through, and contains enough information to be able to successfully implement the USM. It still does not, however, fully explain the mathematics behind this model, which would be essential in carrying out further scientific research. Also, there is a discrepancy between the USM found in [Altman, 1972] and the one found in [Chodas, 1981]. The only other paper found in literature, [Raol and Sinha, 1985], uses the same model as [Chodas, 1981], but without any justification. The change in the model proposed by [Chodas, 1981] appears to be more correct as an incorrect assumption in [Altman, 1972] was pointed out, and the explanation of the USM in [Chodas, 1981] contains more detail on implementation than [Altman, 1972].

An MSc thesis study cannot consist of only a derivation. Therefore, different applications have been found for the USM. In this regard, this thesis study can be viewed as a conceptual design document for the USM. There is no single physical problem fully solved, such as an optimization of a trajectory to a certain planet or asteroid. However, all of the information presented here can help in setting up a program that can solve such a problem.

Other than the derivation of the traditional USM, a new method that uses Modified Rodrigues Parameters (MRP) instead of quaternions has been proposed. The numerical performance of the traditional USM and the modified version of the USM has been compared to the Cartesian coordinates. Other than traditional Runge Kutta (RK) forms of integration, Taylor Series Integration (TSI) has also been explained and implemented. Finally, the navigation performance of the USMs has been compared with Cartesian coordinates.

The thesis has been divided into the following chapters:

The USM was a novel concept when it was first presented and it still has not been used to its full potential. One of the reasons for this is a dearth of the mathematical background material available on the USM. For this reason, the first part of this report deals with the theoretical set up of the USM. The USM uses quaternions and hodographic parameters in order to express the location of a spacecraft. Thus, kinematics, various attitude parameters, and hodographs are treated. To understand orbits, it is essential to have an overview of astrodynamics and the space environment with the various perturbations. With all this background material, it is possible to easily follow the derivation and the applications of the USM.

2. **Kinematics and Reference Frames,** where the kinematics of classical mechanics is addressed, with an emphasis on rotational motion.

3. **Attitude Parameters,** where quaternions, Euler angles and the Modified Rodrigues Parameters are described.

4. **Astrodynamics and Orbital Mechanics,** where the dynamics and theory required for an understanding of space flight is addressed.

5. **Hodograph Theory,** where hodographs in velocity, acceleration, and jerk spaces are derived.

After addressing the theory required to understand the USM, it is finally derived in second part.

6. **Unified State Model,** where the traditional USM, and the new alternative proposed here are extensively derived.

7. **Guide for Applying the Unified State Model,** where the conversions from the USM to other oft-used parameter sets for spacecraft state expression, and vice-versa are presented. A guide on how to implement the USM is also given.

Finally, all the theory that is presented is implemented in three different applications to compare results with corresponding Cartesian implementations.

8. **Orbit Propagation,** where various orbital trajectories are simulated using the RK methods with the USMs and Cartesian coordinates.

9. **Re-entry Dynamics,** where an atmospheric re-entry is simulated using the RK methods with the USMs and Cartesian coordinates.

10. **Recursive Filtering Techniques,** where the various estimation techniques are introduced with the main focus being on Kalman and Divided Difference filters.

11. **Ground Station Tracking,** where the state of a spacecraft is estimated using measurements from a ground station. Various filters are used with the USMs and Cartesian coordinates. Also, a Particle Swarm Optimizer (PSO) is implemented to automate the tuning of the various filters.

Since Taylor Series Integration is a whole topic all by itself, it is dealt with at the end. Theory and applications of orbit propagation will be discussed. This is followed by the Conclusions of this thesis study.

12. **Taylor Series Integration,** where the method of implementing TSI is shown along with the specific equations for the USM7 and Cartesian coordinates, for various cases.

13. **Application of the Taylor Series Integration,** where all the TSI equations presented in Chapter 12 are applied to get some results.

14. **Conclusions and Future Work,** the most important conclusions and recommendations spread throughout the thesis study are presented in a compact manner here.

Chapter 2

Kinematics and Reference Frames

Mechanics is the branch of physics that deals with the study of motion. In Aerospace problems, we are almost always dealing with systems that are constantly in motion. This is especially true for spacecraft, which travel with velocities in the order of kilometers per second. Therefore, it is very important to deal with fundamental mathematics such that the more complex theories can be thoroughly understood. It is unfortunately not possible to do justice to all the intricacies of the theory behind mechanics in one chapter. Thus, the reader is referred to the excellent sources [Török, 2000; Hughes, 1986] for more information.

Mechanics can be split up into the following two categories

Kinematics Description of the motion without paying attention to any resultant forces or the mass

Dynamics Motion under the influence of external forces, and their causes

The mechanics dealt with in this section are Newtonian mechanics, which are based on Newton's 3 laws of motion for a point mass in an inertial frame. An inertial frame of reference is a reference frame that undergoes no acceleration. Newton's famous 3 laws are

First Law A particle will have a constant velocity, or remain at rest, as long as there is no resultant force acting on it.

Second Law The resultant force acting on a particle is directly proportional to the time rate of change of its linear momentum.

Third Law The forces that 2 particles exert on each other lie on the line joining the 2 particles. These forces are equal in magnitude, but opposite in direction.

The kinematics required to be able to describe the orbital motion of the USM can be found in this chapter. This includes inertial reference frames, ways to transform one reference frame to another, and non-inertial reference frames.

2.1 Reference Frames

A reference frame is always required to describe a position, or a motion. For this three dimensional world, a set of 3 noncoplanar vectors is necessary to form a reference frame, $\mathcal{F}_i$. In this way, the direction of any vector can be expressed as components of the 3 reference vectors.

Instead of using any 3 noncoplanar vectors, the standard method is to chose a *dextral*, *orthonormal triad* as the reference vectors. This means that the set of vectors is right-handed and all the

vectors are mutually perpendicular and of unit length. In this reference frame, a vector can be expressed as the product of the magnitude of the vector with the direction cosines of this vector with each of the reference vectors. Thus, the vector $\mathbf{v}$ in $\mathcal{F}_a$ is expressed as

$$\mathbf{v} = v\left(c_1\hat{\mathbf{a}}_1 + c_2\hat{\mathbf{a}}_2 + c_3\hat{\mathbf{a}}_3\right) \tag{2.1}$$

In Eq. (2.1), c_i is the cosine of the angle between $\mathbf{v}$ and $\hat{\mathbf{a}}_i$.

2.1.1 Rotation Matrices

It is not always possible to work in the same reference frame and thus, it is necessary to expresses vectors given in one reference frame, in another with the same origin. This can be done with the use of a rotation matrix, which is also known as a Direction Cosine Matrix (DCM). If there are two reference frames $\mathcal{F}_b$ and $\mathcal{F}_a$, the reference vectors of $\mathcal{F}_b$ can be related to the reference vectors of $\mathcal{F}_a$ in the following manner

$$\hat{\mathbf{b}}_1 = c_{1,1}\hat{\mathbf{a}}_1 + c_{1,2}\hat{\mathbf{a}}_2 + c_{1,3}\hat{\mathbf{a}}_3 \tag{2.2a}$$
$$\hat{\mathbf{b}}_2 = c_{2,1}\hat{\mathbf{a}}_1 + c_{2,2}\hat{\mathbf{a}}_2 + c_{2,3}\hat{\mathbf{a}}_3 \tag{2.2b}$$
$$\hat{\mathbf{b}}_3 = c_{3,1}\hat{\mathbf{a}}_1 + c_{3,2}\hat{\mathbf{a}}_2 + c_{3,3}\hat{\mathbf{a}}_3 \tag{2.2c}$$

In Eq. (2.2), $c_{i,j}$ is the cosine of the angle between $\hat{\mathbf{b}}_i$ and $\hat{\mathbf{a}}_j$.

A DCM $\mathbf{C}_{b,a}$ can now be constructed from all the direction cosines as

$$\mathbf{C}_{b,a} = \begin{bmatrix} c_{1,1} & c_{1,2} & c_{1,3} \\ c_{2,1} & c_{2,2} & c_{2,3} \\ c_{3,1} & c_{3,2} & c_{3,3} \end{bmatrix} \tag{2.3}$$

TheDCM is orthonormal, meaning that its columns are orthogonal unit vectors. According to [Hughes, 1986], this gives it the following interesting and useful properties:

$$\mathbf{C}\mathbf{C}^T = \mathbf{I} \tag{2.4a}$$
$$\mathbf{C}^T\mathbf{C} = \mathbf{I} \tag{2.4b}$$
$$\mathbf{C}^T = \mathbf{C}^{-1} \tag{2.4c}$$
$$\det \mathbf{C} = 1 \tag{2.4d}$$
$$\tag{2.4e}$$

2.1.2 Euler's Theorem

Euler's theorem states that a rotation between any two frames can be expressed by a rotation of the angle Φ about an axis through their common origin. The angle Φ is known as the Euler angle. The axis has the direction of a unit vector $\hat{\mathbf{a}}$ and is known as the Euler axis. This can be seen in Fig. 2.1.

To find the Euler axis and angle, it is necessary to carry out an eigenvalue analysis on $\mathbf{C}$. Let $\mathbf{C}$ have the eigenvalue λ and the corresponding eigenvector $\mathbf{e}$.

$$\mathbf{C}\mathbf{e} = \lambda\mathbf{e} \tag{2.5}$$

Both sides of Eq. (2.5) are left multiplied by their respective Hermitian conjugates. The Hermitian conjugate is found by taking the transpose of the matrix and then by using the complex

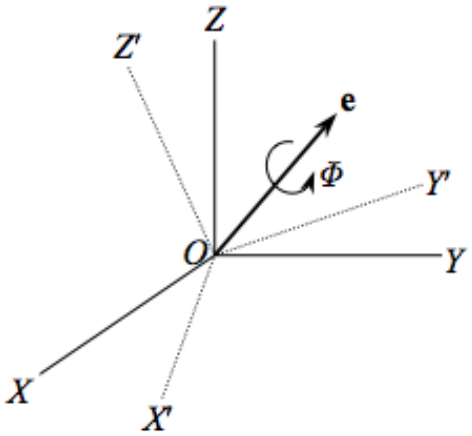


Figure 2.1: Euler's Theorem [Tewari, 2007]

conjugate of each element. To illustrate this, let there be a matrix $\mathbf{A}$ with complex components defined as

$$\mathbf{A} = \begin{bmatrix} a+bi & c-di \\ e-fi & g+hi \end{bmatrix} \tag{2.6}$$

The Hermitian conjugate of $\mathbf{A}$ is

$$\mathbf{A}^H = \begin{bmatrix} a-bi & e+fi \\ c+di & g-hi \end{bmatrix} \tag{2.7}$$

This gives

$$\begin{aligned} (\mathbf{Ce})^H (\mathbf{Ce}) &= \bar{\lambda}\lambda \mathbf{e}^H \mathbf{e} \\ \mathbf{e}^H \mathbf{C}^\mathbf{T} \mathbf{Ce} &= \bar{\lambda}\lambda \mathbf{e}^H \mathbf{e} \\ \mathbf{e}^H \mathbf{e} &= \bar{\lambda}\lambda \mathbf{e}^H \mathbf{e} \end{aligned}$$

resulting in

$$\left(\bar{\lambda}\lambda - 1\right) \mathbf{e}^H \mathbf{e} = 0 \tag{2.8}$$

Since $\mathbf{C}$ is orthonormal, it will have 3 different eigenvalues and eigenvectors. Using this information and the fact that $\det \mathbf{C} = 1$, it can be deduced that one of the eigenvalues in Eq. (2.8) is 1, and the other two are complex conjugates of each other. This can be expressed according to [Hughes, 1986] as

$$\lambda_1 = 1 \tag{2.9a}$$

$$\lambda_{2,3} = \exp(\pm i\Phi) \tag{2.9b}$$

Using $\lambda_1 = 1$, Eq. (2.5) reduces to

$$\mathbf{Ce}_1 = \mathbf{e}_1 \tag{2.10}$$

This means that $\mathbf{e}_1$ remains invariant through the rotation expressed by $\mathbf{C}$ and thus, $\mathbf{e}_1$ is the Euler axis, $\hat{\mathbf{a}}$.

For the other two eigenvalues

$$\mathbf{Ce}_{2,3} = e^{\pm i\Phi} \mathbf{e}_{2,3} \tag{2.11}$$

This means that the vectors undergo a rotation of $+\Phi$ and $-\Phi$ about $\hat{\mathbf{a}}$. Since all the eigenvectors are mutually orthogonal, Φ is the angle of rotation and thus, the Euler angle.

To be able to use Euler's theorem, it is necessary to be able to extract the Euler axis and angle from a given rotation matrix. Using the fact that the trace of a matrix is equal to the sum of its eigenvalues, [Hughes, 1986] shows that the trace of a rotation matrix can be defined to be

$$\sigma = Tr\mathbf{C} = c_{1,1} + c_{2,2} + c_{3,3} = \lambda_1 + \lambda_2 + \lambda_3 = 1 + e^{i\Phi} + e^{-i\Phi} = 1 + 2\cos\Phi \tag{2.12}$$

The cosine of the Euler angle can thus be found from the trace to be

$$\cos\Phi = \frac{1}{2}(\sigma - 1) \tag{2.13}$$

There are two cases where special care has to be taken. These are

$\sigma = -1$: this is the case when $\Phi = \pm\pi, \pm 3\pi, ...$

$\sigma = 3$: this is the case when $\Phi = 0, \pm 2\pi, \pm 4\pi, ...$

When $\sigma \neq 3, -1$, the Euler axis is found to be

$$\mathbf{a} = \frac{1}{2\sin\Phi}\begin{bmatrix} c_{2,3} - c_{3,2} \\ c_{3,1} - c_{1,3} \\ c_{1,2} - c_{2,1} \end{bmatrix} \tag{2.14}$$

When $\sigma = 3$, either no rotation occurs or numerous complete rotations occur and so, the Euler axis cannot be computed. However, if $\sigma = -1$, the Euler axis can be calculated in the following way

$$\begin{aligned} a_1 &= \pm\sqrt{\frac{1 + c_{1,1}}{2}} \\ a_2 &= \pm\sqrt{\frac{1 + c_{2,2}}{2}} \\ a_3 &= \pm\sqrt{\frac{1 + c_{3,3}}{2}} \\ a_1 a_2 &= \frac{c_{1,2}}{2} \\ a_2 a_3 &= \frac{c_{2,3}}{2} \\ a_3 a_1 &= \frac{c_{3,1}}{2} \end{aligned} \tag{2.15}$$

The last three equations in Eq. (2.15) allow for solving the disambiguation of the signs.

The DCM can be expressed in terms of the Euler axis and angle, and can be seen in Eq. (2.16), where c and s stand for cos and sin, respectively.

$$\mathbf{C} = \begin{bmatrix} \mathrm{c}\Phi + a_1^2(1 - \mathrm{c}\Phi) & a_1 a_2(1 - \mathrm{c}\Phi) + a_3\mathrm{s}\Phi & a_1 a_3(1 - \mathrm{c}\Phi) - a_2\mathrm{s}\Phi \\ a_1 a_2(1 - \mathrm{c}\Phi) - a_3\mathrm{s}\Phi & \mathrm{c}\Phi + a_2^2(1 - \mathrm{c}\Phi) & a_2 a_3(1 - \mathrm{c}\Phi) + a_1\mathrm{s}\Phi \\ a_1 a_3(1 - \mathrm{c}\Phi) + a_2\mathrm{s}\Phi & a_2 a_3(1 - \mathrm{c}\Phi) - a_1\mathrm{s}\Phi & \mathrm{c}\Phi + a_3^2(1 - \mathrm{c}\Phi) \end{bmatrix} \tag{2.16}$$

and in a more compact form as

$$\mathbf{C} = \mathbf{I_3}\cos\Phi + (1 - \cos\Phi)\mathbf{a}\mathbf{a}^T - [\mathbf{a}\times]\sin\Phi \tag{2.17}$$

In Eq. (2.17), the term $[\mathbf{a}\times]$ is the skewsymmetric form of the Euler axis and is defined as

$$[\mathbf{a}\times] = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \tag{2.18}$$

2.1.3 Two Successive Rotations

A reference frame, $\mathcal{F}_a$ undergoes a rotation expressed by $\mathbf{C}_{ba}$ to form the frame $\mathcal{F}_b$ and then undergoes another rotation expressed by $\mathbf{C}_{cb}$ to form the frame $\mathcal{F}_c$. $\mathbf{C}_{ba}$ has the Euler axis $\mathbf{a}_1$ and the Euler angle Φ_1. $\mathbf{C}_{cb}$ has the Euler axis $\mathbf{a}_2$ and the Euler angle Φ_2. Using Eq. (2.17), the rotation matrix from $\mathcal{F}_a$ to $\mathcal{F}_c$ can be expressed as

$$\begin{aligned}\mathbf{C}_{c,a} = &\left[\mathbf{I_3}\cos\Phi_1 + (1-\cos\Phi_1)\mathbf{a}_1\mathbf{a}_1^T - [\mathbf{a}_1\times]\sin\Phi_1\right] \\ &\times\left[\mathbf{I_3}\cos\Phi_2 + (1-\cos\Phi_2)\mathbf{a}_2\mathbf{a}_2^T - [\mathbf{a}_2\times]\sin\Phi_2\right]\end{aligned} \tag{2.19}$$

It is also possible to extract the Euler axis and angle corresponding to $\mathbf{C}_{ca}$. For this, it is first necessary to define the cosine of the angle between the two Euler axes. This is

$$\cos\gamma = \mathbf{a}_1^T\mathbf{a}_2 \tag{2.20}$$

Expanding Eq. (2.19) and then computing the trace gives

$$\begin{aligned}Tr\mathbf{C}_{c,a} = &\cos\Phi_1 + \cos\Phi_2 + \cos\Phi_1\cos\Phi_2 \\ &+ (1-\cos\Phi_1)(1-\cos\Phi_2)\cos^2\gamma - 2\sin\Phi_1\sin\Phi_2\cos\gamma\end{aligned} \tag{2.21}$$

After using Eq. (2.13), the value of Φ_3 in terms of half-angles is

$$\cos\frac{\Phi_3}{2} = \cos\frac{\Phi_1}{2}\cos\frac{\Phi_2}{2} - \sin\frac{\Phi_1}{2}\sin\frac{\Phi_2}{2}\cos\gamma \tag{2.22}$$

Ultimately, the final Euler axis can be found to be

$$\begin{aligned}2\sin\Phi_3\mathbf{a}_3 = &\left[\sin\Phi_1(1+\cos\Phi_2) - \sin\Phi_2(1-\cos\Phi_1)\cos\gamma\right]\mathbf{a}_1 \\ &+ \left[\sin\Phi_2(1+\cos\Phi_1) - \sin\Phi_1(1-\cos\Phi_2)\cos\gamma\right]\mathbf{a}_2 \\ &+ \left[\sin\Phi_1\sin\Phi_2 - (1-\cos\Phi_1)(1-\cos\Phi_2)\cos\gamma\right][\mathbf{a}_1\times]\,\mathbf{a}_2\end{aligned} \tag{2.23}$$

and in terms of half-angles as

$$\mathbf{a}_3\sin\frac{\Phi_3}{2} = \left(\mathbf{a}_1\sin\frac{\Phi_1}{2}\cos\frac{\Phi_2}{2} + \mathbf{a}_2\cos\frac{\Phi_1}{2}\sin\frac{\Phi_2}{2} + [\mathbf{a}_1\times]\,\mathbf{a}_2\sin\frac{\Phi_1}{2}\sin\frac{\Phi_2}{2}\right) \tag{2.24}$$

It should be noted that Eqs. (2.23) and (2.24) only hold for the case when $\sin\Phi_3 \neq 0$. The Euler axis for the case when $\sin\Phi_3 = 0$ and Φ_3 is defined, so if $\Phi_3 = \pm\pi, \pm 3\pi, ...$ is

$$\begin{aligned}\sin^2\gamma\mathbf{a}_3 = &\left(\cos^2\frac{\Phi_1}{2} + \cos^2\gamma\sin^2\frac{\Phi_1}{2}\right)(\mathbf{a}_2 - \cos\gamma\mathbf{a}_1) \\ &+ \left(\cos^2\frac{\Phi_2}{2} + \cos^2\gamma\sin^2\frac{\Phi_2}{2}\right)(\mathbf{a}_1 - \cos\gamma\mathbf{a}_2) \\ &+ \left(1 - \cos^2\frac{\Phi_1}{2} - \cos^2\frac{\Phi_2}{2}\right)\sin\gamma\,[\mathbf{a}_2\times]\,\mathbf{a}_1\end{aligned} \tag{2.25}$$

Eq. (2.25) is only useful when $\sin\gamma \neq 0$. If $\sin\gamma = 0$, $\mathbf{a}_1$ and $\mathbf{a}_2$ are either parallel or antiparallel, which means that $\mathbf{a}_3 = \mathbf{a}_1$.

2.2 Non-Inertial Reference Frames

Not all reference frames are inertial. Sometimes it is very useful to have reference frames that are fixed to a body or a point that is accelerating, and still be able to express the motion in an inertial frame of reference. This section is included here because the motion expressed in a non-inertial frame does not translate into the same motion in an inertial frame. There will always be some extra terms that can be thought of as occurring due to the result of apparent forces.

2.2.1 Time Rate of Change of a Vector

A reference frame is made up of unit vectors. Thus, it is necessary to find out the time rate of change of a unit vector. Assume that there is a vector, $\mathbf{v}$, that lies in the direction of the unit vector $\hat{\mathbf{e}}$, has a magnitude v, undergoes a rotation, and the magnitude is not constant.

$$\mathbf{v} = v\hat{\mathbf{e}} \tag{2.26}$$

The time rate of change of the magnitude only causes v to change, and the rotation causes the orientation of $\mathbf{v}$ to change, expressed by a change in $\hat{\mathbf{e}}$. Mathematically, the time rate of change of $\mathbf{v}$ can be found by using the product rule to be

$$\dot{\mathbf{v}} = \dot{v}\hat{\mathbf{e}} + v\dot{\hat{\mathbf{e}}} \tag{2.27}$$

If the angular velocity vector, which is a measure of the rotation rate of the vector, is $\boldsymbol{\omega}$, the time rate of change of the unit vector is

$$\dot{\hat{\mathbf{e}}} = \boldsymbol{\omega} \times \hat{\mathbf{e}} \tag{2.28}$$

Thus, the time rate of change of $\mathbf{v}$ becomes

$$\dot{\mathbf{v}} = \dot{v}\hat{\mathbf{e}} + v(\boldsymbol{\omega} \times \hat{\mathbf{e}}) \tag{2.29}$$

2.2.2 Motion Relative to a Non-Inertial Reference Frame

Let there be an inertial frame of reference defined by the unit vectors, $\hat{\mathbf{I}}$, $\hat{\mathbf{J}}$, and $\hat{\mathbf{K}}$. Let there be a non-inertial reference frame $\mathcal{F}_O$ defined by the unit vectors, $\hat{\mathbf{i}}$, $\hat{\mathbf{j}}$, and $\hat{\mathbf{k}}$. This situation can be seen in Fig. 2.2. The absolute position of a point P, $\mathbf{r}_P$, is given by

$$\begin{aligned}\mathbf{r}_P &= \mathbf{r}_O + \mathbf{r}_{rel} \\ &= \mathbf{r}_O + x\hat{\mathbf{i}} + y\hat{\mathbf{j}} + z\hat{\mathbf{k}}\end{aligned} \tag{2.30}$$

where $\mathbf{r}_O$ is the position of the origin of $\mathcal{F}_O$ expressed in the inertial reference frame and $\mathbf{r}_{rel}$ is the position of P expressed in $\mathcal{F}_O$.

The position is differentiated with respect to time to give the velocity.

$$\begin{aligned}\mathbf{v}_P &= \mathbf{v}_O + \boldsymbol{\omega} \times \mathbf{r}_{rel} + \dot{x}\hat{\mathbf{i}} + \dot{y}\hat{\mathbf{j}} + \dot{z}\hat{\mathbf{k}} \\ &= \mathbf{v}_O + \boldsymbol{\omega} \times \mathbf{r}_{rel} + \mathbf{v}_{rel}\end{aligned} \tag{2.31}$$

The velocity is again differentiated with respect to time to give the acceleration.

$$\begin{aligned}\mathbf{a}_P &= \mathbf{a}_O + \boldsymbol{\alpha} \times \mathbf{r}_{rel} + \boldsymbol{\omega} \times (\boldsymbol{\omega} \times \mathbf{r}_{rel}) + 2\boldsymbol{\omega} \times \mathbf{v}_{rel} + \ddot{x}\hat{\mathbf{i}} + \ddot{y}\hat{\mathbf{j}} + \ddot{z}\hat{\mathbf{k}} \\ &= \mathbf{a}_O + \boldsymbol{\alpha} \times \mathbf{r}_{rel} + \boldsymbol{\omega} \times (\boldsymbol{\omega} \times \mathbf{r}_{rel}) + 2\boldsymbol{\omega} \times \mathbf{v}_{rel} + \mathbf{a}_{rel}\end{aligned} \tag{2.32}$$

In the above equations, the parameters can be split into absolute and relative quantities. The absolute quantities are

$\mathbf{a}_O$, the absolute acceleration of the moving origin.

$\mathbf{v}_O$, the absolute velocity of the moving origin.

$\boldsymbol{\alpha}$, the angular acceleration of the moving reference frame.

$\boldsymbol{\omega}$, the angular velocity of the moving reference frame.

The relative position, velocity, and acceleration are

$$\begin{aligned}\mathbf{r}_{rel} &= x\hat{\mathbf{i}} + y\hat{\mathbf{j}} + z\hat{\mathbf{k}} \\ \mathbf{v}_{rel} &= \dot{x}\hat{\mathbf{i}} + \dot{y}\hat{\mathbf{j}} + \dot{z}\hat{\mathbf{k}} \\ \mathbf{a}_{rel} &= \ddot{x}\hat{\mathbf{i}} + \ddot{y}\hat{\mathbf{j}} + \ddot{z}\hat{\mathbf{k}}\end{aligned} \tag{2.33}$$

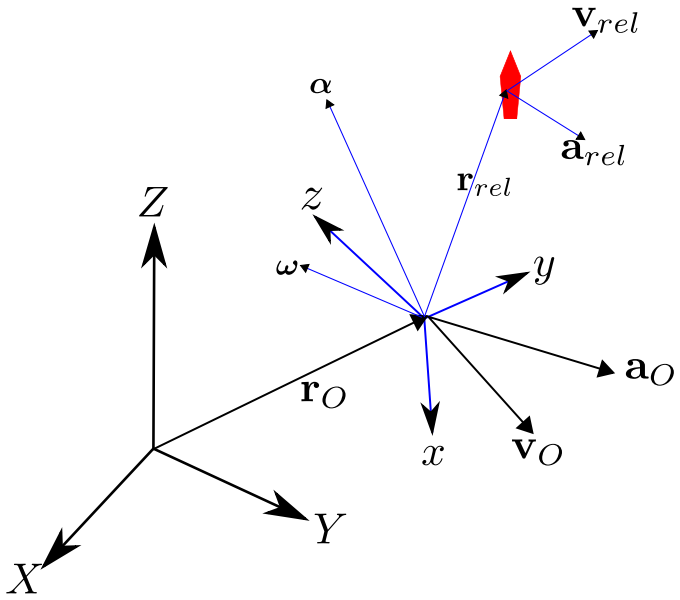


Figure 2.2: Motion in a non-inertial frame, with black vectors expressed in inertial frame, and blue vectors in non-inertial frame

2.2.3 Angular Velocity

The way the motion can be described in a non-inertial reference frame has been discussed in Section 2.2.2. It can be seen that the angular velocity plays a very important role in these equations. In this subsection, the link between the angular velocity and the rotation matrix, Euler axis, and Euler angle is presented. Only the equations are presented, because the derivation is beyond the scope of this report. For the derivations, the reader is referred to [Hughes, 1986]. It is again assumed that there are two reference frames, as found in Fig. 2.2. For ease of explanation, the inertial frame is called $\mathcal{F}_I$ and the non-inertial frame $\mathcal{F}_B$. The angular velocity $\boldsymbol{\omega}$ is the angular velocity of $\mathcal{F}_B$ with respect to $\mathcal{F}_I$ expressed in $\mathcal{F}_B$.

According to [Hughes, 1986], the relationship between the rotation matrix and the angular velocity can be found, to be

$$\dot{\mathbf{C}}_{B,I} + [\boldsymbol{\omega}\times]\,\mathbf{C}_{B,I} = \mathbf{0} \tag{2.34}$$

$$[\boldsymbol{\omega}\times] = -\dot{\mathbf{C}}_{B,I}\mathbf{C}_{B,I}^T = \mathbf{C}_{B,I}\dot{\mathbf{C}}_{B,I}^T \tag{2.35}$$

$$\boldsymbol{\omega} = \begin{bmatrix} \dot{c}_{2,1}c_{3,1} + \dot{c}_{2,2}c_{3,2} + \dot{c}_{2,3}c_{3,3} \\ \dot{c}_{3,1}c_{1,1} + \dot{c}_{3,2}c_{1,2} + \dot{c}_{3,3}c_{1,3} \\ \dot{c}_{1,1}c_{2,1} + \dot{c}_{1,2}c_{2,2} + \dot{c}_{1,3}c_{2,3} \end{bmatrix} \tag{2.36}$$

To find the relation between the angular velocity and the Euler axis and angle, Eqs. (2.17) and (2.35) have to be used. This gives the angular velocity to be

$$\boldsymbol{\omega} = \dot{\Phi}\hat{\mathbf{a}} - (1-\cos\Phi)\,[\hat{\mathbf{a}}\times]\,\dot{\hat{\mathbf{a}}} + \sin\Phi\dot{\hat{\mathbf{a}}} \tag{2.37}$$

At any instant in time, the rotation can be assumed to be about a fixed axis. Using this, the time derivative of the Euler angle becomes

$$\dot{\Phi} = \hat{\mathbf{a}}^T\boldsymbol{\omega} \tag{2.38}$$

The time derivative of the Euler axis is

$$\dot{\hat{\mathbf{a}}} = \frac{1}{2}\left[[\hat{\mathbf{a}}\times] - \cot\frac{\Phi}{2}\,[\hat{\mathbf{a}}\times]\,[\hat{\mathbf{a}}\times]\right]\boldsymbol{\omega} \tag{2.39}$$

2.2.4 Euler's Equation

For a rigid body, the angular acceleration is related to the angular velocity, external torque applied, and the inertia tensor by Euler's equation. Strictly speaking, this is referred to as attitude dynamics, but is still included in the kinematics chapter for the sake of completeness for the angular velocity.

$$\mathbf{J}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega} = \mathbf{M} \tag{2.40}$$

Eq. (2.40) is expressed completely in a reference frame fixed to the rigid body itself. $\mathbf{M}$ is the applied torque, and $\mathbf{J}$ is the inertia tensor

$$\mathbf{J} = \begin{bmatrix} J_{xx} & J_{xy} & J_{xz} \\ J_{xy} & J_{yy} & J_{yz} \\ J_{xz} & J_{yz} & J_{zz} \end{bmatrix} \tag{2.41}$$

The diagonals of $\mathbf{J}$ are known as the *moments of inertia*, and the off-diagonal terms are known as the *products of inertia*. The inertia tensor is a measure of the resistance of the rigid body to changes in its angular velocity. It is also possible to pick the body-fixed reference frame in such a way that the products of inertia become 0. The axes are then known as *principal axes*, and Eq. (2.40) simplifies to

$$J_{xx}\dot{\omega}_x + \omega_y\omega_z(J_{zz} - J_{yy}) = M_x \tag{2.42a}$$
$$J_{yy}\dot{\omega}_y + \omega_x\omega_z(J_{xx} - J_{zz}) = M_y \tag{2.42b}$$
$$J_{zz}\dot{\omega}_z + \omega_x\omega_y(J_{yy} - J_{xx}) = M_z \tag{2.42c}$$

The next chapter presents the techniques to express the orientation of one reference frame with respect to another. These techniques include quaternions, which are essential in understanding the USM.

Chapter 3

Attitude Parameters

The previous chapter defined reference frames and showed that there can be many different kinds of reference frames. Attitude is synonymous with orientation and in the aerospace world it almost always means the orientation of a spacecraft or an aircraft. The attitude of a spacecraft relative to a reference frame actually means the attitude of a reference frame fixed to the body of the spacecraft with respect to another reference frame. Thus, attitude is technically the orientation of one reference frame to another. Attitude parameters are just parameters that can describe the orientation and these will be discussed in this chapter. Thus, the rotation matrix, Euler axis, and Euler angle are all attitude parameters. However, they have been explained in Chapter 2, because they are fundamental in understanding the concepts of reference frames and rotations. There are 4 fundamental truths, taken from [Schaub and Junkins, 2002], about attitude coordinates.

1. *A minimum of three coordinates is required to describe the relative angular displacement between two reference frames $\mathcal{F}_1$ and $\mathcal{F}_2$.*

2. *Any minimal set of three attitude coordinates will contain at least one geometrical orientation where the coordinates are singular, namely at least two coordinates are undefined or not unique.*

3. *At or near such a geometric singularity, the corresponding kinematic differential equations are also singular.*

4. *The geometric singularities and associated numerical difficulties can be avoided altogether through a regularization. Redundant sets of four or more coordinates exist which are universally determined and contain no geometric singularities.*

The rotation matrix consists of nine elements, while the Euler axis and Euler angle together are four elements. Thus, it can be concluded that all rotations can be described using the rotation matrix or a combination of the Euler axis and Euler angle.

In this chapter, the following three important attitude parameters will be dealt with:

- Euler Angles
- Quaternions
- Modified Rodrigues Parameters (MRP)

Only an overview of the various attitude parameters is presented here so that the derivation of the USM can be understood. All the information in this section is based on [Schaub and Junkins, 2002], and the reader is referred to it for more information.

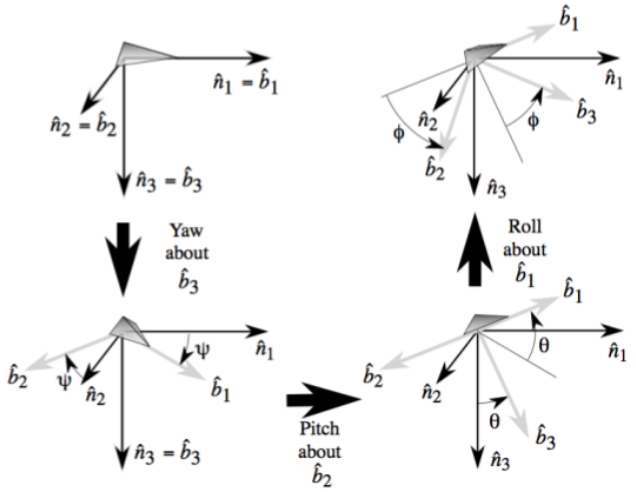


Figure 3.1: A (3-2-1) rotation sequence commonly used to describe the orientation of aerospace vehicles [Schaub and Junkins, 2002]

3.1 Euler Angles

Euler angles are a set of three angles, $(\theta_1, \theta_2, \theta_3)$, that can describe the rotation. It can be concluded, however, that since there are only three elements that describe the rotation, there are bound to be singularities in using this representation. Each of the euler angles represents a rotation about a principal axis of the rotating reference frame. What this means is that when a frame undergoes a rotation, there are actually three rotations. The first two rotations result in intermediate reference frames and the final rotation results in the final reference frame. It is important to label the sequence of the rotation, as different sequences will yield different results. For example, let us assume that there are two reference frames, $\mathcal{F}_a$ and $\mathcal{F}_b$, and the Euler angle set, $(\theta_1, \theta_2, \theta_3)$, describes the rotation from $\mathcal{F}_a$ to $\mathcal{F}_b$. The sequence of rotation describes the principal axis used for the rotation. In the example the sequence of rotation is (3-2-1), which is a common rotation sequence to describe the attitude of aerospace vehicles. When the (3-2-1) sequence is used, the Euler angles are called yaw, pitch, and roll, respectively. The Euler angles are then written as (ψ, θ, ϕ) and the rotation can be seen in Fig. 3.1.

The rotation is as follows:

- Euler rotation with an Euler angle of ψ (θ_1) and Euler axis $\hat{\mathbf{a}}_3$ to yield $\mathcal{F}'_a$
- Euler rotation with an Euler angle of θ (θ_2) and Euler axis $\hat{\mathbf{a}}'_2$ to yield $\mathcal{F}''_a$
- Euler rotation with an Euler angle of ϕ (θ_3) and Euler axis $\hat{\mathbf{a}}''_1$ to yield $\mathcal{F}_b$

For orbital mechanics, a (3-1-3) rotation sequence results in the Euler angles being some of the Keplerian elements that describe an orbit. The Keplerian elements are described later in Chapter 4. The rotation and the Euler angles can be seen in Fig. 3.2. The Euler angles are then written as (Ω, i, ω) and are the right ascension of ascending node Ω, inclination i, and the argument of pericenter ω. The Euler angles of the (3-2-1) sequence, which are usually used to describe the attitude of a body, are an asymmetric set because each rotation is about a separate axis. The Euler angles of the (3-1-3) sequence, which are used to describe orbits, form a symmetric set because 2 of the rotations are about the same axis.

There is a specific rotation matrix for a rotation about each axis. These rotation matrices are known as the principal rotation matrices. Let $\mathbf{C}_i(\theta)$ be an Euler rotation about axis i with angle θ. This is also referred to as an unit-axis rotation.

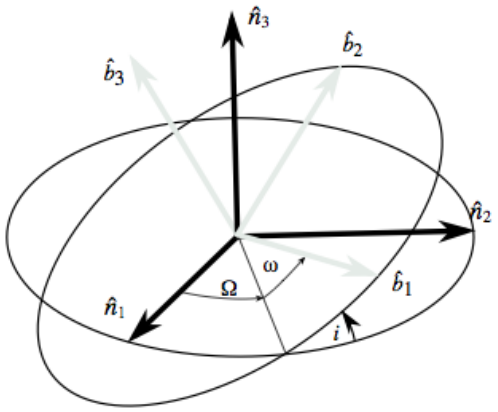


Figure 3.2: Euler Angles to Describe Orbits [Schaub and Junkins, 2002]

$$\mathbf{C}_1(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) \\ 0 & -\sin(\theta) & \cos(\theta) \end{bmatrix} \tag{3.1}$$

$$\mathbf{C}_2(\theta) = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{3.2}$$

$$\mathbf{C}_3(\theta) = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

The rotation matrix of a rotation sequence can then be derived by multiplying the principal rotation matrices in the specified order. In the following equations, the angles are shown next to the rotation matrix to give more insight into the rotations. This notation will not be used afterwards. The (3-2-1) rotation sequence, shown in Fig. 3.1, would yield

$$\mathbf{C}(\psi, \theta, \phi) = \mathbf{C}_1(\phi)\mathbf{C}_2(\theta)\mathbf{C}_3(\psi) \tag{3.4}$$

and the (3-1-3) rotation sequence would yield

$$\mathbf{C}(\Omega, i, \omega) = \mathbf{C}_3(\omega)\mathbf{C}_1(i)\mathbf{C}_3(\Omega) \tag{3.5}$$

Eq. 3.5 can be expanded, according to [Schaub and Junkins, 2002], to give

$$\mathbf{C}(\Omega, i, \omega) = \begin{bmatrix} \mathrm{c}\omega\mathrm{c}\Omega - \mathrm{s}\omega ci\mathrm{s}\Omega & \mathrm{c}\omega\mathrm{s}\Omega + \mathrm{s}\omega ci\mathrm{c}\Omega & \mathrm{s}\omega si \\ -\mathrm{s}\omega\mathrm{c}\Omega - \mathrm{c}\omega ci\mathrm{s}\Omega & -\mathrm{s}\omega\mathrm{s}\Omega + \mathrm{c}\omega ci\mathrm{c}\Omega & \mathrm{c}\omega si \\ si\mathrm{s}\Omega & -si\mathrm{c}\Omega & ci \end{bmatrix} \tag{3.6}$$

If the rotation matrix is given, the Euler angles of the (3-1-3) rotation sequence can be extracted in the following manner

$$\Omega = \arctan\left(-\frac{c_{3,1}}{c_{3,2}}\right) \tag{3.7}$$

$$i = \arccos\left(c_{3,3}\right) \tag{3.8}$$

$$\omega = \arctan\left(\frac{c_{1,3}}{c_{2,3}}\right) \tag{3.9}$$

Since Euler angles use only three elements to express a rotation, there is always an inherent singularity where 2 of the angles cannot be extracted uniquely from the rotation matrix. For the Keplerian angles, this singularity occurs when $i = 0°, 180°$. This is the case for equatorial orbits when $c_{3,1} = c_{3,2} = c_{1,3} = c_{2,3} = 0$. From astrodynamics, it is known that Ω is not defined for equatorial orbits and thus, it can be set to 0. When this is done, ω can be found by

$$\omega = \arctan\left(-\frac{c_{2,1}}{c_{1,1}}\right) \tag{3.10}$$

For true orbits, ω is undefined when the eccentricity of the orbit is 0. This is due to the dynamics and not a singularity in the Euler angles and will therefore not be discussed further.

The kinematic relations for different sets of Euler angles will be different based on their rotation sequence. The kinematic differential equations of the (3-1-3) Euler angles will be presented here. The angular velocity of the rotating frame with respect to the original frame, according to [Schaub and Junkins, 2002], is

$$\boldsymbol{\omega} = \begin{bmatrix} \sin\omega\sin i & \cos\omega & 0 \\ \cos\omega\sin i & -\sin\omega & 0 \\ \cos i & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\Omega} \\ \dot{i} \\ \dot{\omega} \end{bmatrix} \tag{3.11}$$

and the inverse relationship is found to be

$$\begin{bmatrix} \dot{\Omega} \\ \dot{i} \\ \dot{\omega} \end{bmatrix} = \frac{1}{\sin i} \begin{bmatrix} \sin\omega & \cos\omega & 0 \\ \cos\omega\sin i & -\sin\omega\sin i & 0 \\ -\sin\omega\cos i & -\cos\omega\cos i & \sin i \end{bmatrix} \boldsymbol{\omega} \tag{3.12}$$

As mentioned previously, all sets of Euler angles have singularities. This singularity always occurs at certain values of θ_2. In the case of the (3-1-3) rotation, the singularity occurs at $i = 0$ and $i = 180$ degrees. For the (3-2-1) rotation, the singularity occurs at $\theta = \pm 90°$. This singularity is known as *gimbal lock* and occurs because two of the three axes line up. An example is given with the $x - y - z$ notation and the (3-2-1) sequence. The first rotation is carried out by an arbitrary angle around the z-axis. The second rotation by $\pm 90°$ around the intermediate y-axis will cause the intermediate x-axis and z-axis to be parallel to each other. The third rotation around the intermediate z-axis can now be recreated by a rotation around the intermediate x-axis, and so there is a loss of a degree of freedom.

The Euler angles are faster to integrate than the full rotation matrix, but there are still a many trigonometric functions involved, which are tedious for the computer. Also, the behavior close to the singularities will be undesirable. Therefore in the next section, quaternions, which do not possess these disadvantages, are presented.

3.2 Quaternions

A quaternion is a four dimensional hyper-complex number, which was discovered by Sir William Rowan Hamilton. A quaternion consists of one real number and three imaginary numbers. The imaginary numbers are different square roots of -1 and obey the following constraint

$$i^2 = j^2 = k^2 = ijk = -1 \tag{3.13}$$

Quaternions are a very vast topic, with their own special algebra and properties making an exhaustive treatment beyond the scope of this report. Quaternions having unit magnitude can be used to describe rotations and when considered in this way, instead of being pure mathematical concepts, they are also called *Euler Parameters*.

The quaternion can be defined in terms of the Euler axis, $\mathbf{a}$ and the Euler angle, Φ. The vector part is defined as

$$\boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \end{bmatrix} = \mathbf{a} \sin \frac{\Phi}{2} \tag{3.14}$$

and the scalar part as

$$\eta = \cos \frac{\Phi}{2} \tag{3.15}$$

The rotation specified by the quaternion is expressed as $(\boldsymbol{\epsilon}, \eta)$. The 4 quaternion elements are not mutually independent because they satisfy the following constraint.

$$\boldsymbol{\epsilon}^T \boldsymbol{\epsilon} + \eta^2 = \epsilon_1^2 + \epsilon_2^2 + \epsilon_3^2 + \eta^2 = 1 \tag{3.16}$$

This shows that the quaternion describes a unit 3-sphere, meaning a sphere in 4 dimensional space. Any rotation is therefore a trajectory on the surface of the 3-sphere. Also, the orientation specified by $(\boldsymbol{\epsilon}, \eta)$ is the same as the orientation specified by $(-\boldsymbol{\epsilon}, -\eta)$. This is because the second quaternion describes the Euler rotation with Euler axis $-\mathbf{a}$ and Euler angle $-\Phi$. This means that if $(\boldsymbol{\epsilon}, \eta)$ describes the shortest rotation then, $(-\boldsymbol{\epsilon}, -\eta)$ describes the longest rotation. To ensure that the shortest rotation is taken, a necessary condition is that $\eta > 0$. To visualize this, quaternions can be viewed in another manner. The norm of the quaternion is 1, so the 4 quaternion elements represent the radial vector of a point on a 4-dimensional unit sphere. Since it is very difficult to imagine a 4-dimensional sphere, we can consider the vector part $\boldsymbol{\epsilon}$ to express the radial vector of a point on a 3-dimensional sphere. The radius of this 3-dimensional sphere, the norm of $\boldsymbol{\epsilon}$, can be found using the unit norm constraint of the quaternion in the following manner:

$$|\boldsymbol{\epsilon}| = \sqrt{1 - \eta^2} \tag{3.17}$$

Thus, the sphere expressing $\boldsymbol{\epsilon}$ changes with values of η. This will be a unit 3-dimensional sphere if $\eta = 0$, and only a point if $\eta = \pm 1$. Since η is in the fourth dimension, the 3-dimensional spheres of $\boldsymbol{\epsilon}$ corresponding to η can be thought to be present at various times like an animation. This can also be shown as a sphere with a linear offset from a point. Each offset corresponds to a certain value of η and all the points on the surface of the sphere express the possible values of $\boldsymbol{\epsilon}$. This can be seen in Fig. 3.3 along with a specific orientation.

3.2.1 Relationship with the Direction Cosine Matrix

The DCM $\mathbf{C}$, in terms of the quaternion can be found by substituting Eqs.(3.14) and (3.15) into Eq.(2.16).

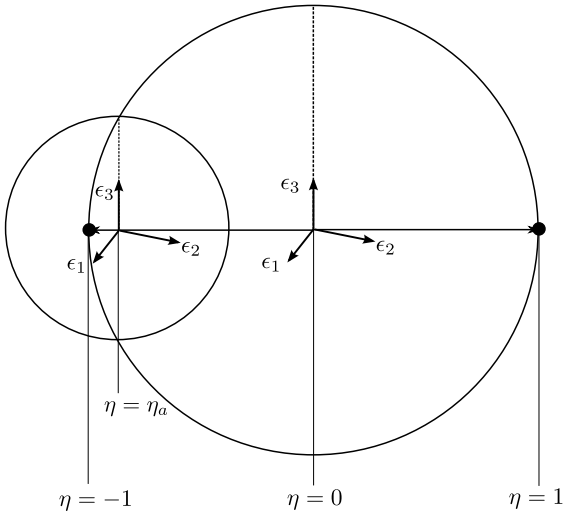
$$\mathbf{C} = \begin{bmatrix} 1 - 2\left(\epsilon_2^2 + \epsilon_3^2\right) & 2\left(\epsilon_1\epsilon_2 + \epsilon_3\eta\right) & 2\left(\epsilon_1\epsilon_3 - \epsilon_2\eta\right) \\ 2\left(\epsilon_2\epsilon_1 - \epsilon_3\eta\right) & 1 - 2\left(\epsilon_3^2 + \epsilon_1^2\right) & 2\left(\epsilon_2\epsilon_3 + \epsilon_1\eta\right) \\ 2\left(\epsilon_3\epsilon_1 + \epsilon_2\eta\right) & 2\left(\epsilon_3\epsilon_2 - \epsilon_1\eta\right) & 1 - 2\left(\epsilon_1^2 + \epsilon_2^2\right) \end{bmatrix} \tag{3.18}$$

$$\mathbf{C} = \left(\eta^2 - \boldsymbol{\epsilon}^T\boldsymbol{\epsilon}\right)\mathbf{I}_3 + 2\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T - 2\eta\left[\boldsymbol{\epsilon}\times\right] \tag{3.19}$$

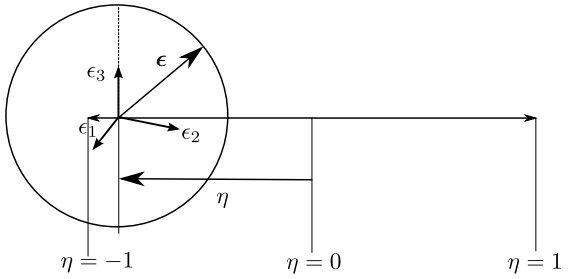
Inversely, there are two methods to extract a quaternion from a DCM. The first method is very compact, but has a singularity at $\eta = 0$. In this method, the unit magnitude property of a quaternion is used to find the η from the DCM found in Eq. (3.18).

$$\eta = \pm\frac{1}{2}\sqrt{Tr\mathbf{C} + 1} \tag{3.20}$$

$$\begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \end{bmatrix} = \frac{1}{4\eta}\begin{bmatrix} c_{2,3} - c_{3,2} \\ c_{3,1} - c_{1,3} \\ c_{1,2} - c_{2,1} \end{bmatrix} \tag{3.21}$$



(a) 3 dimensional spheres expressing the constraint for $\boldsymbol{\epsilon}$ for various values of η



(b) Expression of a specific orientation ($\boldsymbol{\epsilon}$,η)

Figure 3.3: Visualizing quaternions

However, is a long workaround for the case when $\eta = 0$. After substituting $\eta = 0$ into the DCM in Eq. (3.18), the various matrix elements are then functions of only ϵ_1, ϵ_2, and ϵ_3. This results in nine equations for 3 unknowns and thus, the remaining elements of the quaternion can be found.

In Eq. (3.20), the sign can be chosen arbitrarily. This is because the orientation expressed by the quaternion $(\boldsymbol{\epsilon}, \eta)$ is the same as the orientation expressed by $(-\boldsymbol{\epsilon}, -\eta)$. As seen in Eq. (3.21), the signs of the elements of $\boldsymbol{\epsilon}$ change according to the sign of η. The appropriate sign should be chosen from the quaternion history. If the present step is considered to be k, the quaternion element with the largest magnitude in step $k-1$ should have the same sign in step k. For there to have been a smooth switch of sign for all the quaternion elements in one time step, the time step has to be extremely large because of the unit-magnitude constraint. If 3 of the quaternion elements pass through 0, one element still has to be approximately 1 or -1. Thus, for reasonably sized time steps and smooth uni-directional motion, only a maximum of 3 quaternion elements may change sign simultaneously.

The second method, without singularities, can be found in [Stanley, 1978]. In this method, the squares of the vector and scalar parts of the quaternion should first be computed using the following equations:

$$4\epsilon_i^2 = 1 - Tr\mathbf{C} + 2c_{i,i} \tag{3.22a}$$

$$4\eta^2 = 1 - Tr\mathbf{C} + 2Tr\mathbf{C} \tag{3.22b}$$

Starting from the quaternion element with the largest square in Eq. (3.22), the following relations

can be used to compute the remaining quaternion elements:

$$4\epsilon_i\epsilon_j = c_{i,j} + c_{j,i} \tag{3.23a}$$
$$4\epsilon_i\eta = c_{j,k} - c_{k,j} \tag{3.23b}$$

In Eqs. (3.22) and (3.23), the subscripts (i, j, k) are cyclic permutations of $(1, 2, 3)$. By default, the positive square root can be taken and then, the elements can get the appropriate sign based on the quaternion history.

Euler angles and Quaternions Quaternions are normally used for calculating the attitude of an aerospace vehicle. It is, however, easier to visualize the attitude using the (3-2-1) sequence of Euler angles. According to [Schaub and Junkins, 2002], the conversion between the Euler angles and quaternion is

$$\begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \eta \end{bmatrix} = \begin{bmatrix} \sin\frac{\phi}{2}\cos\frac{\theta}{2}\cos\frac{\psi}{2} - \cos\frac{\phi}{2}\sin\frac{\theta}{2}\sin\frac{\psi}{2} \\ \cos\frac{\phi}{2}\sin\frac{\theta}{2}\cos\frac{\psi}{2} + \sin\frac{\phi}{2}\cos\frac{\theta}{2}\sin\frac{\psi}{2} \\ \cos\frac{\phi}{2}\cos\frac{\theta}{2}\sin\frac{\psi}{2} - \sin\frac{\phi}{2}\sin\frac{\theta}{2}\cos\frac{\psi}{2} \\ \cos\frac{\phi}{2}\cos\frac{\theta}{2}\cos\frac{\psi}{2} - \sin\frac{\phi}{2}\sin\frac{\theta}{2}\sin\frac{\psi}{2} \end{bmatrix} \tag{3.24}$$

According to [Schaub and Junkins, 2002], the conversion from the quaternion to Euler angles is

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \arctan\frac{2(\epsilon_1\eta+\epsilon_2\epsilon_3)}{1-2\left(\epsilon_1^2+\epsilon_2^2\right)} \\ \arcsin\left(2\left(\epsilon_2\eta - \epsilon_1\epsilon_3\right)\right) \\ \arctan\frac{2(\epsilon_3\eta+\epsilon_1\epsilon_2)}{1-2\left(\epsilon_2^2+\epsilon_3^2\right)} \end{bmatrix} \tag{3.25}$$

Even though quaternions are singularity free, the conversion to Euler angles in Eq. (3.25) cannot be carried out if $\sin\theta = \pm 1$, which means that $\epsilon_2\eta - \epsilon_3\epsilon_1 = \pm 0.5$, because of the gimbal lock situation. The would occur, for example, when $(\epsilon_1, \epsilon_2, \epsilon_3, \eta) = (0, \frac{\sqrt{2}}{2}, 0, \frac{\sqrt{2}}{2})$. To still be able to extract some value for the Euler angles, one of the other two Euler angles, ϕ and ψ, should be set to 0. The remaining angle is computed using from $\arctan(\epsilon_1/\eta)$[1]. If one checks the history, the evolution of the quaternion elements will be smooth at this point, but the Euler angles will have a discontinuity.

3.2.2 Successive Rotations

If there are 2 successive rotations, $(\boldsymbol{\epsilon}', \eta')$ followed by $(\boldsymbol{\epsilon}'', \eta'')$, the full rotation, $(\boldsymbol{\epsilon}, \eta)$, can be expressed as

$$\boldsymbol{\epsilon} = \eta''\boldsymbol{\epsilon}' + \eta'\boldsymbol{\epsilon}'' + [\boldsymbol{\epsilon}'\times]\,\boldsymbol{\epsilon}'' \tag{3.26}$$
$$\eta = \eta'\eta'' - \boldsymbol{\epsilon}'^T\boldsymbol{\epsilon}'' \tag{3.27}$$

This can be written as

$$\begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \eta \end{bmatrix} = \begin{bmatrix} \eta'' & \epsilon_3'' & -\epsilon_2'' & \epsilon_1'' \\ -\epsilon_3'' & \eta'' & \epsilon_1'' & \epsilon_2'' \\ \epsilon_2'' & -\epsilon_1'' & \eta'' & \epsilon_3'' \\ -\epsilon_1'' & -\epsilon_2'' & -\epsilon_3'' & \eta'' \end{bmatrix} \begin{bmatrix} \epsilon_1' \\ \epsilon_2' \\ \epsilon_3' \\ \eta' \end{bmatrix} \tag{3.28}$$

or alternatively as

$$\begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \eta \end{bmatrix} = \begin{bmatrix} \eta' & -\epsilon_3' & \epsilon_2' & \epsilon_1' \\ \epsilon_3' & \eta' & -\epsilon_1' & \epsilon_2' \\ -\epsilon_2' & \epsilon_1' & \eta' & \epsilon_3' \\ -\epsilon_1' & -\epsilon_2' & -\epsilon_3' & \eta' \end{bmatrix} \begin{bmatrix} \epsilon_1'' \\ \epsilon_2'' \\ \epsilon_3'' \\ \eta'' \end{bmatrix} \tag{3.29}$$

[1] http://www.euclideanspace.com/maths/geometry/rotations/conversions/quaternionToEuler/Quaternions.pdf

3.2.3 Kinematic Differential Equation

To find the kinematic differential equation of the quaternion, it is necessary to first differentiate the rotation matrix found in Eq. (3.18) with respect to time:

$$\dot{\mathbf{C}} = \begin{bmatrix} -4(\epsilon_2\dot{\epsilon}_2 + \epsilon_3\dot{\epsilon}_3) & 2(\dot{\epsilon}_1\epsilon_2 + \epsilon_1\dot{\epsilon}_2 + \dot{\epsilon}_3\eta + \epsilon_3\dot{\eta}) & \\ 2(\dot{\epsilon}_2\epsilon_1 + \epsilon_2\dot{\epsilon}_1 - \dot{\epsilon}_3\eta - \epsilon_3\dot{\eta}) & -4(\epsilon_3\dot{\epsilon}_3 + \epsilon_1\dot{\epsilon}_1) & \cdots \\ 2(\dot{\epsilon}_3\epsilon_1 + \epsilon_3\dot{\epsilon}_1 + \dot{\epsilon}_2\eta + \epsilon_2\dot{\eta}) & 2(\dot{\epsilon}_3\epsilon_2 + \epsilon_3\dot{\epsilon}_2 - \dot{\epsilon}_1\eta - \epsilon_1\dot{\eta}) & \\ & & \begin{matrix} 2(\dot{\epsilon}_1\epsilon_3 + \epsilon_1\dot{\epsilon}_3 - \dot{\epsilon}_2\eta - \epsilon_2\dot{\eta}) \\ \cdots \quad 2(\dot{\epsilon}_2\epsilon_3 + \epsilon_2\dot{\epsilon}_3 + \dot{\epsilon}_1\eta + \epsilon_1\dot{\eta}) \\ -4(\epsilon_1\dot{\epsilon}_1 + \epsilon_2\dot{\epsilon}_2) \end{matrix} \end{bmatrix} \tag{3.30}$$

This leads to a relation between the angular velocity, the quaternion, and the time derivative of the quaternion. To get this relation, elements of the DCM in terms of the quaternion elements from Eq. (3.18), and the elements of the time derivative of the DCM in terms of quaternion elements and their time derivatives from Eq. (3.30) have to be filled into Eq. (2.36). The resultant relation for the angular velocity vector, after some simplification, is

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} = 2 \begin{bmatrix} \dot{\epsilon}_1\eta + \dot{\epsilon}_2\epsilon_3 - \dot{\epsilon}_3\epsilon_2 - \dot{\eta}\epsilon_1 \\ \dot{\epsilon}_2\eta + \dot{\epsilon}_3\epsilon_1 - \dot{\epsilon}_1\epsilon_3 - \dot{\eta}\epsilon_2 \\ \dot{\epsilon}_3\eta + \dot{\epsilon}_1\epsilon_2 - \dot{\epsilon}_2\epsilon_1 - \dot{\eta}\epsilon_3 \end{bmatrix} \tag{3.31}$$

There are 4 quaternion elements and so a fourth equation is necessary. This can be found by differentiating the unit constraint of the quaternion, found in Eq. (3.17), with respect to time:

$$2(\epsilon_1\dot{\epsilon}_1 + \epsilon_2\dot{\epsilon}_2 + \epsilon_3\dot{\epsilon}_3 + \eta\dot{\eta}) = 0 \tag{3.32}$$

Using Eqs. (3.32) and (3.31) , the kinematic differential equation can now be written canonically as

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ 0 \end{bmatrix} = 2 \begin{bmatrix} \eta & \epsilon_3 & -\epsilon_2 & -\epsilon_1 \\ -\epsilon_3 & \eta & \epsilon_1 & -\epsilon_2 \\ \epsilon_2 & -\epsilon_1 & \eta & -\epsilon_3 \\ \epsilon_1 & \epsilon_2 & \epsilon_3 & \eta \end{bmatrix} \begin{bmatrix} \dot{\epsilon}_1 \\ \dot{\epsilon}_2 \\ \dot{\epsilon}_3 \\ \dot{\eta} \end{bmatrix} \tag{3.33}$$

or, after rearranging, as

$$\begin{bmatrix} \dot{\epsilon}_1 \\ \dot{\epsilon}_2 \\ \dot{\epsilon}_3 \\ \dot{\eta} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & \omega_3 & -\omega_2 & \omega_1 \\ -\omega_3 & 0 & \omega_1 & \omega_2 \\ \omega_2 & -\omega_1 & 0 & \omega_3 \\ -\omega_1 & -\omega_2 & -\omega_3 & 0 \end{bmatrix} \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \eta \end{bmatrix} \tag{3.34}$$

This can be written in a more compact form as

$$\begin{bmatrix} \dot{\boldsymbol{\epsilon}} \\ \dot{\eta} \end{bmatrix} = \frac{1}{2}\boldsymbol{\Omega} \begin{bmatrix} \boldsymbol{\epsilon} \\ \eta \end{bmatrix} \tag{3.35}$$

with $\boldsymbol{\Omega}$ being defined as

$$\boldsymbol{\Omega} = \begin{bmatrix} -[\boldsymbol{\omega}\times] & \boldsymbol{\omega} \\ \boldsymbol{\omega}^T & 0 \end{bmatrix} \tag{3.36}$$

The kinematic differential equation for the quaternion is linear and has no transcendental functions. Thus, the computational load is lower than the case with Euler angles.

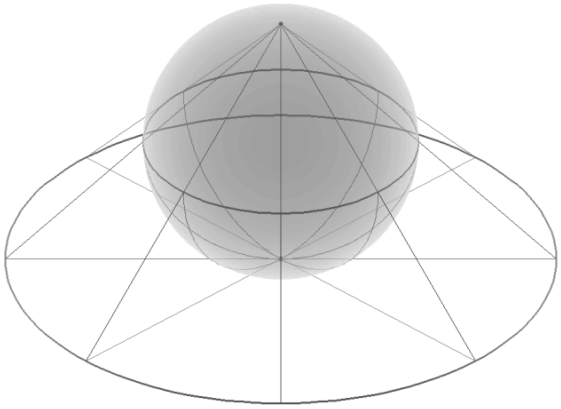


Figure 3.4: Stereographic Projection of a 2-Sphere [Howison, 2007]

3.3 Modified Rodrigues Parameters

Modified Rodrigues Parameters (MRPs) are a modification of the classical Rodrigues parameters, as can be deduced from the name. A very extensive treatment of both Classical, and Modified Rodrigues parameters can be found in [Schaub and Junkins, 2002]. Classical Rodrigues parameters will not be dealt with in this section, as the MRP are an improvement and they will be used in further chapters. An MRP, $\boldsymbol{\sigma}$, vector is made up of three MRPs and can be defined in terms of the Euler axis and angle as

$$\boldsymbol{\sigma} = \hat{\mathbf{a}} \tan(\Phi/4) \tag{3.37}$$

As an MRP vector is of dimension 3, there will be a singularity present. From Eq. (3.37) it can be seen that this singularity occurs at $\Phi = \pm 360°$. There are ways to get around this singularity by switching to the so-called Shadow Modified Rodrigues Parameters (SMRPs), which are explained further on in this section.

3.3.1 Relationship between Modified Rodrigues Parameters and Quaternions

MRPs are very closely linked to quaternions. They are a stereographic projection of the quaternion unit 3-sphere. Stereographic projection is a way of describing the points of an n-sphere onto a tangent n-dimensional surface using a mapping point. An example using a 2-sphere can be seen in Fig. 3.4. If the sphere in Fig. 3.4 is considered to be the Earth, then the mapping point in this case would be the north pole and the tangent plane would be tangent to the south pole. The mapping is carried out as if there is a point light source at the mapping point emanating rays. The projection of any point on the sphere would be the intersection of the ray, from the mapping point to the point of interest, and the tangent plane. This means that any point on the surface of the sphere, except the mapping point, can be projected onto the tangent plane. This same concept can be applied to the unit 3-sphere of the quaternion.

MRP are very closely related to quaternions and are extracted from a DCM by first extracting the quaternion and then converting the quaternion to MRP. For the MRP, the projection point for the stereographic projection is $\begin{bmatrix} \boldsymbol{\epsilon}^T, & \eta \end{bmatrix} = \begin{bmatrix} 0, & 0, & 0, & -1 \end{bmatrix}$. The tangent hyperplane is at $\eta = 0$ and normal to the η-axis. All the rotations can be expressed with a 3-dimensional vector, except for a rotation with $\eta = -1$. This occurs, as stated previously, at $\Phi = \pm 360°$. Since the four quaternion elements lie in 4 different dimensions, a cross section can be taken of any two dimensions to view them in 2 dimensions as they are all orthogonal. To view the stereographic projection of any one element of $\boldsymbol{\epsilon}$, ϵ_i, the $\epsilon_i - \eta$ plane can be viewed as seen in Fig. 3.5. In this

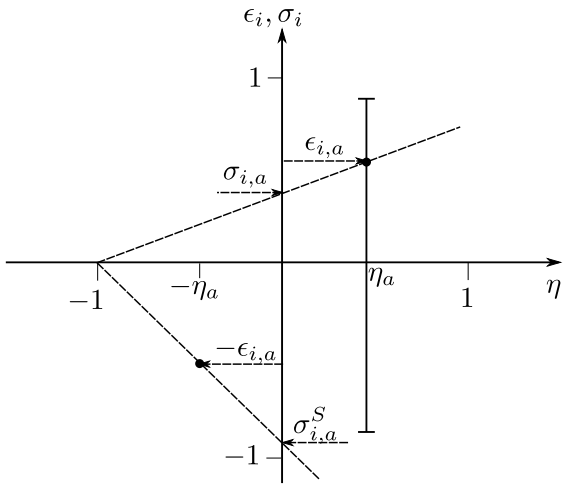


Figure 3.5: Stereographic projection of an arbitrary ϵ_i

2-dimensional plane, the hyperplane at $\eta = 0$ is the ϵ_i-axis. The set of all possible ϵ_i is a line of length equal to $|\boldsymbol{\epsilon}|$ found in Eq. (3.17). The stereographic projection of ϵ_i, which is an element of the quaternion ($\boldsymbol{\epsilon}_a$, η_a) can be seen in Fig. 3.5.

The MRP vector, $\boldsymbol{\sigma}$, can be extracted from a quaternion with the following expression:

$$\boldsymbol{\sigma} = \frac{\boldsymbol{\epsilon}}{1+\eta}, \forall \eta \neq -1 \tag{3.38}$$

The magnitude of the MRP vector can be defined as

$$\sigma = \sqrt{\boldsymbol{\sigma}^T \boldsymbol{\sigma}} \tag{3.39}$$

The MRP vector can then be converted back to a quaternion using

$$\boldsymbol{\epsilon} = \frac{2}{1+\sigma^2}\boldsymbol{\sigma} \tag{3.40a}$$

$$\eta = \frac{1-\sigma^2}{1+\sigma^2} \tag{3.40b}$$

MRP are always extracted from a quaternion, because of their close relation. Thus, to convert to any other attitude parameters, the conversion to quaternion has to be carried out first.

3.3.2 Shadow Modified Rodrigues Parameters

To avoid the singularity ($\eta = -1$ $\Phi = \pm 360°$), an alternative set of parameters known as the Shadow Modified Rodrigues Parameters (SMRP), $\boldsymbol{\sigma}^S$, is defined. The property of quaternions that the orientation expressed by $(\boldsymbol{\epsilon}, \eta)$ is the same as the orientation expressed by $(-\boldsymbol{\epsilon}, -\eta)$ is used. Thus, the SMRP vector is a stereographic projection of $(-\boldsymbol{\epsilon}, -\eta)$, which can be seen in Fig. 3.5. These SMRP can be extracted from Euler parameters using

$$\boldsymbol{\sigma}^S = \frac{-\boldsymbol{\epsilon}}{1-\eta}, \forall \eta \neq 1 \tag{3.41}$$

The MRP can be converted to SMRP by

$$\boldsymbol{\sigma}^S = \frac{-\boldsymbol{\sigma}}{\sigma^2} \tag{3.42}$$

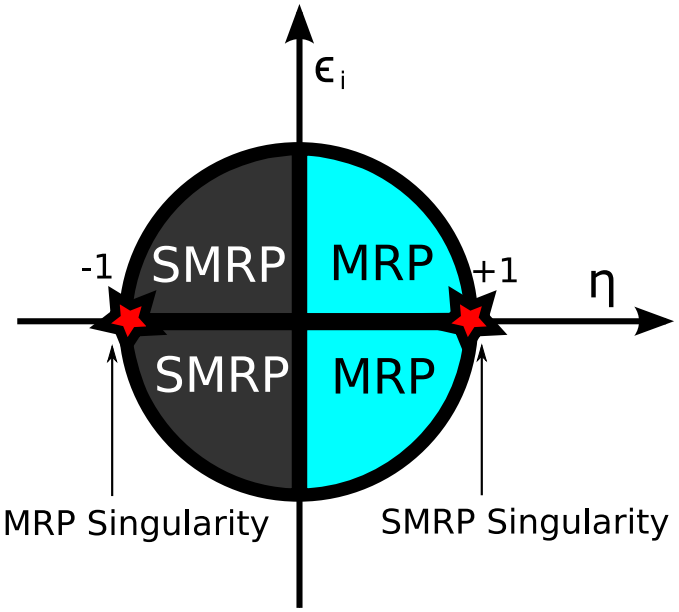


Figure 3.6: Range of Euler parameters where MRP and SMRP are used

The SMRP can extracted from the Euler axis and angle by

$$\boldsymbol{\sigma}^S = \mathbf{a} \tan\left(\frac{\Phi - 2\pi}{4}\right) \tag{3.43}$$

Since the SMRP uses $(-\boldsymbol{\epsilon}, -\eta)$, there will be no singularity occurring at $\eta = -1$. The singularity will instead occur at $\eta = 1$. Since it is possible to switch between MRP and SMRP using Eq. (3.42), the singularity can always be avoided. An important property is that the kinematic differential relations and other equations are the same for MRP and SMRP. Thus, it is only necessary to keep track of which set of parameters is being used and not use different sets of kinematic equations during computations.

The switching between MRP and SMRP can be carried out arbitrarily. However, it can be seen from Fig. 3.4 that $\boldsymbol{\sigma} \leq 1$ when $\eta \geq 0$ and that $\boldsymbol{\sigma}^S \leq 1$ when $\eta \leq 0$. This means that it is best to switch when $\eta = 0$, which also simplifies Eq. (3.42) to

$$\boldsymbol{\sigma}^S = -\boldsymbol{\sigma} \tag{3.44}$$

It is not always possible to know exactly when $\eta = 0$. Thus, it is best to switch as soon as the norm of the current MRP or SMRP is greater than 1. It should be noted, however, that the MRP and SMRP do not have the unit magnitude constraint. This ensures that the numbers are always finite and also that the singularity is always approximately 180° away in terms of the Euler angle. The range where MRP and where SMRP are used can be seen in Fig. 3.6.

3.3.3 Properties

The DCM can be expressed in terms of MRP as

$$\mathbf{C} = \mathbf{I}_3 - \frac{4\left(1-\sigma^2\right)}{\left(1+\sigma^2\right)^2}\left[\boldsymbol{\sigma}\times\right] + \frac{8}{\left(1+\sigma^2\right)^2}\left[\boldsymbol{\sigma}\times\right]^2 \tag{3.45}$$

The method of extracting the MRP directly from a DCM was not found in literature. It is required to first extract the quaternion from the DCM and then convert the DCM to MRP or SMRP as desired.

If a rotation of $\boldsymbol{\sigma}'$ is followed by a rotation of $\boldsymbol{\sigma}''$ The total rotation can be expressed in the following way

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}'' \otimes \boldsymbol{\sigma}' = \frac{\left(1-(\sigma')^2\right)^2 \boldsymbol{\sigma}'' + \left(1-(\sigma'')^2\right)^2 \boldsymbol{\sigma}' - 2\left[\boldsymbol{\sigma}''\times\right]\boldsymbol{\sigma}'}{1+(\sigma'')^2(\sigma')^2 - 2\boldsymbol{\sigma}''^T\boldsymbol{\sigma}'} \tag{3.46}$$

The relation in Eq. (3.46), using the symbol $\otimes$, is defined as MRP multiplication. Some interesting properties of MRP multiplication, deduced from [Schaub and Junkins, 2002], are

$$\mathbf{0} = -\boldsymbol{\sigma} \otimes \boldsymbol{\sigma} \tag{3.47a}$$
$$\boldsymbol{\sigma}' = (-\boldsymbol{\sigma}'') \otimes \boldsymbol{\sigma} \tag{3.47b}$$
$$\boldsymbol{\sigma}'' = \boldsymbol{\sigma} \otimes (-\boldsymbol{\sigma}') \tag{3.47c}$$

The relations found in Eq. (3.47) can be used when an initial and a final orientation is given, and the rotation between the two has to be found.

3.3.4 Kinematic Differential Equation

The kinematic differential equation for the MRP is

$$\dot{\boldsymbol{\sigma}} = \frac{1}{4}\begin{bmatrix} 1-\sigma^2+2\sigma_1^2 & 2(\sigma_1\sigma_2-\sigma_3) & 2(\sigma_1\sigma_3+\sigma_2) \\ 2(\sigma_2\sigma_1+\sigma_3) & 1-\sigma^2+2\sigma_2^2 & 2(\sigma_2\sigma_3-\sigma_1) \\ 2(\sigma_3\sigma_1-\sigma_2) & 2(\sigma_3\sigma_2+\sigma_1) & 1-\sigma^2+2\sigma_3^2 \end{bmatrix}\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} \tag{3.48}$$

This can rewritten as

$$\dot{\boldsymbol{\sigma}} = \frac{1}{4}\left[\left(1-\sigma^2\right)\mathbf{I}_3 + 2\left[\boldsymbol{\sigma}\times\right] + 2\boldsymbol{\sigma}\boldsymbol{\sigma}^T\right]\boldsymbol{\omega} \tag{3.49}$$

The inverse relation is

$$\boldsymbol{\omega} = \frac{4}{\left(1+\sigma^2\right)^2}\left[\left(1-\sigma^2\right)\mathbf{I}_3 - 2\left[\boldsymbol{\sigma}\times\right] + 2\boldsymbol{\sigma}\boldsymbol{\sigma}^T\right]\dot{\boldsymbol{\sigma}} \tag{3.50}$$

The kinematic differential relations are the same for both MRP and SMRP. However, if the time rate of change of MRP has to be converted to the time rate of change of SMRP, the following relation can be used

$$\dot{\boldsymbol{\sigma}}^S = -\frac{\dot{\boldsymbol{\sigma}}}{\sigma^2} + \frac{1}{2}\left(\frac{1+\sigma^2}{\sigma^4}\right)\boldsymbol{\sigma}\boldsymbol{\sigma}^T\boldsymbol{\omega} \tag{3.51}$$

The MRP provide a good alternative to quaternions. They have a singularity, which can be bypassed by switching parameters. The dimension is 1 less than that of Euler parameters. The kinematic differential equation has no transcendental numbers, and only a quadratic non-linearity.

The various attitude parameters have been dealt with in the chapter. The most important for this thesis study were the quaternions and the MRP as they will be used extensively for the USM. The goal of the USM is to express spacecraft orbit. Therefore, the next chapter deals with *why* and *how* bodies move in space.

Chapter 4

Astrodynamics and Orbital Mechanics

In the previous chapter, the different methods used to express the orientation of a reference frame with respect to another were shown. The theory and equations about quaternions and MRP are especially important for the USM. In this chapter, the reason for the motion of bodies in space is presented. As important as the reason for motion is the motion itself, and this is also presented here.

Astrodynamics is a vast and very old topic with centuries of mathematicians and physicians working on it. This chapter only provides a basic overview of the dynamics applicable to space flight. The information presented here is based on [Wakker, 2007a] and [Wakker, 2007b].

4.1 Gravity

Gravity is the main driving force for motion in space. Therefore, it is essential to understand this force. In this section, only the definition of gravity in classical mechanics is provided and used. The Law of Gravitation was first stated by Sir Isaac Newton as

> *Two particles attract each other with a force directly proportional to their masses and inversely proportional to the square of the distance between them.*

The magnitude of this force, F, is

$$F = G\frac{m_1 m_2}{r^2} \tag{4.1}$$

In Eq. 4.1 the various components are

- G is the *Universal Gravity Constant* with a value of 6.668×10^{-11} $\mathrm{Nm^2kg^{-2}}$.
- m_1 is the mass of the first particle
- m_2 is the mass of the second particle
- r is the distance between the two particles

The gravity force is an attracting force and acts along the line connecting the two bodies. The force on particle 1 can be written in vectorial format as

$$\mathbf{F}_1 = -G\frac{m_1 m_2}{r^3}\mathbf{r}_{2\to 1} \tag{4.2}$$

The acceleration experienced by particle 1, the force per unit mass, is

$$\mathbf{g}_1 = -G\frac{m_2}{r^3}\mathbf{r}_{2\to 1} \tag{4.3}$$

The acceleration due to gravity can also be thought of as the *field strength*, at the position of m_1, of the *gravitational field* generated by m_2. The gravitational field can also be expressed in the form of partial differentiation of a scalar function with respect to the position. This scalar function is known as the *potential*, U. The field strength can be written in terms of the potential as

$$\mathbf{g} = \nabla U \tag{4.4}$$

and the potential being

$$U = -G\frac{m_2}{r} \tag{4.5}$$

The potential of a gravity field is negative everywhere and 0 at inifinity.

The equations shown above are for point masses. Point masses are particles that have a mass, but no size. Spacecraft orbit celestial bodies that are obviously not point masses. This does not mean that the analysis with point masses is wrong. It turns out that the analysis with point masses is a good first order approximation to celestial bodies. For this approximation, the assumption that the mass distribution of the celestial body is radially symmetric is made. With this assumption, the celestial body behaves as if it is a point mass located at its center. This means that the equations derived for point mass are valid for a first order approximation of reality.

4.2 Gravitational Influence of Many Bodies

So far, it has been assumed that there are only two bodies present. In reality, there is an infinite number of bodies in the universe that all attract each other. The equations of motion of a particular point, i, in a set of n bodies, relative to an inertial frame of reference at the center of mass of the system, is

$$m_i\ddot{\mathbf{r}}_i = -G\sum_{j=1}^{n,j\neq i}\frac{m_i m_j}{r_{j\to i}^3}\mathbf{r}_{j\to i} \tag{4.6}$$

Thus, a summation is taken of the gravitational attraction of particle i with respect to all other particles, except for itself. This summation has to be done for every single particle and then integrated to describe the motion. This integration will have to be carried out numerically most of the time.

The many body problem is insightful for understanding the dynamics, but is not always suitable for realistic usage. This is because for space flight applications, it is only necessary to find the motion of a certain body around another body. An example would be the motion of a satellite, body i, in an orbit around the Earth, body k, with the Moon, Sun, and other planets also influencing the motion. The motion needs to be expressed in a non-rotating inertial frame with its origin at the location of k, which undergoes only translational accelerations. The motion would be

$$\ddot{\mathbf{r}}_i = -G\frac{m_i + m_k}{r_i^3}\mathbf{r}_i + G\sum_{j=1}^{n,j\neq i,k} m_j\left(\frac{\mathbf{r}_j - \mathbf{r}_i}{r_{i\to j}^3} - \frac{\mathbf{r}_j}{r_j^3}\right) \tag{4.7}$$

The significance of Eq. (4.7) is that the motion can be expressed as a simple two body problem between body i and body k, with the rest of the bodies providing a perturbing acceleration. This

perturbing acceleration is the difference between the acceleration, due to bodies other than k, on i and the acceleration on k. The potential for this scenario is

$$U = -G\frac{m_k + m_i}{r_i} - G\sum_{j=1}^{n,j\neq i,k} m_j \left(\frac{1}{r_{i\to j}} - \frac{\mathbf{r}_i^T \mathbf{r}_j}{r_j^3}\right) \tag{4.8}$$

The left half of the right-hand side of Eq. (4.8) is known as the *primary potential*, and the right half is known as the *perturbing potential*. Perturbations will be further explained in Appendix B.

4.3 Two-Body Problem

It was shown that for a satellite in orbit around the Earth, the influence of the other solar system bodies can be considered to be perturbations. Thus, a first-order analysis can be carried out considering only the satellite and the Earth. The mass of the Earth is much larger than the mass of the satellite. This means that the effect of the satellite mass on the motion is negligible. Thus, the motion of the satellite can be expressed, in a non-rotating frame fixed to the Earth, by

$$\ddot{\mathbf{r}}_i = -\frac{\mu_k}{r_i^3}\mathbf{r}_i \tag{4.9}$$

where μ is the gravitational parameter and is defined as

$$\mu = Gm_k \tag{4.10}$$

For the case described above, there are certain parameters that are constant. These constants can be derived by some manipulation and integration. The first constant is the energy per unit mass of body i.

$$\mathcal{E} = \frac{1}{2}v^2 - \frac{\mu}{r_i} \tag{4.11}$$

Eq. (4.11) shows that the sum of the kinetic energy per unit mass and the potential energy per unit mass is constant. Another constant is the angular momentum, or the *Second Laplace Vector*.

$$\mathbf{h} = \mathbf{r} \times \mathbf{v} \tag{4.12}$$

Since the angular momentum is constant, the motion of i is on a plane perpendicular to $\mathbf{h}$.

4.4 Orbits

This section continues the analysis of the 2-body problem. After further mathematical analysis, three important laws about the motion of i about k can be found. These laws were first stated by Johannes Kepler, who discovered them empirically. These laws are

1. The motion of i around k is a conic section with k at one of the foci and period, T, semi-major axis , a, and eccentricity, e.
2. A line joining i and k sweeps out a constant area per unit time.
3. The period, T, squared is directly proportional to the semi-major axis, a, cubed.

It is important to note that Kepler's laws only hold for elliptical orbits.

The distance from k to i in the orbit can be described by the following relation

$$r_i = \frac{h^2}{\mu\,(1 + e\cos\nu)} = \frac{p}{1 + e\cos\nu} \tag{4.13}$$

In Eq. (4.13),

p is the semi-latus rectum of the conical section

ν is the true anomaly and is measured from the pericenter, which is the closest point in the orbit

Integrating Eq. (4.9) gives rise to a constant vector, the magnitude of which is the eccentricity. This vector is

$$\mathbf{e} = \frac{1}{\mu_k}\left[\left(\mathbf{v}_i^T\mathbf{v}_i - \frac{\mu}{r_i}\right)\mathbf{r}_i - \left(\mathbf{r}_i^T\mathbf{v}_i\right)\mathbf{v}_i\right] \tag{4.14}$$

Various values of e describe different types of orbits.

4.4.1 Elliptical Orbits

When $0 \leq e < 0$, the orbit of i around n forms a complete ellipse, and the orbital energy is negative, $\mathcal{E} < 0$. This means that i will be able to completely orbit k and return to the pericenter repeatedly. There exists a special case when $e = 0$, where the orbit is circular. In a circular orbit, the distance between i and n remains constant.

For elliptical orbits, the equation describing the orbit becomes

$$r_i = \frac{a(1-e^2)}{1 + e\cos\nu} \tag{4.15}$$

The pericenter, i.e., the smallest separation , occurring at $\nu = 0°$ and the apocenter, i.e., the largest separation, occurring at $\nu = 180°$, distances can be found via

$$r_p = a(1-e) \tag{4.16a}$$
$$r_a = a(1+e) \tag{4.16b}$$

The inverse relations to get a and e are

$$a = \frac{r_a + r_p}{2} \tag{4.17a}$$
$$e = \frac{r_a - r_p}{r_a + r_p} \tag{4.17b}$$

The semi-major axis can be related to the orbital energy in the following way

$$a = \frac{\mu/2}{\frac{\mu}{r_i} - \frac{v_i^2}{2}} = -\frac{\mu}{2\mathcal{E}} \tag{4.18}$$

The velocity at any point in the orbit can be found using the *Vis-Viva Integral*, which is

$$v_i^2 = \mu_k\left(\frac{2}{r_i} - \frac{1}{a}\right) \tag{4.19}$$

The vis-viva integral provides some useful insights on elliptical orbits. The square of the velocity is inversely proportional to the distance to the central body. So, the velocity will be greatest at the pericenter where the distance is the smallest. This can also be deduced from Kepler's second law. At the pericenter, the line joining i and k is the shortest. Thus, it has to move the fastest to sweep the same area. The opposite occurs at the apocenter and the velocity is the lowest there.

The period of an elliptical orbit is

$$T = 2\pi\sqrt{\frac{a^3}{\mu_k}} \tag{4.20}$$

A mean angular motion, n, can now be defined from the period as

$$n = \frac{2\pi}{T} = \sqrt{\frac{\mu_k}{a^3}} \tag{4.21}$$

In a circular orbit, the velocity of the orbiting body is in the orbital plane and is always perpendicular to the radial vector. Since the radius is constant, it is equal to the semi-major axis:

$$r_i = a \tag{4.22}$$

It is important to note that a pure circular orbit will never occur in reality as it is impossible to make sure that the eccentricity is exactly 0. The Circular velocity is an important factor that will be used later on. Circular velocity is the magnitude of the velocity of i in a circular orbit at distance r_i.

$$v_c = \sqrt{\frac{\mu_k}{r_i}} \tag{4.23}$$

4.4.2 Parabolic Orbits

When $e = 1$, a special kind of orbit called a parabolic orbit occurs. The orbital energy of a body in a parabolic orbit is 0, $\mathcal{E} = 0$. A parabolic orbit is a theoretical phenomenon because it is impossible to have an eccentricity exactly equal to 1. A parabolic orbit is open, which means that if i is in a parabolic orbit, it will pass through the pericenter only once. The apocenter is at an infinite distance and thus, the semi-major axis is not defined. Since $e = 1$, the orbital equation simplifies to

$$r_i = \frac{p}{1 + \cos\nu} \tag{4.24}$$

The pericenter distance of a parabolic orbit is

$$r_p = \frac{p}{2} \tag{4.25}$$

The concept of the circular velocity defined in Eq. (4.23) is very useful for parabolic orbits. This is because the magnitude of the velocity at any point in the parabolic orbit is

$$v_i = \sqrt{\frac{2\mu}{r_i}} = \sqrt{2}v_c \tag{4.26}$$

It is clear that any body in a parabolic orbit will travel to an infinite distance away from the central body and thus, escaping the gravity field. Thus, the velocity at any point of a parabolic orbit is the escape velocity at that point. This is the lowest velocity required to escape the central gravity field because the velocity at an infinite distance is 0. Thus, the orbiting body would be able to just escape.

$$v_{esc} = \sqrt{2}v_c \tag{4.27}$$

Since the parabolic orbit is open, it is impossible to define a period for it. However, a special kind of period, which is the time between two consecutive crossings of the semi-latus rectum, is defined as

$$T_p = \frac{4}{3}\sqrt{\frac{p^3}{\mu_k}} \tag{4.28}$$

4.4.3 Hyperbolic Orbits

An orbit with $e > 1$ is known as a hyperbolic orbit. The orbital energy for hyperbolic orbits is greater than 0, $\mathcal{E} > 0$. In a hyperbolic orbit, the path of i around n would become asymptotic after a certain value of the true anomaly. After this value for the true anomaly, the orbit degenerates into a rectilinear orbit. The limit of the true anomaly is

$$\cos\nu > -\frac{1}{e} \tag{4.29}$$

Unlike the parabolic orbit, the semi-major axis of a hyperbolic orbit is defined. It is, however, negative and can be found using

$$a = \frac{p}{1-e^2} \tag{4.30}$$

With this, the rest of the orbital relations are the same as for elliptical orbits.

With hyperbolic orbits, body i would not only be able to escape the gravity field of body k, it would still have a velocity at an infinite distance. This velocity can be derived by using the vis-viva integral, Eq. (4.19), and taking the limit of r_i to infinity. This gives a value of

$$v_\infty^2 = -\frac{\mu_k}{a} \tag{4.31}$$

The velocity at any point in the hyperbolic orbit will be greater than the escape velocity at that point. A mathematical relation for this velocity difference is

$$v^2 - v_{esc}^2 = v_\infty^2 \tag{4.32}$$

4.5 Reference Frames

Now that the characteristics and dynamics of orbits have been treated, the reference frames in which the orbits are considered will be presented. There are many different reference frames available and used for modern space flight. It is beyond the scope of this work to provide an exhaustive overview. The reference frames treated here will be the ones based on the Earth. Both, inertial and non-inertial reference frames will be treated. All the inertial reference frames are actually pseudo-inertial. This is because a true inertial frame would be located at the center of the universe and thus, is not known and would be impractical to use.

4.5.1 Earth Centered Inertial

The Earth Centered Inertial (ECI) reference frame, $\mathcal{F}_{ECI}$, is a non-rotating reference frame with its origin at the center of mass of the Earth. The x-axis lies on the equator and points towards the vernal equinox. The vernal equinox is defined as the intersection of the Earth's equatorial plane and the ecliptic. The ecliptic is the plane in which the Earth orbits the Sun. The z-axis is the axis of rotation of the Earth, and the y-axis completes the dextral triad. The orbits of Earth satellites are expressed in this frame. A realization of this frame is the J2000 frame, which freezes the directions of the x-axis and z-axis at the respective directions they had at 12:00 on January 1 2000. This was necessary because the location of the vernal equinox and the rotation changes with time.

4.5.2 Earth Centered Earth Fixed

The Earth Centered Earth Fixed (ECEF) reference frame, $\mathcal{F}_{ECEF}$, is a rotating reference frame with its origin at the center of mass of the Earth. The x-axis lies on the equator and points

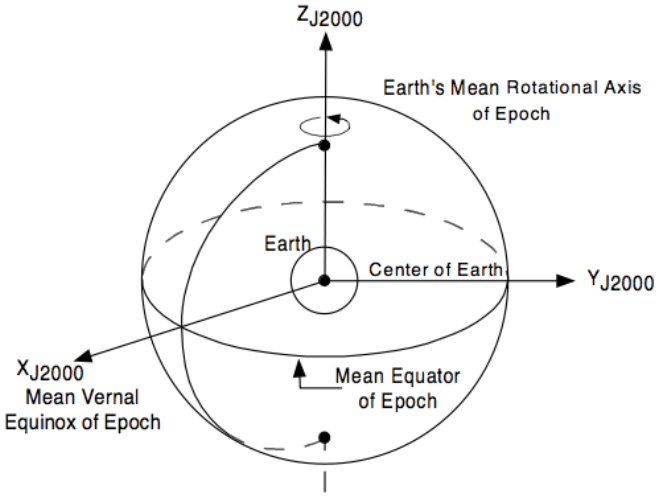


Figure 4.1: The J2000 Reference Frame [Rose, 1998]

towards a reference meridian. This reference meridian is the International Earth Rotation and Reference Systems Service (IERS) reference meridian . The z-axis lies along the Earth's axis of rotation. The y-axis completes the dextral triad. For the rest of this report, it is assumed that the direction of the rotation axis of the Earth remains constant. For this case, the rotation matrix from ECI to ECEF is

$$\begin{aligned} \mathbf{C}_{ECEF,ECI} &= \mathbf{C}_3(\theta_{J2000} + \omega_E \cdot \Delta t) \\ &= \begin{bmatrix} \cos(\theta_{J2000} + \omega_E \cdot \Delta t) & \sin(\theta_{J2000} + \omega_E \cdot \Delta t) & 0 \\ -\sin(\theta_{J2000} + \omega_E \cdot \Delta t) & \cos(\theta_{J2000} + \omega_E \cdot \Delta t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{4.33}$$

In Eq. (4.33), $\theta_{J2000} \approx 100°$ is the angular separation between the x-axes of ECI and ECEF at 12:00 on January 1 2000 and Δt is the time since then till present.

In reality, the rotation axis of the Earth changes with time, which can be seen in Fig. 4.2. There is a long-term change called *Precession*, and a short term change called *Nutation*. There is also an additional motion known as *Polar Motion*, which is not shown in Fig. 4.2, that occurs due to complicated dynamics and nutation errors that are not modeled. The precession occurs because the hemisphere of Earth that points towards the Sun during aphelion points away from the Sun during perihelion. This is the reason for seasons not coinciding in the northern and southern hemispheres. Nutation is caused due to the gravitational effects of the Sun and the Moon. Common ways of computing the nutation include the IERS 1996 Theory of Nutation, or the 1980 International Astronomical Union (IAU) Theory of Nutation, of which the IERS theory is more accurate. In this thesis study, however, the rotation axis of the Earth is assumed to be fixed.

World Geodetic System 1984

A realization of the ECEF reference frame is the World Geodetic System 1984 (WGS 84). This frame is the standard for Global Positioning System (GPS) and thus, is the frame that is most frequently used. The WGS 84 models the Earth as a ellipsoid of revolution instead of a sphere, has its own gravity model, and it defines its own set of fundamental constants. Some of the most important constants can be found in Table 4.1.

The ellipsoid is known as the Geodetic ellipse and can be seen in Fig. 4.3. In Fig. 4.3, N is the radius of curvature in the prime vertical, h is the height above the ellipse along the direction of the prime vertical, and ϕ_{gd} is the geodetic latitude, and ϕ_{gc} is the geocentric latitude. The

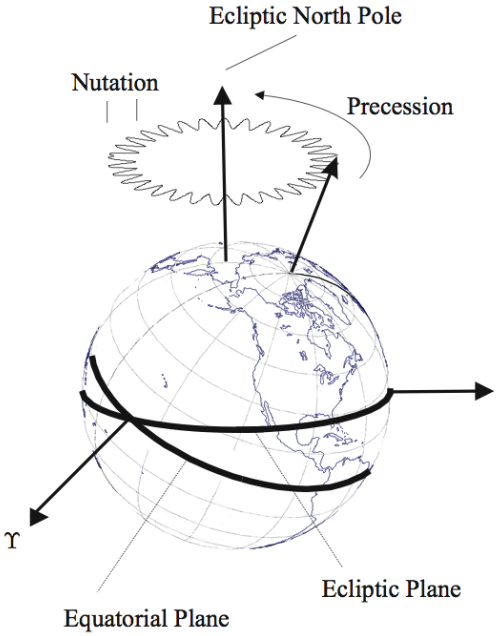


Figure 4.2: Motion of Earth's spin axis [Hughes, 2009]

position given in by a GPS measurement unit is usually h, ϕ_{gd}, and the longitude λ. For the remainder of the report, the latitude used is always the geodetic one. The geocentric latitude is only used if the spherical planet assumption is explicitly stated.

To convert these spherical coordinates to Cartesian coordinates, the ellipsoid flatness, f, has to be found first.

$$f_E = \frac{a_E - b_E}{a_E} = 3.3528107 \times 10^{-3} \tag{4.34}$$

The eccentricity of the ellipse is then found using

$$e_E = \sqrt{f_E\,(2 - f_E)} = 8.1819191 \times 10^{-2} \tag{4.35}$$

The ellipsoid flatness and the eccentricity are constants and thus, have to be calculated only once. The radius of curvature in the prime vertical varies with the latitude and can be found in the following way

$$N = \frac{a_E}{\sqrt{1 - e_E^2 \sin^2 \lambda}} \tag{4.36}$$

Table 4.1: Some important parameters about the Earth defined in the World Geodetic System 1984 [Grewal et al., 2001]

Parameter	Symbol	Value
Semi-major axis of Earth	a_E	6378.1370000 km
Semi-minor axis of Earth	b_E	6356.7523142 km
Rotation rate of Earth	ω_E	$7.2921151467 \times 10^{-5}$ rad/s
Gravitational parameter of Earth	μ_E	3.986005×10^5 km^3 / s^2

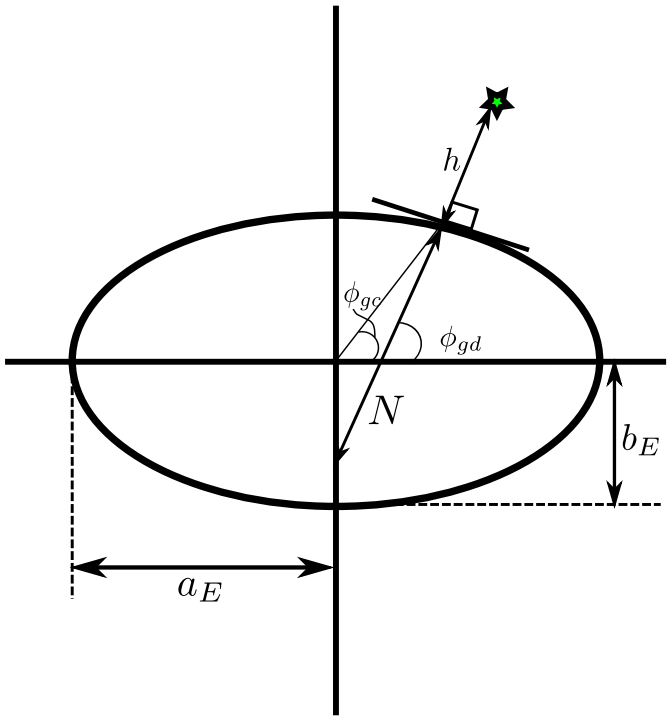


Figure 4.3: Shape of the geodetic ellipse

Finally the cartesian coordinates are

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{ECEF} = \begin{bmatrix} (h+N)\cos\phi_{gd}\cos\lambda \\ (h+N)\cos\phi_{gd}\sin\lambda \\ \left(h+\left(1-e_E^2\right)N\right)\sin\phi_{gd} \end{bmatrix} \tag{4.37}$$

The inverse conversion to find h, ϕ_{gd}, and λ is more involved. This is especially true for ϕ_{gd}, where iteration has to be used. The following conversion from Cartesian coordinates in $\mathcal{F}_{ECEF}$ to h, ϕ_{gd}, and λ is taken from [Hughes, 2009]. The longitude can be calculated using

$$\lambda = \text{atan2}(y, x) \tag{4.38}$$

For ϕ_{gd}, an initial guess is computed using

$$\phi_{gd_0} = \text{atan2}(z, \sqrt{x^2+y^2}) \tag{4.39}$$

An iteration is now started till $\Delta\phi_{gd}$ becomes smaller than a certain specified value. At each step

$$\phi_{gd_k} = \text{atan2}\left(z + \frac{a_E \cdot e_E^2 \sin^2\phi_{gd_{k-1}}}{\sqrt{1-e_E^2\sin\phi_{gd_{k-1}}}}, \sqrt{x^2+y^2}\right) \tag{4.40}$$

$$\Delta\phi_{gd} = \left|\phi_{gd_k} - \phi_{gd_{k-1}}\right| \tag{4.41}$$

Finally, the altitude is

$$h = \frac{\sqrt{x^2+y^2}}{\cos\phi_{gd}} - \frac{a_E}{\sqrt{1-e_E^2\sin^2\phi_{gd}}} \tag{4.42}$$

The difference between the geodetic and geocentric latitude quite small. The maximum value of the difference occurs when $\phi_{gc} \approx 45°$ and is approximately 0.2°, and the average difference is approximately 0.1°.

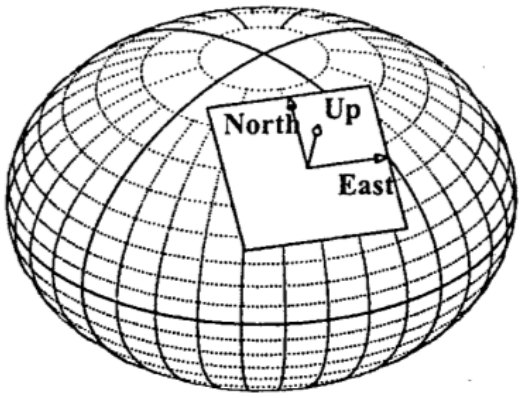


Figure 4.4: Definition of the local tangent plane. Modified from [Grewal et al., 2001]

Local Tangent Plane

In the ECEF, a local tangent plane can be defined as being tangential to the plane at the specified longitude and latitude. A reference frame can be defined on this tangential plane with the axes pointing East, North, and Up. An example of this frame, $\mathcal{F}_{ENU}$, is found in Fig. 4.4. The first two axes lie on the tangential plane to the East and North respectively, and the third axis points up along the radius of curvature in the prime vertical. The rotation matrix from $\mathcal{F}_{ECEF}$ to $\mathcal{F}_{ENU}$ is

$$\begin{aligned}\mathbf{C}_{ENU,ECEF} &= \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \mathbf{C}_2(\phi_{gd})\mathbf{C}_3(\lambda) \\ &= \begin{bmatrix} -\sin\lambda & \cos\lambda & 0 \\ -\cos\lambda\sin\phi_{gd} & -\sin\phi_{gd}\sin\lambda & \cos\phi_{gd} \\ \cos\phi_{gd}\cos\lambda & \cos\phi_{gd}\sin\lambda & \sin\phi_{gd} \end{bmatrix}\end{aligned} \tag{4.43}$$

Sometimes, an alternate version of the local tangent plane is used with the first two axes pointing North and East respectively, and the third axis pointing down along the radius of curvature in the prime vertical. The rotation matrix between $\mathcal{F}_{ENU}$ and $\mathcal{F}_{NED}$ is

$$\mathbf{C}_{NED,ENU} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{4.44}$$

4.5.3 International Celestial Reference Frame

To describe interplanetary orbits, or orbits of other solar system bodies around the Sun, it is impractical to have an inertial frame centered on the Earth. Thus, the International Celestial Reference Frame, $\mathcal{F}_{ICRF}$, is defined with its origin at the Solar System barycenter. The axes are defined with respect to objects that are outside the Milky Way. The plane defined by the x-axis and y-axis is the same as the J2000 ECI equatorial plane. Since the reference bodies for the axes are very far away, this frame is as close as possible to a true inertial frame. This reference frame is now used to express the orbits of the planets in the solar system.

A Heliocentric reference frame is also used to express interplanetary trajectories.

4.6 Describing Orbits

Orbits can be expressed in the inertial coordinates using simple Cartesian coordinates. The full state consists of three position and three velocity elements and they are normally used for numerical computations. Spherical coordinates can also be used to describe orbits, but they will not be treated here. The position and velocity, in Cartesian coordinates, in the inertial frame would be

$$\mathbf{r} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{4.45a}$$

$$\mathbf{v} = \begin{bmatrix} v_x \\ v_y \\ v_z z \end{bmatrix} \tag{4.45b}$$

The one major drawback of this method is that it is almost impossible to be able to imagine an orbit. There is a set of 6 elements known as the Keplerian elements that make imagining an orbit easy. The Keplerian elements are described in the ECI frame, but can be used for any other inertial frame. Three of the elements describe the orientation of the orbital plane in 3-dimensional space. These 3 elements are Euler angles with the (3-1-3) sequence and can be seen in Figure 3.2. The Euler angles, (Ω, i, ω) mean

- Ω: Right ascension of ascending node is the angle, along the equatorial plane, between the vernal equinox and the ascending node.
- i: The inclination is the angle between the orbital plane and the equatorial plane. The inclination can range between 0 and 180 degrees. When $i < 90$ degrees the orbit is known as prograde, so the orbit is in the same sense as the rotation of the Earth. When $i > 90$ degrees the orbit is retrograde, so the orbit is in the opposite sense as the rotation of the Earth.
- ω: The argument of periapsis is the angle between the ascending node and the periapsis, along the orbital plane.

Now that the orientation of the orbit is fixed, the shape of the orbit on the 2 dimensional plane has to be determined. For this purpose, 2 elements are used. They are the semi-major axis, a, and the eccentricity, e. Finally, the location of the satellite in the orbit has to be fixed. This is described using the true anomaly, ν. It is necessary to be able to switch between the Keplerian elements and Cartesian coordinates.

4.6.1 Cartesian Coordinates to Keplerian Elements

The conversion from Cartesian coordinates to Keplerian elements is now shown. The method of [Curtis, 2005] is used. The conversion deals with closed orbits, so only elliptical and circular orbits. The orbit can be split into two halves. One half is the trajectory from the periapsis to the apoapsis, where the separation between the central body and the orbiting body is increasing. The second half is the trajectory from the apoapsis to the periapsis, where the separation between the central body and the orbiting body increases. The conversion procedure is now outlined.

The radial velocity should be computed. This result indicates the sector of the orbit the satellite is in. If the the radial velocity is negative, the satellite is traveling from the apoapsis to the periapsis. If the radial velocity is positive, the satellite is traveling from the periapsis to the apoapsis.

$$v_r = \frac{\mathbf{r}^T \mathbf{v}}{r} \tag{4.46}$$

The semi-major axis is first computed

$$a = \frac{\mu/2}{\frac{\mu}{r} - \frac{v^2}{2}} \tag{4.47}$$

The angular momentum can be calculated by taking the vector product of the position and velocity:

$$\mathbf{h} = \begin{bmatrix} h_x & h_y & h_z \end{bmatrix}^T = \mathbf{r} \times \mathbf{v} \tag{4.48}$$

with

$$h = |\mathbf{h}| \tag{4.49}$$

The eccentricity vector can be computed using

$$\mathbf{e} = \frac{1}{\mu}\left[\left(\mathbf{v}^T\mathbf{v} - \frac{\mu}{r}\right)\mathbf{r} - \left(\mathbf{r}^T\mathbf{v}\right)\mathbf{v}\right] \tag{4.50}$$

The inclination can be calculated as

$$i = \arccos\left(\frac{h_z}{h}\right) \tag{4.51}$$

An intermediate vector, $\mathbf{N}$, is defined as

$$\mathbf{N} = \begin{bmatrix} N_1 \\ N_2 \\ N_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times \mathbf{h} \tag{4.52}$$

Using $\mathbf{N}$, the right ascension of ascending node can be computed. The tangent and cosine of Ω are

$$\cos\Omega = \frac{N_1}{N} \tag{4.53a}$$

$$\tan\Omega = \frac{N_2}{N_1} \tag{4.53b}$$

This depends on the value of N_2. If $N_2 \geq 0$

$$\Omega = \arccos\left(\frac{N_1}{N}\right) \tag{4.54}$$

If $N_2 < 0$

$$\Omega = 2\pi - \arccos\left(\frac{N_1}{N}\right) \tag{4.55}$$

The argument of perigee depends on the third element of the eccentricity vector. If $e_3 \geq 0$

$$\omega = \arccos\left(\frac{\mathbf{N}^T\mathbf{e}}{N \cdot e}\right) \tag{4.56}$$

If $e_3 < 0$

$$\omega = 2\pi - \arccos\left(\frac{\mathbf{N}^T\mathbf{e}}{N \cdot e}\right) \tag{4.57}$$

Finally, the true anomaly can be computed and this depends on the sign of the radial velocity. If $v_r \geq 0$

$$\nu = \arccos\left(\frac{\mathbf{e}^T\mathbf{r}}{e \cdot r}\right) \tag{4.58}$$

If $v_r < 0$

$$\nu = 2\pi - \arccos\left(\frac{\mathbf{e}^T\mathbf{r}}{e \cdot r}\right) \tag{4.59}$$

Now that the conversion from Cartesian coordinates to Keplerian elements as found in [Curtis, 2005] has been shown, the inverse conversion needs to be shown.

4.6.2 Keplerian Elements to Cartesian Coordinates

The inverse conversion is based on the method found in [Curtis, 2005; Wakker, 2007a] The orbital plane is described by a set of (3-1-3) Euler angles, Ω, i, ω, with the rotation matrix

$$\mathbf{C}_{OP,ECI} = \mathbf{C}_3(\omega)\mathbf{C}_1(i)\mathbf{C}_3(\Omega) = \begin{bmatrix} \mathrm{c}\omega\mathrm{c}\Omega - \mathrm{s}\omega c i\mathrm{s}\Omega & \mathrm{c}\omega\mathrm{s}\Omega + \mathrm{s}\omega c i\mathrm{c}\Omega & \mathrm{s}\omega\mathrm{s}i \\ -\mathrm{s}\omega\mathrm{c}\Omega - \mathrm{c}\omega c i\mathrm{s}\Omega & -\mathrm{s}\omega\mathrm{s}\Omega + \mathrm{c}\omega c i\mathrm{c}\Omega & \mathrm{c}\omega\mathrm{s}i \\ \mathrm{s}i\mathrm{s}\Omega & -\mathrm{s}i\mathrm{c}\Omega & \mathrm{c}i \end{bmatrix} \tag{4.60}$$

The rotation matrix defines the transformation from the ECI frame to a frame in the orbital plane, $\mathcal{F}_{OP}$. In this new frame the first axis points towards the periapsis, the third axis lies along the angular momentum vector, and the second axis completes the dextral triad, which can be seen in Fig. 3.2 on page 15. The position in this orbital plane frame is

$$\mathbf{r}_{OP} = r \begin{bmatrix} \cos\nu \\ \sin\nu \\ 0 \end{bmatrix} \tag{4.61}$$

The velocity in this frame is

$$\mathbf{v}_{OP} = \begin{bmatrix} \dot{r}\cos\nu - r\dot{\nu}\sin\nu \\ \dot{r}\sin\nu + r\dot{\nu}\cos\nu \\ 0 \end{bmatrix} = \frac{\mu}{h} \begin{bmatrix} -\sin\nu \\ e + \cos\nu \\ 0 \end{bmatrix} \tag{4.62}$$

The magnitude of the angular momentum can be computed using

$$h = \sqrt{\mu a(1 - e^2)} \tag{4.63}$$

The position and velocity now need to be transformed from the orbital plane frame to the planet centered inertial frame. For this, it is necessary to know the inverse of the DCM, which is simply the transpose.

$$\mathbf{C}_{ECI,OP} = \begin{bmatrix} \mathrm{c}\omega\mathrm{c}\Omega - \mathrm{s}\omega c i\mathrm{s}\Omega & -\mathrm{s}\omega\mathrm{c}\Omega - \mathrm{c}\omega c i\mathrm{s}\Omega & 0 \\ \mathrm{c}\omega\mathrm{s}\Omega + \mathrm{s}\omega c i\mathrm{c}\Omega & -\mathrm{s}\omega\mathrm{s}\Omega + \mathrm{c}\omega c i\mathrm{c}\Omega & 0 \\ \mathrm{s}\omega\mathrm{s}i & \mathrm{c}\omega\mathrm{s}i & 0 \end{bmatrix} \tag{4.64}$$

The last column of the matrix in Eq. (4.64) is a zero vector because there are no out of plane positions or velocities in an orbit. The position and velocity in the Cartesian coordinates of the ECI are

$$\mathbf{r}_{ECI} = \mathbf{C}_{ECI,OP}\mathbf{r}_{OP} \tag{4.65a}$$

$$\mathbf{v}_{ECI} = \mathbf{C}_{ECI,OP}\mathbf{v}_{OP} \tag{4.65b}$$

The USM is based on orbital theory and thus, it is vital to present astrodynamics, the cause of this orbital motion. Based on orbital theory, it is possible to derive hodographs, which are an important part of the USM and will be presented in the following chapter.

Chapter 5

Hodograph Theory

Based on the Astrodynamics presented the in the previous chapter, some special interesting properties of orbital motion can be derived in the form of hodographs. Hodographs, greek for path-writing, are also known as velocity diagrams. They were invented by Sir Hamilton, who also invented the quaternions, a subset of which are the Euler parameters. The various orbits in position space have been dealt with, and in this section orbits in velocity, acceleration, and jerk spaces will be shown. The orbits in the velocity space show some very interesting characteristics, which are essential for the Unified State Model. An excellent derivation of hodographs is shown in [Altman, 1967a] and [Eades, 1968] and the reader is referred to it, if more information about this topic is necessary. Only unperturbed orbits are considered in this chapter.

5.1 Velocity Hodograph

The method of [Eades, 1968] for deriving the equations of the velocity hodograph is shown here. To derive the velocity hodograph, 3 different reference frames are used. They are

$\mathcal{F}_I$ is an inertial reference frame like the ECI

$\mathcal{F}_P$ is an inertial frame embedded in the orbital plane, which was used to convert Keplerian elements to ECI.

$\mathcal{F}_O$ is a rotating reference frame on the orbital plane

$\mathcal{F}_P$ can be fixed with respect to $\mathcal{F}_I$ using the (3-1-3) sequence of Euler angles (Ω, i, ω). $\mathcal{F}_P$ consists of the unit vectors $\hat{\mathbf{e}}_x$, $\hat{\mathbf{e}}_y$, and $\hat{\mathbf{e}}_z$. The unit vector $\hat{\mathbf{e}}_z$ points along the angular momentum vector, $\hat{\mathbf{e}}_x$ points to the pericenter, and $\hat{\mathbf{e}}_y$ completes the reference frame. $\mathcal{F}_O$ consists of the unit vectors $\hat{\mathbf{e}}_r$, $\hat{\mathbf{e}}_\nu$, and $\hat{\mathbf{e}}_z$. The unit vector $\hat{\mathbf{e}}_r$ lies along the radial vector of the orbiting body, and $\hat{\mathbf{e}}_\nu$ completes the reference frame. $\mathcal{F}_O$ can be found by carrying out an Euler rotation of $\mathcal{F}_P$ with the Euler axis $\hat{\mathbf{e}}_z$ and the Euler angle ν, which is the true anomaly. These two reference frames can be seen in Fig. 5.1.

To derive the equations for the velocity hodograph, the equations of motion due to the central gravity field is right multiplied by the angular momentum, $\mathbf{h}$.

$$\ddot{\mathbf{r}} \times \mathbf{h} = -\frac{\mu}{r^2}\hat{\mathbf{e}}_r \times \mathbf{h} = -\frac{\mu}{r^2}\hat{\mathbf{e}}_r \times (\mathbf{r} \times \dot{\mathbf{r}}) \tag{5.1}$$

This can be simplified to

$$\ddot{\mathbf{r}} \times \mathbf{h} = \mu\dot{\hat{\mathbf{e}}}_r \tag{5.2}$$

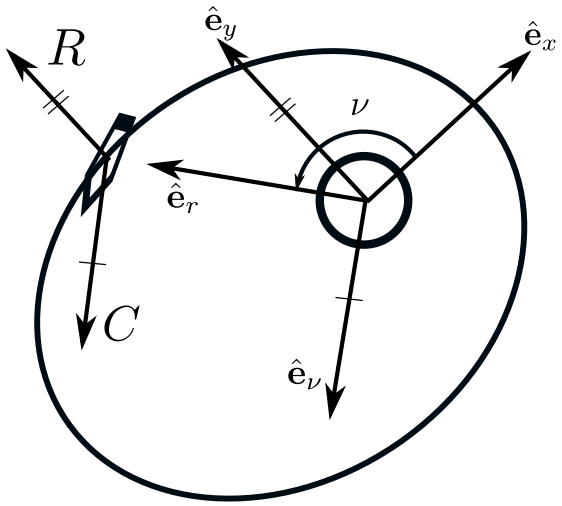


Figure 5.1: A 2-dimensional image of the two reference frames $\mathcal{F}_P$ and $\mathcal{F}_O$, used for the derivation of a hodograph and the hodographic velocities

According to [Wakker, 2007a], after integration, the result is

$$\dot{\mathbf{r}} \times \mathbf{h} = \mu(\hat{\mathbf{e}}_r + \mathbf{e}) \tag{5.3}$$

In Eq. (5.3), $\mathbf{e}$ is the eccentricity vector. It is directed towards the periapsis, and so it can be written in $\mathcal{F}_P$ as

$$\mathbf{e} = e\hat{\mathbf{e}}_x \tag{5.4}$$

Eq. (5.3) can be left multiplied by $\mathbf{h}$ to give

$$\mathbf{h} \times (\dot{\mathbf{r}} \times \mathbf{h}) = \mathbf{h} \times (\mu(\hat{\mathbf{e}}_r + e\hat{\mathbf{e}}_x)) \tag{5.5}$$

This calculation is simplified due to the fact that $v_z = 0$ and $\mathbf{h} = h\hat{\mathbf{e}}_z$. After carrying carrying out the computation

$$\dot{\mathbf{r}} = \frac{\mu}{h}(\hat{\mathbf{e}}_\nu + e\hat{\mathbf{e}}_y) \tag{5.6}$$

Eq. (5.6) can be written as

$$\dot{\mathbf{r}} = C\hat{\mathbf{e}}_\nu + R\hat{\mathbf{e}}_y \tag{5.7}$$

with C and R being

$$C = \frac{\mu}{h} \tag{5.8}$$

$$R = \frac{\mu}{h}e = Ce \tag{5.9}$$

This means that the orbital velocity is made up of the sum of C, normal to the radial vector and in the orbital plane, and R, lying 90° ahead of periapsis in the orbital plane. The directions of C and R can be seen in Fig. 5.1. C and R are constants when there are no perturbations. The velocity along an axis at any time in the orbit now only varies with the true anomaly. The components of velocity can be derived using simple trigonometry, notably

$$v_x = -C\sin\nu \tag{5.10a}$$

$$v_y = C(\cos\nu + e) = R + C\cos\nu \tag{5.10b}$$

$$v_z = 0 \tag{5.10c}$$

$$v_r = Ce\sin\nu = R\sin\nu \tag{5.11a}$$
$$v_\nu = C(1 + e\cos\nu) = C + R\cos\nu \tag{5.11b}$$
$$v_z = 0 \tag{5.11c}$$

The information presented here can now be used to define the velocity hodograph. There are two types of hodographs

Classical Expressed in $\mathcal{F}_P$

Polar Expressed in $\mathcal{F}_O$

Equation 5.7 equated to the velocity in $\mathcal{F}_P$ is

$$\dot{\mathbf{r}} = C\hat{\mathbf{e}}_\nu + R\hat{\mathbf{e}}_y = v_x\hat{\mathbf{e}}_x + v_y\hat{\mathbf{e}}_y \tag{5.12}$$

Combining similar terms gives

$$C\hat{\mathbf{e}}_\nu = v_x\hat{\mathbf{e}}_x + (v_y - R)\hat{\mathbf{e}}_y \tag{5.13}$$

Taking the square of the magnitude of the vectorial equation gives

$$v_x^2 + (v_y - R)^2 = C^2 \tag{5.14}$$

Eq. (5.14) shows that the classical velocity hodograph is a circle with its center at $(v_x = 0, v_y = R)$ and a radius C. This classical velocity hodograph for various types of orbits with the same value of C can be seen in Fig. 5.2. The effect of increasing the eccentricity is that R increases, and thus the center of the hodograph circle goes up, along the v_y axis.

Eq. (5.7) equated to the velocity in $\mathcal{F}_O$ is

$$\dot{\mathbf{r}} = C\hat{\mathbf{e}}_\nu + R\hat{\mathbf{e}}_y = v_r\hat{\mathbf{e}}_r + v_\nu\hat{\mathbf{e}}_\nu \tag{5.15}$$

Combining similar terms gives

$$R\hat{\mathbf{e}}_y = v_r\hat{\mathbf{e}}_r + (v_\nu - C)\hat{\mathbf{e}}_\nu \tag{5.16}$$

Taking the square of the magnitude of the vectorial equation gives

$$v_r^2 + (v_\nu - C)^2 = R^2 \tag{5.17}$$

Eq. (5.17) shows that the polar velocity hodograph is a circle with its center at $(v_r = 0, v_\nu = R)$ and a radius R. The polar velocity hodograph for various types of orbits with the same value of C can be seen in Figure 5.3. The effect of increasing the eccentricity is that R increases, and thus the radius of the hodograph circle increases. As a comparison, these orbits in position space can be seen in Fig. 5.4.

5.2 Acceleration Hodograph

For acceleration, there is no polar hodograph as the acceleration only acts in the radial direction. Using C, the trajectory of the conic section can be written as

$$r = \frac{h^2}{\mu(1 + e\cos\nu)} = \frac{\mu}{C(C + R\cos\nu)} \tag{5.18}$$

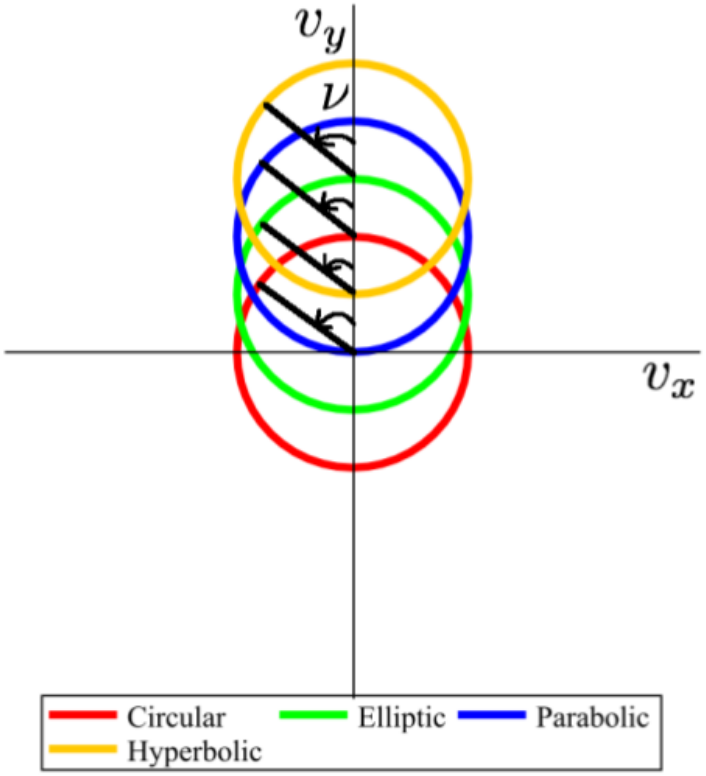


Figure 5.2: Classical velocity hodograph for various types of orbits

The acceleration due to the central gravity field is

$$\ddot{\mathbf{r}} = -\frac{\mu}{r^2}\hat{\mathbf{e}}_r \tag{5.19}$$

Plugging in r from Eq. (5.18) into Eq. (5.19) gives

$$\ddot{\mathbf{r}} = -\frac{C^2}{\mu}(C + R\cos\nu)^2\hat{\mathbf{e}}_r \tag{5.20}$$

The acceleration hodograph in Cartesian coordinates can be derived by multiplying with $\sin\nu$ and $\cos\nu$ respectively. The acceleration in the x direction is

$$a_x = \frac{C^2}{\mu}(C + R\cos\nu)^2\cos\nu \tag{5.21a}$$

$$a_y = \frac{C^2}{\mu}(C + R\cos\nu)^2\sin\nu \tag{5.21b}$$

The Cartesian acceleration hodograph can be seen in Fig. 5.5.

5.3 Jerk Hodograph

In theory, it is possible to continue getting the hodographs in higher order spaces. Out of scientific curiosity, a hodograph in jerk space is made. Jerk is the time derivative of the acceleration and thus, the third time derivative of the position. The acceleration can also be found by taking the time derivative of the velocity expressed in $\mathcal{F}_O$. The time derivatives of the unit vectors in $\mathcal{F}_O$ are

$$\dot{\hat{\mathbf{e}}}_r = \dot{u}\hat{\mathbf{e}}_u \tag{5.22a}$$

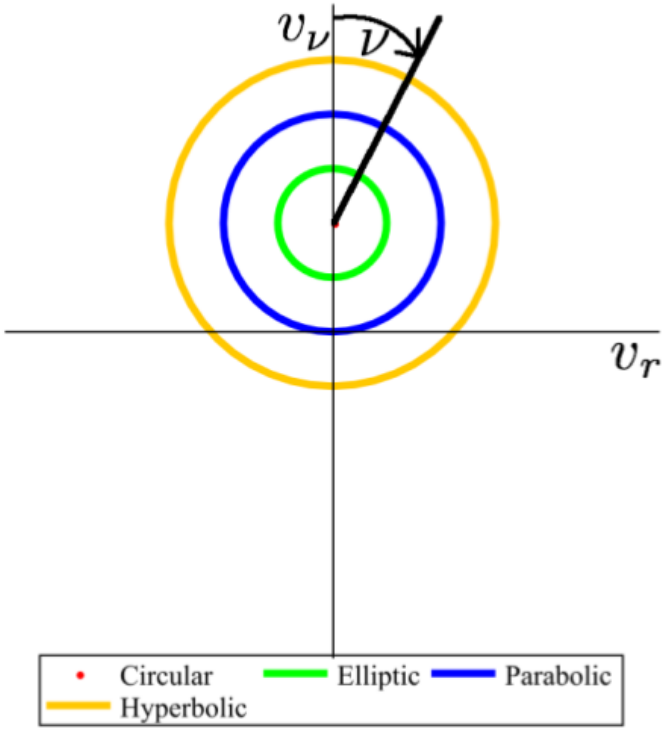


Figure 5.3: Polar velocity hodograph for various types of orbits

$$\dot{\hat{\mathbf{e}}}_u = -\dot{u}\hat{\mathbf{e}}_r \tag{5.22b}$$

The acceleration is

$$\mathbf{a} = \dot{v}_r\hat{\mathbf{e}}_r + v_r\dot{\hat{\mathbf{e}}}_r + \dot{v}_\nu\hat{\mathbf{e}}_\nu + v_\nu\dot{\hat{\mathbf{e}}}_\nu \tag{5.23}$$

After expanding, the value of acceleration becomes

$$\mathbf{a} = (\dot{R}\sin\nu - C\dot{\nu})\hat{\mathbf{e}}_r + (\dot{C} + \dot{R}\cos\nu)\hat{\mathbf{e}}_\nu \tag{5.24}$$

Since there are no perturbations, $\dot{R}$ and $\dot{C}$ are 0. Thus, Eq. (5.24) can be simplified to

$$\mathbf{a} = -C\dot{\nu}\hat{\mathbf{e}}_r \tag{5.25}$$

By setting Eq. (5.25) to be equal to Eq. (5.20), the value of $\dot{\nu}$ can be found to be

$$\dot{\nu} = \frac{C}{\mu}(C + R\cos\nu)^2 \tag{5.26}$$

The time derivative of the scalar, r, is found to be

$$\dot{r} = CR\sin\nu \tag{5.27}$$

Taking the derivative of the acceleration gives

$$\dot{\mathbf{a}} = \frac{2\mu\dot{r}}{r^3}\hat{\mathbf{e}}_r - \frac{\mu\dot{\nu}}{r^2}\hat{\mathbf{e}}_\nu \tag{5.28}$$

After filling in the corresponding variables, the jerk is

$$\dot{\mathbf{a}} = 2\frac{C^3R}{\mu^2}(C + R\cos\nu)^3\sin\nu\hat{\mathbf{e}}_r - \frac{C^3}{\mu^2}(C + R\cos\nu)^4\hat{\mathbf{e}}_\theta \tag{5.29}$$

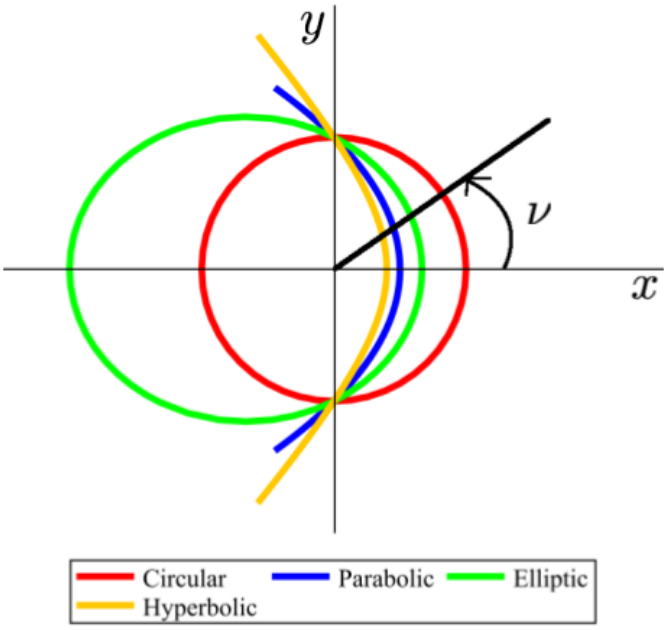


Figure 5.4: Various orbits in position space

The jerk hodograph expressed in $\mathcal{F}_O$ can be seen in Fig. 5.6. There is no polar jerk present for a circular orbit because the acceleration is constant and only in the radial direction. The jerk can be expressed in terms of the unit vectors of $\mathcal{F}_P$ as

$$\dot{\mathbf{a}} = (j_r \cos\nu - j_\nu \sin\nu)\hat{\mathbf{e}}_x + (j_r \sin\nu + j_\nu \cos\nu)\hat{\mathbf{e}}_y \tag{5.30}$$

The symbol for jerk can be either $\dot{a}$, or j. The jerk hodograph expressed in $\mathcal{F}_P$ can be seen in Fig. 5.7.

All the theory necessary for the USM: quaternions, astrodynamics, and the hodograph theory has now been presented. Therefore, the USM can finally be derived in the next chapter.

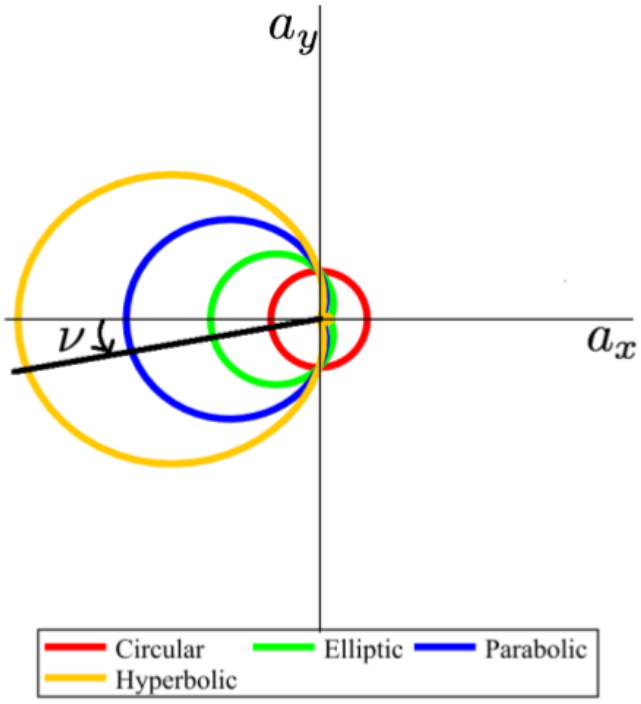


Figure 5.5: The acceleration hodograph of various orbits

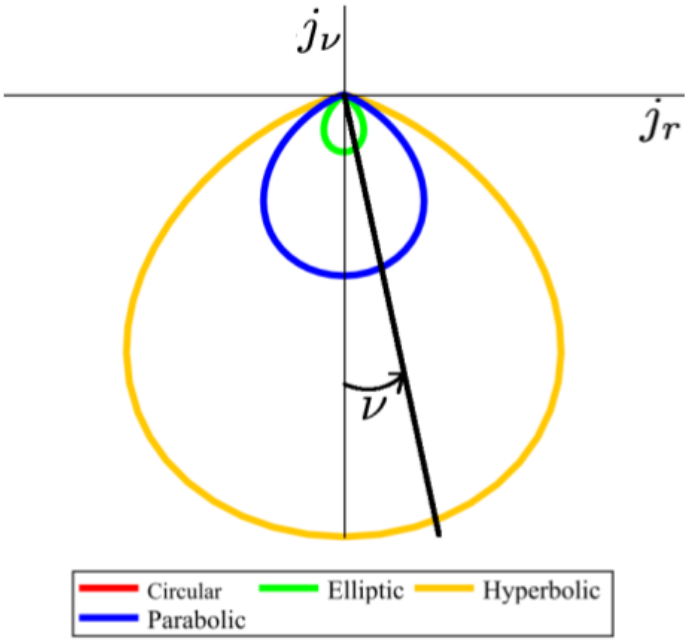


Figure 5.6: The polar jerk hodograph of various orbits

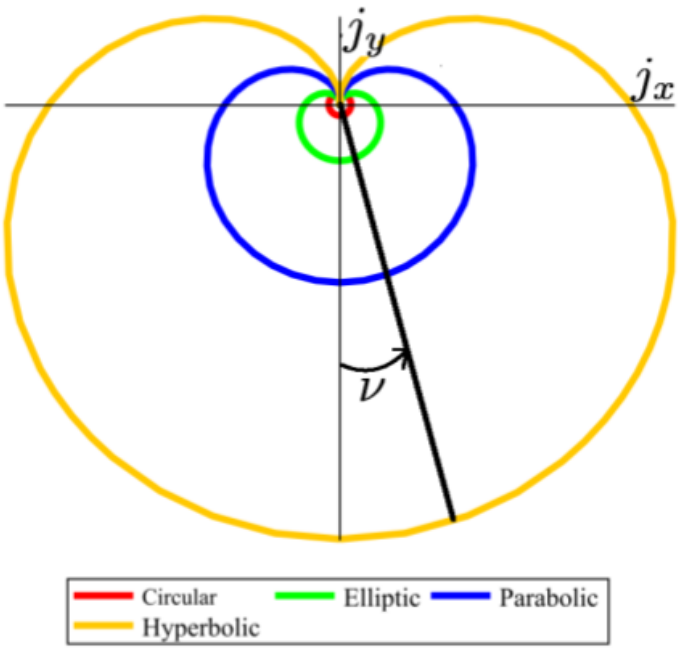


Figure 5.7: The classical jerk hodograph of various orbits

Chapter 6

Unified State Model

All the background theory required for understanding the USM has been presented in the preceding chapters. In this chapter the Unified State Model (USM) proposed in [Altman, 1972] is presented and derived. The USM describes an orbit using seven parameters and it does so in a local frame of reference. Three of the seven parameters are related to the velocity hodograph and the remaining four are the quaternion elements that represent the orientation of the local orbital reference frame with respect to the inertial frame. The USM is a very interesting model that treats orbital mechanics in a similar manner as rigid body mechanics. It was decided to rigorously derive the USM to make this model more transparent to the author and any other interested parties for future work with the model. This was a very important factor as the USM is a very elegant and promising model, on which a very small amount of work has been carried out so far. Also, this would ensure that any errors in the USM in either [Altman, 1972], or [Chodas, 1981] would be identified and corrected. Apart from the traditional USM, another version of the USM using MRP has been proposed in this thesis study.

The velocity parameters that will be used are $\mathbf{C}$ and $\mathbf{R}$. C is the radius of the velocity hodograph of the orbit, found in Section 5.1 on page 39, and R is the displacement of the center from the origin. These values can also be expressed as vectors. $\mathbf{C}$ has the magnitude C and lies along the positive orbit normal. $\mathbf{R}$ has the magnitude R and lies 90° ahead of the perifocus, on the orbital frame. These velocity parameters have been derived in Chapter 5 and their orientations can be seen in Fig. 5.1.

It is assumed for the remainder of the chapter that the orbit is closed, so $e < 1$. This is because the derivation starts with Keplerian elements that are only valid for closed orbits. This is, however, only a limitation of the Keplerian elements and not of the USM.

6.1 Local Orbital Frame

The local orbital frame is called $\mathcal{F}_e$. The three reference vectors of this frame are

- $\hat{\mathbf{e}}_1$ along the position vector
- $\hat{\mathbf{e}}_2$ along the direction of flight
- $\hat{\mathbf{e}}_3$ along the positive orbit normal

A series of 2 successive rotations has to be undergone to get from the inertial reference frame to $\mathcal{F}_e$. The inertial reference frame, $\mathcal{F}_g$, is rotated about a line lying in the $\hat{\mathbf{g}}_1$-$\hat{\mathbf{g}}_2$ plane, from the origin to the ascending node of the orbit. This rotation forms a new reference frame, $\mathcal{F}_f$, with reference vectors, $\hat{\mathbf{f}}_1$, $\hat{\mathbf{f}}_2$, and $\hat{\mathbf{f}}_3$. $\hat{\mathbf{f}}_1$ and $\hat{\mathbf{f}}_2$ lie in the orbital plane, and $\hat{\mathbf{f}}_3$ lies along the positive

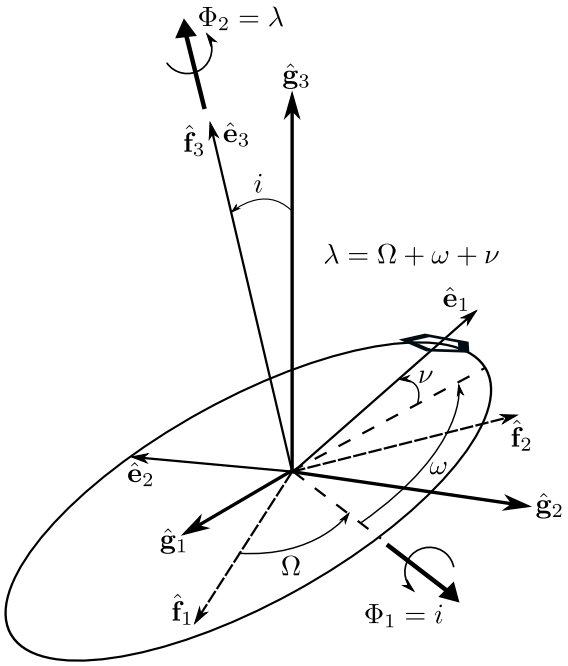


Figure 6.1: The relationship between various reference frames in the USM

orbit normal. The reference frame, $\mathcal{F}_f$ is then rotated along $\hat{\mathbf{f}}_3$ by λ to give $\mathcal{F}_e$. λ is the angle between $\hat{\mathbf{f}}_1$ and the position vector and can be expressed in Keplerian elements as

$$\lambda = \Omega + u \tag{6.1}$$

where u is the argument of latitude, which is the sum of the argument of perigee and the true anomaly.

$$u = \omega + \nu \tag{6.2}$$

All these rotations can be seen in Fig. 6.1, which gives an overview of the various reference frames.

The rotation to get from $\mathcal{F}_g$ to $\mathcal{F}_f$ can be expressed as a rotation with the Euler axis, $\mathbf{a}_1$, and the Euler angle Φ_1. It should be noted that for equatorial orbits, Ω is not defined. The derivation of the USM, and the equations remain the same in that case, but the value of Ω should be set to 0. This Euler axis and angle rotation can be seen in Fig. 6.2. The values of the Euler axis and angle are

$$\mathbf{a}_1 = \begin{bmatrix} \cos\Omega \\ \sin\Omega \\ 0 \end{bmatrix} \tag{6.3}$$

$$\Phi_1 = i \tag{6.4}$$

The rotation matrix of this rotation can be calculated by using Eq. (2.17) found on page 8.

$$\mathbf{C}_{f,g} = \begin{bmatrix} \cos i + \cos^2\Omega(1-\cos i) & \cos\Omega\sin\Omega(1-\cos i) & -\sin\Omega\sin i \\ \cos\Omega\sin\Omega(1-\cos i) & \cos i + \sin^2\Omega(1-\cos i) & \cos\Omega\sin i \\ \sin\Omega\sin i & -\cos\Omega\sin i & \cos i \end{bmatrix} \tag{6.5}$$

The Euler parameters for this rotation, $(\boldsymbol{\epsilon}_1, \eta_1)$, are found by using Eqs. (3.14) and (3.15).

$$\boldsymbol{\epsilon}_1 = \begin{bmatrix} \cos\Omega\sin\frac{i}{2} \\ \sin\Omega\sin\frac{i}{2} \\ 0 \end{bmatrix} \tag{6.6}$$

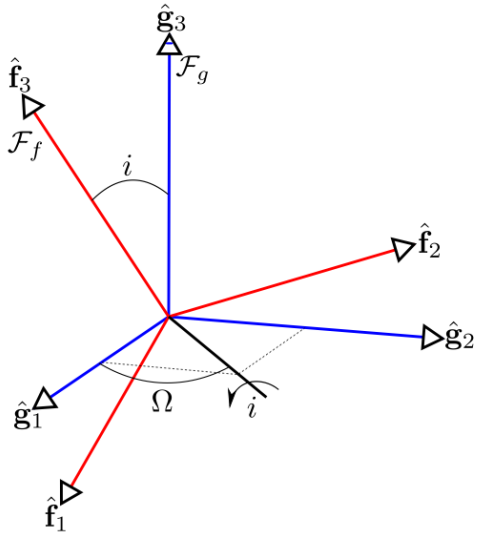


Figure 6.2: The rotation from $\mathcal{F}_g$ (blue) to $\mathcal{F}_f$ (red), with an Euler axis (black) and angle (i) rotation

$$\eta_1 = \cos\frac{i}{2} \tag{6.7}$$

The rotation to get from $\mathcal{F}_f$ to $\mathcal{F}_e$ can be expressed as a rotation with the Euler axis, $\mathbf{a}_2$, and the Euler angle Φ_2. In the case of circular orbits, ω is not defined, but the USM equations still hold with ω being set to 0. If the orbit is circular and equatorial, both Ω and ω are not defined. In that case, the equations of the USM should be used with both those values being set to 0. In this case, the true anomaly will be counted from the vernal equinox. The Euler axis and angle rotation for the transformation from $\mathcal{F}_f$ to $\mathcal{F}_e$ can be seen in Fig. 6.3. The values of the Euler axis and angle are

$$\mathbf{a}_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{6.8}$$

$$\Phi_2 = \lambda \tag{6.9}$$

The Euler parameters for this rotation, $(\boldsymbol{\epsilon}_2, \eta_2)$, are found by again using Eqs. (3.14) and (3.15).

$$\mathbf{C}_{e,f} = \begin{bmatrix} \cos\lambda & \sin\lambda & 0 \\ -\sin\lambda & \cos\lambda & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.10}$$

$$\boldsymbol{\epsilon}_2 = \begin{bmatrix} 0 \\ 0 \\ \sin\frac{\lambda}{2} \end{bmatrix} \tag{6.11}$$

$$\eta_2 = \cos\frac{\lambda}{2} \tag{6.12}$$

The goal now is to find the quaternion, $(\boldsymbol{\epsilon}_3, \eta_3)$, that represents the rotation from $\mathcal{F}_g$ to $\mathcal{F}_e$. For this purpose, Eqs. (3.26) and (3.27) are used.

$$\begin{aligned} \eta_3 &= \eta_1\eta_2 - \boldsymbol{\epsilon}_1^T\boldsymbol{\epsilon}_2 \\ &= \cos\frac{i}{2}\cos\frac{\lambda}{2} \end{aligned}$$

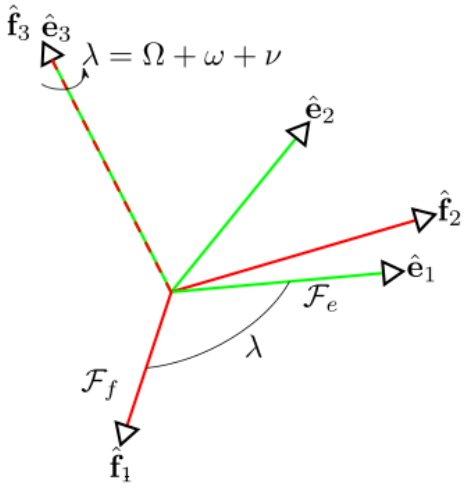


Figure 6.3: The rotation from $\mathcal{F}_f$ (red) to $\mathcal{F}_e$ (green), with an Euler axis $\hat{\mathbf{f}}_3$ and angle (λ) rotation

$$= \cos\frac{i}{2}\cos\frac{\Omega+u}{2} \tag{6.13}$$

$$\begin{aligned}
\boldsymbol{\epsilon}_3 &= \eta_2\boldsymbol{\epsilon}_1 + \eta_1\boldsymbol{\epsilon}_2 + [\boldsymbol{\epsilon}_1\times]\,\boldsymbol{\epsilon}_2 \\
&= \begin{bmatrix} \cos\Omega\sin\frac{i}{2}\cos\frac{\lambda}{2} \\ \sin\Omega\sin\frac{i}{2}\cos\frac{\lambda}{2} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \cos\frac{i}{2}\sin\frac{i}{2} \end{bmatrix} + \begin{bmatrix} \sin\Omega\sin\frac{i}{2}\sin\frac{\lambda}{2} \\ -\cos\Omega\sin\frac{i}{2}\sin\frac{\lambda}{2} \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} \sin\frac{i}{2}\left(\cos\Omega\cos\frac{\lambda}{2} + \sin\Omega\sin\frac{\lambda}{2}\right) \\ \sin\frac{i}{2}\left(\sin\Omega\cos\frac{\lambda}{2} - \cos\Omega\sin\frac{\lambda}{2}\right) \\ \cos\frac{i}{2}\sin\frac{\lambda}{2} \end{bmatrix}
\end{aligned} \tag{6.14}$$

The value found for $\boldsymbol{\epsilon}_3$ in Eq. (6.14) can be simplified further to get to the form found in [Altman, 1972]. For simplification, the sum of angles trigonometric formulae found in Appendix A have to be used. The following simplifications can be made

$$\begin{aligned}
\cos\Omega\cos\frac{\lambda}{2} + \sin\Omega\sin\frac{\lambda}{2} &= \left[\cos^2\frac{\Omega}{2} - \sin^2\frac{\Omega}{2}\right]\left[\cos\frac{\Omega}{2}\cos\frac{u}{2} - \sin\frac{\Omega}{2}\sin\frac{u}{2}\right] \\
&\quad + \left[2\sin\frac{\Omega}{2}\cos\frac{\Omega}{2}\right]\left[\sin\frac{\Omega}{2}\cos\frac{u}{2} + \cos\frac{\Omega}{2}\sin\frac{u}{2}\right] \\
&= \cos\frac{\Omega}{2}\cos\frac{u}{2} + \sin\frac{\Omega}{2}\sin\frac{u}{2} \\
&= \cos\left(\frac{\Omega-u}{2}\right)
\end{aligned}$$

$$\begin{aligned}
\sin\Omega\cos\frac{\lambda}{2} - \cos\Omega\sin\frac{\lambda}{2} &= \left[2\sin\frac{\Omega}{2}\cos\frac{\Omega}{2}\right]\left[\cos\frac{\Omega}{2}\cos\frac{u}{2} - \sin\frac{\Omega}{2}\sin\frac{u}{2}\right] \\
&\quad + \left[\sin^2\frac{\Omega}{2} - \cos^2\frac{\Omega}{2}\right]\left[\sin\frac{\Omega}{2}\cos\frac{u}{2} + \cos\frac{\Omega}{2}\sin\frac{u}{2}\right] \\
&= \sin\frac{\Omega}{2}\cos\frac{u}{2} - \cos\frac{\Omega}{2}\sin\frac{u}{2} \\
&= \sin\left(\frac{\Omega-u}{2}\right)
\end{aligned}$$

The simplified form of Eq. 6.14 can now be written as

$$\boldsymbol{\epsilon}_3 = \begin{bmatrix} \sin\frac{i}{2}\cos\left(\frac{\Omega-u}{2}\right) \\ \sin\frac{i}{2}\sin\left(\frac{\Omega-u}{2}\right) \\ \cos\frac{i}{2}\sin\left(\frac{\Omega+u}{2}\right) \end{bmatrix} \tag{6.15}$$

These Euler parameters describe the orientation of the orbital frame with respect to the inertial frame and are thus given the subscript, O.

$$\begin{bmatrix} \boldsymbol{\epsilon}_O \\ \eta_O \end{bmatrix} = \begin{bmatrix} \sin\frac{i}{2}\cos\left(\frac{\Omega-u}{2}\right) \\ \sin\frac{i}{2}\sin\left(\frac{\Omega-u}{2}\right) \\ \cos\frac{i}{2}\sin\left(\frac{\Omega+u}{2}\right) \\ \cos\frac{i}{2}\cos\left(\frac{\Omega+u}{2}\right) \end{bmatrix} \tag{6.16}$$

6.2 Velocity Components

The four quaternion elements of the USM have already been derived in the previous section. It was previously stated that the velocities, $\mathbf{C}$ and $\mathbf{R}$, are also important parameters of the USM. The vector $\mathbf{C}$ is in the direction of the positive orbital normal and can be expressed in terms of the reference vectors of $\mathcal{F}_e$ as

$$\mathbf{C} = C\hat{\mathbf{e}}_3 \tag{6.17}$$

At any instant in time, the velocity of an orbiting body out of the osculating orbital plane is 0. Thus, the total velocity of the orbiting body is only described by velocities in the orbital plane. $\mathbf{R}$ is already defined in the orbital plane but $\mathbf{C}$ is defined to be out of plane. However, from the definition of the hodograph, the velocity C contributes to the total velocity in the direction perpendicular to the radial vector and in the orbital plane. This corresponds to the direction of $\hat{\mathbf{e}}_2$. To ensure that the defined hodographic velocity vectors give the correct velocity value, the following relations between the constituent unit vectors of a right-handed reference frame are used:

$$\hat{\mathbf{e}}_i = \hat{\mathbf{e}}_j \times \hat{\mathbf{e}}_k \tag{6.18}$$

where $(i,\ j,\ k)$ is a cyclic permutation of (1, 2, 3). So, the contribution of $\mathbf{C}$ to the velocity is $\mathbf{C} \times \mathbf{e}_1$. Therefore, the total velocity is related to the hodographic velocities and the reference vectors of $\mathcal{F}_e$ in the following way

$$\mathbf{v} = \mathbf{R} + \mathbf{C} \times \hat{\mathbf{e}}_1 \tag{6.19}$$

To find the acceleration, the derivative of both sides has to be taken. During the differentiation, it should be noted that $\mathcal{F}_e$ is not an inertial reference frame.

$$\begin{aligned} \dot{\mathbf{v}} = \mathbf{a} &= \dot{\mathbf{R}} + \dot{\mathbf{C}} \times \hat{\mathbf{e}}_1 + \mathbf{C} \times \dot{\hat{\mathbf{e}}}_1 \\ &= \dot{\mathbf{R}} + \dot{\mathbf{C}} \times \hat{\mathbf{e}}_1 + C\hat{\mathbf{e}}_3 \times \boldsymbol{\omega} \times \hat{\mathbf{e}}_1 \end{aligned}$$

The following calculations can then be carried out

$$\begin{aligned} \hat{\mathbf{e}}_3 \times (\boldsymbol{\omega} \times \hat{\mathbf{e}}_1) &= \hat{\mathbf{e}}_3 \times (\omega_3\hat{\mathbf{e}}_2 - \omega_2\hat{\mathbf{e}}_3) \\ &= \omega_3\hat{\mathbf{e}}_3 \times \hat{\mathbf{e}}_2 - \omega_2\hat{\mathbf{e}}_3 \times \hat{\mathbf{e}}_3 \\ &= -\omega_3\hat{\mathbf{e}}_1 \end{aligned} \tag{6.20}$$

Finally, the equation for the acceleration is

$$\mathbf{a} = \dot{\mathbf{R}} + \dot{\mathbf{C}} \times \hat{\mathbf{e}}_1 - C\omega_3\hat{\mathbf{e}}_1 \tag{6.21}$$

where

$\mathbf{a}$ is the acceleration of the orbital body

ω_3 is the angular velocity of $\mathcal{F}_e$ along $\mathbf{e}_3$

The angular velocity, ω_3, was stated to be $\dot{\lambda}$ in [Altman, 1972]. It was pointed out in [Chodas, 1981] that this was wrong and that the correct value of ω_3 should be

$$\omega_3 = \dot{\lambda} + (\cos i - 1)\,\dot{\Omega} \tag{6.22}$$

The value shown in [Chodas, 1981] is the correct one. The value derived in [Altman, 1972] occurs when one considers only the rotation from $\mathcal{F}_f$ to $\mathcal{F}_e$ and not the complete rotation from $\mathcal{F}_g$ to $\mathcal{F}_f$ to $\mathcal{F}_e$. To derive Eq. (6.22), the complete rotation matrix from $\mathbf{C}_{e,g} = \mathbf{C}_{e,f}\mathbf{C}_{f,g}$ has to be calculated. $\boldsymbol{\omega}$ can then be calculated from the components of the rotation matrix. This is Eq. (2.36) found on page 11.

$$\boldsymbol{\omega} = \begin{bmatrix} \dot{c}_{2,1}c_{3,1} + \dot{c}_{2,2}c_{3,2} + \dot{c}_{2,3}c_{3,3} \\ \dot{c}_{3,1}c_{1,1} + \dot{c}_{3,2}c_{1,2} + \dot{c}_{3,3}c_{1,3} \\ \dot{c}_{1,1}c_{2,1} + \dot{c}_{1,2}c_{2,2} + \dot{c}_{1,3}c_{2,3} \end{bmatrix}$$

ω_3 is then the third component, which is

$$\omega_3 = \dot{c}_{1,1}c_{2,1} + \dot{c}_{1,2}c_{2,2} + \dot{c}_{1,3}c_{2,3} \tag{6.23}$$

This calculation was carried out by the author to check and confirm that Eq. (6.22) indeed is correct. The method outlined above is a very lengthy process. Fortunately, there is a simpler method to get an expression for ω_3. The only angular quantity in the orbital plane, perpendicular to $\hat{\mathbf{e}}_3$, is λ. Thus, it would be logical to assume that ω_3 is equal to $\dot{\lambda}$ and this was the result found in [Altman, 1972]. However, it is important to note that even though $\dot{\Omega}$ acts along $\hat{\mathbf{g}}_3$, it will have a projection and therefore an influence along $\hat{\mathbf{e}}_3$. The projection of one vector onto another is its product with the cosine of the angle between the two vectors. The angle between $\hat{\mathbf{g}}_3$ and $\hat{\mathbf{e}}_3$ is i, thus the angular velocity contribution of $\dot{\Omega}$ to ω_3 is $\dot{\Omega}\cos i$.

The velocity components that the USM uses are C, R_{f1}, and R_{f2}. R_{f1} and R_{f2} are the components of R along $\hat{\mathbf{f}}_1$ and $\hat{\mathbf{f}}_2$ respectively. R is defined to lie in the orbital plane and so $R_{f3} = 0$.The velocity of the orbiting body can now be written in $\mathcal{F}_e$. It is clear from Eq. (6.19) that C only contributes to the velocity component along $\hat{\mathbf{e}}_2$. Thus, only $\mathbf{R}$ contributes to v_{e1}. Using simple trigonometry, the velocity expressed in $\mathcal{F}_e$ is

$$v_{e1} = R_{f1}\cos\lambda + R_{f2}\sin\lambda \tag{6.24}$$

$$v_{e2} = C - R_{f1}\sin\lambda + R_{f2}\cos\lambda \tag{6.25}$$

$$v_{e3} = 0 \tag{6.26}$$

v_{e3} is 0 because the orbiting body has no out of plane velocity. The velocity can be expressed in canonical form as

$$\begin{bmatrix} v_{e1} \\ v_{e2} \end{bmatrix} = \begin{bmatrix} 0 \\ C \end{bmatrix} + \begin{bmatrix} \cos\lambda & \sin\lambda \\ -\sin\lambda & \cos\lambda \end{bmatrix} \begin{bmatrix} R_{f1} \\ R_{f2} \end{bmatrix} \tag{6.27}$$

Eq. (6.27) can easily be inverted to solve for R_{f1} and R_{f2}.

$$\begin{bmatrix} R_{f1} \\ R_{f2} \end{bmatrix} = \begin{bmatrix} \cos\lambda \\ \sin\lambda \end{bmatrix} v_{e1} + \begin{bmatrix} -\sin\lambda \\ \cos\lambda \end{bmatrix} v_{e2} + \begin{bmatrix} \sin\lambda \\ -\cos\lambda \end{bmatrix} C \tag{6.28}$$

As can be seen in Eqs. (6.27) and (6.28), it is necessary to be able to calculate the value of $\sin\lambda$ and $\cos\lambda$. This can be extracted from the quaternion. A few of the quaternion elements in Eq. (6.16) on page 51 can be rewritten as

$$\epsilon_{O3} = \cos\frac{i}{2}\sin\frac{\lambda}{2} \tag{6.29}$$

$$\eta_O = \cos\frac{i}{2}\cos\frac{\lambda}{2} \tag{6.30}$$

This can be rearranged to give

$$\sin\frac{\lambda}{2} = \frac{\epsilon_{O3}}{\cos\frac{i}{2}}$$
$$\cos\frac{\lambda}{2} = \frac{\eta_O}{\cos\frac{i}{2}}$$

Some basic trigonometric equations and the half-angle formulae found in Appendix A can be used to get the following intermediate results

$$\tan\frac{\lambda}{2} = \frac{\sin\frac{\lambda}{2}}{\cos\frac{\lambda}{2}} = \frac{\epsilon_{O3}}{\cos\frac{i}{2}}\frac{\cos\frac{i}{2}}{\eta_O} = \frac{\epsilon_{O3}}{\eta_O}$$

$$\tan\lambda = \frac{2\tan\frac{\lambda}{2}}{1-\tan^2\frac{\lambda}{2}} = \frac{2\frac{\epsilon_{O3}}{\eta_O}}{1-\left(\frac{\epsilon_{O3}}{\eta_O}\right)^2} = \frac{2\epsilon_{O3}\eta_O}{(\eta_O^2-\epsilon_{O4}^2)}$$

$$\cos^2\frac{\lambda}{2} = \frac{1}{2}(1+\cos\lambda) = \left(\frac{\eta_O}{\cos\frac{i}{2}}\right)^2$$

$$\sin^2\frac{\lambda}{2} = \frac{1}{2}(1-\cos\lambda) = \left(\frac{\epsilon_{O3}}{\cos\frac{i}{2}}\right)^2$$

$$\tan^2\frac{\lambda}{2} = \frac{\sin^2\frac{\lambda}{2}}{\cos^2\frac{\lambda}{2}} = \left(\frac{\epsilon_{O3}}{\eta_O}\right)^2 = \frac{1-\cos\lambda}{1+\cos\lambda}$$

Solving for $\cos\lambda$ gives

$$\cos\lambda = \frac{\eta_O^2-\epsilon_{O3}^2}{\epsilon_{O3}^2+\eta_O^2} \tag{6.31}$$

$$\sin\lambda = \tan\lambda\cos\lambda = \frac{2\epsilon_{O3}\eta_O}{(\eta_O^2-\epsilon_{O4}^2)}\frac{\eta_O^2-\epsilon_{O3}^2}{\epsilon_{O3}^2+\eta_O^2}$$

$$\sin\lambda = \frac{2\epsilon_{O3}\eta_O}{\epsilon_{O3}^2+\eta_O^2} \tag{6.32}$$

The sine and cosine can together be expressed as

$$\begin{bmatrix}\sin\lambda\\ \cos\lambda\end{bmatrix} = \frac{1}{\epsilon_{O3}^2+\eta_O^2}\begin{bmatrix}2\epsilon_{O3}\eta_O\\ \eta_O^2-\epsilon_{O3}^2\end{bmatrix} \tag{6.33}$$

6.3 Kinematics and Dynamics

To be able to utilize the USM, the kinematics and dynamics have to be derived as well. The dynamic equations are the time derivative of the USM parameters and can be split into two parts. The first part of the dynamics equation deals with the time rate of change of the three velocity parameters, C, R_{f1}, and R_{f2}. The second part of the dynamics equation deals with the time rate of change of the Euler parameters.

It is useful to first find the vector from the center of the attracting body to the orbiting body. This is the radius vector $\mathbf{r}$, and according to the definition of $\mathcal{F}_e$, it lies in the direction of $\hat{\mathbf{e}}_1$.

$$\mathbf{r} = r\hat{\mathbf{e}}_1 \tag{6.34}$$

In Eq. (6.34), r is the magnitude of the radius vector. This parameter can also be derived from the given information. The angular momentum, $\mathbf{h}$, lies along $\hat{\mathbf{e}}_3$, and the orbiting body has no velocity component along $\hat{\mathbf{e}}_3$. Using this information, the following derivation can be carried out in $\mathcal{F}_e$

$$\begin{aligned} h\hat{\mathbf{e}}_3 &= r\hat{\mathbf{e}}_1 \times (v_{e1}\hat{\mathbf{e}}_1 + v_{e2}\hat{\mathbf{e}}_2) \\ &= rv_{e2}\hat{\mathbf{e}}_3 \iff \end{aligned}$$

$$r = h/v_{e2} = \mu/(Cv_{e2})$$

Thus, $\mathbf{r}$ is

$$\mathbf{r} = \mu/(Cv_{e2})\hat{\mathbf{e}}_1 \tag{6.35}$$

The acceleration that the orbiting body experiences can be split up into the central gravitational attraction, $\mathbf{a}_g$, and the perturbing acceleration, $\mathbf{a}_e$. This perturbing acceleration is expressed in $\mathcal{F}_e$. The acceleration due to uniform gravity is

$$\mathbf{a}_g = -\mu/r^3\mathbf{r} = -\mu/r^2\hat{\mathbf{e}}_1 = -C^2v_{e2}^2/\mu\hat{\mathbf{e}}_1 \tag{6.36}$$

and $\mathbf{a}_e$ is defined as

$$\mathbf{a}_e = a_{e1}\hat{\mathbf{e}}_1 + a_{e2}\hat{\mathbf{e}}_2 + a_{e3}\hat{\mathbf{e}}_3 \tag{6.37}$$

The total acceleration is

$$\mathbf{a} = \left(a_{e1} - C^2v_{e2}^2/\mu\right)\hat{\mathbf{e}}_1 + a_{e2}\hat{\mathbf{e}}_2 + a_{e3}\hat{\mathbf{e}}_3 \tag{6.38}$$

For the kinematics and dynamics, expressions for the time derivative of the quaternion and the hodograph velocities, respectively, have to be found. The quaternion time derivative is simply the kinematic relation found in Eq. (3.34) with the only unknown quantities being the angular velocities expressed in $\mathcal{F}_e$. It is important to note that the velocity in the inertial frame $\mathcal{F}_g$ always has components along $\hat{\mathbf{e}}_1$ and $\hat{\mathbf{e}}_2$. However, the frame $\mathcal{F}_e$ is a rotating frame. Thus, there will be apparent forces that have to be taken into account when taking the time derivatives.

We begin the derivation by taking the time derivative of the velocity in $\mathcal{F}_e$. Since $v_{e3} = 0$, the velocity of the orbiting body is

$$\mathbf{v} = v_{e1}\hat{\mathbf{e}}_1 + v_{e2}\hat{\mathbf{e}}_2 + 0\hat{\mathbf{e}}_3 \tag{6.39}$$

The expression for the velocity found in Eq. (6.39) holds for every point in the orbit of the satellite. Because the frame is non-inertial, the direction of $\hat{\mathbf{e}}_1$ and $\hat{\mathbf{e}}_2$ in the inertial reference frame changes along with the value of the scalars v_{e1} and v_{e2}. Therefore, the time derivative of Eq. (6.39) should also take into account the time derivate of the unit vectors.

$$\dot{\mathbf{v}} = (\dot{v}_{e1} - v_{e2}\omega_3)\,\hat{\mathbf{e}}_1 + (\dot{v}_{e2} + v_{e1}\omega_3)\,\hat{\mathbf{e}}_2 + (\dot{v}_{e3} + v_{e2}\omega_1 - v_{e1}\omega_2)\,\hat{\mathbf{e}}_3 \tag{6.40}$$

Knowing that the time derivative of the velocity in Eq. (6.40) has to equal the total acceleration in Eq. (6.38), the scalar velocity time derivative components can be found to be

$$\dot{v}_{e1} = a_{e1} - (C^2v_{e2}^2/\mu) + v_{e2}\omega_3 \tag{6.41}$$

$$\dot{v}_{e2} = a_{e2} - v_{e1}\omega_3 \tag{6.42}$$

$$\dot{v}_{e3} = a_{e3} - v_{e2}\omega_1 + v_{e1}\omega_2 \tag{6.43}$$

Similarly, the position vector from Eq. (6.35) can also be differentiated with respect to time.

$$\begin{aligned} \dot{\mathbf{r}} = &-\mu\left(\dot{C}/\left(C^2v_{e2}\right) + \dot{v}_{e2}/\left(Cv_{e2}^2\right)\right)\hat{\mathbf{e}}_1 \\ &+ \mu\Big(\omega_3/\left(Cv_{e2}\right)\Big)\hat{\mathbf{e}}_2 - \mu\Big(\omega_2/\left(Cv_{e2}\right)\Big)\hat{\mathbf{e}}_3 \end{aligned} \tag{6.44}$$

The derivative of the position is the velocity and by using the fact that Eq. (6.44) is equal to Eq. (6.39), the values of certain unknown parameters can be found.

$$\omega_2 = 0 \tag{6.45}$$
$$\omega_3 = Cv_{e2}^2/\mu \tag{6.46}$$
$$\dot{C} = -Ca_{e2}/(v_{e2}) \tag{6.47}$$

To have the same notation as [Altman, 1972], a new parameter p is defined:

$$p = C/v_{e2} \tag{6.48}$$

This results in $\dot{C}$ being

$$\dot{C} = -pa_{e2} \tag{6.49}$$

For ω_1, the time derivative of the angular momentum, also known as the torque, is used. ω_1 is only found in the time derivative of the velocity, which occurs in the torque, but not in the angular momentum itself. It is known that the angular momentum acts along $\hat{\mathbf{e}}_3$ and the magnitude can be found from C.

$$\mathbf{h} = (\mu/C)\hat{\mathbf{e}}_3 \tag{6.50}$$

One way to find the torque, $\dot{\mathbf{h}}$, is by differentiating Eq. (6.50) with respect to time. The unit vector $\hat{\mathbf{e}}_3$ remains fixed for unperturbed orbits because the orbital plane does not change its orientation. However, the general case with perturbations should be considered here and thus the time derivative of $\hat{\mathbf{e}}_3$ also has to be taken. The resulting expression for the torque is

$$\dot{\mathbf{h}} = 0\hat{\mathbf{e}}_1 - \Big(\mu\omega_1/C\Big)\hat{\mathbf{e}}_2 + \Big(\mu a_{e2}/(Cv_{e2})\Big)\hat{\mathbf{e}}_3 \tag{6.51}$$

Another way to express the torque is to differentiate the definition of the angular momentum:

$$\begin{aligned}\dot{\mathbf{h}} &= \frac{\mathrm{d}}{\mathrm{d}t}(\mathbf{r}\times\mathbf{v}) \\ &= 0\hat{\mathbf{e}}_1 - \Big(\mu a_{e3}/(Cv_{e2})\Big)\hat{\mathbf{e}}_2 + \Big(\mu a_{e2}/(Cv_{e2})\Big)\hat{\mathbf{e}}_3\end{aligned} \tag{6.52}$$

The two different expressions found in Eq. (6.51) and Eq. (6.52) both represent the torque and thus, they should be equivalent. Equating Eq. (6.51) and Eq. (6.52) gives then the value for ω_1:

$$\omega_1 = a_{e3}/v_{e2} \tag{6.53}$$

All the angular velocities of $\mathcal{F}_e$ are now known, and thus the time derivative of the Euler parameters can be shown using Eq. (3.34) on page 20 to be

$$\boxed{\begin{bmatrix}\dot{\varepsilon}_{O1}\\ \dot{\varepsilon}_{O2}\\ \dot{\varepsilon}_{O3}\\ \dot{\eta}_O\end{bmatrix} = \frac{1}{2}\begin{bmatrix}0 & \omega_3 & 0 & \omega_1\\ -\omega_3 & 0 & \omega_1 & 0\\ 0 & -\omega_1 & 0 & \omega_3\\ -\omega_1 & 0 & -\omega_3 & 0\end{bmatrix}\begin{bmatrix}\varepsilon_{O1}\\ \varepsilon_{O2}\\ \varepsilon_{O3}\\ \eta_O\end{bmatrix}} \tag{6.54}$$

However, the dynamics of the two remaining USM elements has to still be derived, i.e., $\dot{R}_{f1}$ and $\dot{R}_{f2}$. First, the time derivative of Eq. (6.28) has to be taken.

$$\dot{R}_{f1} = \dot{v}_{e1}\cos\lambda + \Big(\dot{C} - \dot{v}_{e2}\Big)\sin\lambda - \dot{\lambda}R_{f2} \tag{6.55}$$

$$\dot{R}_{f2} = \dot{v}_{e1}\sin\lambda + \Big(\dot{v}_{e2} - \dot{C}\Big)\cos\lambda + \dot{\lambda}R_{f1} \tag{6.56}$$

In Eq. (6.55) and Eq. (6.56), all parameters except for $\dot{\lambda}$ are known. The cosine and sine of λ are known and are based on the quaternion elements. The time derivative of the quaternion is known from Eq. (6.54) and so, the time derivatives of the cosine and sine of λ can be computed. It is only necessary to compute the time derivative in terms of USM elements of one of the trigonometric functions. For example, if the time derivative of $\sin\lambda$ is computed, $\dot{\lambda}$ can be computed using the reversed chain rule in the following way

$$\dot{\lambda} = \frac{1}{\cos\lambda}\frac{\mathrm{d}}{\mathrm{d}t}(\sin\lambda) \tag{6.57}$$

In Eq. (6.57), the time derivative of $\sin\lambda$ can be computed, using the fact that $\sin\lambda$ depends on on ϵ_{O3} and η, in the following manner:

$$\frac{\mathrm{d}}{\mathrm{d}t}(\sin\lambda) = \frac{\partial\sin\lambda}{\partial\epsilon_{O3}}\dot{\epsilon}_{O3} + \frac{\partial\sin\lambda}{\partial\eta_O}\dot{\eta}_O \tag{6.58}$$

The partials of $\sin\lambda$ with respect to ϵ_{O3} and η_O can also be found in [Chodas, 1981], where they are used to find the Jacobians for the EKF. According to [Chodas, 1981], they are

$$\frac{\partial\sin\lambda}{\partial\epsilon_{O3}} = 2\eta_O\left(\eta_O^2 - \epsilon_{O3}^2\right)/\left(\epsilon_{O3}^2 + \eta_O^2\right)^2 \tag{6.59a}$$

$$\frac{\partial\sin\lambda}{\partial\eta_O} = -2\epsilon_{O3}\left(\eta_O^2 - \epsilon_{O3}^2\right)/\left(\epsilon_{O3}^2 + \eta_O^2\right)^2 \tag{6.59b}$$

The time derivates of the two required quaternion elements are

$$\dot{\epsilon}_{O3} = \frac{1}{2}\left(-\omega_1\epsilon_{O2} + \omega_3\eta_O\right) = -\frac{1}{2}\frac{a_{e3}\epsilon_{O2}}{v_{e2}} + \frac{1}{2}\frac{Cv_{e2}^2\eta_O}{\mu} \tag{6.60a}$$

$$\dot{\eta}_O = \frac{1}{2}\left(-\omega_1\epsilon_{O1} - \omega_3\epsilon_{O3}\right) = -\frac{1}{2}\frac{a_{e3}\epsilon_{O1}}{v_{e2}} - \frac{1}{2}\frac{Cv_{e2}^2\epsilon_{O3}}{\mu} \tag{6.60b}$$

Filling in the partials from Eq. (6.59), the time derivatives of the individual quaternion elements found in Eq. (6.60), and $\cos\lambda$ into Eq. (6.57) followed by some algebraic manipulation leads to

$$\dot{\lambda} = \frac{\epsilon_{O1}\epsilon_{O3} - \epsilon_{O2}\eta_O}{\epsilon_{O3}^2 + \eta_O^2}\frac{a_{e3}}{v_{e2}} + \omega_3 \tag{6.61}$$

For ease of notation and to be consistent with the equations found in [Chodas, 1981], a new parameter is introduced.

$$\gamma = \frac{\epsilon_{O1}\epsilon_{O3} - \epsilon_{O2}\eta_O}{\epsilon_{O3}^2 + \eta_O^2} \tag{6.62}$$

Finally, after substitution of Eq. (6.61) and Eq. (6.62) into Eq. (6.55) and Eq. (6.56), the time derivatives of R_{f1} and R_{f2} can be written as

$$\dot{R}_{f1} = a_{e1}\cos\lambda - a_{e2}(1+p)\sin\lambda - a_{e3}\left(\gamma R_{f2}/v_{e2}\right) \tag{6.63}$$

$$\dot{R}_{f2} = a_{e1}\sin\lambda + a_{e2}\left(1+p\right)\cos\lambda + a_{e3}\left(\gamma R_{f1}/v_{e2}\right) \tag{6.64}$$

The dynamics of the velocity parameters of the USM can be expressed in a canonical form as

$$\boxed{\frac{d}{dt}\begin{bmatrix} C \\ R_{f1} \\ R_{f2} \end{bmatrix} = \begin{bmatrix} 0 & -p & 0 \\ \cos\lambda & -(1+p)\sin\lambda & -\gamma R_{f2}/v_{e2} \\ \sin\lambda & (1+p)\cos\lambda & \gamma R_{f1}/v_{e2} \end{bmatrix}\begin{bmatrix} a_{e1} \\ a_{e2} \\ a_{e3} \end{bmatrix}} \tag{6.65}$$

The original model found in [Altman, 1972], as mentioned previously, wrongly assumes that $\dot{\lambda} = \omega_3$. This erroneous assumption removes the dependency of the hodograph parameters on the out-of-plane perturbations. This error was pointed out in [Chodas, 1981] and the correct equations were also shown in [Raol and Sinha, 1985]. The dependance of $\dot{\lambda}$ on a_{e3} follows from the fact that $\dot{\lambda}$ depends on $\dot{\epsilon}_{O3}$ and $\dot{\eta}_O$, which both depend on ω_1, which ultimately depends on a_{e3}. It was also seen in Eq. (6.22) that $\dot{\lambda}$ depends on $\dot{\Omega}$. A changing Ω means that the orientation of the orbital plane changes. It can be seen in any book on astrodynamics that an out-of-plane force is required to change the out-of-plane Keplerian elements. Thus, it can be deduced that $\dot{\lambda}$ has to depend on a_{e3}, which is the only out-of-plane acceleration.

6.4 Reflections on the Unified State Model

In this chapter, the Unified State Model has been derived. The USM has some attractive properties that make it suitable for numerical integration. The fact that the Euler parameters have a unit norm makes for a convenient way to check for numerical errors during integration. During numerical integration, it is suggested to normalize the quaternion according to the method found in Appendix C.4. Three of the seven state elements vary only in the presence of perturbations, which is similar to Encke's method of integration. This also means that during any orbit, only 4 of the 7 state elements vary rapidly. Also, the 4 rapidly varying elements are bounded between -1 and 1. It is also possible to scale the velocity parameters by dividing by C_0, which is the initial value of C. This will ensure that these parameters also stay in the unit region. It is realistic for most simulations that during the orbit C, R_{f1}, and R_{f2} do not increase by an order of magnitude. In case the attitude of the spacecraft is also simulated, the usage of Euler parameters for all 6 degrees of freedom may decrease the computation load.

Even though the Euler parameters are free from singularities, the USM is unfortunately not. There are 2 types of orbits where the USM breaks down. This can be seen in the mathematics whenever the denominator of a fraction is 0. There are 3 denominators, which are:

Case 1 $\epsilon_{O3}^2 + \eta_O^2$ from Eq. (6.33)

Case 2 h from the definition of C

Case 3 v_{e2}, present in many equations

Case 2, when $h = 0$, is a more generalized version of the case, $v_{e2} = 0$. They both signify that the motion is rectilinear. This is not the case for normal trajectories in celestial mechanics. This might be the case for hyperbolic orbits when the true anomaly limit is reached. At the true anomaly limit the orbit degenerates into a rectilinear orbit. In the absence of perturbing accelerations, this would cause both ω_1 and ω_2 to become 0. Thus, the Euler parameters and the velocity parameters of the USM would remain constant and the state would be undeterminable, but without a singularity. This is logical because at this point the spacecraft has escaped from the central gravity field and the underlying assumption of the USM is motion in a central gravity field, albeit perturbed. This will not be a problem for orbital work because, using the notion of sphere of influence, the orbit should be out of the sphere of influence of the planet, and in orbit around the Sun. Escapes from the Solar System are not cases that occur very often for artificial spacecraft. The rectilinear limit might also cause a problem for re-entry vehicles. According to [Altman, 1972], the USM should be able to handle these as well. This will also be investigated further in the thesis study in Chapter 9.

The other singularity occurs when

$$\epsilon_{O3}^2 + \eta_O^2 = 0 \tag{6.66}$$

Using the condition found in Eq. 6.66 and the condition of unit norm for Euler parameters results in

$$\epsilon_{O1}^2 + \epsilon_{O2}^2 = 1 \tag{6.67}$$

It can be seen that this relation can be inserted directly in Eq. (7.54) for the inclination on page 69. This results in

$$i = \arccos(-1) = 180^\circ \tag{6.68}$$

Another way of deducing this is to use the definition of the Euler parameters in terms of the Keplerian elements.

$$\epsilon_{O3}^2 + \eta_O^2 = \cos^2\left(\frac{i}{2}\right) = 0 \tag{6.69}$$

The half angle identity from Eq. A.8 can be used to find that i is 180°. The reason for the singularity can be found in the rotation matrix from $\mathcal{F}_g$ to $\mathcal{F}_f$, $\mathbf{C}_{f,g}$, found in Eq. 6.5 on page 48:

$$\mathbf{C}_{f,g} = \begin{bmatrix} \cos i + \cos^2\Omega(1-\cos i) & \cos\Omega\sin\Omega(1-\cos i) & -\sin\Omega\sin i \\ \cos\Omega\sin\Omega(1-\cos i) & \cos i + \sin^2\Omega(1-\cos i) & \cos\Omega\sin i \\ \sin\Omega\sin i & -\cos i\sin i & \cos i \end{bmatrix}$$

When the case of $i = 180°$ is filled in, the resulting $\mathbf{C}_{f,g}$ is

$$\mathbf{C}_{f,g} = \begin{bmatrix} 2\cos^2\Omega - 1 & 2\cos\Omega\sin\Omega & 0 \\ 2\cos\Omega\sin\Omega & 2\sin^2\Omega - 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{6.70}$$

As can be seen in Eq. 6.70, different values of Ω result in different orientations of $\mathcal{F}_f$. This is obviously not the case in reality and thus, the singularity. According to [Chodas, 1981], this singularity can be removed by defining an alternate $\mathcal{F}_f$, which is based on a rotation about the descending node. This translates mathematically into a rotation from $\mathcal{F}_g$ to $\mathcal{F}_f$ using the Euler axis and angle

$$\mathbf{a}_1 = \begin{bmatrix} \cos(\Omega+\pi) \\ \sin(\Omega+\pi) \\ 0 \end{bmatrix} = -\begin{bmatrix} \cos\Omega \\ \sin\Omega \\ 0 \end{bmatrix} \tag{6.71a}$$

$$\Phi_1 = i \tag{6.71b}$$

The singularity of the pure-retrograde orbit is also seen in Eq. (7.39) on page 65, which is the method used in [Altman, 1972] and [Chodas, 1981] to extract the Euler parameters from Cartesian coordinates. In Eq. (7.39), all the Euler parameters have $h(h + h_z)$ in their denominators. This means that there is a singularity if $h = 0$, corresponding to a rectilinear orbit, or if $h + h_z = 0$. When $h + h_z = 0$, $h_z = -h$, which means that the angular momentum is in the direction of the $-z$-axis in $\mathcal{F}_g$, corresponding to a pure-retrograde orbit.

This concludes the derivation of the USM, which is the main focus of this report. It has been shown that there was indeed an erroneous assumption made in [Altman, 1972], which was successfully corrected in [Chodas, 1981]. In the aforementioned works, only the final equations of USM were given. In this chapter, each parameter was successfully defined and derived. Thus, any reader with a limited knowledge of rotational dynamics and celestial mechanics can fully comprehend the intricacies of this model. Hopefully, the model has been demystified, and one of the goals of this thesis study satisfied.

6.5 Unified State Model using Modified Rodrigues Parameters

The USM has 7 elements, of which 4 are the elements of a quaternion. Since the quaternion is used to describe the orientation of the orbital frame, any other set of attitude parameters should also suffice. According to [Schaub and Junkins, 2002], any attitude parameter set that has fewer than 4 elements will have a singularity. MRP, however, have the benefit of having shadow parameters that allow bypassing this singularity. Thus, a method is proposed here of using MRP to reduce the number of elements of the USM theory to 6. Since the classical USM has 7 elements, it will be referred to as USM7 from here on, and the USM using MRP will be referred to as USM6. The properties of MRP and their relationship with quaternions can be found in Section 3.3.

The derivation is the same as in the case of quaternions. Since MRP are a direct mapping of the quaternion, the equations of USM derived with quaternions will simply be converted to MRP

and SMRP. The conversion from quaternion to MRP and SMRP and vice versa is done using Eq. (3.37) and Eq. (3.42).

It is now possible to convert the quaternion of the Orbital frame to an MRP and SMRP vector. In terms of MRP it would be

$$\boldsymbol{\sigma}_O = \left(1 + \cos\left(\frac{i}{2}\right)\cos\left(\frac{\Omega+u}{2}\right)\right)^{-1} \begin{bmatrix} \sin\left(\frac{i}{2}\right)\cos\left(\frac{\Omega-u}{2}\right) \\ \sin\left(\frac{i}{2}\right)\sin\left(\frac{\Omega-u}{2}\right) \\ \cos\left(\frac{i}{2}\right)\sin\left(\frac{\Omega+u}{2}\right) \end{bmatrix} \tag{6.72}$$

The SMRP would then be

$$\boldsymbol{\sigma}_O^S = \left(\cos\left(\frac{i}{2}\right)\cos\left(\frac{\Omega+u}{2}\right) - 1\right)^{-1} \begin{bmatrix} \sin\left(\frac{i}{2}\right)\cos\left(\frac{\Omega-u}{2}\right) \\ \sin\left(\frac{i}{2}\right)\sin\left(\frac{\Omega-u}{2}\right) \\ \cos\left(\frac{i}{2}\right)\sin\left(\frac{\Omega+u}{2}\right) \end{bmatrix} \tag{6.73}$$

The rest of the equations will be the same, because they represent the dynamics in the orbital frame, which has the same orientation expressed in quaternions and MRP. The only differences will be the components that are computed from the quaternions. These are: λ, γ, and the kinematic differential equation. For the sine and cosine of λ, the quaternion elements in Eq. (6.33) should be substituted with the MRP elements found in Eq. (3.40). For the kinematic differential equation, Eq. (3.48) should be used. However, all the elements corresponding to ω_2 can be removed because ω_2 is known to be 0 for the USM. In terms of MRP, these equations become

$$\begin{bmatrix} \sin\lambda \\ \cos\lambda \end{bmatrix} = \frac{1}{4\sigma_3^2 + \left(1-\sigma^2\right)^2} \begin{bmatrix} 4\sigma_3\left(1-\sigma^2\right) \\ \left(1-\sigma^2\right)^2 - 4\sigma_3^2 \end{bmatrix} \tag{6.74}$$

$$\dot{\boldsymbol{\sigma}} = \frac{1}{4} \begin{bmatrix} \left(1-\sigma^2+2\sigma_1^2\right)\omega_1 + 2(\sigma_1\sigma_3+\sigma_2)\omega_3 \\ 2(\sigma_2\sigma_1+\sigma_3)\omega_1 + 2(\sigma_2\sigma_3-\sigma_1)\omega_3 \\ 2(\sigma_3\sigma_1-\sigma_2)\omega_1 + \left(1-\sigma^2+2\sigma_3^2\right)\omega_3 \end{bmatrix} \tag{6.75}$$

The kinematic differential equation can be written in a more compact form as found in Eq. (3.49)

The parameter γ is required for the dynamics of R_{f1} and R_{f2}. In terms of MRP, the time derivatives of R_{f1} and R_{f2} do not have the same simple form as found for the quaternion case. Thus, it is not possible to simply substitute the quaternion elements with the MRP elements to find γ. The time derivatives of R_{f1} and R_{f2} are derived from Eq. (6.55) and Eq. (6.56). The terms $\dot{C}$, $\dot{v}_{e1}$, and $\dot{v}_{e2}$ can be calculated from Eq. (6.49), Eq. (6.41), and Eq. (6.42), respectively. It is, however, still required to calculate the time derivative of λ. The procedure of computing $\dot{\lambda}$ is the same as for USM7 and can be carried out in the following way:

$$\dot{\lambda} = \frac{1}{\cos\lambda}\left(\frac{\partial \sin\lambda}{\partial\sigma_1}\dot{\sigma_1} + \frac{\partial \sin\lambda}{\partial\sigma_2}\dot{\sigma_2} + \frac{\partial \sin\lambda}{\partial\sigma_3}\dot{\sigma_3}\right) \tag{6.76}$$

Because the magnitude of the MRP or SMRP vector is required to find $\sin\lambda$ and $\cos\lambda$, the partials depend on all three elements of the MRP/SMRP vector. The partials required in Eq. (6.76) can be found by differentiation of Eq. (6.74).

$$\frac{\partial \sin\lambda}{\partial\sigma_1} = -\frac{8\sigma_1\sigma_3}{4\sigma_3^2 + \left(1-\sigma^2\right)^2} + \frac{16\sigma_1\sigma_3\left(1-\sigma^2\right)^2}{\left(4\sigma_3^2 + \left(1-\sigma^2\right)^2\right)^2} \tag{6.77a}$$

$$\frac{\partial \sin\lambda}{\partial\sigma_2} = -\frac{8\sigma_2\sigma_3}{4\sigma_3^2 + \left(1-\sigma^2\right)^2} + \frac{16\sigma_2\sigma_3\left(1-\sigma^2\right)^2}{\left(4\sigma_3^2 + \left(1-\sigma^2\right)^2\right)^2} \tag{6.77b}$$

$$\frac{\partial \sin\lambda}{\partial\sigma_3} = \frac{4\left(1-\sigma^2\right)}{4\sigma_3^2 + \left(1-\sigma^2\right)^2} - \frac{8\sigma_3^2}{4\sigma_3^2 + \left(1-\sigma^2\right)^2} - \frac{4\sigma_3\left(1-\sigma^2\right)^2\left(8\sigma_3 - 4\left(1-\sigma^2\right)\sigma_3\right)}{\left(4\sigma_3^2 + \left(1-\sigma^2\right)^2\right)^2} \tag{6.77c}$$

The resulting equation for $\dot{\lambda}$ is very long and is therefore not presented. The equations presented above for the MRP also hold for the SMRP, because both the time derivative equations and the equations for the sine and cosine of λ are the same in terms of MRP and SMRP.

It is not enough to simply have the equations for the USM. The main goal of an engineer is to be able to implement the model. The additional information required to fully use the USM like the transformations to and from Cartesian coordinates, etc. are presented in the next chapter.

Chapter 7

Guide for Applying the Unified State Model

In Chapter 6, the traditional USM, USM7, and the alternative using MRP, USM6 have been derived. However, more information about the model is required to be able to fully utilize it. The initial position and velocity of a satellite is most often expressed in classical Keplerian elements or in Cartesian coordinates. Thus, the conversions between the USM and these other models need to be known. The conversion between Keplerian elements and Cartesian coordinates was presented in Section 4.6 and therefore, will not be presented here. In this chapter, only the transformations to and from USM7 will be shown. For the conversions to USM6, it is simplest to use the conversions to USM7 and then to extract the MRP vector from the quaternion. To find the inverse relations, it also best to convert the MRP vector to a quaternion and then use the USM7 conversions.

7.1 Conversion to the USM7 Elements

The conversion from both the Keplerian elements and the Cartesian coordinates to the USM elements is shown here.

7.1.1 Conversion from Classic Keplerian Elements to USM7 Elements

The classic Keplerian elements that are used here are (Ω,i,ω,ν,a,e). The first step is to extract the velocity parameters C and R. C can be found by using Eqs. (4.63) and (5.8).

$$C = \frac{\mu}{h} = \sqrt{\frac{\mu}{a\left(1-e^2\right)}} \tag{7.1}$$

R can be found by using Eqs. (7.1) and (5.9).

$$R = eC = e\sqrt{\frac{\mu}{a\left(1-e^2\right)}} \tag{7.2}$$

The quaternion elements can be calculated using Eq. (6.16), which is repeated here for the sake of completeness.

$$\begin{bmatrix} \boldsymbol{\epsilon}_O \\ \eta_O \end{bmatrix} = \begin{bmatrix} \sin\frac{i}{2}\cos\left(\frac{\Omega-u}{2}\right) \\ \sin\frac{i}{2}\sin\left(\frac{\Omega-u}{2}\right) \\ \cos\frac{i}{2}\sin\left(\frac{\Omega+u}{2}\right) \\ \cos\frac{i}{2}\cos\left(\frac{\Omega+u}{2}\right) \end{bmatrix} \tag{7.3}$$

R has to be expressed in the $\mathcal{F}_f$ to give R_{f1} and R_{f2}. The vector $\mathbf{R}$ lies in the orbital plane and is 90° ahead of the perifocus, this means that in $\mathcal{F}_e$, $\mathbf{R}$ can be written as

$$\mathbf{R}_e|_{\nu=90^\circ} = \begin{bmatrix} R, & 0, & 0 \end{bmatrix}^T \tag{7.4}$$

At the location where $\mathbf{R}$ is expressed, $\nu = 90^\circ$ and λ is

$$\lambda|_{\nu=90^\circ} = \Omega + \omega + 90^\circ \tag{7.5}$$

The rotation matrix $\mathbf{C}_{f,e}$ is the inverse of $\mathbf{C}_{e,f}$ and is

$$\mathbf{C}_{f,e}|_{\nu=90^\circ} = \begin{bmatrix} \cos\lambda & -\sin\lambda & 0 \\ \sin\lambda & \cos\lambda & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7.6}$$

$\mathbf{R}$ can be converted to $\mathcal{F}_f$ using the definitions of λ and u from Eqs. (6.1) and (6.2).

$$\begin{aligned} \begin{bmatrix} R_{f1} \\ R_{f2} \\ 0 \end{bmatrix} &= \begin{bmatrix} \cos((\Omega+\omega)+90^\circ) & -\sin((\Omega+\omega)+90^\circ) & 0 \\ \sin((\Omega+\omega)+90^\circ) & \cos((\Omega+\omega)+90^\circ) & 0 \\ 0 & 0 & 1 \end{bmatrix}\Bigg|_{\nu=90^\circ} \times \begin{bmatrix} R \\ 0 \\ 0 \end{bmatrix}\Bigg|_{\nu=90^\circ} \\ &= \begin{bmatrix} -\sin(\Omega+\omega) & -\cos(\Omega+\omega) & 0 \\ \cos(\Omega+\omega) & -\sin(\Omega+\omega) & 0 \\ 0 & 0 & 1 \end{bmatrix}\Bigg|_{\nu=90^\circ} \times \begin{bmatrix} R \\ 0 \\ 0 \end{bmatrix}\Bigg|_{\nu=90^\circ} \\ &= \begin{bmatrix} -R\sin(\Omega+\omega) \\ R\cos(\Omega+\omega) \\ 0 \end{bmatrix} \end{aligned} \tag{7.7}$$

This concludes the conversion from classic Keplerian elements to the USM elements.

7.1.2 Conversion from Cartesian Coordinates to the USM7 Elements

There is a method for conversion from Cartesian coordinates to the USM7 provided in [Altman, 1972] and [Chodas, 1981], which are both similar. Another method that is presented here is proposed by the author.

Original Method

The original method is the conversion provided by [Chodas, 1981]. The conversion found in [Altman, 1972] is also correct, but the one from [Chodas, 1981] is easier to follow. The state in Cartesian coordinates consists of the position given in (x,y,z), and the velocity (v_x,v_y,v_z). The position and velocity are written as vectors $\mathbf{r}$ and $\mathbf{v}$, respectively. The first step is to calculate the angular momentum, which is required for computing C.

$$\mathbf{h} = \begin{bmatrix} h_x \\ h_y \\ h_z \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \times \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} yv_z - zv_y \\ zv_x - xv_z \\ xv_y - yv_x \end{bmatrix} \tag{7.8}$$

C can be found from

$$C = \frac{\mu}{h} \tag{7.9}$$

where h is the magnitude of the angular momentum found in Eq. (7.8).

The velocity can be split up into $\mathbf{v}_{e1}$ and $\mathbf{v}_{e2}$ in the following way

$$\mathbf{v}_{e1} = \frac{\mathbf{r}^T\mathbf{v}}{r^2}\mathbf{r} = \frac{xv_x + yv_y + zv_z}{x^2+y^2+z^2} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{7.10a}$$

$$\mathbf{v}_{e2} = \mathbf{v} - \mathbf{v}_{e1} \tag{7.10b}$$

In Eq. (7.10), $\mathbf{v}_{e1}$ is simply the projection of $\mathbf{v}$ along the radial vector $\mathbf{r}$. In Eq. (7.10), $\mathbf{v}_{e1}$ and $\mathbf{v}_{e2}$ are still expressed $\mathcal{F}_g$. They are simply computed in vectorial form to be able to extract the scalar magnitudes v_{e1} and v_{e2}. The angle λ is not explicitly required, but its cosine and sine are. They can be found with the following expressions

$$\cos\lambda = \frac{1}{r}\left(x - \frac{h_x}{h+h_z}z\right) \tag{7.11a}$$

$$\sin\lambda = \frac{1}{r}\left(y - \frac{h_y}{h+h_z}z\right) \tag{7.11b}$$

The method to find the sine and cosine of λ is now outlined.

It is shown in [Chodas, 1981] that the axes of $\mathcal{F}_e$ are

$$\hat{\mathbf{e}}_1 = \frac{1}{r}\begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{7.12}$$

$$\hat{\mathbf{e}}_3 = \frac{1}{h}\begin{bmatrix} h_x \\ h_y \\ h_z \end{bmatrix} \tag{7.13}$$

$$\hat{\mathbf{e}}_2 = \hat{\mathbf{e}}_3 \times \hat{\mathbf{e}}_1 = -\frac{1}{rh}\begin{bmatrix} zh_y - yh_z \\ xh_z - zh_x \\ yh_x - xh_y \end{bmatrix} \tag{7.14}$$

The equations above follow from the elementary facts that $\hat{\mathbf{e}}_1$ lies along the radial vector and $\hat{\mathbf{e}}_3$ lies along the angular momentum vector.

According to [Chodas, 1981], the axes of $\mathcal{F}_f$ are

$$\hat{\mathbf{f}}_1 = \frac{1}{h}\begin{bmatrix} h_z + \frac{h_y^2}{h+h_z} \\ -\frac{h_x h_y}{h+h_z} \\ -h_x \end{bmatrix} \tag{7.15}$$

$$\hat{\mathbf{f}}_2 = \frac{1}{h}\begin{bmatrix} \frac{-h_x h_y}{h+h_z} \\ h_z + \frac{h_x^2}{h+h_z} \\ -h_y \end{bmatrix} \tag{7.16}$$

λ is the angle between $\hat{\mathbf{f}}_2$ and $\hat{\mathbf{e}}_2$ and so the cosine and sine of λ are

$$\cos\lambda = \hat{\mathbf{f}}_2^T\hat{\mathbf{e}}_2 = \frac{1}{r}\left(x - \frac{h_x}{h+h_z}z\right) \tag{7.17}$$

$$\sin\lambda = -\hat{\mathbf{f}}_1^T\hat{\mathbf{e}}_2 = \frac{1}{r}\left(y - \frac{h_y}{h+h_z}z\right) \tag{7.18}$$

All the steps presented in [Chodas, 1981] are logical, but there is no explanation available on how to find the axes of $\mathcal{F}_f$. This will be presented here in the spirit of this study, which aims to explain every single detail of the USM.

The method of deriving these axes is to transform the axes of $\mathcal{F}_g$ to $\mathcal{F}_f$ by using the DCM $\mathbf{C}_{f,g}$ from Eq. (6.5) found on page 48.

$$\mathbf{C}_{f,g} = \begin{bmatrix} \cos i + \cos^2\Omega(1-\cos i) & \cos\Omega\sin\Omega(1-\cos i) & -\sin\Omega\sin i \\ \cos\Omega\sin\Omega(1-\cos i) & \cos i + \sin^2\Omega(1-\cos i) & \cos\Omega\sin i \\ \sin\Omega\sin i & -\cos\Omega\sin i & \cos i \end{bmatrix}$$

It can be seen that to calculate the elements of the rotation matrix, the trigonometric functions of Ω and i are required. The cosine and sine of Ω can be found from Eq. (4.53) on page 36.

$$\cos\Omega = -\frac{h_y}{\sqrt{h_x^2 + h_y^2}} \tag{7.19}$$

$$\sin\Omega = \frac{h_x}{\sqrt{h_x^2 + h_y^2}} \tag{7.20}$$

The cosine of i is known from Eq. (4.51) on page 36, which yields

$$i = \arccos\left(\frac{h_z}{h}\right) \tag{7.21}$$

The inclination is the angle between the angular momentum vector and the z-axis. The sine of this angle can then be found by using the components of the angular momentum lying in the x-y plane. This trigonometric values is

$$\sin i = \frac{\sqrt{h_x^2 + h_y^2}}{h} \tag{7.22}$$

With this information, each element of the DCM $\mathbf{C}_{f,g}$ can be calculated using the following simplification

$$\frac{h - h_z}{h_x^2 + h_y^2} = \frac{h - h_z}{h^2 - h_z^2} = \frac{h - h_z}{(h + h_z)(h - h_z)} = \frac{1}{h + h_z}$$

The elements of $\mathbf{C}_{f,g}$ are

$$\begin{aligned} c_{1,1} &= \cos i + \cos^2\Omega\,(1 - \cos i) = \frac{1}{h}\left(h_z + \frac{h_y^2\,(h - h_z)}{h_x^2 + h_y^2}\right) \\ &= \frac{1}{h}\left(h_z + \frac{h_y^2}{h + h_z}\right) \end{aligned} \tag{7.23}$$

$$c_{1,2} = \cos\Omega\sin\Omega(1 - \cos i) = -\frac{1}{h}\left(\frac{h_x h_y\,(h - h_z)}{h_x^2 + h_y^2}\right) = -\frac{h_x h_y}{h\,(h + h_z)} \tag{7.24}$$

$$c_{1,3} = -\sin\Omega\sin i = -\frac{h_x}{h} \tag{7.25}$$

$$\tag{7.26}$$

$$\begin{aligned} c_{2,1} &= \cos\Omega\sin\Omega(1 - \cos i) = c_{1,2} = -\frac{1}{h}\left(\frac{h_x h_y\,(h - h_z)}{h_x^2 + h_y^2}\right) \\ &= -\frac{h_x h_y}{h\,(h + h_z)} \end{aligned} \tag{7.27}$$

$$\begin{aligned} c_{2,2} &= \cos i + \sin^2\Omega(1 - \cos i) = \frac{1}{h}\left(h_z + \frac{h_x^2\,(h - h_z)}{h_x^2 + h_y^2}\right) \\ &= \frac{1}{h}\left(h_z + \frac{h_x^2}{h + h_z}\right) \end{aligned} \tag{7.28}$$

$$c_{2,3} = \cos\Omega\sin i = -\frac{h_y}{h} \tag{7.29}$$

$$c_{3,1} = \sin\Omega\sin i = -c_{1,3} = \frac{h_x}{h} \tag{7.30}$$

$$c_{3,2} = -\cos\Omega \sin i = -c_{2,3} = \frac{h_y}{h} \tag{7.31}$$

$$c_{3,3} = \cos i = \frac{h_z}{h} \tag{7.32}$$

Finally, the unit vectors of $\mathcal{F}_f$ can be calculated using the definition of the DCM from Eq. (2.2) on page 6

$$\hat{\mathbf{f}}_1 = c_{1,1}\begin{bmatrix}1\\0\\0\end{bmatrix} + c_{1,2}\begin{bmatrix}0\\1\\0\end{bmatrix} + c_{1,3}\begin{bmatrix}0\\0\\1\end{bmatrix} = \frac{1}{h}\begin{bmatrix} h_z + \frac{h_y^2}{h+h_z} \\ -\frac{h_x h_y}{h+h_z} \\ -h_x \end{bmatrix} \tag{7.33}$$

$$\hat{\mathbf{f}}_2 = c_{2,1}\begin{bmatrix}1\\0\\0\end{bmatrix} + c_{2,2}\begin{bmatrix}0\\1\\0\end{bmatrix} + c_{2,3}\begin{bmatrix}0\\0\\1\end{bmatrix} = \frac{1}{h}\begin{bmatrix} \frac{-h_x h_y}{h+h_z} \\ h_z + \frac{h_x^2}{h+h_z} \\ -h_y \end{bmatrix} \tag{7.34}$$

$$\hat{\mathbf{f}}_3 = c_{3,1}\begin{bmatrix}1\\0\\0\end{bmatrix} + c_{3,2}\begin{bmatrix}0\\1\\0\end{bmatrix} + c_{3,3}\begin{bmatrix}0\\0\\1\end{bmatrix} = \frac{1}{h}\begin{bmatrix} h_x \\ h_y \\ h_z \end{bmatrix} \tag{7.35}$$

The 3 unit vectors making up $\mathcal{F}_f$ have now been derived successfully. The first two vectors $\hat{\mathbf{f}}_1$ and $\hat{\mathbf{f}}_2$ equal to the ones found in [Chodas, 1981]. The third vector $\hat{\mathbf{f}}_3$ is equal to $\hat{\mathbf{e}}_3$ and lies in the direction of the angular momentum as predicted. This is a good check to show that the method and the equations used above have been correct.

If required, λ can be calculated using the atan2 command found in most computing languages. It would be possible to use the atan function, but atan2 makes sure that there are no quadrant ambiguities.

$$\lambda = \text{atan2}\left(\sin\lambda, \cos\lambda\right) \tag{7.36}$$

The velocities R_{f1} and R_{f2} can be found using

$$R_{f1} = v_{e1}\cos\lambda - (v_{e2} - C)\sin\lambda \tag{7.37a}$$

$$R_{f2} = v_{e1}\sin\lambda + (v_{e2} - C)\cos\lambda \tag{7.37b}$$

To compute the quaternion elements, the half angle trigonometric functions of λ are required. These can be computed from Eq. (7.11) by using the trigonometric identities found in Appendix A. More specifically, Eqs. (A.8) and (A.9) are used.

$$\sin\frac{\lambda}{2} = \sqrt{\frac{1-\cos\lambda}{2}} \tag{7.38a}$$

$$\cos\frac{\lambda}{2} = \sqrt{\frac{1+\cos\lambda}{2}} \tag{7.38b}$$

The quaternion elements are

$$\epsilon_{O1} = \frac{1}{2\sqrt{h\,(h+h_z)}}\left(h_x\sqrt{2}\sin\frac{\lambda}{2} - h_y\sqrt{2}\cos\frac{\lambda}{2}\right) \tag{7.39a}$$

$$\epsilon_{O2} = \frac{1}{2\sqrt{h\,(h+h_z)}}\left(h_x\sqrt{2}\cos\frac{\lambda}{2} + h_y\sqrt{2}\sin\frac{\lambda}{2}\right) \tag{7.39b}$$

$$\epsilon_{O3} = \frac{h+h_z}{2\sqrt{h\,(h+h_z)}}\left(\sqrt{2}\sin\frac{\lambda}{2}\right) \tag{7.39c}$$

$$\eta_O = \frac{h+h_z}{2\sqrt{h\,(h+h_z)}}\left(\sqrt{2}\cos\frac{\lambda}{2}\right) \tag{7.39d}$$

As can in be seen in Eq. (7.39), the half angle versions of λ are always found in a product with $\sqrt{2}$. There are also sign ambiguities because of taking square roots in Eq. (7.39), which can be resolved as in [Chodas, 1981]. When $\cos\lambda \geq 0$

$$\sqrt{2}\cos\frac{\lambda}{2} = \sqrt{1+\cos\lambda} \tag{7.40a}$$

$$\sqrt{2}\sin\frac{\lambda}{2} = \frac{\sin\lambda}{\sqrt{2}\cos\frac{\lambda}{2}} \tag{7.40b}$$

When $\cos\lambda < 0$

$$\sqrt{2}\cos\frac{\lambda}{2} = \frac{\sin\lambda}{\sqrt{2}\sin\frac{\lambda}{2}} \tag{7.41a}$$

$$\sqrt{2}\sin\frac{\lambda}{2} = \text{sign}\left(\sin\lambda\right)\sqrt{1-\cos\lambda} \tag{7.41b}$$

which saves the computation of a square root. It should be noted that the quaternion computed in Eq. (7.39) may not always have the same sign as the quaternion found by propagating the state in USM. This because the orientation expressed by ($-\boldsymbol{\epsilon}_O$,$-\eta_O$) is the same as the orientation expressed by ($\boldsymbol{\epsilon}_O$,η_O). Thus, to have a smooth ephemeris when converting from Cartesian coordinates, it is necessary to have a logic that ensures that all the quaternion elements switch signs when they approach 0.

Alternative Method

An alternative method proposed by the author is based on first creating the DCM from $\mathcal{F}_g$ to $\mathcal{F}_e$ and subsequently extracting the quaternion. The three unit vectors that make $\mathcal{F}_g$ are

$$\hat{\mathbf{g}}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \qquad \hat{\mathbf{g}}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \qquad \hat{\mathbf{g}}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{7.42}$$

The three unit vectors that make $\mathcal{F}_e$ can be found from Eqs (7.12 - 7.14).

$$\hat{\mathbf{e}}_1 = \frac{1}{r}\begin{bmatrix} x \\ y \\ z \end{bmatrix} \qquad \hat{\mathbf{e}}_2 = -\frac{1}{rh}\begin{bmatrix} zh_y - yh_z \\ xh_z - zh_x \\ yh_x - xh_y \end{bmatrix} \qquad \hat{\mathbf{e}}_3 = \frac{1}{h}\begin{bmatrix} h_x \\ h_y \\ h_z \end{bmatrix}$$

From the definition of a DCM found in Section 2.1.1, the elements of $\mathbf{C}_{e,g}$, $c_{i,j}$, can be found in the following manner

$$\hat{\mathbf{e}}_1 = c_{1,1}\hat{\mathbf{g}}_1 + c_{1,2}\hat{\mathbf{g}}_2 + c_{1,3}\hat{\mathbf{g}}_3 \tag{7.43a}$$

$$\hat{\mathbf{e}}_2 = c_{2,1}\hat{\mathbf{g}}_1 + c_{2,2}\hat{\mathbf{g}}_2 + c_{2,3}\hat{\mathbf{g}}_3 \tag{7.43b}$$

$$\hat{\mathbf{e}}_3 = c_{3,1}\hat{\mathbf{g}}_1 + c_{3,2}\hat{\mathbf{g}}_2 + c_{3,3}\hat{\mathbf{g}}_3 \tag{7.43c}$$

Solving for all the elements of the rotation matrix results in

$$\mathbf{C}_{e,g} = \frac{1}{rh}\begin{bmatrix} xh & yh & zh \\ zh_y - yh_z & xh_z - zh_x & yh_x - xh_y \\ rh_x & rh_y & rh_z \end{bmatrix} \tag{7.44}$$

The quaternion can then be extracted using Eqs. (3.22) and (3.23) on page 19. These equations are presented again, but with the elements of the DCM filled in. The squares of the quaternion elements are

$$\epsilon_{O1}^2 = \frac{1}{4rh}\left(1 + xh - xh_z + zh_x - rh_z\right) \tag{7.45a}$$

$$\epsilon_{O2}^2 = \frac{1}{4rh}\left(1 - xh + xh_z - zh_x - rh_z\right) \tag{7.45b}$$

$$\epsilon_{O3}^2 = \frac{1}{4rh}\left(1 - xh - xh_z + zh_x + rh_z\right) \tag{7.45c}$$

$$\eta_O^2 = \frac{1}{4rh}\left(1 + xh + xh_z - zh_x + rh_z\right) \tag{7.45d}$$

$$\tag{7.45e}$$

After taking the square root of the largest square, the rest of the quaternion elements can be found from the following relations

$$\epsilon_{O1}\epsilon_{O2} = \frac{1}{4rh}\left(yh + zh_x - yh_z\right) \tag{7.46a}$$

$$\epsilon_{O1}\epsilon_{O3} = \frac{1}{4rh}\left(rh_x + zh\right) \tag{7.46b}$$

$$\epsilon_{O1}\eta_O = \frac{1}{4rh}\left(yh_x - xh_y - rh_y\right) \tag{7.46c}$$

$$\epsilon_{O2}\epsilon_{O3} = \frac{1}{4rh}\left(yh_x - xh_y + rh_y\right) \tag{7.46d}$$

$$\epsilon_{O2}\eta_O = \frac{1}{4rh}\left(rh_x - zh\right) \tag{7.46e}$$

$$\epsilon_{O3}\eta_O = \frac{1}{4rh}\left(yh - zh_y + yh_z\right) \tag{7.46f}$$

After extracting the quaternion, λ can be found using Eq. (6.33) and R_{f1} and R_{f2} can be found using Eq. (7.37).

This concludes the conversion from Cartesian coordinates to the USM7 elements. The author's method provides more insight into the conversion process and thus, is easier to follow for someone unfamiliar with the USM. Also, it is shown how to compute the DCM directly from the Cartesian coordinates, and a standard quaternion extraction function can be consistently used throughout the thesis. This function can also be used when it is required to extract a quaternion from rotation matrices for other applications like the vehicle attitude.

7.2 Conversion from the USM7 Elements

The conversion from the USM7 elements to both the Keplerian elements and the Cartesian coordinates is shown here.

7.2.1 Conversion to Cartesian Coordinates

The first conversion that is dealt with is the conversion to the Cartesian coordinates as this is quite straight forward. The USM elements available are (R_{f1},R_{f2},C,ϵ_{O1},ϵ_{O2},ϵ_{O3},η_O). Using the quaternion, the sine and cosine of λ can be calculated by using Eq. (6.33).

$$\begin{bmatrix} \sin\lambda \\ \cos\lambda \end{bmatrix} = \frac{1}{\epsilon_{O3}^2 + \eta_O^2}\begin{bmatrix} 2\epsilon_{O3}\eta_O \\ \eta_O^2 - \epsilon_{O3}^2 \end{bmatrix}$$

Using the sine and cosine of λ, and the velocity components C, R_{f1}, and R_{f2}, the velocity in $\mathcal{F}_e$ can be computed using Eq. (6.27).

$$\begin{bmatrix} v_{e1} \\ v_{e2} \end{bmatrix} = \begin{bmatrix} 0 \\ C \end{bmatrix} + \begin{bmatrix} \cos\lambda & \sin\lambda \\ -\sin\lambda & \cos\lambda \end{bmatrix}\begin{bmatrix} R_{f1} \\ R_{f2} \end{bmatrix}$$

The position in $\mathcal{F}_e$ is found from Eq. (6.35).

$$\mathbf{r} = \frac{\mu}{Cv_{e2}}\hat{\mathbf{e}}_1$$

The Cartesian coordinates are expressed in $\mathcal{F}_g$. The DCM from $\mathcal{F}_g$ to $\mathcal{F}_e$ can be found from the quaternion, from Eq. (3.18), to be

$$\mathbf{C}_{e,g} = \begin{bmatrix} 1-2\left(\epsilon_{O2}^2+\epsilon_{O3}^2\right) & 2\left(\epsilon_{O1}\epsilon_{O2}+\epsilon_{O3}\eta_O\right) & 2\left(\epsilon_{O1}\epsilon_{O3}-\epsilon_{O2}\eta_O\right) \\ 2\left(\epsilon_{O2}\epsilon_{O1}-\epsilon_{O3}\eta_O\right) & 1-2\left(\epsilon_{O3}^2+\epsilon_{O1}^2\right) & 2\left(\epsilon_{O2}\epsilon_{O3}+\epsilon_{O1}\eta_O\right) \\ 2\left(\epsilon_{O3}\epsilon_{O1}+\epsilon_{O2}\eta_O\right) & 2\left(\epsilon_{O3}\epsilon_{O2}-\epsilon_{O1}\eta_O\right) & 1-2\left(\epsilon_{O1}^2+\epsilon_{O2}^2\right) \end{bmatrix} \tag{7.47}$$

If the USM6 is being used, $C_{e,g}$ can be directly constructed using Eq. (3.45). The DCM from $\mathcal{F}_e$ back to $\mathcal{F}_g$, $\mathbf{C}_{g,e}$, is simply the transpose of $\mathbf{C}_{e,g}$. The position and velocity in the Cartesian coordinates can be found by converting the position and velocities from $\mathcal{F}_e$ to $\mathcal{F}_g$.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \frac{\mu}{Cv_{e2}}\mathbf{C}_{e,g}^T \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tag{7.48}$$

$$\begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \mathbf{C}_{e,g}^T \begin{bmatrix} v_{e1} \\ v_{e2} \\ 0 \end{bmatrix} \tag{7.49}$$

7.2.2 Conversion to Keplerian Elements

The conversion to Keplerian elements is not as straightforward as the conversion to Cartesian coordinates. One way would be to first convert the USM elements to Cartesian coordinates, and then use the method from Section 4.6 to convert the Cartesian coordinates to Keplerian elements. Another way is to continue working with the USM elements, which requires derivation but will ensure that all the conversions are independent and no unnecessary computations are carried out. The technique used, however, is based on Section 4.6.

The eccentricity of the orbit, e, can then be found by taking the quotient of R and C. C is already present in the USM state vector, and R can be found by taking the norm of R_{f1} and R_{f2}.

$$e = R/C \tag{7.50}$$

Eq. (7.50) is singular when $C = 0$. This does not add any more singularities because the USM is singular for rectilinear trajectories, which is the case when $C = 0$. The semi-major axis, a, can be found from the vis-viva integral, which is

$$a = \frac{\mu}{2\left(\frac{\mu}{r}-\frac{v^2}{2}\right)} \tag{7.51}$$

The total velocity, v, in Eq. (7.51) can be found by taking the norm of v_{e1} and v_{e2}. The magnitude of the radial vector, r, can be found from Eq. (6.35). Filling in all the appropriate values in Eq. (7.51) gives the expression for the semi-major axis to be

$$a = \frac{\mu}{2Cv_{e2} - (v_{e1}^2 + v_{e2}^2)} \tag{7.52}$$

To find $\mathbf{h}$ in $\mathcal{F}_g$, the property that $\mathbf{h}$ in $\mathcal{F}_e$ is

$$\mathbf{h}_e = (\mu/C)\hat{\mathbf{e}}_3$$

is used. The inverse of $\mathbf{C}_{e,g}$ is used to express $\mathbf{h}$ in $\mathcal{F}_g$. After conversion, $\mathbf{h}$ becomes

$$\begin{aligned} \mathbf{h}_g =& (\mu/C)\Big(2\left(\epsilon_{O1}\epsilon_{O3}+\epsilon_{O2}\eta_O\right)\hat{\mathbf{g}}_1 + 2\left(\epsilon_{O2}\epsilon_{O3}-\epsilon_{O1}\eta_O\right)\hat{\mathbf{g}}_2 \\ &+ \left(1-2\left(\epsilon_{O1}^2+\epsilon_{O2}^2\right)\right)\hat{\mathbf{g}}_3\Big) \end{aligned} \tag{7.53}$$

The inclination of the orbit is the angle between the angular momentum vector and $\hat{\mathbf{g}}_3$. The cosine of this angle can be found by taking the inner product of the unit vector in the direction of $\mathbf{h}$ from Eq. (7.53) and $\hat{\mathbf{g}}_3$, which is simply $\begin{bmatrix} 0, & 0, & 1 \end{bmatrix}^T$. Thus, the inclination can be found via the following relation:

$$i = \arccos\left(1 - 2\left(\epsilon_{O1}^2 + \epsilon_{O2}^2\right)\right) \tag{7.54}$$

To compute Ω, it is necessary to find the line of nodes, $\mathbf{N}$. The line of nodes can be computed by taking the cross product of $\hat{\mathbf{g}}_3$ and $\mathbf{h}$ from Eq. (7.53). This results in the line of nodes expressed in $\mathcal{F}_g$ to be

$$\mathbf{N}_g = 2(\mu/C)\Big(\left(\epsilon_{O1}\eta_O - \epsilon_{O2}\epsilon_{O3}\right)\hat{\mathbf{g}}_1 + \left(\epsilon_{O1}\epsilon_{O3} + \epsilon_{O2}\eta_O\right)\hat{\mathbf{g}}_2\Big) \tag{7.55}$$

Ω is the angle between $\hat{\mathbf{g}}_1$ and $\mathbf{N}$. Thus, the cosine of Ω is the inner product of the unit vector in the direction of $\mathbf{N}$ and $\hat{\mathbf{g}}_1$, which is simply $\begin{bmatrix} 1, & 0, & 0 \end{bmatrix}^T$. The cosine of Ω is

$$\cos\Omega = \left(\frac{\epsilon_{O1}\eta_O - \epsilon_{O2}\epsilon_{O3}}{\sqrt{\left(\epsilon_{O1}^2 + \epsilon_{O2}^2\right)\left(\eta_O^2 + \epsilon_{O3}^2\right)}}\right) \tag{7.56}$$

The true anomaly is the angle between the eccentricity vector, $\mathbf{e}$, and the position vector. The eccentricity vector points towards the periapsis of the orbit and has a magnitude equal to e. According to [Curtis, 2005], the eccentricity vector is found in the following way:

$$\mathbf{e} = \left(\mathbf{v} \times \mathbf{h}\right)/\mu - \left(\mathbf{r}/r\right) \tag{7.57}$$

The eccentricity vector from Eq. (7.57) expressed in $\mathcal{F}_e$ after some algebraic manipulation yields:

$$\mathbf{e}_e = \Big(\left(v_{e2} - C\right)\hat{\mathbf{e}}_1 - v_{e1}\hat{\mathbf{e}}_2\Big)/C \tag{7.58}$$

Using the eccentricity vector from Eq. (7.58) and the position vector in $\mathcal{F}_e$ from Eq. (6.35), the cosine of the true anomaly is the inner product and is

$$\cos\nu = \left(v_{e2} - C\right)/R \tag{7.59}$$

The angle $\lambda = \Omega + \omega + \nu$, of which the sine and cosine have already been calculated during the conversion. With λ, Ω, and ν calculated, ω can be calculated in the following manner

$$\omega = \lambda - \Omega - \nu \tag{7.60}$$

The cosine of ω can also be found directly since it is the angle between the line of nodes to the eccentricity vector along the orbital plane. However, the expression found in that way is longer and we find the method of Eq. (7.60) to be simpler.

This concludes all the conversions regarding the USM elements. There is now enough information to extract the USM elements from either the Cartesian coordinates or the classic Keplerian elements. The USM elements can also be converted back to those two coordinate systems for ease of comparison.

7.3 Implementation of the USM

All the equations and theory required for the USM have been treated in Chapter 6 and the conversions in Sections 7.1 and 7.2. The implementation can be seen in Table 7.1. Only the implementation of USM7 is shown, because the method for USM6 is identical.

The equations required for the implementation of USM7 are spread out over many pages in the report. They are all shown here together for the sake of completeness. The reader is referred to the page numbers given in Table 7.1 for information regarding the derivation. The conversions

Table 7.1: The implementation procedure for the USM

Step	Derived Where
Initialize the USM state	Section 7.1, page 61
Find λ	Section 6.2, page 53, Eq. (6.33)
Compute v_{e1}, v_{e2}	Section 6.2, page 52, Eq. (6.27)
Find perturbing accelerations	Appendix B, page 191
Compute p	Section 6.3, page 55, Eq. (6.48)
Compute γ	Section 6.3, page 56, Eq. (6.62)
Compute ω_1	Section 6.3, page 55, Eq. (6.53)
Compute ω_3	Section 6.3, page 55, Eq. (6.46)
Propagate velocities, C, R_{f1}, and R_{f3}	Section 6.3, page 56, Eq. (6.65)
Propagate quaternion	Section 6.3, page 55, Eq. (6.54)
Repeat process	

are not shown here as they can be found in a relatively compact manner starting from page 61. The propagations are carried out using numerical integration methods found in Section C.1.

$$\begin{bmatrix} \sin\lambda \\ \cos\lambda \end{bmatrix} = \frac{1}{\epsilon_{O3}^2 + \eta_O^2} \begin{bmatrix} 2\epsilon_{O3}\eta_O \\ \eta_O^2 - \epsilon_{O3}^2 \end{bmatrix}$$

$$\lambda = \text{atan2}\left(\sin\lambda, \cos\lambda\right)$$

$$\begin{bmatrix} v_{e1} \\ v_{e2} \end{bmatrix} = \begin{bmatrix} 0 \\ C \end{bmatrix} + \begin{bmatrix} \cos\lambda & \sin\lambda \\ -\sin\lambda & \cos\lambda \end{bmatrix} \begin{bmatrix} R_{f1} \\ R_{f2} \end{bmatrix}$$

$$p = \frac{C}{v_{e2}}$$

$$\gamma = \frac{\epsilon_{O1}\epsilon_{O3} - \epsilon_{O2}\eta_O}{\epsilon_{O3}^2 + \eta_O^2}$$

$$\omega_1 = \frac{a_{e3}}{v_{e2}}$$

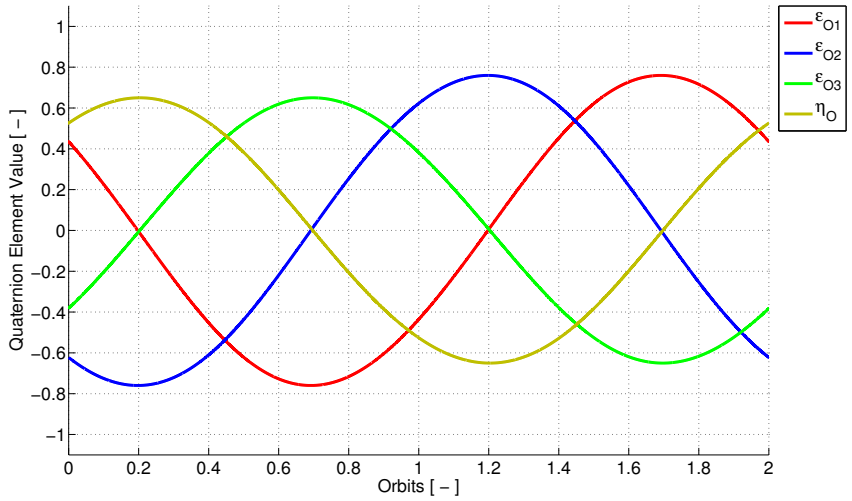
$$\omega_3 = \frac{C v_{e2}^2}{\mu}$$

$$\begin{bmatrix} \dot{C} \\ \dot{R}_{f1} \\ \dot{R}_{f2} \end{bmatrix} = \begin{bmatrix} 0 & -p & 0 \\ \cos\lambda & -(1+p)\sin\lambda & -\gamma R_{f2}/v_{e2} \\ \sin\lambda & (1+p)\cos\lambda & \gamma R_{f1}/v_{e2} \end{bmatrix} \begin{bmatrix} a_{e1} \\ a_{e2} \\ a_{e3} \end{bmatrix}$$

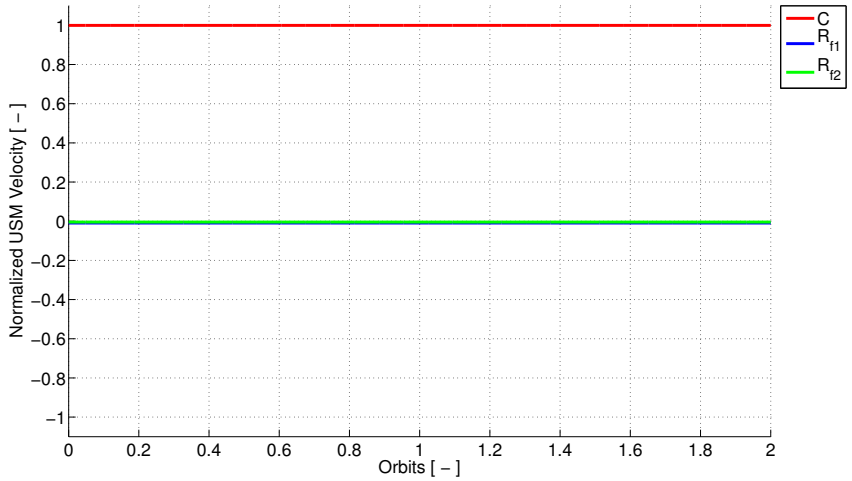
$$\begin{bmatrix} \dot{\epsilon}_{O1} \\ \dot{\epsilon}_{O2} \\ \dot{\epsilon}_{O3} \\ \dot{\eta}_O \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & \omega_3 & 0 & \omega_1 \\ -\omega_3 & 0 & \omega_1 & 0 \\ 0 & -\omega_1 & 0 & \omega_3 \\ -\omega_1 & 0 & -\omega_3 & 0 \end{bmatrix} \begin{bmatrix} \epsilon_{O1} \\ \epsilon_{O2} \\ \epsilon_{O3} \\ \eta_O \end{bmatrix}$$

7.4 Visualizing Orbits using USM

Specialists in astronautics are accustomed to visualizing orbits using classic Keplerian elements. An orbital trajectory is also easy to see when plotted in Cartesian coordinates. There are no available visualizations, in existing literature, of orbits using the USM elements. Therefore, the behavior of the USM elements for various orbits are shown here. Only unperturbed orbits will be presented.



(a) Behavior of the quaternion elements



(b) Behavior of the normalized USM velocities

Figure 7.1: The behavior of the USM elements over 2 orbits of the SARSAT with $C \approx 7434$ m/s, $R_{f1} \approx -68$ m/s, and $R_{f2} \approx -30$ m/s

7.4.1 Orbits expressed in USM7

Some of the orbits that will be used during the remainder of this thesis study are plotted. The quaternion elements are bounded by 1, but C, R_{f1}, and R_{f2} are many orders of magnitude larger. To ensure that all the USM elements fit in one plot, the velocity parameters are scaled by using the initial value of C. Two orbits are plotted instead of one to show the periodicity of the quaternion elements.

The first plot to be shown is the orbit of the SARSAT, which is a Search and Rescue satellite in an almost circular polar orbit, can be seen in Fig. 7.1. The orbital elements can be found in Tab. 8.1 on pg. 78. SARSAT is on an almost polar, retrograde, and slightly eccentric orbit.

The second plot is the orbit of HEOS-2, which can be seen in Figure 7.2. The orbital elements can be found in Tab. 8.1 on page 78. HEOS-2 is on a highly eccentric polar orbit.

The plots show that after a full orbit, all the quaternion elements have the same magnitude, but

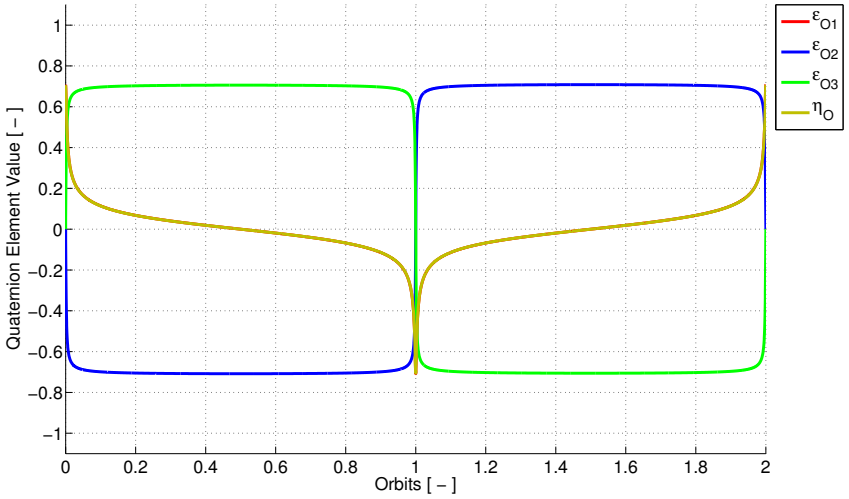
the opposite sign. This is accurate because when $\nu = 360°$, the rotation is the same as when $\nu = 0°$, but the long way around. For the next orbit, the behavior is the same but with the signs reversed. This is because the quaternion elements use the half angle of the Euler angle and thus, are periodic over 720°. To be very consistent, the sign of the quaternion elements should be switched using some control logic once $\nu > 360°$. This would, however, not influence the results and it would add an unnecessary discontinuity in the ephemeris.

7.4.2 Orbits expressed in USM6

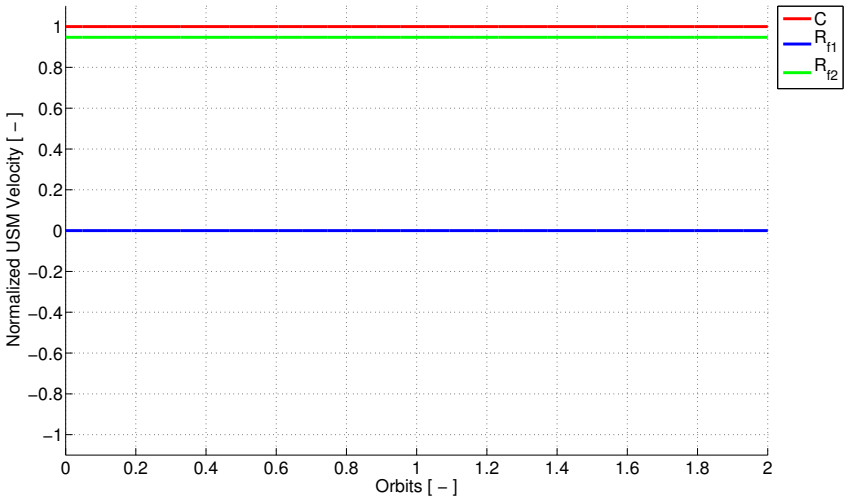
The same orbits as found in Section 7.4.1 are shown here. However, the orientation of $\mathcal{F}_e$ is expressed using MRP. Since C, R_{f1}, and R_{f2} are the same for both USM7 and USM6, they are not shown in the plots. The evolution of the MRP elements for the SARSAT orbit can be seen in Fig. 7.3 and for the HEOS2 orbit in Fig. 7.4.

Technically, the MRP should be periodic over four orbits. However, the MRP is checked frequently during the orbital integration to make sure that the norm of the vector does become more than 1. If the norm is indeed larger than 1, the MRP is converted to SMRP and vice versa.

This chapter shows all the information required to implement the USM. Thus, some tests using numerical integration for orbit propagation are carried out in the next chapter. The goal is to compare the performance of the USM with Cartesian coordinates.



(a) Behavior of the quaternion elements with $\epsilon_{O1} \approx \eta_O$



(b) Behavior of the normalized USM velocities

Figure 7.2: The behavior of the USM elements over 2 orbits of the HEOS-2 with $C \approx 5495$ m/s, $R_{f1} = 0$ m/s, and $R_{f2} \approx 5207$ m/s

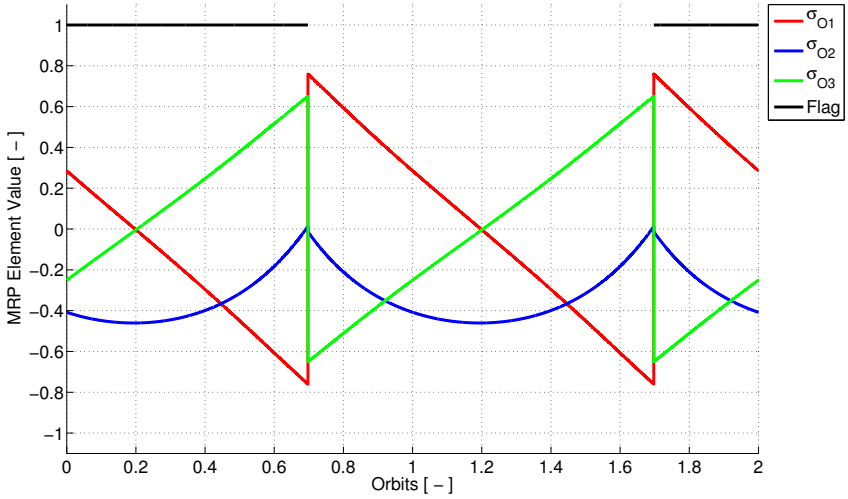


Figure 7.3: Behavior of the MRP elements over 2 orbits of SARSAT with MRP being used when the Flag equals 1 and SMRP otherwise

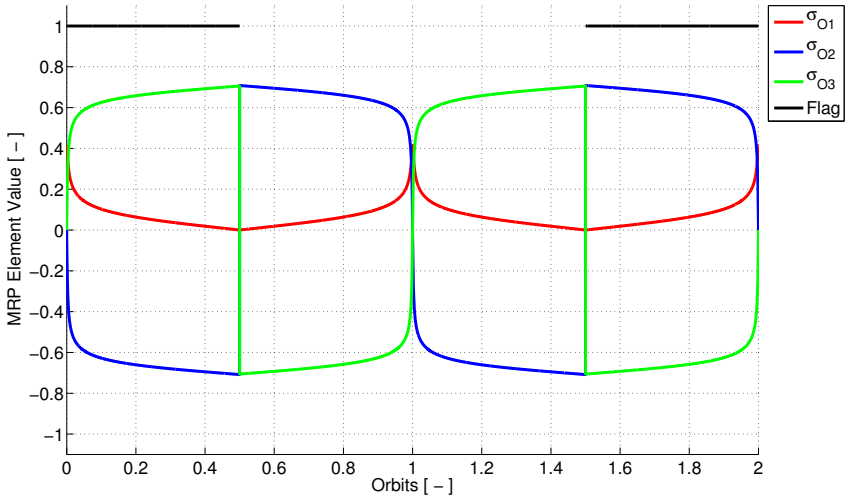


Figure 7.4: Behavior of the MRP elements over 2 orbits of HEOS2 with MRP being used when the Flag equals 1 and SMRP otherwise

Chapter 8

Orbit Propagation

In the previous chapter, the equations required to fully implement the USM were shown. It is important to apply all the theoretical equations to practice to see what the numerical performance of the USM is. The simulations are carried out using USM7, USM6, and Cartesian coordinates, with the model based on Cartesian coordinates being the benchmark. Simulations of unperturbed Keplerian orbits are neither shown, nor compared as that is not a realistic scenario for missions. Some simulations of unperturbed orbits were carried out to check and validate the USM and it was found that the error is many orders of magnitude lower for the USM than for the Cartesian coordinates. To be able to fully apply the USM, tests should be carried out with perturbed orbits. Low thrust propulsion is the focus of much work these days, and there are analytical ways to compute the trajectories with certain constraints on the thrust profile. Numerically integrating these low-thrust orbits, however, takes a large amount of time as these trajectories have a very long time of flight. Thus, any possibility of reducing the CPU time is desirable.

8.1 Numerical Integration Method

All the simulations carried out require a true orbit, which cannot be determined analytically for perturbed orbits. Thus, a high-order RK8(7) integrator, as found in Appendix C.1.3 on pg. 201, is used to first generate the so-called truth model. A lower-order RK5(4) integrator, as found in Appendix C.1.2 on pg. 200, is then used to find the error with respect to the truth model. Simulations are normally carried out using variable step-size integrators, but a fixed time-step integrator can show the stability of the model.

It is possible to derive all the various perturbations in terms of the USM as shown in [Altman, 1972]. However, users have perturbations already available in the Cartesian form. To ensure that the USM can be easily implemented without much overhead and because the time required to convert between the USM and Cartesian coordinates is very small compared to the computation time required to compute the perturbations, perturbation models expressed in Cartesian coordinates have been used. The perturbations applied are: atmospheric drag, $J_{2,0}$, $J_{2,2}$, solar and lunar 3^{rd} body perturbation, and solar-radiation pressure. According to [Montenbruck and Gill, 2005], these perturbations are the major sources of perturbation for LEO satellites. Details of the perturbations can be found in Appendix B. The time derivative for both USM7 and USM6 was carried out as shown in Fig. 8.1. This shows that if perturbations are to be used, the USM state is first converted to Cartesian coordinates prior to computing the perturbations.

The integration procedure for USM7 and USM6 can be seen in Fig. 8.2. The *USM function evaluations* block in Fig. 8.2 refers to the flowchart seen in Fig. 8.1. It is important to note that the tolerances were checked during the USM integration by converting the propagated state to Cartesian coordinates. This way, a consistent time step-size adjuster is used for integration with USM and Cartesian coordinates. This was done because the traditional way of step-size control,

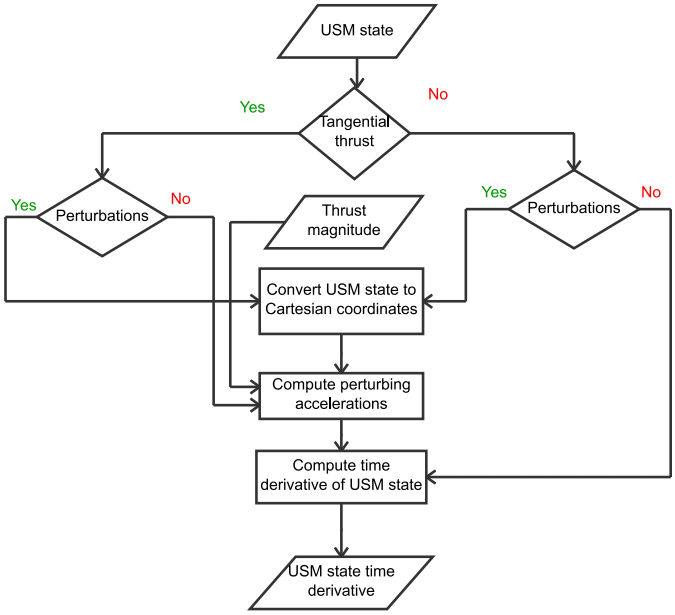


Figure 8.1: The procedure for a finding the state time derivative when carrying out integration using USM7 or USM6

which checks the tolerance on each individual state element does not function well with the USM. This was later confirmed independently in [Mooij, 2010].

The variable step-size integrator requires two tolerance values ε_{pos} and ε_{vel}. After an integration step, both the actual 5^{th}-order $\mathbf{y}$, and the embedded 4^{th}-order solution $\mathbf{y}_{em}$ are used. If the integration is carried out using the USM7 or USM6, they are converted to Cartesian coordinates. The difference between the two vectors in Cartesian coordinates Δ, is taken and then divided by the present time step-size h:

$$\Delta = \begin{bmatrix} \Delta\mathbf{r} \\ \Delta\mathbf{v} \end{bmatrix} = \frac{\mathbf{y} - \mathbf{y}_{em}}{h} \tag{8.1}$$

In case $|\Delta\mathbf{r}| \leq \varepsilon_{pos}$ and $|\Delta\mathbf{v}| \leq \varepsilon_{vel}$, the solution is accepted. Otherwise, the integration is repeated using a new time step-size h_{new}. Even if the solution is accepted, h_{new} is computed for the next integration step.

Two more values are defined to be

$$\delta r = 0.84 \left(\frac{\varepsilon_{pos}}{|\Delta\mathbf{r}|} \right)^{1/4} \tag{8.2a}$$

$$\delta v = 0.84 \left(\frac{\varepsilon_{vel}}{|\Delta\mathbf{v}|} \right)^{1/4} \tag{8.2b}$$

The difference δ to be used to find h_{new} is the smallest out of δr and δv. Finally, h_{new} is computed using one of the following three ways: If $\delta \leq 0.1$

$$h_{new} = 0.1 \cdot h \tag{8.3}$$

if $\delta \geq 5$

$$h_{new} = 5 \cdot h \tag{8.4}$$

and for all other cases

$$h_{new} = \delta \cdot h \tag{8.5}$$

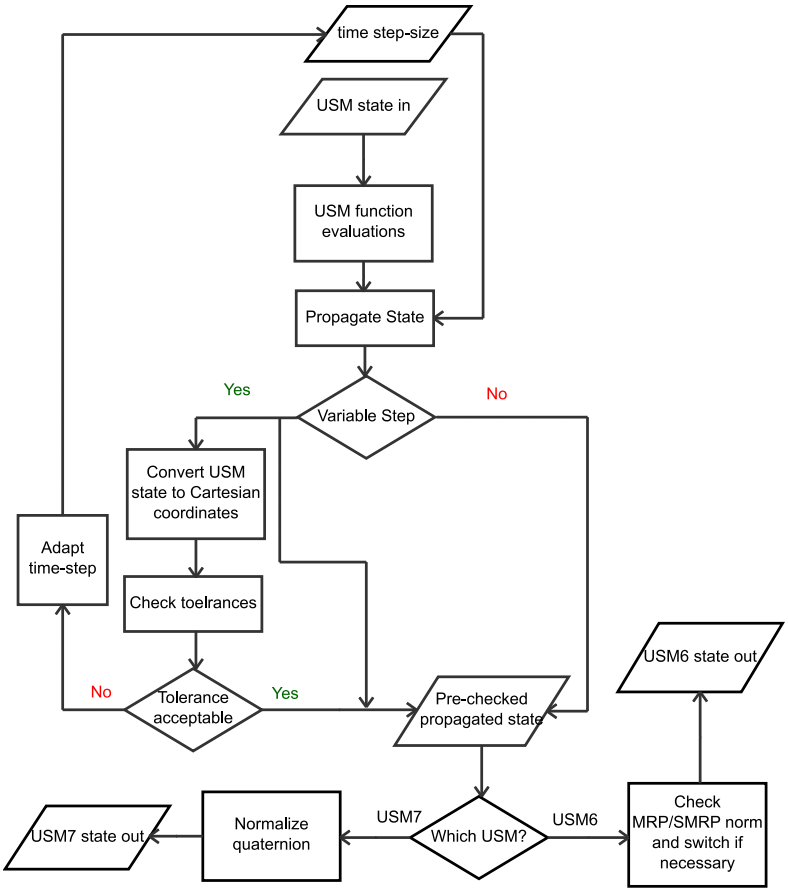


Figure 8.2: The integration procedure for USM7 and USM6

Satellite	a [km]	e [-]	i [deg]	Ω [deg]	ω [deg]	ν [deg]
SARSAT	7213	0.01	98.9	269	205	174
HEOS-2	125853	0.95	90.2	0	0	0

Table 8.1: Keplerian elements of SARSAT and HEOS-2

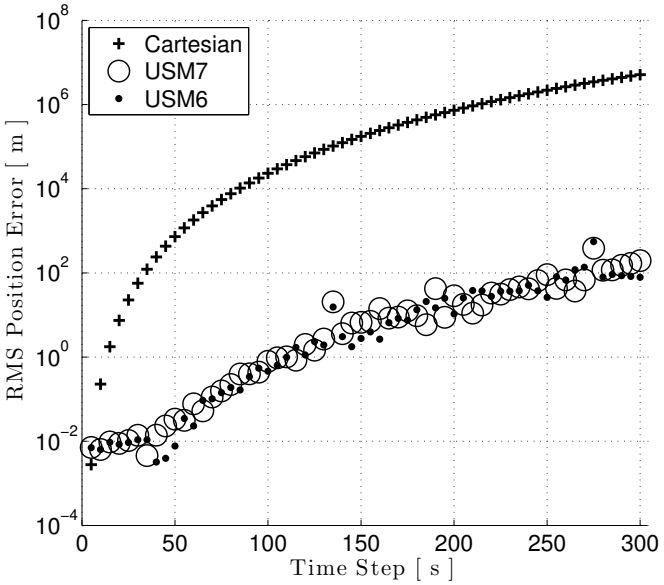


Figure 8.3: Error in position for 100 orbital revolutions of SARSAT with a fixed step-size integrator and all perturbations

8.2 Perturbed Orbits without Thrust

The orbits of SARSAT and HEOS-2 are simulated. The SARSAT orbit has a low eccentricity, and is an almost polar LEO, while HEOS-2 is on a highly elliptical orbit that has a perigee of 400 km and an apogee of almost 1/3 AU. The orbit of SARSAT was chosen, because it was used in [Chodas, 1981] for orbit determination tests and HEOS-2 has an orbit that passes through all of the various perturbing forces possible. The Keplerian elements of the two orbits can be seen in Table 8.1.

Since SARSAT has a LEO with an altitude of approximately 800 km, the major perturbation is due to the $J_{2,0}$ effect. The next largest perturbations are due to drag and $J_{2,2}$, but their effect is many orders of magnitude smaller than $J_{2,0}$. To investigate the effect of very minor perturbations on the numerical stability of the quaternion, simulations were carried out using only $J_{2,0}$ as one case, and with $J_{2,0}$, $J_{2,2}$, and atmospheric drag as another case. The RMS of the position error using all the perturbations for 100 orbital periods of SARSAT using a fixed step-size integrator with various step sizes can be seen in Fig. 8.3, and the results using a variable step-size integrator with various tolerances can be seen in Fig. 8.4. The results for using only the J_2 perturbations can be seen in Figs. 8.5 and 8.6.

For fixed time step integration in Fig. 8.3, the RMS position error for the USM6 and the USM7 is around 10^{-2} m for time step-sizes between 5 s and 40 s. For a time step-size of 40 s, the RMS position error for the USM7 and the USM6 decreases to approximately 5×10^{-3} m, and then continues increasing till the RMS position error is approximately 100 m for time step-sizes of 300 s. As the time step-size increases, there is a sawtooth-like behavior in the RMS position error curve. For the Cartesian coordinates, the RMS error is approximately 5×10^{-3} m when the time step-size is 5 s, and then increases continuously and smoothly till the error is approximately

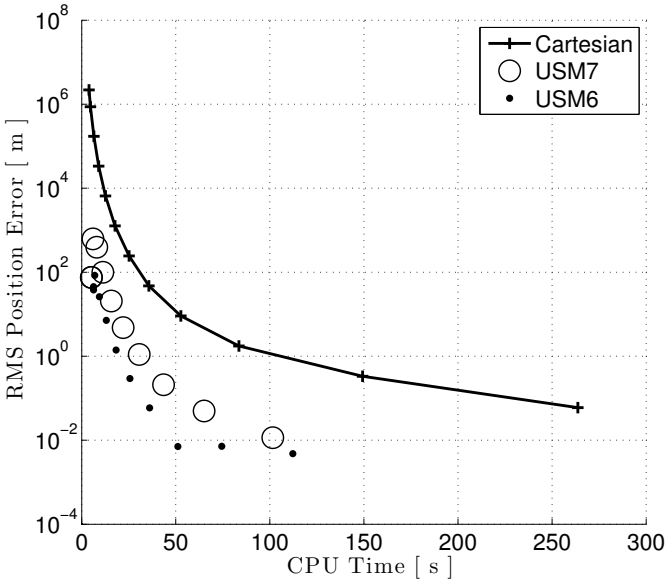


Figure 8.4: Error in position for 100 orbital revolutions of SARSAT with a variable step-size integrator and all perturbations

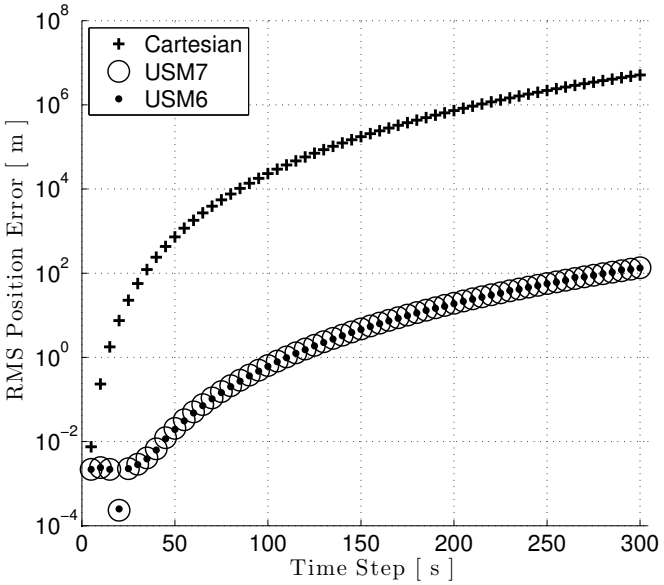


Figure 8.5: Error in position for 100 orbital revolutions of SARSAT with a fixed step-size integrator and perturbation due to $J_{2,0}$ only

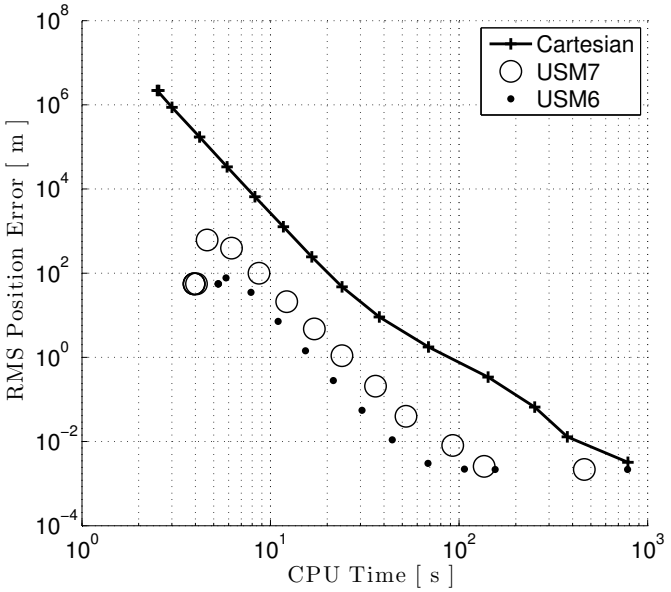


Figure 8.6: Error in position for 100 orbital revolutions of SARSAT with a variable step-size integrator and perturbation due to $J_{2,0}$ only

5×10^6 m for time step-sizes of 300 s. This shows that at a time step-size of 5 s for Cartesian coordinates and 35 s for the the USMs, the solution has reached the accuracy of the so-called truth model. The behavior in Fig. 8.5 is almost the same as in Fig. 8.3. This time, however, there is no sawtooth-like behavior. The USMs reach the accuracy of the so-called truth model at a time step of 20 s instead of 35 s. The error for time step-sizes between 5 - 15 s is around 3×10^{-3} m for the USMs. The error for the Cartesian coordinates at a time step-size of 5 s is approximately 9×10^{-3} m.

For a low eccentricity orbit like the SARSAT orbit, the performance of the USM models is clearly superior to the Cartesian model. When fixed-step integration is used, the position error for Cartesian and USM models is almost the same for small step sizes. However, as the size of the integration step increases, the position error of the USM models remains bounded and in the order of 10^2 m. The position error of the Cartesian model increases unboundedly and even reaches the order of 10^6 m. In Fig. 8.3, there is a minor saw-tooth behavior present in the USM integration. This is not present in Fig. 8.5, because only the largest perturbation, J_2, is used. This saw-tooth behavior is due to the effect of very small perturbations on the quaternion and MRP element values. The position is very sensitive to changes in the quaternion and MRP elements. Thus, small changes caused by small perturbations act like a sort of noise to give the saw-tooth behavior.

During the variable step-size integration, the results of the Cartesian model form a front in the position error - CPU time space, as seen in Figs. 8.4 and 8.5, that is like the Pareto front used for optimization. Any USM model results below or to the left of this front can be considered to perform better than the Cartesian results. As can be seen in Figs. 8.4 and 8.6, all of the USM results are better than the Cartesian results. For a given RMS position error, the CPU time for the USM7 is approximately a fourth of the CPU time for Cartesian coordinates and the CPU time for the USM6 is approximately a fifth. The performance of both USM models is similar for fixed-step integration. However, the performance of USM6 is better than the performance of USM7 for variable step-size integration. Like the Cartesian results, if the USM7 results are also made to form a front, all of the USM6 results are superior.

Only two orbits of HEOS-2 are simulated for the fixed time step-size integration since the orbital period is significantly larger than the orbital period of SARSAT. However, ten orbits are simulated

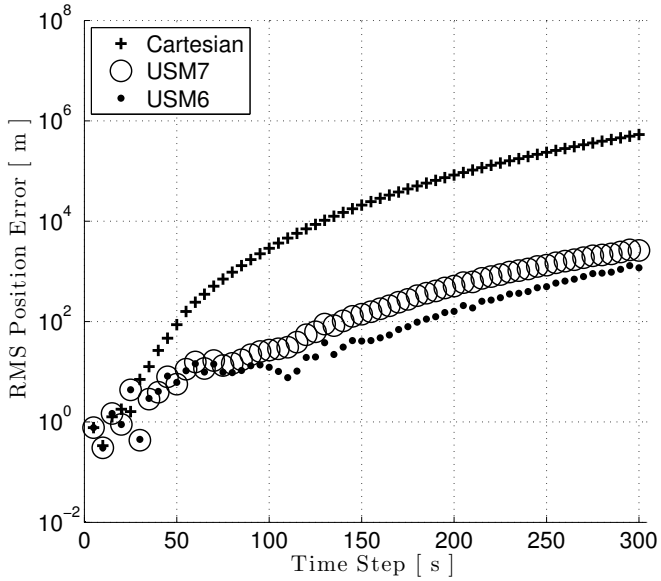


Figure 8.7: Error in position for 2 orbital revolutions of HEOS-2 with a fixed step-size integrator and all perturbations

for the variable step-size integration to emphasize the difference between the models. The results of the fixed step-size integration can be seen in Fig. 8.7 and of the variable step-size integration can be seen in Fig. 8.8.

For the extremely eccentric HEOS-2 orbit, the performance of the fixed step-size integration using all the models is almost the same for time step-sizes smaller than 15 s at around 1 - 3 m. For time step-sizes greater than 15 s, the position errors in the Cartesian model increase faster than the position errors in the USM. By time step-sizes of 80 s, the error for both the USMs is approximately 10 m and the error for Cartesian coordinates is approximately 700 m. For time step-sizes greater than 80 s, the position errors for both USM are two orders of magnitude lower than the Cartesian model. The position errors continuously increase for all the models. At a time step-size of 300 s, the position error for Cartesian coordinates is approximately 7×10^5 m, the position error for the USM6 is approximately 1000 m, and the position error for the USM7 is approximately 2000 m. When compared to USM7, the errors in USM6 are lower for time steps greater than 75 s. For the variable step-size simulations, it can be seen in Fig. 8.8 that the Cartesian simulation provides better results than the USM. If a Pareto front is again formed with the results of the Cartesian coordinates, all the USM results are to the right of it. For small tolerances, larger time step-sizes are taken during the integration. For these cases the error in the Cartesian model is very high, while the error in the USM remains in the order of 10^2 m. However, each integration step takes less time in the Cartesian model than in the USM, and so the Cartesian model performs better even for larger tolerances when the time step-sizes become small. For this case, the performance of USM7 is superior to the performance of USM6. This is because the same relationship exists between USM6 and USM7. Even though USM6 has fewer elements than USM7 and is marginally more accurate for a given time step-size, each integration takes longer for the USM6 than for the USM7, because the kinematic equation for a quaternion is linear while the kinematic equation for MRP is quadratic.

An option for a new type of integration in this case would be to switch between the various models during the variable step-size integration. Cartesian coordinates perform better for the almost linear portions of the orbit, while the USM performs better for the highly dynamic portion of the orbit near the perigee. Thus, one of the USM models can be used for the integration near the perigee and Cartesian coordinates can be used in the remaining portion of the orbit.

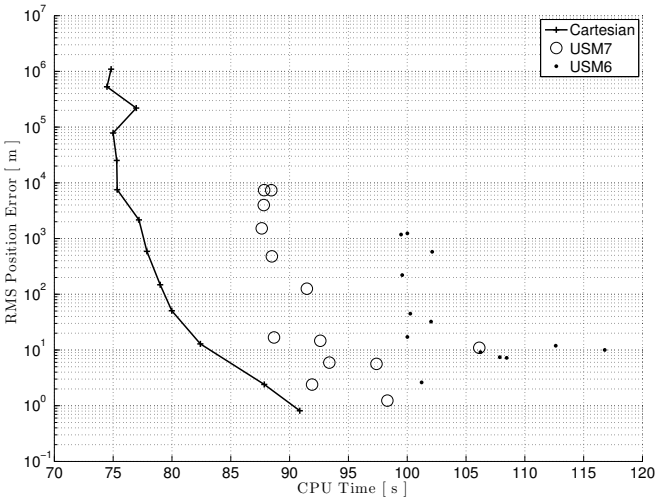


Figure 8.8: Error in position for 10 orbital revolutions of HEOS-2 with a variable step-size integrator and all perturbations

8.3 Orbits with Thrust

Many orbital trajectories also utilize thrust to change orbital parameters. A very common example of this would be a Geostationary Transfer Orbit (GTO) from a LEO to a GEO. If high-thrust, i.e., chemical propulsion is used, two burns are required. One burn increases the orbital energy and puts the satellite in the GTO, and the second burn at the GTO circularizes the orbit into a circular GEO. Another important simulation case is a low-thrust propulsion satellite. There are a few analytical solutions for low-thrust orbits such as exponential sinusoids [Petropoulos and Longuski, 2004] and inverse polynomials [Wall and Conway, 2009]. These analytical solutions suffice for a first-order approximation, but to get more accurate results, optimizers have to be used that simulate the whole trajectory. However, low-thrust orbits take a long time to simulate. Therefore, the USM could be readily applied if the CPU time for simulations would decrease. To test the performance of the USM, the following three thrusting scenarios have been utilized:

Scenario 1 One circular parking LEO at an altitude of 185 km is simulated, followed by the first burn, an elliptic GTO, a second burn, and finally one revolution at GEO. The spacecraft is under the influence of all perturbations.

Scenario 2 A circular parking orbit at an altitude of 838 km is used as the initial orbit and then simulated for 100 original orbit periods with a continuous tangential thrust of 0.0005 g (0.004905 m/s^2). No perturbations are used.

Scenario 3 A circular parking orbit at an altitude of 1000 km is used as the initial orbit. A continuous tangential thrust of 0.001 g (0.00981 m/s^2) is then applied till the spacecraft reaches GEO altitude. The spacecraft is under the influence of all perturbations.

The results for the fixed step-size integration for Scenario 1 can be seen in Fig. 8.9 and for the variable step-size integration can be seen in Fig. 8.10. The behavior of the fixed step-size integration is very similar to the HEOS-2 case. For small time step-sizes, the error is similar for all three models. When a time step-size of 5 s is used, the error of all three models is approximately 0.1 m. For time step-sizes between 10 - 40 s, the error for all three models is almost the same at around 10 m. However, for larger time step-sizes the error of the Cartesian model is again approximately two orders of magnitude larger than for the USM. For the last simulation, when a time step-size of 300 s is used, the error of the USMs is around 1000 m and the error for the

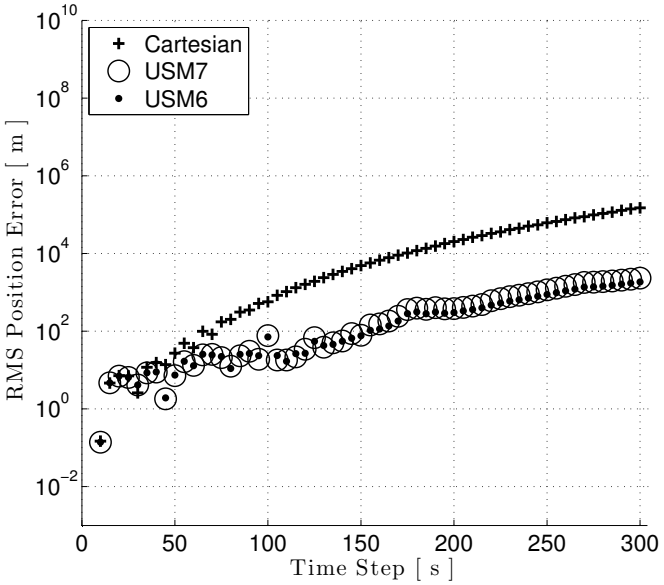


Figure 8.9: Error in position for a fixed step-size integration for the trajectory using high-thrust and all perturbations

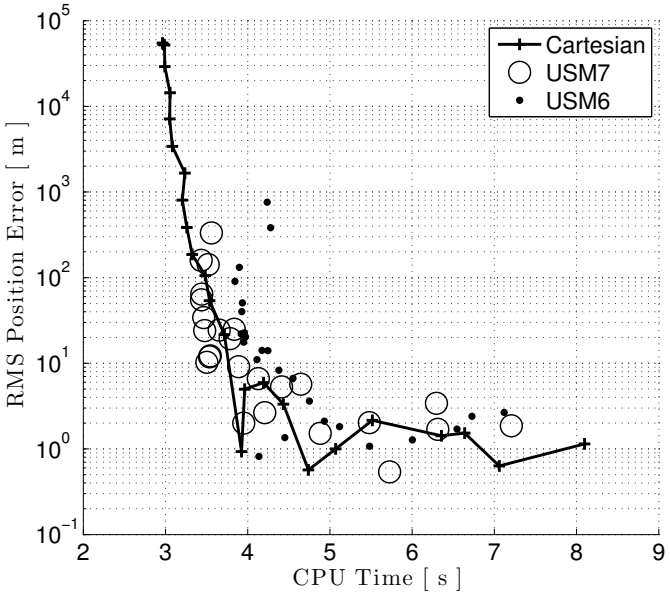


Figure 8.10: Error in position for a variable step-size integration for the trajectory using high-thrust and all perturbations

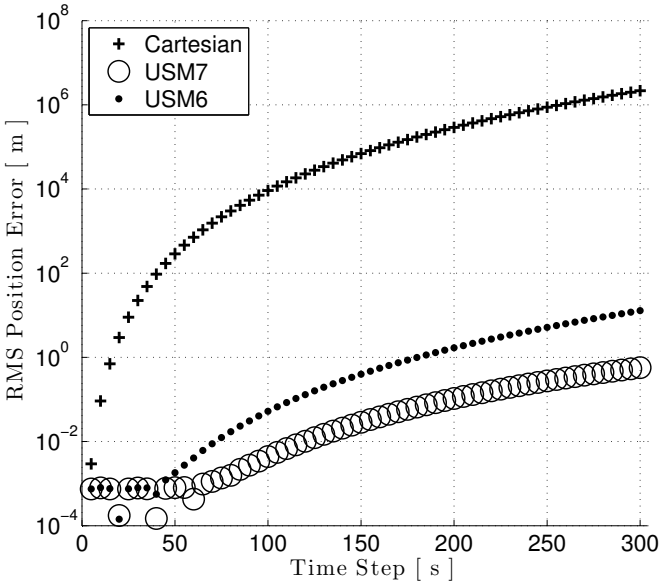


Figure 8.11: Error in position for tangential low-thrust orbit with a fixed step-size integrator

Cartesian coordinates is around 10^5. The error for the USM6 is again marginally lower than the error for the USM7. The results for the variable step-size integration show that the error is similar for USM7 and the Cartesian model, with the results of the USM7 being distributed equally on both sides of the front. The front, and all the USM results are quite erratic because of the guidance law enforced. This does not allow the variable step-size integrator to freely pick the suitable step-sizes. Since each time step takes longer for USM6 when compared to USM7, the points are shifted slightly to the right. Thus, most of the results can be considered to be inferior to the USM7 and the Cartesian model.

For Scenario 2, no other perturbations were used so that the abilities of the USM to handle low-thrust orbits could be showcased. For the low thrust case, no conversions have to be made to the Cartesian model to find the perturbations as seen in Fig. 8.2. The results of the fixed step integrator can be seen in Fig. 8.11 and of the variable step size integrator can be seen in Fig. 8.12.

For the case of low-thrust orbits with tangential thrusting, the position error using a fixed step-size integrator is many orders of magnitude smaller for the USM than for the Cartesian model. For time steps up to 55 s, the error for the USM7 integration is at a converged value of around 10^{-3}. For the USM6, this is the case up till a time step of 40 s. For time steps larger than this, the error for both the USMs increases. However, the error for the USM6 is higher than the error for the USM7. At a time step-size of 300 s, the error for the USM6 is roughly 10 m and the error for the USM7 is approximately 0.8 m. At a time step-size of 300 s, the error with the Cartesian coordinates is roughly 2×10^6.

For the variable step-size integrations, the position error for USM7 and USM6 converge after a CPU time of approximately 4 s and 10 s respectively. For integration using the Cartesian model, the error has not yet converged after a CPU time of approximately 100 s. To emphasize the difference in performance between the Cartesian model and USM, the evolution of the specific energy error of the Cartesian model for various time step-sizes divided by the average USM7 specific energy error for that particular time step-size is shown in Fig. 8.13. For these time step-sizes, the USM7 energy is in the order of 10^{-2} J/kg. For time step-sizes of 5 s, 15 s, and 30 s, the error in the energy for integration using cartesian coordinates is 1, 3, and 5 orders of magnitude higher than the average energy error in USM7, respectively. This shows that the USM is more suited for simulating trajectories under the influence of continuous low-thrust. For the variable step-size integration, if a front is formed by the USM7 results, it can be seen that the

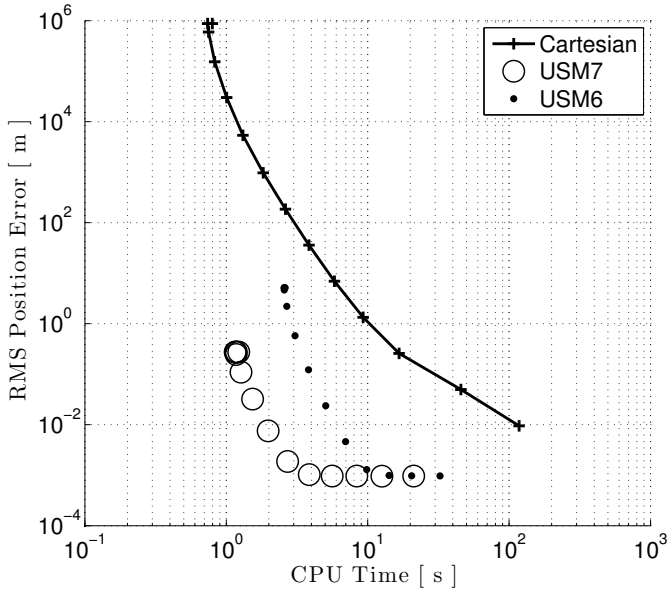


Figure 8.12: Error in position for tangential low-thrust orbit with a variable step-size integrator

USM6 results are to the right. For a given accuracy, the variable step-size integration using the USM6 is roughly 3 time slower than the USM7. The integration using Cartesian coordinates is 1 - 2 orders of magnitude slower than the USM7.

In reality, low thrust orbits also have perturbations. Thus, a full test of the USM would be to have a low-thrust orbit with perturbations as found in Scenario 3. The results of the fixed step-size integration can be seen in Fig. 8.14 and of the variable step-size integration can be seen in Fig. 8.15.

The results with low thrust and other perturbations are very similar to the results found for only low-thrust propulsion. Again, the USMs perform better than the Cartesian model. For Cartesian coordinates and the USM7, the behavior of both fixed-step and variable step-size integration is the same as for the only low-thrust propulsion case. For the USM6, however, the performance for the fixed step-size integration is much better than for the only low-thrust propulsion case. For fixed step-size integration, the error is even smaller than for the USM7. At a time step-size of 300 s, the error for the USM6 is roughly 0.1 m compared to the 1 m for the USM7. For time step-sizes up till 65 s, the error for both the USMs is at a converged value of roughly 10^{-3}. When the error is at this converged value, both the USMs exhibit a sawtooth-like behavior. Once the time step-sizes increase, the error of the USM7 starts increasing more than the error of the USM6. It should be noted that the USM6 exhibits sawtooth-like behavior for almost all the time step-sizes.

For the variable step-size integration, the accuracy of the USMs is again much higher than that of Cartesian coordinates. For a given accuracy, the CPU time of the USMs is an order of magnitude less than that of Cartesian coordinates. The performance of both the USMs is very similar, with the USM6 being marginally more accurate.

8.4 Optimization of a Solar Sailing Mission

Optimization of the trajectory of a mission to the Solar poles using Solar sailing has been in carried out [Mooij et al., 2006]. This work will be repeated using in USM7 in [Mooij et al., 2010]. The setup is presented here along with a few tentative results. It should be noted that the majority of the work, and all of the implementation for [Mooij et al., 2010] will not be carried out by the author, but by [Mooij, 2010].

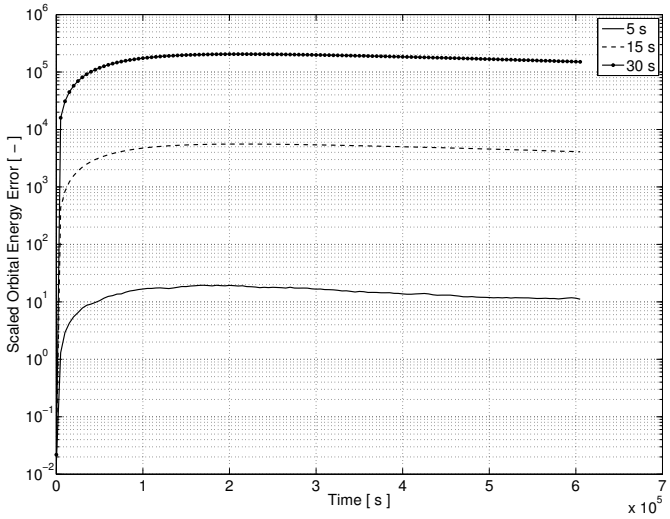


Figure 8.13: The evolution of the energy error in integration using the Cartesian model scaled with respect to the average USM7 energy error for time step-sizes of 5 s, 15 s, and 30 s

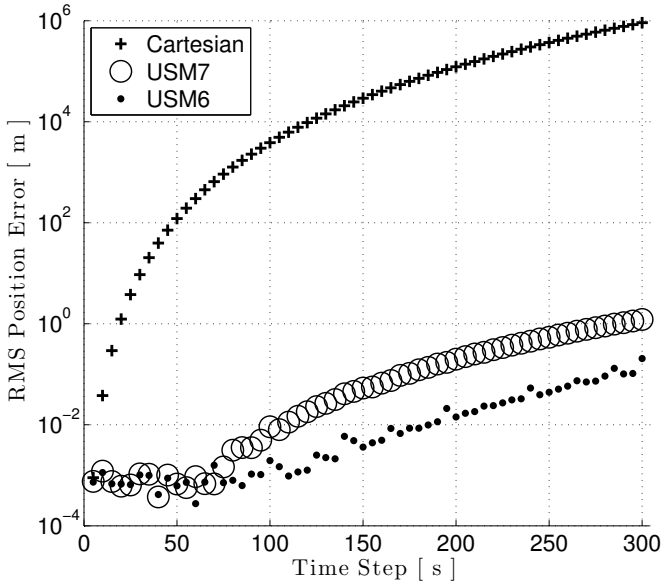


Figure 8.14: Error in position for a low-thrust orbit raising maneuver with a fixed step-size integrator and all perturbations

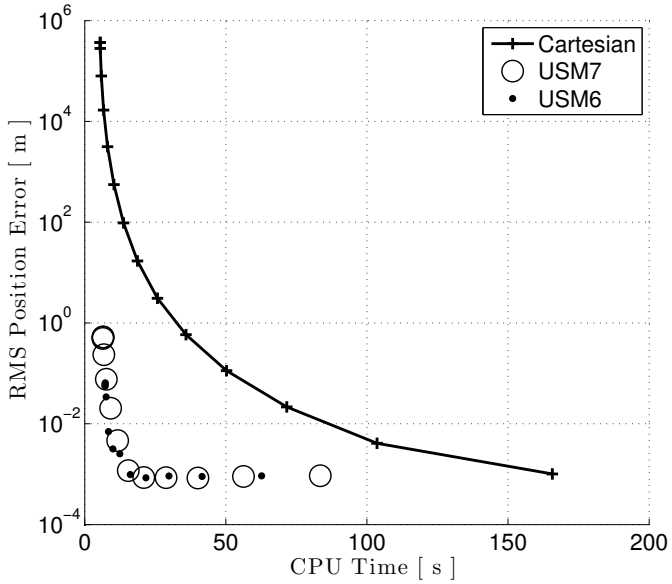


Figure 8.15: Error in position for a low-thrust orbit raising maneuver with a variable step-size integrator and all perturbations

Work Carried Out

[Mooij et al., 2006] uses Evolutionary Optimization (EO) instead of Genetic Algorithms (GA), and uses a realistic model for the solar sail. It is shown that using a perfect solar sail model gives solutions that are more optimistic, as can be seen in Fig .8.16.

EO is very similar to optimization using GA because they both follow the Darwinian philosophy of "survival of the fittest". There is, however, one major difference between the two. GA techniques represent the positions in the search space in terms of binary code, while EO uses the actual floating numbers. According to [Michalewicz, 1996], EO techniques are usually more efficient and produce more realistic results when compared to GA. The objective for the optimizer to minimize is the total mission cost, while still satisfying the constraint of arriving at a solar polar orbit by 2020.

This mission has the spacecraft starting off in a GTO , followed by an spiraling orbit to escape from the Earth in using solar propulsion. Finally, there is an heliocentric phase that cranks the orbit and increases the inclination to achieve the desired polar orbit and makes the spacecraft spiral inwards towards the sun.

There are many more specific factors that need to be taken into account to fully implement this mission. They include perturbations, guidance of the sail, modeling of the sail, etc. The reader is referred to [Mooij et al., 2006] for further information. The final result of the optimization process is a vehicle of 218.05 kg with a sail area of 6424 m and a travel time of 9.46 years. The launch date is January 2011 and the arrival date is July 2020. The geocentric and heliocentric phases of the optimum trajectory can be seen in Fig. 8.17 and Fig. 8.18, respectively.

Work in Progress

As mentioned previously, [Mooij et al., 2006] finds an optimum trajectory with a flight time of 9.46 years which is approximately 3×10^8 s. Throughout this whole trajectory, the spacecraft is using solar sailing as a method of low-thrust propulsion. As has been shown so far in this chapter, integration of trajectories utilizing low-thrust propulsion is faster using the USM than Cartesian coordinates. Thus, in [Mooij et al., 2010] the USM7 will be used to optimize the

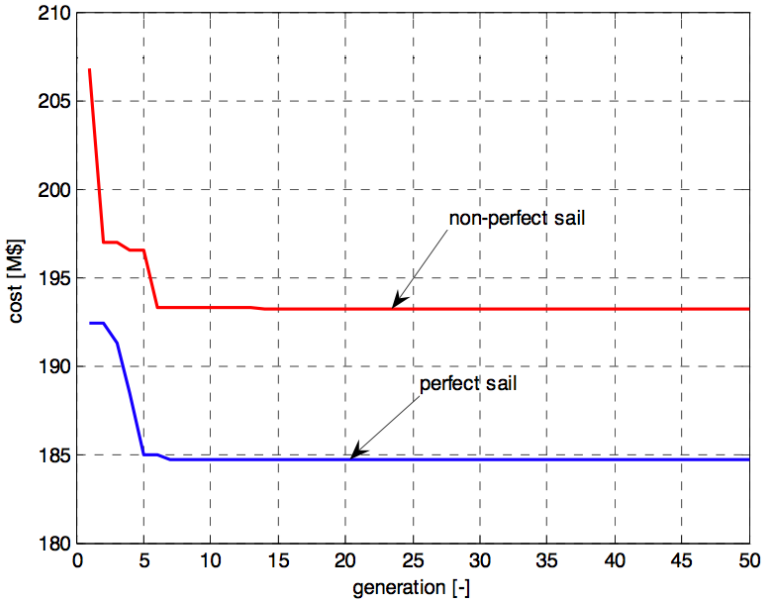


Figure 8.16: Differences in mission cost between a realistic and a perfect solar sail model [Mooij et al., 2006]

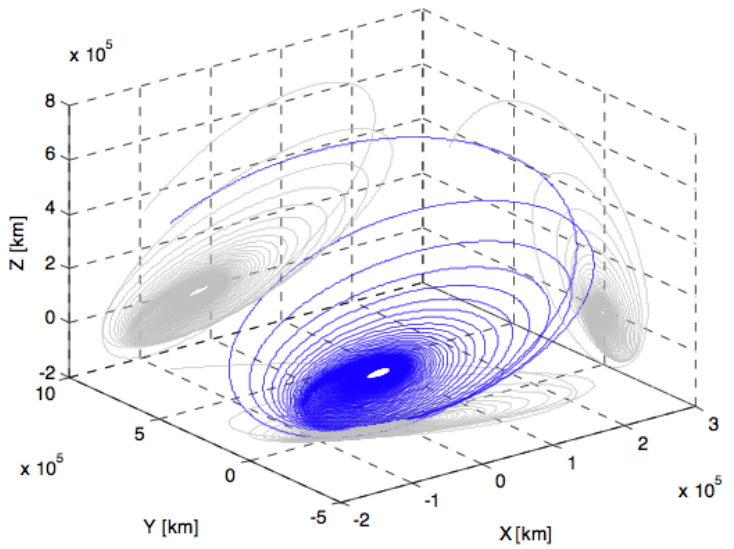


Figure 8.17: Optimum trajectory in the Geocentric frame [Mooij et al., 2006]

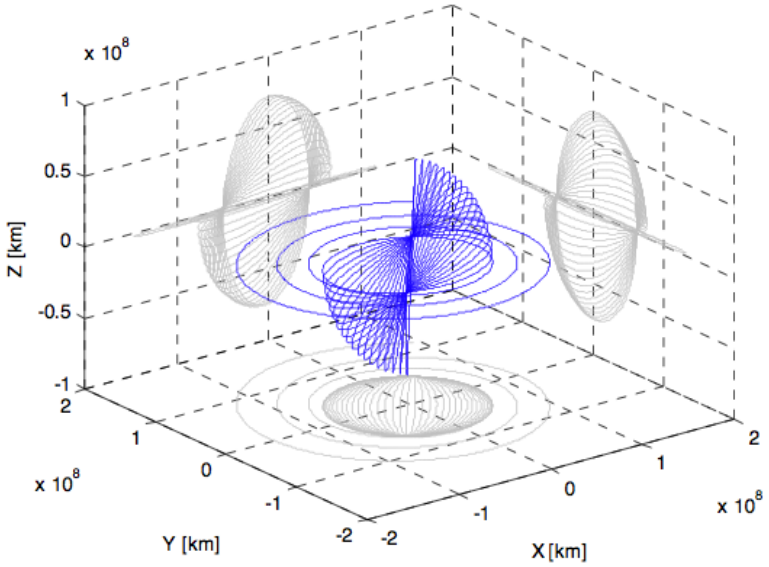


Figure 8.18: Optimum trajectory in the Heliocentric frame [Mooij et al., 2006]

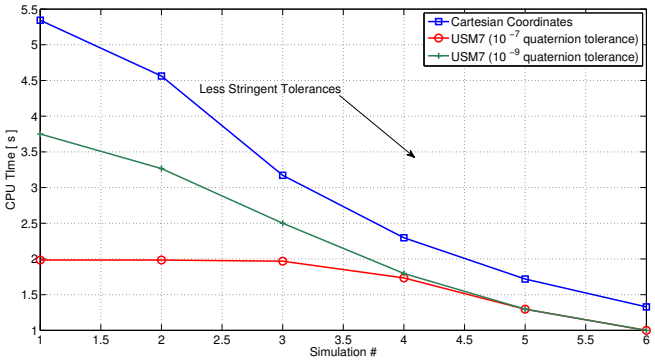


Figure 8.19: The CPU time for various simulations of a solar sailing mission

mission from [Mooij et al., 2006] and the results will be compared. Since the integration of the trajectory can be carried out faster using USM7, the optimization can be carried out faster.

A so-called truth model was created of a trajectory using Cartesian coordinates and an RK7(8) integrator. This integrator has a relative tolerance of 10^{-9} and a maximum time step-size of 1000 s. This gives a total Time of Flight (TOF) of approximately 3652.5 days, which is approximately 10 years. The same trajectory was integrated again using the same guidance laws, but this time an RK5(6) integrator was used instead of the RK7(8). The integration was carried out with Cartesian coordinates and with the USM7 for various tolerances. This integrator uses an absolute and a relative tolerance. The maximum time step-size was 10^7 s. For Cartesian coordinates, the velocity and position absolute tolerances are 0.1. For the the USM7, there were two cases used. Both the cases had an absolute tolerance of 0.1 for the hodographic velocity components. The first case of the USM7 had an absolute tolerance of 10^{-7} for the quaternion elements and the second case of the USM7 has an absolute tolerance of 10^{-9}. For the Cartesian and the two USM7 cases, the relative tolerance is changed from 10^{-9} to 10^{-4} and the integration is carried out. The CPU Time for the various cases can be seen in Fig. 8.19. The TOF for the various cases can be seen in Fig. 8.20 and the TOF mismatch when compared to the TOF of the truth model can be seen in Fig. 8.21.

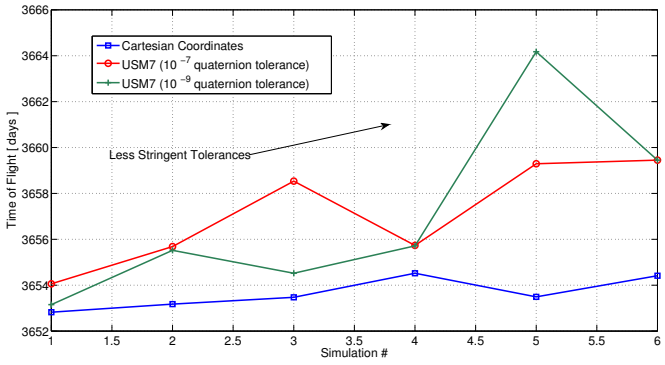


Figure 8.20: The time of flight for various simulations of a solar sailing mission

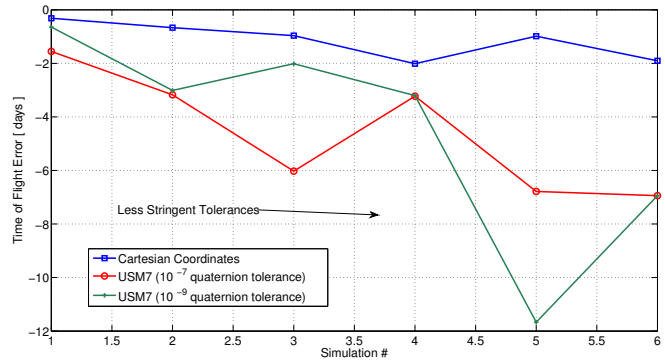


Figure 8.21: The time of flight mismatch for various simulations of a solar sailing mission

In Fig. 8.19, it can be seen that making the tolerance more stringent increases the CPU time. For a relative tolerance of 10^{-4}, the CPU time of both the USM7 cases is approximately 1 s and of the Cartesian case is approximately 1.3 s. As the tolerance becomes more stringent, the CPU times for all cases increase, as expected. However, for a tolerance including 10^{-7} and lower, the CPU time for the USM with an absolute tolerance of 10^{-7} for the quaternion elements tapers off at around 2 s. For the USM with an absolute tolerance of 10^{-9} for the quaternion elements requires increasing CPU times as the tolerances get more stringent and a final CPU time of 3.75 s for a relative tolerance of 10^{-9}. The Cartesian coordinates case also requires increasing CPU times as the tolerances get more stringent and a final CPU time of 5.3 s. From this graph, it can be concluded that when a normal step-size controller is used for the USM7, the absolute tolerance of the quaternion elements is very important. Making this tolerance more stringent forces the integrator to take smaller time steps, especially when more stringent relative tolerances are also used. When the relative tolerance is not very stringent, making the absolute tolerance more stringent does not have any effect. With the most stringent relative tolerance, the Cartesian coordinates case takes 2.65 times the CPU time of the USM7 case with an absolute quaternion tolerance of 10^{-7}. The Cartesian coordinates case takes 1.41 times the CPU time of the USM7 case with an absolute quaternion tolerance of 10^{-9}.

The same information can be gotten out of Figs. 8.20 and 8.21. All the cases have a larger TOF than the truth case. This, however, does not necessarily mean that there is an error. The integration method checks if the present state has passed the stop criterion, e.g. 0.4 AU. If the present state of the integrator has not passed the stop criterion, the next integration step is made. The maximum time step-size is 10^7 s, which is approximately 116 days. This means that the integration step can have a large time step-size and therefore, the TOF when the integration stops can be much higher than that of the truth model. Once the integration step has been made, it is of course possible to interpolate back to the correct stop criterion location in time. This was not carried out due to time constraints. This overshooting is a bigger problem for less stringent tolerances as the time step-sizes will be larger. Also, the overshooting is more for the USM7 cases than for the Cartesian coordinates, which hints that larger time step-sizes are being used.

To see how accurate the simulations are, the TOF should be kept constant and the position or velocity errors should be checked against the truth model. As the work on solar sailing is ongoing, this can be done in the future.

8.5 Conclusions and Recommendations

It was found that the USM7 and USM6 can be used for all the applications that the Cartesian coordinates are used for, with very few adaptations. For unperturbed orbits, the behavior of the USM is excellent because the orbital energy and the angular momentum are conserved. The USM can also be used for parabolic and hyperbolic trajectories. Hyperbolic trajectories are well represented within the sphere of influence of planets and thus, the patched conic method can be used for the USM for interplanetary trajectories. Other than the singularity present for pure retrograde orbits, there are no other theoretical scenarios where the USM cannot be used. However, the true anomaly limit for hyperbolic orbits causes a practical limit due to the slow change in the true anomaly near this limit.

For perturbed orbits, the USM performs better than Cartesian coordinates for both fixed-step and variable-step integration. For very small time step sizes, the USM and Cartesian results are very similar. However, the error of the USM is much lower when larger time steps are used. This can also be seen in the results for the variable-step integration as the USM performs much better than the Cartesian model with smaller simulation times. The exception to this is the highly eccentric HEOS-2 orbit. In this case, the satellite spends much more time in an almost linear trajectory. Integration in Cartesian coordinates is inherently better suited for this as each state advancement is linear. For many commonly used orbits, however, the USM still performs better than Cartesian coordinates. For example, a GTO, with an eccentricity of around 0.7, still has

more accurate results when the USM is used.

The scenario where the USM truly shines is low-thrust propulsion. When the thrust is the only perturbing force, the USM7 has position errors that are 5 - 6 orders of magnitude lower than the Cartesian model, when using a variable step size integrator. When also other perturbations are present, the performance of the USM6 is slightly superior to the performance of the USM7, and both the USM have errors many orders of magnitude lower than the Cartesian model.

As the USM performs much better than Cartesian coordinates for low eccentricity and continuous thrust orbits, its usage in optimizers is highly recommended. For an optimization problem many different trajectories have to be simulated, which make the computation time very large. Thus, using the USM would help in reducing this CPU load. In particular, low thrust trajectories with electric propulsion or solar sailing would be the optimal target group of the USM. Because of the stability, usage of the USM for long-term orbit simulation is also recommended. The fact that the USM performs better for low eccentricity orbits should not be a deterrent as orbits of most satellites and debris are circular. When perturbations that exert a very small and erratic acceleration, there is a sawtooth-like behavior shown for the fixed step-size simulations using the USM. However, the error of the USM models is still many orders of magnitude lower than the Cartesian model. Thus, the sawtooth-like behavior should not pose much of a problem. Between USM7 and USM6, it is recommend to use the original USM7 model for numerical integration of orbits with only low-thrust propulsion, highly dynamic short term simulations like the high-thrust orbit scenario, and for very highly eccentric orbits like the HEOS-2. For low-thrust orbits with other perturbations and low eccentricity orbits with perturbations, the use of USM6 is recommended.

When implementing the dynamics function for the USM6 and the USM7, it is important to validate them with respect to an unperturbed analytical, and a perturbed orbit before they can be used to generate trajectories. For Cartesian coordinates the dynamics function is quite simple to implement and choosing an unperturbed, eccentric, and inclined orbit will sufficiently excite all the time derivatives. For the USM6 and USM7 however, an unperturbed orbit only causes a change in the MRP and quaternion, respectively. Unless a perturbed orbit is chosen, $\dot{C}$, $\dot{R}_{f1}$, $\dot{R}_{f2}$, and ω_1 will not be excited. Thus, any bug in the dynamics regarding those elements would not be found.

In the future, investigation must be carried out to find out exactly why the traditional step-size control method is not so effective for the USM. Also, if it is decided to fully switch to the USM7 or the USM6, the perturbation models should be converted to the USM. This would obviate the need to switch to Cartesian coordinates.

This concludes the chapter about numerical simulations of orbital trajectories using the USM with the RK method of integrating. However, to get a complete overview of the USM's applicability to numerical simulations, atmospheric re-entry should also be investigated. This is done in the following chapter.

Chapter 9

Re-entry Dynamics

The USM was shown to have superior performance in numerical integration when compared to Cartesian coordinates for most orbit propagation cases. In [Altman, 1972], there is an assertion that the USM can also be used for the simulation of atmospheric re-entry. Therefore, a simulation of an atmospheric reentry is carried out here to check if this is indeed possible, and also to see whether the USM can outperform Cartesian coordinates. This chapter presents a brief investigation carried out on applicability of the USM for re-entry.

The atmospheric re-entry is a highly complex and dynamic environment. This problem becomes even more complex when a winged vehicle is utilized. Normally, a full 6 DOF simulator is used along with guidance algorithms to get a meaningful trajectory that satisfies the requirements. To circumvent the need to build this simulator and the guidance algorithms, a profile of the aerodynamic angles and control surface deflections was taken from [Mooij, 1998] and [Mooij and Chu, 2002]. The reference vehicle for the trajectory is the HORUS-2B, which can be seen in Fig. 9.1. The HORUS-2B is an unpowered winged re-entry vehicle.

The characteristics of the HORUS-2B can be found in [Grallert, 1988] and a few important ones are presented in Table 9.1. The profile of all the aerodynamic angles and the control surface deflections can be seen in Fig. 9.2, and the altitude vs. relative velocity plot can be seen in Fig. 9.3.

9.1 Re-entry Dynamics

In this work, only a 3 DOF simulator is considered. Thus, all the dynamics felt during the trajectory because of the attitude of the vehicle are only introduced as perturbing accelerations to the vehicle. The integration is carried out numerically using Cartesian coordinates in ECI, or using the USMs. The procedure is as follows:

First, the position of the vehicle is converted from ECI to ECEF using the DCM $\mathbf{C}_{ECEF,ECI}$ from Eq. (4.33) on page 31. This equation is repeated below for the sake of convenience of the reader. In this chapter, $\mathcal{F}_{ECEF}$ will be referred to as $\mathcal{F}_R$ since it is a rotating frame of reference.

$$\mathbf{C}_{ECEF,ECI} = \begin{bmatrix} \cos\left(\theta_{J2000} + \omega_E \cdot \Delta t\right) & \sin\left(\theta_{J2000} + \omega_E \cdot \Delta t\right) & 0 \\ -\sin\left(\theta_{J2000} + \omega_E \cdot \Delta t\right) & \cos\left(\theta_{J2000} + \omega_E \cdot \Delta t\right) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Both in ECI and ECEF, it can be assumed that the Earth is rotating with the following angular velocity vector

$$\boldsymbol{\omega}_E = \begin{bmatrix} 0 \\ 0 \\ \omega_E \end{bmatrix} \tag{9.1}$$

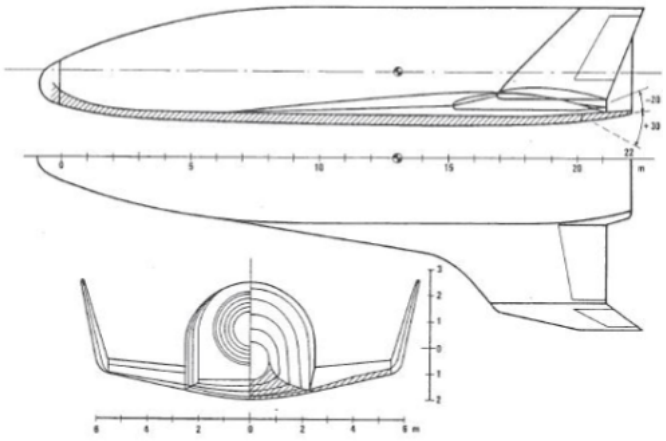


Figure 9.1: The Horus-2B [Grallert, 1988]

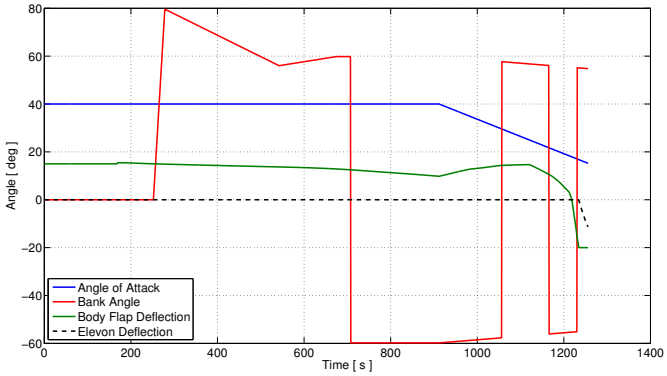


Figure 9.2: The profile of the aerodynamic angles and control surfaces deflections provided by [Mooij, 1998]

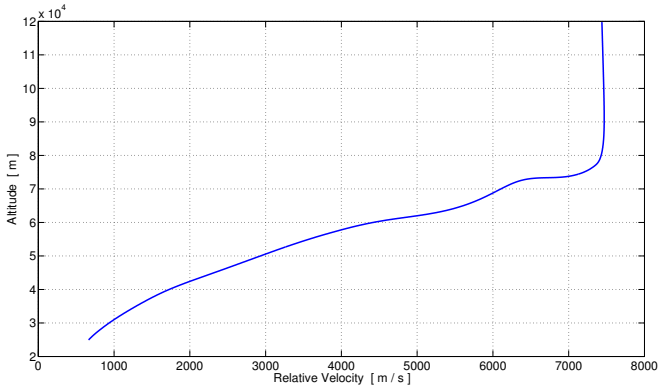


Figure 9.3: Altitude against relative velocity in the ECEF frame of the trajectory provided by [Mooij, 1998]

In Eq. (9.1), the value of ω_E can be found in Table 4.1 on page 32. Because ECEF is not an

Table 9.1: Characteristics of the HORUS-2B [Grallert, 1988]

Property	Value
Vehicle Length	25 m
Fuselage Length	23 m
Maximum Fuselage Width	5.4 m
Height	4.5 m
Wingspan, b_{ref}	13 m
Wing chord, c_{ref}	23 m
Wing area, S_{ref}	110 m^2
Center-of-mass	13 m from the nose
Maximum payload mass	7,000 kg
Re-entry mass	26,029 kg
I_{xx}	119,000 kg m^2
I_{yy}	769,000 kg m^2
I_{zz}	806,000 kg m^2
I_{xy}	0 kg m^2
I_{xz}	−20,372 kg m^2
I_{yz}	0 kg m^2

inertial frame, the relative velocity in ECEF of the vehicle can be found in the following way

$$\mathbf{v}_R = \mathbf{C}_{ECEF,ECI} \times (\mathbf{v}_I - \boldsymbol{\omega}_E \times \mathbf{r}_I) \tag{9.2}$$

In Eq. (9.2), $\mathbf{r}_I$ and $\mathbf{v}_I$ are the position and velocity vectors of the vehicle in ECI, respectively. The vectors $\mathbf{r}_R$ and $\mathbf{v}_R$ are the position and the relative velocity ECEF, respectively.

The assumption of a spherical Earth is made for this simulation. The latitude and longitude can now be found in the following way

$$\phi_{gc} = \sin\left(\frac{z_R}{\sqrt{x_R^2 + y_R^2 + z_R^2}}\right) \tag{9.3a}$$

$$\lambda_{gc} = \arctan(y_R/x_R) \tag{9.3b}$$

The following step is to compute the DCM from $\mathcal{F}_R$ to the local horizontal frame $\mathcal{F}_V$. The local horizontal frame is analogous to $\mathcal{F}_{NED}$ found in section 4.5.2 on page 34. The DCM $\mathbf{C}_{V,R}$ can be found from the product of $\mathbf{C}_{ENU,ECEF}$ and $\mathbf{C}_{NED,ENU}$ found in Eqs. (4.43) and (4.44). Motion in the local horizontal plane can be seen in Fig. 9.4.

$$\mathbf{C}_{V,R} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \times \begin{bmatrix} -\sin\lambda & \cos\lambda & 0 \\ -\cos\lambda\sin\phi_{gd} & -\sin\phi_{gd}\sin\lambda & \cos\phi_{gd} \\ \cos\phi_{gd}\cos\lambda & \cos\phi_{gd}\sin\lambda & \sin\phi_{gd} \end{bmatrix} \tag{9.4}$$

The next step is to convert the relative velocity $\mathbf{v}_R$ to $\mathcal{F}_V$ to give $\mathbf{v}_V$. The three orthogonal velocity components of $\mathbf{v}_V$ are $[v_N, v_E, v_D]^T$. Using these velocities, the heading angle with respect to the ground based trajectory, χ_G, and the flight path angle with respect to the ground based trajectory, γ_G can be calculated. These angles can be seen in Fig. 9.4.

$$\chi_G = \arctan(v_E/v_N) \tag{9.5}$$

$$\gamma_G = -\arcsin\left(\frac{v_D}{\sqrt{\mathbf{v}_R^T\mathbf{v}_R}}\right) \tag{9.6}$$

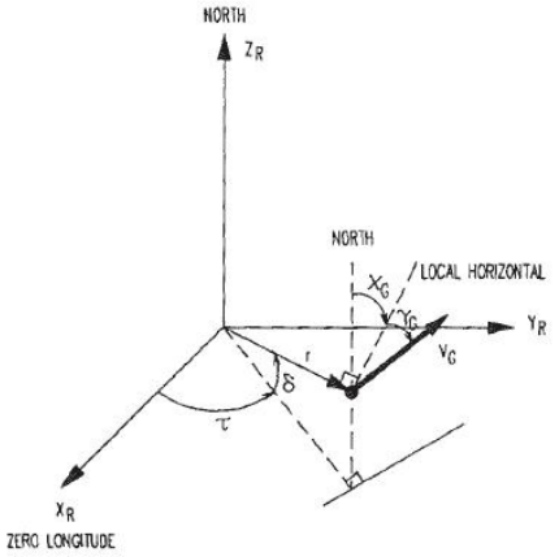


Figure 9.4: Motion in the local horizontal plane [Mooij, 1998]

A new reference frame called the ground based trajectory frame, $\mathcal{F}_{TG}$ can be defined using χ_G and γ_G. The DCM for this reference frame from $\mathcal{F}_V$ is calculated using

$$\mathbf{C}_{TG,V} = \mathbf{C}_2(\gamma_G)\mathbf{C}_3(\chi_G) \tag{9.7}$$

During this work, it is assumed that there is no wind. Therefore, the following elementary relations exists for the flight path angle with respect to the aerodynamic trajectory and the heading angle with respect to the aerodynamic trajectory.

$$\chi_A = \chi_G$$
$$\gamma_A = \gamma_G$$

To compute the aerodynamic forces, it is essential to know the atmospheric density, ρ, and the Mach number, M. The density can be found from a standard atmospheric model such as the U.S. Standard Atmosphere 1976. M is the ratio between the current aerodynamic velocity, v_A, and the speed of sound v_a. Because it is assumed that there is no wind, v_A is equal to the magnitude of $\mathbf{v}_R$. The speed of sound can also be found from the U.S. Standard Atmosphere 1976 data. The Mach number is then

$$M = \frac{v_A}{v_a} \tag{9.8}$$

The dynamic pressure has to also be calculated based on the density and the relative velocity.

$$q_{dyn} = \frac{1}{2}\rho v_A^2 \tag{9.9}$$

Using the dynamic pressure and assuming that there is no sideslip, the perturbing accelerations felt by the vehicle in the aerodynamic reference frame can be calculated using

$$\mathbf{a}_{aero,A} = -\frac{q_{dyn}S_{ref}}{m}\begin{bmatrix} C_D \\ 0 \\ C_L \end{bmatrix} \tag{9.10}$$

In Eq. (9.10), C_D and C_L are the drag and lift coefficients of the vehicle, respectively. For real vehicles, they are not constant but depend on many parameters. They can be written as

$$C_D = C_{D0}(\alpha, M) + C_{D\delta_{bf}}(\alpha, \delta_{bf}, M) + C_{D\delta_e}(\alpha, \delta_e, M) \tag{9.11a}$$

$$C_L = C_{L0}(\alpha, M) + C_{L\delta_{bf}}(\alpha, \delta_{bf}, M) + C_{L\delta_e}(\alpha, \delta_e, M) \tag{9.11b}$$

The angles in Eq. (9.11) are:

α	Angle of attack
δ_{bf}	Body flap deflection
δ_e	Elevon deflection
C_{D0}	The drag coefficient of the vehicle in the default configuration
C_{L0}	The lift coefficient of the vehicle in the default configuration
$C_{D\delta bf}$	The extra drag coefficient of the vehicle due to a deflection of the body flap
$C_{L\delta bf}$	The extra lift coefficient of the vehicle due to a deflection of the body flap
$C_{D\delta e}$	The extra drag coefficient of the vehicle due to a deflection of the elevon
$C_{L\delta e}$	The extra lift coefficient of the vehicle due to a deflection of the elevon

The C_{D0} and C_{L0} only depend on the angle of attack and the Mach number. The $C_{D\delta bf}$ and $C_{L\delta bf}$ depend on the body flap deflection angle as well as the angle of attack and Mach number. The $C_{D\delta e}$ and $C_{L\delta e}$ depend on the elevon deflection angle as well as the angle of attack and Mach number. The total lift and drag coefficients are the sum of the lift and drag coefficients due to the default vehicle configuration, the body flap deflection, and the elevon deflection. The angle of attack, body flap deflection, and the elevon deflection are interpolated from the reference trajectory, based on the time since start of simulation. The lift and drag coefficients from Eq. (9.11) are then found using data tables taken from [Mooij, 1998].

The accelerations computed in Eq. (9.10) need to be converted from $\mathcal{F}_A$ to $\mathcal{F}_I$ to be used in the integrators. According to [Mooij, 1997], the DCM to convert from $\mathcal{F}_A$ to $\mathcal{F}_V$ is

$$\mathbf{C}_{V,A} = \mathbf{C}_3(-\chi_A)\mathbf{C}_2(-\gamma_A)\mathbf{C}_1(\sigma) \tag{9.12}$$

In Eq. (9.12), $\mathbf{C}_i(\theta)$ means an principal axis rotation with axis i being the Euler axis and angle θ being the Euler angle. The acceleration can be converted to $\mathcal{F}_I$ in the following way

$$\mathbf{a}_{aero,I} = \mathbf{C}_{I,R}\mathbf{C}_{R,V}\mathbf{C}_{V,A}\mathbf{a}_{aero,A} \tag{9.13}$$

9.2 Simulations

For the simulations, the same procedure was used as in Chapter 8. The perturbations implemented were J_2 and the atmospheric drag as explained in section 9.1. To calculate the re-entry dynamics, a function was made that takes the present state in Cartesian coordinates as the input, and outputs the acceleration due to the aerodynamic forces in the inertial frame. Thus, this function could be used in the same manner as all the other orbital perturbations. Tests were then carried out using the fifth order variable step-size and fixed step-size integrators using Cartesian coordinates and the USMs.

The results of the fixed step simulation can be seen in Fig. 9.5. The time step-sizes were varied from 1 s to 11 s. When a time step-size of 1 s is used, the RMS position error for all three models is approximately 0.3 m. The reference trajectory and angles were all also given in 1 s intervals. Once the time step-size increased to 2 s, the RMS error jumps to 100 m for all the models. As the time step-size keeps increasing, the position error for all models also increases. At the largest time step-size of 11 s, the RMS position error is approximately 10^4 m for all models. At all time step-sizes the error for all the models is roughly the same. For time step-sizes larger than 6 s, there is a difference between the models but there is no pattern. For some time step-sizes, the Cartesian coordinates are more accurate than the USMs and vice-versa.

The results of the variable time step-size simulations can be seen in Figs. 9.6 and 9.7. If a front is formed by the Cartesian coordinate results in Fig. 9.6, the results of both the USMs are to

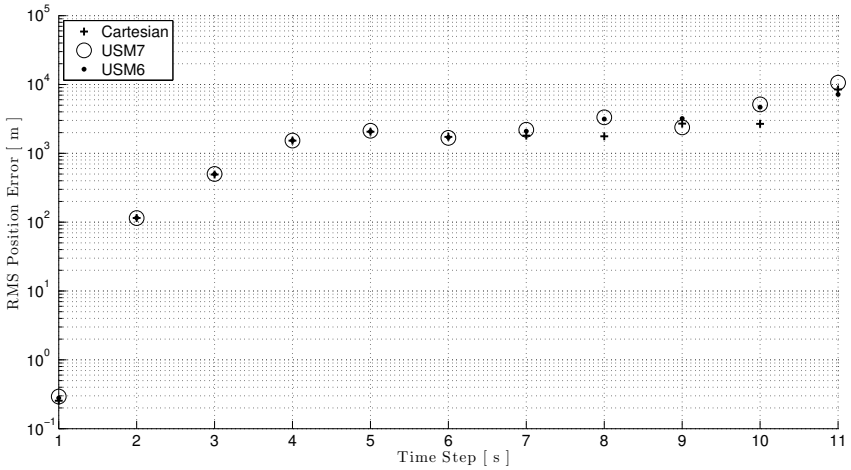


Figure 9.5: RMS of position error again time step-size for an atmospheric re-entry simulation using a fixed step integrator

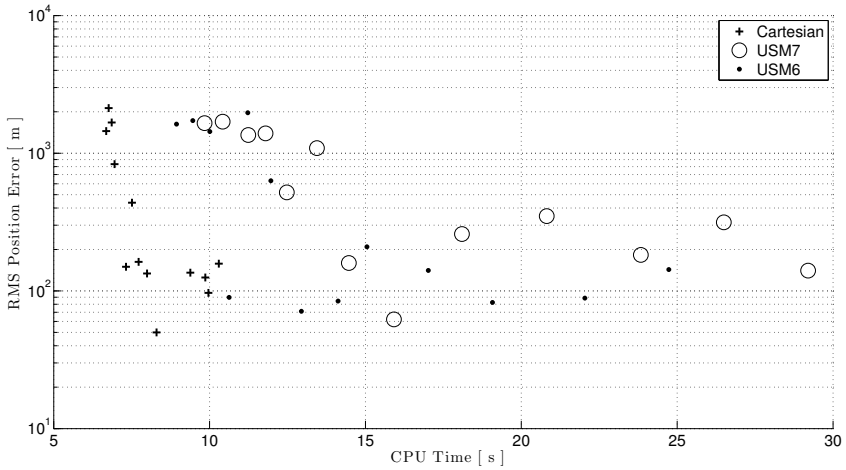


Figure 9.6: RMS of position error against CPU time for an atmospheric re-entry simulation using a variable step-size integrator

the right. This means that the USMs are take more CPU time than the Cartesian coordinates to achieve the same accuracy. On average, the USMs take twice as much CPU time as the Cartesian coordinates. Fig. 9.7 shows the average time step-sizes taken by the variable step-size integrators. The ellipses show where the results for the different models are roughly located. The maximum average step-size taken by the USMs is around 7.5 s, while the Cartesian coordinates can have an average time step-size of up to 9.5 s. It is already known that one integration step using Cartesian coordinates is faster than one integration step using any of the USMs. If the variable step-size integrator using Cartesian coordinates have larger average time step-sizes, it means that they are making fewer integration steps. Thus, integration with the Cartesian coordinates is faster for re-entry than with the USMs.

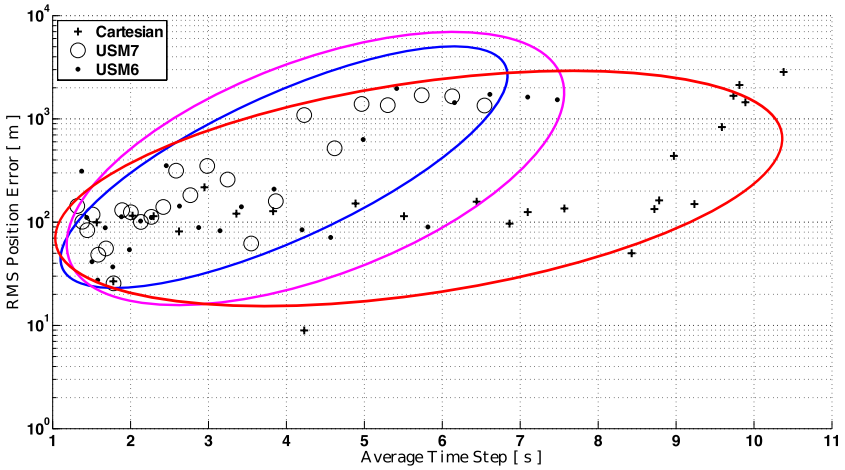


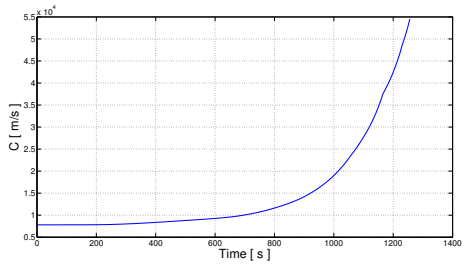
Figure 9.7: RMS of position error against average time step-size for an atmospheric re-entry simulation using a variable step-size integrator. The results of the Cartesian coordinates, USM7, and USM6 are approximately bounded by the red, blue, and magenta outlines, respectively

9.3 Conclusions

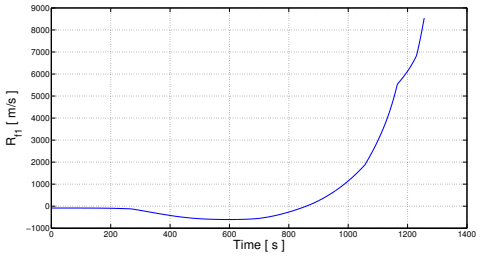
It is stated in [Altman, 1972] that the USM can be used for orbit propagation and atmospheric re-entry. The conjecture in [Altman, 1972] has been shown to be true in this chapter. Indeed, it is possible to use both the USM7 and the USM6 to carry out simulations of a vehicle re-entering the atmosphere as can be seen in Figs. 9.5 and 9.6. For simulations using a fixed time step-size integrator, the RMS of the position error is approximately the same for Cartesian coordinates, USM7, and USM6. However, it can be seen in Fig. 9.6 that the CPU time for USM7 and USM6 is higher than for Cartesian coordinates when a variable time step-size integrator is used. This would also be the case for the fixed time step-size integration because each integration step takes longer for the USM than for Cartesian coordinates.

Furthermore, it can be seen in Fig. 9.7 that the integrator using Cartesian coordinates is capable of using larger step-sizes than the integrator using any of the USMs. Fig. 9.7 plots the RMS position error against the average time-step, which is the mean of the step-sizes during a variable step-size integration run, for various tolerances. This is the case because atmospheric re-entry is a highly dynamic environment for the vehicle to experience. This highly dynamic environment makes the trajectory of the vehicle very *un-orbitlike*. Due to this, the same reasons that make the USM highly effective for orbital trajectory simulation, make the USM uncompetitive for re-entry. Cartesian coordinates are a very generic form for expressing position and velocity. Given an origin, any position and velocity can be expressed relative to it. The USM, however, is based on fundamental assumption of a body in orbit. Three of the state elements are from the velocity hodograph for orbital motion, and the other 4 elements express the orientation of the orbital frame. During an actual orbit, these elements have a physical meaning and vary relatively slowly. During re-entry however, the vehicle is no longer in an orbit. Thus, the USM elements may behave in a more chaotic manner as can be seen in Figs. 9.8 and 9.9. These factors combined make the USM less suitable than Cartesian coordinates for integration of re-entry trajectories. It is perfectly possible to continue using the USM for a spacecraft going from the orbital phase to the re-entry phase. However, using Cartesian coordinates would decrease CPU time.

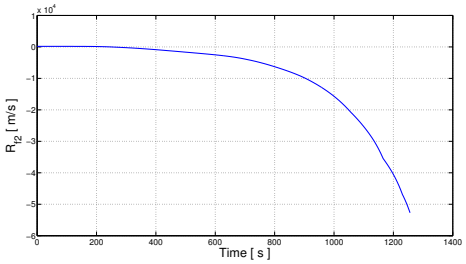
This re-entry problem also uses guidance history that is provided at a certain output. If this history is smaller than the default time step-sizes taken by the integrator, the integrator is either forced to take small time step-sizes or some of the commands get missed. If the integrator is forced to take smaller time step-sizes, the USMs will lose their advantage over the Cartesian coordinates. If some guidance commands get missed, then the error in the simulation will be



(a) Evolution of C during the re-entry trajectory

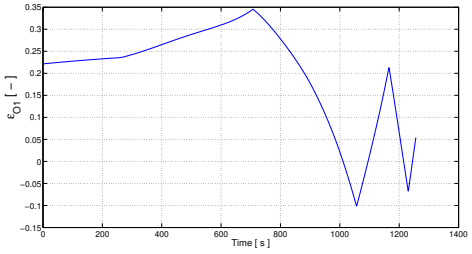


(b) Evolution of R_{f1} during the re-entry trajectory

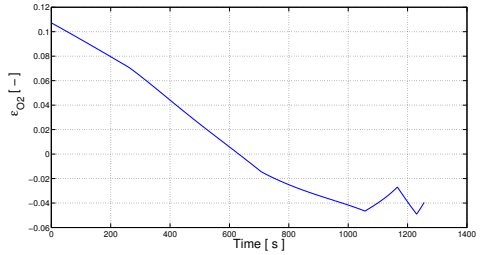


(c) Evolution of R_{f2} during the re-entry trajectory

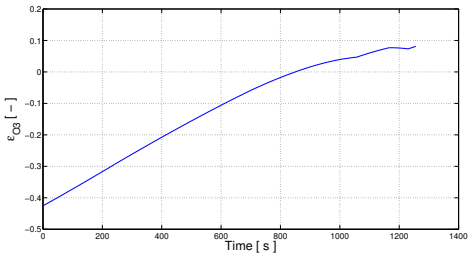
Figure 9.8: Evolution of the hodographic velocities during the re-entry trajectory



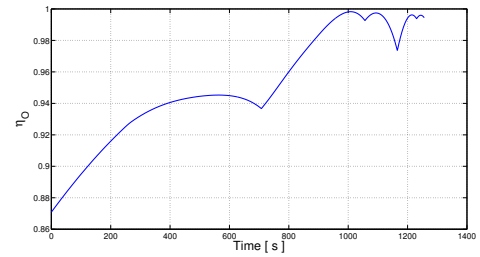
(a) Evolution of ϵ_{O1} during the re-entry trajectory



(b) Evolution of ϵ_{O2} during the re-entry trajectory



(c) Evolution of ϵ_{O3} during the re-entry trajectory



(d) Evolution of η_O during the re-entry trajectory

Figure 9.9: Evolution of the quaternion elements during the re-entry trajectory

large for all models, which is also not desirable.

Between the USM7 and the USM6, it is recommended to use USM6. For variable step-size integrations, it is capable of utilizing larger time step-sizes and still achieving the same accuracy as the USM7. Each integration step using the USM6 takes more CPU time than an integration step using the USM7. However, the benefit of the USM6 is that the MRP use only three elements to describe an orientation, instead of four like the quaternion. This means that all the MRP parameters are independent of each other and thus, the effect of numerical inaccuracies is smaller when compared to quaternions, which have the problem of deviating from unit norm.

This chapter has shown the performance of the USM for the numerical integration of an atmospheric re-entry trajectory. With this, the topic of numerical integration using conventional RK integrators is concluded. The next part of this thesis study focuses on filtering and navigation, so that aspects of the USM, other than numerical integration performance, can be investigated.

Chapter 10

Recursive Filtering Techniques

The previous chapter on atmospheric re-entry was the last chapter to use RK methods for integration. It was concluded that the performance of the USM is inferior to Cartesian coordinates for re-entry dynamics. The focus of the thesis study for this chapter and the next is on filtering and estimation.

Theoretical equations always give exact answers. In real life, however, not everything is so straight forward. When a system is modeled, there are always some discrepancies compared to real life. It is not possible to predict everything exactly. The goal of navigation is to be able to accurately estimate the state of a spacecraft. The word, estimate, is very important here, because it is also not possible to measure the state exactly. Through all their efforts, engineers are only able to estimate, albeit with a very high accuracy, the state of the spacecraft. Since Navigation is extremely important for satellites, the properties of the USM for Navigation should be investigated. Filtering techniques play a pivotal part in Navigation and they are therefore presented here.

If the estimation does not have to be carried out in real time, a method such as least squares can be used. There are plenty of cases where it is necessary to get estimated parameters in real time. For this purpose, it is much more efficient to use recursive filtering techniques. Thus, only recursive filtering techniques will be dealt with in this section. The main reason why estimation is such an important process is because of noise. Noise is the randomness that occurs in the real world, which does not provide exact measurements. This is also the cause for the uncertainty in the modeling of systems. The most famous filters for real-time estimation are the Kalman Filter (KF) and its derivatives. In recent years there has been development on a new type of filter known as the Divided Difference Filter (DDF). In this chapter, these two types of filters will be dealt with. It is assumed that the reader is familiar with some basic probability theory, as this is essential for the understanding of these filters.

10.1 Kalman Filter

The Kalman filter was the brainchild of Rudolf Emil Kalman. It has since then become the most widely used estimation technique, especially in the aerospace field. The original Kalman filter is the Linear Kalman Filter (LKF), which, as the name suggests, can only be applied to linear systems. Most systems are, however, quite non-linear. Thus, there have been additions to the KF that extended the domain of validity to nonlinear systems. One of the additions is the Extended Kalman Filter (EKF), which is the de facto estimator in the aerospace field. Another more recent addition is the Unscented Kalman Filter (UKF), which is gaining popularity in recent times. For an excellent introduction to Kalman filtering, the reader is referred to [Welch and Bishop, 2001].

10.1.1 Linear Kalman Filter

As the LKF is the original Kalman filter, it will be derived in this section. This derivation is based on the one found in [Welch and Bishop, 2001], but with added detail. The goal of a Kalman filter is to estimate a linear system

$$\mathbf{x}_{k+1} = \mathbf{A}_k\mathbf{x}_k + \mathbf{B}_k\mathbf{u}_{k+1} + \mathbf{w}_k \tag{10.1}$$

with measurements in the form of

$$\mathbf{z}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k \tag{10.2}$$

In Eqs. (10.1) and (10.2), $\mathbf{w}_k$ and $\mathbf{v}_k$ represent the system and measurement noise respectively. They are Gaussian noises, so they have a normal distribution with zero mean. The variables $\mathbf{x}$, $\mathbf{u}$, and $\mathbf{z}$ are the state, control input, and measurement respectively. $\mathbf{A}$ is the state transition matrix, and it defines how the state at the next step is related to the current state. $\mathbf{B}$ describes the relation between the input and the state at the next stop. $\mathbf{H}$ is the measurement matrix and shows how the measurement can be computed from the state.

Both noises are assumed to be uncorrelated to each other. Their probability distributions are

$$p(\mathbf{v}_k) \sim N(0, \mathbf{R}_k) \tag{10.3}$$
$$p(\mathbf{w}_k) \sim N(0, \mathbf{Q}_k) \tag{10.4}$$

$\mathbf{R}$ is the noise present in the measurement device and $\mathbf{Q}$ is the uncertainty present in the modeling of the system. $\mathbf{R}$ is usually available from the manufacturer of the measurement device. $\mathbf{Q}$, on the other hand, can only be derived by testing, or from the intuition of the engineer.

The estimation process can be split into two steps. The first step is to get an *a priori* estimate, and the second step is to get an *a posteriori* estimate. The *a priori* estimate only takes into account the system noise, this means that the next state is estimated using only the model. For navigation purposes, the time derivative of the state elements can sometimes be measured and then it also occurs in the state transition. This occurs, for example, when an IMU, or velocity measurements are used. In this case the measurement noise is incorporated into the system noise. The *a posteriori* estimate then updates and refines the *a priori* estimate using the measurement. The *a priori* and *a posteriori* estimates are expressed as $\hat{\mathbf{x}}_k^-$ and $\hat{\mathbf{x}}_k$ respectively. The *a priori* and *a posteriori* errors can be defined as

$$\mathbf{e}_k^- \equiv \mathbf{x}_k - \hat{\mathbf{x}}_k^- \tag{10.5}$$
$$\mathbf{e}_k \equiv \mathbf{x}_k - \hat{\mathbf{x}}_k \tag{10.6}$$

The error between the true state and the estimated state can be expressed as an uncertainty in terms of a covariance matrix. These error covariance matrices are

$$\mathbf{P}_k^- = \mathrm{E}\left[\mathbf{e}_k^-\mathbf{e}_k^{-T}\right] \tag{10.7}$$
$$\mathbf{P}_k = \mathrm{E}\left[\mathbf{e}_k\mathbf{e}_k^T\right] \tag{10.8}$$

Since the *a priori* estimate of the state is not perfect and there is noise present in the measurements, it is not possible to predict the measurements with full accuracy. To predict the measurement, noise is not taken into account as the expected value of noise is 0. The predicted measurement is

$$\hat{\mathbf{z}}_k = \mathbf{H}_k\hat{\mathbf{x}}_k^- \tag{10.9}$$

The difference between the actual measurement and the predicted measurement is known as the measurement innovation, $\delta\mathbf{z}_k^-$.

$$\delta\mathbf{z}_k^- = \mathbf{z}_k - \hat{\mathbf{z}}_k = \mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_k^- \tag{10.10}$$

Given an initial, or an *a posteriori* estimate, the *a priori* state estimate of the next state is found via

$$\hat{\mathbf{x}}_{k+1}^{-} = \mathbf{A}_k \hat{\mathbf{x}}_k + \mathbf{B}_k \mathbf{u}_k \tag{10.11}$$

The *a priori* covariance can be found using the proper linear covariance propagation law.

$$\mathbf{P}_{k+1}^{-} = \mathbf{A}_k \mathbf{P}_k \mathbf{A}_k^T + \mathbf{Q}_k \tag{10.12}$$

The aim of the Kalman filter is to improve the state estimation by optimally incorporating the information found in the measurement. The update can be expressed in terms of the sum of the *a priori* state and the product of a gain with the measurement innovation.

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^{-} + \mathbf{K}_k \delta \mathbf{z}_k^{-} = \hat{\mathbf{x}}_k^{-} + \mathbf{K}_k \left(\mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_k^{-} \right) \tag{10.13}$$

As can be seen, it is important to find the value of the gain, $\mathbf{K}$, which shall now be derived.

The *a posteriori* error can be expanded as

$$\begin{aligned}
\mathbf{e}_k &= \mathbf{x}_k - \hat{\mathbf{x}}_k \\
&= \mathbf{x}_k - \hat{\mathbf{x}}_k^{-} - \mathbf{K}_k \left(\mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_k^{-} \right) \\
&= -\left(\mathbf{I} - \mathbf{K}_k \mathbf{H}_k \right) \hat{\mathbf{x}}_k^{-} - \mathbf{K}_k \mathbf{z}_k + \mathbf{x}_k \\
&= -\left(\mathbf{I} - \mathbf{K}_k \mathbf{H}_k \right) \hat{\mathbf{x}}_k^{-} - \mathbf{K}_k \left(\mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k \right) + \mathbf{x}_k \\
&= -\hat{\mathbf{x}}_k^{-} - \mathbf{K}_k \mathbf{H}_k \hat{\mathbf{x}}_k^{-} - \mathbf{K}_k \mathbf{H}_k \mathbf{x}_k - \mathbf{K}_k \mathbf{v}_k + \mathbf{x}_k \\
&= \left(\mathbf{x}_k - \hat{\mathbf{x}}_k^{-} \right) + \mathbf{K}_k \mathbf{H}_k \left(\hat{\mathbf{x}}_k^{-} - \mathbf{x}_k \right) - \mathbf{K}_k \mathbf{v}_k \\
&= \left(\mathbf{I} - \mathbf{K}_k \mathbf{H}_k \right) \left(\mathbf{x}_k - \hat{\mathbf{x}}_k^{-} \right) - \mathbf{K}_k \mathbf{v}_k
\end{aligned}$$

Using the definition of the *a priori* error from Eq. (10.6), the *a posteriori* error can be written as

$$\mathbf{e}_k = \left[\mathbf{I} - \mathbf{K}_k \mathbf{H}_k \right] \mathbf{e}_k^{-} - \mathbf{K}_k \mathbf{v}_k \tag{10.14}$$

The *a posteriori* covariance matrix can be written as

$$\mathbf{P}_k = \mathrm{E}\left[\mathbf{e}_k \mathbf{e}_k^T \right] = \left(\mathbf{I}_k - \mathbf{K}_k \mathbf{H}_k \right) \mathbf{P}_k^{-} \left(\mathbf{I} - \mathbf{K}_k \mathbf{H}_k \right)^T + \mathbf{K}_k \mathbf{R}_k \mathbf{K}_k^T \tag{10.15}$$

In the covariance matrix, the diagonal terms express the variance of the state elements. The off-diagonal terms express the covariance between the state elements. To make sure that the estimation is optimal, the variance has to be brought to a minimum value. A measure of the variance of all the state elements is the trace of the covariance matrix. Thus, the trace has to be brought to a minimum value. To get the trace, the relation for the covariance found in Eq. (10.15) has to be expanded.

$$\begin{aligned}
\mathbf{P}_k &= \mathrm{E}\left[\mathbf{e}_k \mathbf{e}_k^T \right] = \left(\mathbf{I}_k - \mathbf{K}_k \mathbf{H}_k \right) \mathbf{P}_k^{-} \left(\mathbf{I} - \mathbf{K}_k \mathbf{H}_k \right)^T + \mathbf{K}_k \mathbf{R}_k \mathbf{K}_k^T \\
&= \left(\mathbf{I} - \mathbf{K}_k \mathbf{H}_k \right) \mathbf{P}_k^{-} \left(\mathbf{I} - \mathbf{H}_k^T \mathbf{K}_k^T \right) + \mathbf{K}_k \mathbf{R}_k \mathbf{K}_k^T \\
&= \left(\mathbf{P}_k^{-} - \mathbf{K}_k \mathbf{H}_k \mathbf{P}_k^{-} \right) \left(\mathbf{I} - \mathbf{H}_k^T \mathbf{K}_k^T \right) + \mathbf{K}_k \mathbf{R}_k \mathbf{K}_k^T \\
&= \mathbf{P}_k^{-} - \mathbf{K}_k \mathbf{H}_k \mathbf{P}_k^{-} - \mathbf{P}_k^{-} \mathbf{H}_k^T \mathbf{K}_k^T + \mathbf{K}_k \mathbf{H}_k \mathbf{P}_k^{-} \mathbf{H}_k^T \mathbf{K}_k^T + \mathbf{K}_k \mathbf{R}_k \mathbf{K}_k^T
\end{aligned}$$

The trace is

$$Tr\left(\mathbf{P}_k \right) = Tr\left(\mathbf{P}_k^{-} \right) - 2 \cdot Tr\left(\mathbf{K}_k \mathbf{H}_k \mathbf{P}_k^{-} \right) + Tr\left(\mathbf{K}_k \left(\mathbf{H}_k \mathbf{P}_k^{-} \mathbf{H}_k^T \right) \mathbf{K}_k^T \right) + Tr\left(\mathbf{K}_k \mathbf{R}_k \mathbf{K}_k^T \right) \tag{10.16}$$

To find the gain that minimizes the trace of the covariance matrix, the derivative has to be 0.

$$\frac{\partial \left(Tr\left(\mathbf{P}_k \right) \right)}{\partial \mathbf{K}_k} = \mathbf{0} \tag{10.17}$$

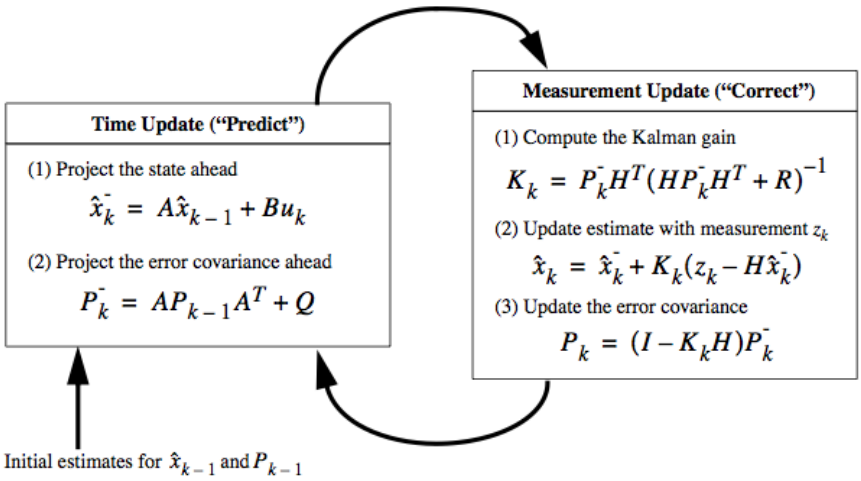


Figure 10.1: Implementation of a Kalman Filter [Welch and Bishop, 2001]

To compute the derivative, the following 2 trace calculation rules are used

$$\frac{\partial Tr\left(\mathbf{M}_1\mathbf{M}_2\mathbf{M}_1^T\right)}{\partial \mathbf{M}_1} = 2\mathbf{M}_1\mathbf{M}_2 \qquad \text{Only applicable for symmetric } \mathbf{M}_2 \tag{10.18}$$

$$\frac{\partial Tr\left(\mathbf{M}_1\mathbf{M}_2\right)}{\partial \mathbf{M}_1} = \mathbf{M}_2^T \tag{10.19}$$

This results in the following condition for the gain

$$\frac{\partial\left(Tr\left(\mathbf{P}_k\right)\right)}{\partial \mathbf{K}_k} = -2\left(\mathbf{H}_k\mathbf{P}_k^-\right)^T + 2\mathbf{K}_k\left(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^T\right) + 2\mathbf{K}_k\mathbf{R}_k = 0 \tag{10.20}$$

Eq. (10.20) can then be rearranged to give the optimal gain.

$$\begin{aligned}
&-2\left(\mathbf{H}_k\mathbf{P}_k^-\right)^T + 2\mathbf{K}_k\left(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^T\right) + 2\mathbf{K}_k\mathbf{R}_k = 0 \\
&\mathbf{K}_k^{-1}\left(\mathbf{H}_k\mathbf{P}_k^-\right)^T = \left(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^T\right) + \mathbf{R}_k \\
&\mathbf{K}_k^{-1} = \left(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^T + \mathbf{R}_k\right)\left(\mathbf{P}_k^-\mathbf{H}_k^T\right)^{-1}
\end{aligned}$$

Taking the inverse, finally gives the value of the optimal gain

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k^T\left(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^T + \mathbf{R}_k\right) \tag{10.21}$$

The optimal gain found in Eq. (10.21) provides a more compact version of the *a posteriori* covariance matrix.

$$\mathbf{P}_k = \left(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k\right)\mathbf{P}_k^- \tag{10.22}$$

This concludes the derivation for the optimal recursive filter for linear systems, the Kalman filter. The way the filter works and the equations can be seen in Fig. 10.1.

10.1.2 Extended Kalman Filter

The Linear Kalman Filter, as its name implies, is only useful for linear systems. Most systems that are modeled are nonlinear, thus rendering the LKF useless. The EKF is one way to adapt the Kalman filtering techniques to nonlinear systems. The EKF works by using a first order Taylor series approximation of the nonlinear system around the current estimate.

This derivation of the EKF is based on the one in [Welch and Bishop, 2001]. The nonlinear system is expressed by the following nonlinear equation

$$\mathbf{x}_{k+1} = \mathbf{f}\left(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k\right) \tag{10.23}$$

with the measurement

$$\mathbf{z}_k = \mathbf{h}\left(\mathbf{x}_k, \mathbf{v}_k\right) \tag{10.24}$$

The noise parameters are the same as in Eqs. (10.3) and (10.4).

$$p(\mathbf{v}_k) \sim N(0, \mathbf{R}_k)$$
$$p(\mathbf{w}_k) \sim N(0, \mathbf{Q}_k)$$

The *a priori* estimate can be derived from the previous *a posteriori* estimate using

$$\hat{\mathbf{x}}_{k+1}^{-} = \mathbf{f}\left(\hat{\mathbf{x}}_k, \mathbf{u}_{k+1}, \mathbf{0}\right) \tag{10.25}$$

The measurement can be estimated using

$$\hat{\mathbf{z}}_k = \mathbf{h}\left(\hat{\mathbf{x}}_k^{-}, \mathbf{0}\right) \tag{10.26}$$

The state and the measurement can be approximated by the linear equations

$$\mathbf{x}_{k+1} \approx \hat{\mathbf{x}}_{k+1}^{-} + \mathbf{A}_k\left(\mathbf{x}_k - \hat{\mathbf{x}}_k\right) + \mathbf{W}_k\mathbf{w}_k \tag{10.27}$$
$$\mathbf{z}_k \approx \hat{\mathbf{z}}_k + \mathbf{H}_k\left(\mathbf{x}_k - \hat{\mathbf{x}}_k^{-}\right) + \mathbf{V}_k\mathbf{v}_k \tag{10.28}$$

In Eqs. 10.27 and 10.28, the terms $\mathbf{A}$, $\mathbf{H}$, $\mathbf{V}$, and $\mathbf{W}$ are Jacobians that are used to linearize the system.

$$\mathbf{A}_k = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{(\hat{\mathbf{x}}_k, \mathbf{u}_k, \mathbf{0})} \tag{10.29}$$
$$\mathbf{H}_k = \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right|_{(\hat{\mathbf{x}}_k^{-}, \mathbf{0})} \tag{10.30}$$
$$\mathbf{V}_k = \left.\frac{\partial \mathbf{h}}{\partial \mathbf{v}}\right|_{(\hat{\mathbf{x}}_k^{-}, \mathbf{0})} \tag{10.31}$$
$$\mathbf{W}_k = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{w}}\right|_{(\hat{\mathbf{x}}_k, \mathbf{u}_{k+1}, \mathbf{0})} \tag{10.32}$$

The Jacobians have to be derived analytically beforehand and then the values have to be computed at each step. There is a way to also compute them numerically by using numerical differentiation techniques such as complex-step differentiation [Martins et al., 2000]. Numerical differentiation obviates the need to derive the Jacobians, but is more computationally intensive and possibly less accurate and stable for application.

A prediction error, which is equivalent to the *a priori* estimation error of the linear Kalman filter, is defined as

$$\hat{\mathbf{e}}_{xk}^{-} \equiv \mathbf{x}_k - \hat{\mathbf{x}}_k^{-} \approx \mathbf{A}_{k-1}\left(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1}\right) + \boldsymbol{\varepsilon}_k \tag{10.33}$$

The measurement residual, which is equivalent to the measurement innovation of the linear Kalman filter, is defined as

$$\hat{\mathbf{e}}_{zk}^{-} \equiv \mathbf{z}_k - \hat{\mathbf{z}}_k \approx \mathbf{H}_k\left(\hat{\mathbf{e}}_{xk}^{-}\right) + \boldsymbol{\eta}_k \tag{10.34}$$

The variables $\boldsymbol{\varepsilon}$ and $\boldsymbol{\eta}$ are the linear approximations of the $\mathbf{w}$ and $\mathbf{v}$ respectively. Using the linear variance propagation laws, the distribution of these two noise parameters is

$$p(\boldsymbol{\varepsilon}_k) \sim N\left(\mathbf{0}, \mathbf{W}_{k-1}\mathbf{Q}_{k-1}\mathbf{W}_{k-1}^T\right) \tag{10.35}$$

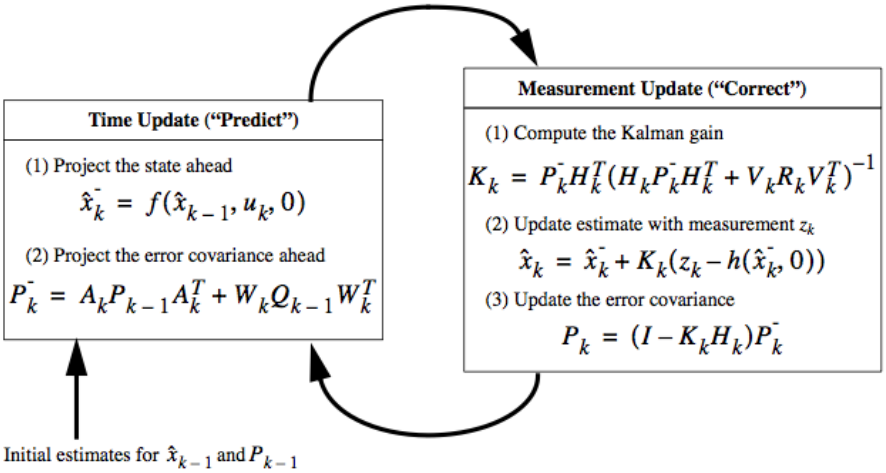


Figure 10.2: Implementation of an Extended Kalman Filter [Welch and Bishop, 2001]

$$p(\boldsymbol{\eta}_k) \sim N\left(\mathbf{0}, \mathbf{V}_k \mathbf{R}_k \mathbf{V}_k^T\right) \tag{10.36}$$

The prediction error estimate is defined as the difference between the *a posteriori* and the *a priori* errors.

$$\hat{\mathbf{e}}_k \equiv \hat{\mathbf{x}}_k - \hat{\mathbf{x}}_k^- \tag{10.37}$$

This prediction error estimate should be equivalent to the state update. Using a Kalman gain, this becomes

$$\hat{\mathbf{e}}_k = \mathbf{K}_k \left(\hat{\mathbf{e}}_{zk}^-\right) = \mathbf{K}_k \left(\mathbf{z}_k - \hat{\mathbf{z}}_k^-\right) \tag{10.38}$$

The *a posteriori* state estimate can now be written as

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k \left(\mathbf{z}_k - \hat{\mathbf{z}}_k^-\right) \tag{10.39}$$

This results in the same kind of equations as the linear Kalman filter with an important difference. The difference being that the noise parameters are replaced by their linearized counterparts. The equation for the covariance propagation is

$$\mathbf{P}_{k+1}^- = \mathbf{A}_k \mathbf{P}_k \mathbf{A}_k^T + \mathbf{W}_k \mathbf{Q}_k \mathbf{W}_k^T \tag{10.40}$$

The gain can then be found by

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T \left(\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{V}_k \mathbf{R}_k \mathbf{V}_k^T\right)^{-1} \tag{10.41}$$

The method used to implement an EKF can be seen in Fig. 10.2.

10.1.3 Unscented Kalman Filter

As stated previously, the EKF is the de-facto filter in the aerospace field, but this does not imply that it is the best. The EKF is only a first degree approximation and it requires that the Jacobians are derived beforehand. It should be noted that it is also possible to compute the Jacobians numerically, but as stated earlier, this tends to lower the performance and possibly accuracy and stability. There is a new kind of filter known as the Unscented Kalman Filter (UKF), which is based on the Unscented Transform (UT). The UKF was first presented in [Julier and Uhlmann, 1997] and excellent information can also be found in [Wan and Van der Merwe,

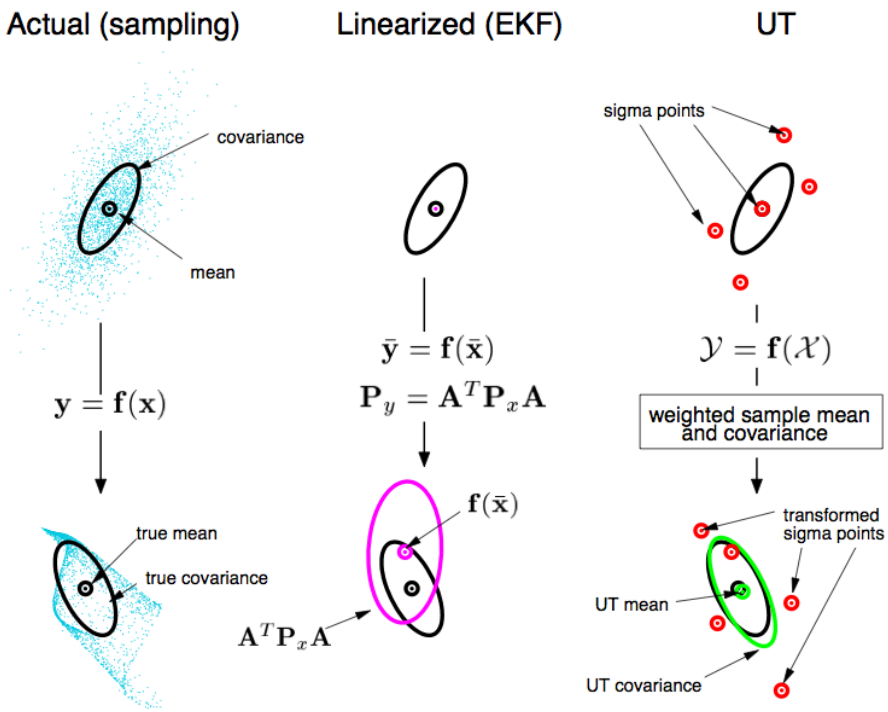


Figure 10.3: Differences in covariance transformation through a non linear function for the actual, linearized, and Unscented cases [Wan and Van der Merwe, 2000]

2000]. According to [Crassidis and Junkins, 2004], *The Unscented filter works on the premise that with a fixed number of parameters it should be easier to approximate a Gaussian distribution than to approximate an arbitrary nonlinear function.* The UT estimates the covariance of a nonlinear function up to the third order. It does this by applying the function to a set of carefully chosen points, called sigma points. The difference between the actual, linearized, and unscented covariance propagations can be seen in Fig. 10.3.

For the UKF, the same kind of nonlinear system and measurement equations are used as with the EKF. They are

$$\mathbf{x}_{k+1} = \mathbf{f}\left(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k\right)$$

$$\mathbf{z}_k = \mathbf{h}\left(\mathbf{x}_k, \mathbf{v}_k\right)$$

$$p(\mathbf{v}_k) \sim N(0, \mathbf{R}_k)$$
$$p(\mathbf{w}_k) \sim N(0, \mathbf{Q}_k)$$

It is assumed that the state is estimated with $\hat{\mathbf{x}}_k$ and covariance $\mathbf{P}_k$. The state is augmented by adding the expected value of the process and the measurement noises. It is important to note that the expected noise is always 0 because normal Gaussian noise is being considered.

$$\hat{\mathbf{x}}_k^a = \begin{bmatrix} \hat{\mathbf{x}}_k \\ \hat{\mathbf{w}}_k \\ \hat{\mathbf{v}}_k \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{x}}_k \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} \tag{10.42}$$

This augmented state has its own covariance matrix, which is

$$\mathbf{P}_k^a = \begin{bmatrix} \mathbf{P}_k & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{R}_k \end{bmatrix} \tag{10.43}$$

The noise sources are considered to be independent of each other, so there is no correlation between them. The length of the augmented state vector is L, which is the sum of the the lengths of the state and the noise vectors. The next step is to calculate the sigma points, which are the most essential points. Each sigma point, $\boldsymbol{\mathcal{X}}_{i,k}$, is a vector with dimension L. In total, $2L+1$ sigma points need to be calculated in the following way

$$\boldsymbol{\mathcal{X}}_{0,k} = \hat{\mathbf{x}}_k^a \tag{10.44a}$$

$$\boldsymbol{\mathcal{X}}_{i,k} = \hat{\mathbf{x}}_k^a + \left(\sqrt{(L+\lambda)\,\mathbf{P}_k^a}\right)_i \quad i = 1,\ldots,L \tag{10.44b}$$

$$\boldsymbol{\mathcal{X}}_{i,k} = \hat{\mathbf{x}}_k^a - \left(\sqrt{(L+\lambda)\,\mathbf{P}_k^a}\right)_{i-L} \quad i = L+1,\ldots,2L \tag{10.44c}$$

In the equation above, the i on the right hand side refers to the i th column, λ is a scaling parameter and is defined as

$$\lambda = \alpha^2\,(L+\kappa) - L \tag{10.45}$$

In Eq. (10.45), α is used to distribute the sigma points around $\hat{\mathbf{x}}^a$. κ is also a scaling parameter, but it is secondary to λ. According to [Wan and Van der Merwe, 2000], α is normally a small positive number like 0.003 and κ is 0.

Each sigma point can now be propagated using

$$\boldsymbol{\mathcal{X}}_{i,k+1}^{\mathbf{x}-} = \mathbf{f}\left(\boldsymbol{\mathcal{X}}_{i,k}^{\mathbf{x}}, \mathbf{u}_{k+1}, \boldsymbol{\mathcal{X}}_{i,k}^{\mathbf{w}}\right) \tag{10.46}$$

In Eq. (10.46), the superscripts refer to the part of the sigma vector referring to the state or noises. The *a priori* state estimate can be found from the expected value of the propagated sigma points.

$$\hat{\mathbf{x}}_{k+1}^- = \sum_{i=0}^{2L} W_i^{(m)} \boldsymbol{\mathcal{X}}_{i,k+1}^{\mathbf{x}-} \tag{10.47}$$

In Eq. (10.47), $W_i^{(m)}$ is a weight factor and is found in the following way

$$W_0^{(m)} = \frac{\lambda}{L+\lambda} \tag{10.48a}$$

$$W_i^{(m)} = \frac{1}{2(L+\lambda)} \quad i = 1,\ldots,2L \tag{10.48b}$$

The *a priori* covariance can be found in the following way

$$\mathbf{P}_{k+1}^- = \sum_{i=0}^{2L} W_i^{(c)} \left(\boldsymbol{\mathcal{X}}_{i,k+1}^{a-} - \hat{\mathbf{x}}_{k+1}^-\right)\left(\boldsymbol{\mathcal{X}}_{i,k+1}^{a-} - \hat{\mathbf{x}}_{k+1}^-\right)^T \tag{10.49}$$

In Eq. (10.49), $W_i^{(c)}$ is a weight and can be found via

$$W_0^{(c)} = \frac{\lambda}{L+\lambda} + (1 - \alpha^2 + \beta) \tag{10.50a}$$

$$W_i^{(c)} = \frac{1}{2(L+\lambda)} \quad i = 1,\ldots,2L \tag{10.50b}$$

with β providing information about the probability distribution function of the state. According to [Wan and Van der Merwe, 2000], the value of β for Gaussian distributions is 2.

For the *a posteriori* update at k, the measurement function has to be applied to the sigma points. This will create a set of estimated measurement sigma points, $\boldsymbol{\mathcal{Z}}_{i,k}$, found by

$$\boldsymbol{\mathcal{Z}}_{i,k} = \mathbf{h}\left(\boldsymbol{\mathcal{X}}_{i,k}^{\mathbf{x}-}, \boldsymbol{\mathcal{X}}_{i,k-1}^{\mathbf{v}}\right) \tag{10.51}$$

The estimated measurement can then be found by taking the expected value of the estimated measurement sigma points.

$$\hat{\mathbf{z}}_k = \sum_{i=0}^{2L} W_i^{(m)} \boldsymbol{\mathcal{Z}}_{i,k} \tag{10.52}$$

The covariance for the estimated measurement is

$$\mathbf{P}_{\mathbf{z}\mathbf{z}_k} = \sum_{i=0}^{2L} W_i^{(c)} \left(\boldsymbol{\mathcal{Z}}_{i,k} - \hat{\mathbf{z}}_k\right) \left(\boldsymbol{\mathcal{Z}}_{i,k} - \hat{\mathbf{z}}_k\right)^T \tag{10.53}$$

The covariance between the state and the estimated measurement is

$$\mathbf{P}_{\mathbf{x}\mathbf{z}_k} = \sum_{i=0}^{2L} W_i^{(c)} \left(\boldsymbol{\mathcal{X}}_{i,k}^{\mathbf{x}-} - \hat{\mathbf{x}}_k^-\right) \left(\boldsymbol{\mathcal{Z}}_{i,k} - \hat{\mathbf{z}}_k\right)^T \tag{10.54}$$

The Kalman gain can be found in the following way

$$\mathbf{K}_k = \mathbf{P}_{\mathbf{x}\mathbf{z}_k} \mathbf{P}_{\mathbf{z}\mathbf{z}_k}^{-1} \tag{10.55}$$

The *a posteriori* state estimate can be found from the *a priori* state estimate using the standard Kalman filter update equation.

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k \left(\mathbf{z}_k - \hat{\mathbf{z}}_k\right) \tag{10.56}$$

The state covariance can finally be updated in the following manner

$$\mathbf{P}_k = \mathbf{P}_k^- - \mathbf{K}_k \mathbf{P}_{\mathbf{z}\mathbf{z}_k} \mathbf{K}_k^T \tag{10.57}$$

10.2 Divided Difference Filter

The Divided Difference Filter (DDF) was first proposed in [Nøgaard et al., 2004]. This method uses Stirling's interpolation formula instead of Taylor's approximation. The definition of Stirling's interpolation method is beyond the scope of this paper and so the reader is referred to [Nøgaard et al., 2004] for more information on this subject. Stirling's interpolation is a very large topic by itself, but it is not important to know the method to be able to use the filter. The important difference between Stirling's and Taylor's method is that Stirling's method uses differences while Taylor's method used derivatives. The result is that Taylor's approximation is very accurate close to the point of linearization, but Sterling's method describes the overall function better. This difference can be seen in Fig. 10.4.

Stirling's 2$^{\text{nd}}$ order approximation of a function $\mathbf{f}(\mathbf{x})$ around $\mathbf{x} = \bar{\mathbf{x}}$ is

$$\mathbf{y} \approx \mathbf{f}(\bar{\mathbf{x}}) + \tilde{\mathbf{D}}_{\Delta\mathbf{x}}\mathbf{f} + \frac{1}{2!}\tilde{\mathbf{D}}_{\Delta\mathbf{x}}^2\mathbf{f} \tag{10.58}$$

In Eq. (10.58), $\tilde{\mathbf{D}}_{\Delta\mathbf{x}}\mathbf{f}$ and $\tilde{\mathbf{D}}_{\Delta\mathbf{x}}^2\mathbf{f}$ are the divided difference operators.

$$\tilde{\mathbf{D}}_{\Delta\mathbf{x}}\mathbf{f} = \frac{1}{h}\left(\sum_{i=1}^{n} \Delta\mathbf{x}_i \boldsymbol{\mu}_i \boldsymbol{\delta}_i\right)\mathbf{f}(\bar{\mathbf{x}}) \tag{10.59}$$

$$\tilde{\mathbf{D}}_{\Delta\mathbf{x}}^2\mathbf{f} = \frac{1}{h^2}\left(\sum_{i=1}^{n} \left(\Delta\mathbf{x}_i\right)^2 \boldsymbol{\delta}_i^2 + \sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n} \Delta\mathbf{x}_i\Delta\mathbf{x}_j \left(\boldsymbol{\mu}_i\boldsymbol{\delta}_i\right)\left(\boldsymbol{\mu}_j\boldsymbol{\delta}_j\right)\right)\mathbf{f}(\bar{\mathbf{x}}) \tag{10.60}$$

In the divided difference operators, $\hat{\mathbf{e}}_i$ refers to the ith unit vector. The partial difference operator, $\boldsymbol{\delta}_i$, and the average operator, $\boldsymbol{\mu}_i$, are

$$\boldsymbol{\delta}_i = \mathbf{f}\left(\bar{\mathbf{x}} + \frac{h}{2}\hat{\mathbf{e}}_i\right) - \mathbf{f}\left(\bar{\mathbf{x}} + \frac{h}{2}\hat{\mathbf{e}}_i\right) \tag{10.61}$$

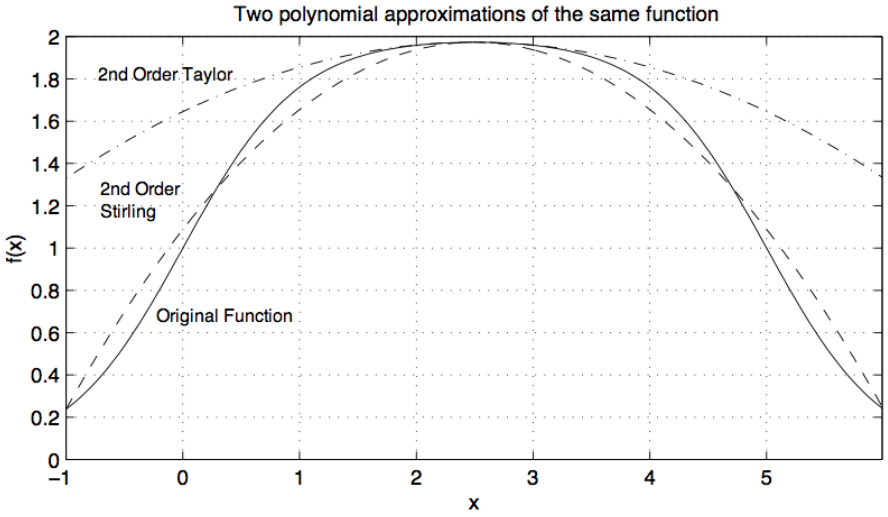


Figure 10.4: Difference between Stirling's and Taylor's approximation methods (Modified from [Nøgaard et al., 2004])

$$\boldsymbol{\mu}_i = \frac{1}{2}\left(\mathbf{f}\left(\bar{\mathbf{x}} + \frac{h}{2}\hat{\mathbf{e}}_i\right) + \mathbf{f}\left(\bar{\mathbf{x}} + \frac{h}{2}\hat{\mathbf{e}}_i\right)\right) \tag{10.62}$$

Finally, h is the interval length. For the first order interpolation, only the first two components of the right hand side of Eq. (10.58) need to be used.

[Nøgaard et al., 2004] have applied this interpolation method to stochastic systems to develop filters. They have developed the following two versions of the DDF

DD1 Using 1st order Stirling's interpolation

DD2 Using 2nd order Stirling's interpolation

Both DD1 and DD2 will be explained in this section. Since both filters are for nonlinear systems, the same system description as for the UKF and EKF is used.

$$\mathbf{x}_{k+1} = \mathbf{f}\left(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k\right)$$

$$\mathbf{z}_k = \mathbf{h}\left(\mathbf{x}_k, \mathbf{v}_k\right)$$

$$p(\mathbf{v}_k) \sim N(0, \mathbf{R}_k)$$
$$p(\mathbf{w}_k) \sim N(0, \mathbf{Q}_k)$$

The concept of the DDFs is similar to that of the UKF in the sense that certain points are used to find the covariance after a nonlinear transformation. The step size, h, has an optimal value of $\sqrt{3}$ according to [Nøgaard et al., 2004]. This value can, however, also be thought of as a tuning parameter. These filtering methods use square roots of the various covariance matrices, which is good for the numerical stability of the matrices. The square root matrix is a lower triangle matrix and is best computed using Cholesky decomposition, found in Section C.2. The square roots of the four important covariance matrices are

$$\mathbf{P}_k = \mathbf{S}_{\mathbf{x}k}\mathbf{S}_{\mathbf{x}k}^T \tag{10.63a}$$
$$\mathbf{P}_k^- = \mathbf{S}_{\mathbf{x}k}^-\left(\mathbf{S}_{\mathbf{x}k}^-\right)^T \tag{10.63b}$$
$$\mathbf{R}_k = \mathbf{S}_{\mathbf{v}k}\mathbf{S}_{\mathbf{v}k}^T \tag{10.63c}$$
$$\mathbf{Q}_k = \mathbf{S}_{\mathbf{w}k}\mathbf{S}_{\mathbf{w}k}^T \tag{10.63d}$$

10.2.1 Divided Difference Filter 1

The DDF1, which uses the first-order Stirling's interpolation method, is presented first. There are four more matrices that are found by applying divided differences. The first is the matrix corresponding to the propagation of the state.

$$\mathbf{S}^{(1)}_{\mathbf{x}\hat{\mathbf{x}}k}(i) = \frac{1}{2h}\left(\mathbf{f}\left(\hat{\mathbf{x}}_k + h\mathbf{S}_{\mathbf{x}k}(i), \mathbf{u}_k, \hat{\mathbf{w}}_k\right) - \mathbf{f}\left(\hat{\mathbf{x}}_k - h\mathbf{S}_{\mathbf{x}k}(i), \mathbf{u}_k, \hat{\mathbf{w}}_k\right)\right) \tag{10.64}$$

In Eq. (10.64), i refers to the ith column of the matrix. This is also true for the rest of the matrices. The matrix for the propagation of the process noise is

$$\mathbf{S}^{(1)}_{\mathbf{x}\mathbf{w}k}(i) = \frac{1}{2h}\left(\mathbf{f}\left(\hat{\mathbf{x}}_k, \mathbf{u}_k, \hat{\mathbf{w}}_k + h\mathbf{S}_{\mathbf{w}k}(i)\right) - \mathbf{f}\left(\hat{\mathbf{x}}_k, \mathbf{u}_k, \hat{\mathbf{w}}_k - h\mathbf{S}_{\mathbf{w}k}(i)\right)\right) \tag{10.65}$$

The matrix corresponding to measurement estimation with respect to the estimated state is

$$\mathbf{S}^{(1)}_{\mathbf{z}\hat{\mathbf{x}}^-k}(i) = \frac{1}{2h}\left(\mathbf{g}\left(\hat{\mathbf{x}}^-_k + h\mathbf{S}^-_{\mathbf{x}k}(i), \hat{\mathbf{v}}_k\right) - \mathbf{g}\left(\hat{\mathbf{x}}^-_k - h\mathbf{S}^-_{\mathbf{x}k}(i), \hat{\mathbf{v}}_k\right)\right) \tag{10.66}$$

The matrix corresponding to measurement estimation with respect to the measurement noise is

$$\mathbf{S}^{(1)}_{\mathbf{z}\mathbf{v}k}(i) = \frac{1}{2h}\left(\mathbf{g}\left(\hat{\mathbf{x}}^-_k, \hat{\mathbf{v}}_k + h\mathbf{S}_{\mathbf{v}k}(i)\right) - \mathbf{g}\left(\hat{\mathbf{x}}^-_k, \hat{\mathbf{v}}_k - h\mathbf{S}_{\mathbf{v}k}(i)\right)\right) \tag{10.67}$$

With the matrices defined in Eqs. (10.64 - 10.67) it is possible to carry out the full state estimation process. For the *a priori* update, the state will be propagated using the normal system equation.

$$\hat{\mathbf{x}}^-_{k+1} = \mathbf{f}\left(\hat{\mathbf{x}}_k, \mathbf{u}_k, \hat{\mathbf{w}}_k\right) \tag{10.68}$$

The *a priori* covariance of the state can be found using

$$\mathbf{P}^-_{k+1} = \mathbf{S}^{(1)}_{\mathbf{x}\hat{\mathbf{x}}k}\left(\mathbf{S}^{(1)}_{\mathbf{x}\hat{\mathbf{x}}k}\right)^T + \mathbf{S}^{(1)}_{\mathbf{x}\mathbf{w}k}\left(\mathbf{S}^{(1)}_{\mathbf{x}\mathbf{w}k}\right)^T \tag{10.69}$$

This gives the covariance matrix of which the square root can again be found by Cholesky decomposition. The square root can also be found immediately from the following matrix

$$\mathbf{S}^-_{\mathbf{x}(k+1)} = \begin{bmatrix} \mathbf{S}^{(1)}_{\mathbf{x}\hat{\mathbf{x}}k} & \mathbf{S}^{(1)}_{\mathbf{x}\mathbf{w}k} \end{bmatrix} \tag{10.70}$$

The matrix in Eq. (10.70) is a rectangular matrix, and must be transformed into a square matrix by using Householder triangularization. Details about the implementation of Householder triangularization can be found in Section C.3 in page 202.

For the *a posteriori* estimate, the measurement can be predicted by using the non linear measurement equation.

$$\hat{\mathbf{z}}_k = \mathbf{g}\left(\hat{\mathbf{x}}^-_k, \hat{\mathbf{v}}_k\right) \tag{10.71}$$

The square root of the output estimation error can be found by the Householder triangularization of

$$\mathbf{S}_{\mathbf{z}k} = \begin{bmatrix} \mathbf{S}^{(1)}_{\mathbf{z}\hat{\mathbf{x}}^-k} & \mathbf{S}^{(1)}_{\mathbf{z}\mathbf{v}k} \end{bmatrix} \tag{10.72}$$

The cross-covariance of measurement and state can be found by

$$\mathbf{P}_{\mathbf{x}\mathbf{z}k} = \mathbf{S}^-_{\mathbf{x}k}\left(\mathbf{S}^{(1)}_{\mathbf{z}\hat{\mathbf{x}}^-k}\right)^T \tag{10.73}$$

The gain can be computed in the following way

$$\mathbf{K}_k = \mathbf{P}_{\mathbf{x}\mathbf{z}k}\left(\mathbf{S}_{\mathbf{z}k}\mathbf{S}^T_{\mathbf{z}k}\right)^{-1} \tag{10.74}$$

The state is updated in the following manner

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k \left(\mathbf{z}_k - \hat{\mathbf{z}}_k \right) \tag{10.75}$$

Finally, the *a posteriori* state covariance can be found by using the gain as

$$\mathbf{P}_k = \left(\mathbf{S}_{\mathbf{x}k}^- - \mathbf{K}_k \mathbf{S}_{\mathbf{zx}k}^{(1)} \right) \left(\mathbf{S}_{\mathbf{x}k}^- - \mathbf{K}_k \mathbf{S}_{\mathbf{zx}k}^{(1)} \right)^T + \mathbf{K}_k \mathbf{S}_{\mathbf{zv}k}^{(1)} \left(\mathbf{K}_k \mathbf{S}_{\mathbf{zv}k}^{(1)} \right)^T \tag{10.76}$$

The square root of the *a posteriori* covariance, $\mathbf{S}_{\mathbf{x}k}$, can either be found from the Cholesky decomposition of Eq. (10.76) or the Householder triangularization of

$$\mathbf{S}_{\mathbf{x}k} = \begin{bmatrix} \mathbf{S}_{\mathbf{x}k}^- - \mathbf{K}_k \mathbf{S}_{\mathbf{zx}k}^{(1)} & \mathbf{K}_k \mathbf{S}_{\mathbf{zv}k}^{(1)} \end{bmatrix} \tag{10.77}$$

Since the DD1 is also first order, it performs like the EKF. The DD1 filter, however, does not require the computation of the Jacobians.

10.2.2 Divided Difference Filter 2

The second order Divided Difference Filter, DD2, works on almost the same principle as DD1. Similar matrices as the ones found in Eqs. (10.64 - 10.67) are computed using second order divided differences. It should be noted that Eqs. (10.64 - 10.67) are required for DD2 in addition to the second order divided differences. The matrices are

$$\begin{aligned} \mathbf{S}_{\mathbf{x}\hat{\mathbf{x}}k}^{(2)}(i) =& \frac{\sqrt{h^2-1}}{2h^2} \times \\ & \left(\mathbf{f}\left(\hat{\mathbf{x}}_k + h\mathbf{S}_{\mathbf{x}k}(i), \mathbf{u}_k, \hat{\mathbf{w}}_k \right) + \mathbf{f}\left(\hat{\mathbf{x}}_k - h\mathbf{S}_{\mathbf{x}k}(i), \mathbf{u}_k, \hat{\mathbf{w}}_k \right) - 2\mathbf{f}\left(\hat{\mathbf{x}}_k, \mathbf{u}_k, \hat{\mathbf{w}}_k \right) \right) \end{aligned} \tag{10.78}$$

$$\begin{aligned} \mathbf{S}_{\mathbf{x}\mathbf{w}k}^{(2)}(i) =& \frac{\sqrt{h^2-1}}{2h^2} \times \\ & \left(\mathbf{f}\left(\hat{\mathbf{x}}_k, \mathbf{u}_k, \hat{\mathbf{w}}_k + h\mathbf{S}_{\mathbf{w}k}(i) \right) + \mathbf{f}\left(\hat{\mathbf{x}}_k, \mathbf{u}_k, \hat{\mathbf{w}}_k - h\mathbf{S}_{\mathbf{w}k}(i) \right) - 2\mathbf{f}\left(\hat{\mathbf{x}}_k, \mathbf{u}_k, \hat{\mathbf{w}}_k \right) \right) \end{aligned} \tag{10.79}$$

$$\begin{aligned} \mathbf{S}_{\mathbf{z}\hat{\mathbf{x}}^-k}^{(2)}(i) =& \frac{\sqrt{h^2-1}}{2h^2} \times \\ & \left(\mathbf{g}\left(\hat{\mathbf{x}}_k^- + h\mathbf{S}_{\mathbf{x}k}^-(i), \hat{\mathbf{v}}_k \right) + \mathbf{g}\left(\hat{\mathbf{x}}_k^- - h\mathbf{S}_{\mathbf{x}k}^-(i), \hat{\mathbf{v}}_k \right) - 2\mathbf{g}\left(\hat{\mathbf{x}}_k^-, \hat{\mathbf{v}}_k \right) \right) \end{aligned} \tag{10.80}$$

$$\begin{aligned} \mathbf{S}_{\mathbf{zv}k}^{(2)}(i) =& \frac{\sqrt{h^2-1}}{2h^2} \times \\ & \left(\mathbf{g}\left(\hat{\mathbf{x}}_k^-, \hat{\mathbf{v}}_k + h\mathbf{S}_{\mathbf{v}k}(i) \right) + \mathbf{g}\left(\hat{\mathbf{x}}_k^-, \hat{\mathbf{v}}_k - h\mathbf{S}_{\mathbf{v}k}(i) \right) - 2\mathbf{g}\left(\hat{\mathbf{x}}_k^-, \hat{\mathbf{v}}_k \right) \right) \end{aligned} \tag{10.81}$$

The *a priori* state estimate can be found in the following way

$$\begin{aligned} \hat{\mathbf{x}}_{k+1} =& \frac{h^2 - n_{\mathbf{x}} - n_{\mathbf{w}}}{h^2} \mathbf{f}\left(\hat{\mathbf{x}}_k, \mathbf{u}_k, \hat{\mathbf{w}}_k \right) \\ & + \frac{1}{2h^2} \sum_{i=1}^{n_{\mathbf{x}}} \mathbf{f}\left(\hat{\mathbf{x}}_k + h\mathbf{S}_{\mathbf{x}k}(i), \mathbf{u}_k, \hat{\mathbf{w}}_k \right) + \mathbf{f}\left(\hat{\mathbf{x}}_k - h\mathbf{S}_{\mathbf{x}k}(i), \mathbf{u}_k, \hat{\mathbf{w}}_k \right) \\ & + \frac{1}{2h^2} \sum_{i=1}^{n_{\mathbf{w}}} \mathbf{f}\left(\hat{\mathbf{x}}_k, \mathbf{u}_k, \hat{\mathbf{w}}_k + h\mathbf{S}_{\mathbf{w}k}(i) \right) + \mathbf{f}\left(\hat{\mathbf{x}}_k, \mathbf{u}_k, \hat{\mathbf{w}}_k - h\mathbf{S}_{\mathbf{w}k}(i) \right) \end{aligned} \tag{10.82}$$

In Eq. (10.82), $n_{\mathbf{x}}$ is the size of the state vector, and $n_{\mathbf{w}}$ is the size of the process noise vector. The *a priori* state estimate of the DD2 is the same as the one for the UKF. The *a priori* covariance estimate is, however, not equal to the one found in the UKF. The covariance can be found by applying Householder triangularization to the following rectangular matrix

$$\mathbf{S}_{\mathbf{x}(k+1)}^- = \begin{bmatrix} \mathbf{S}_{\mathbf{x}\hat{\mathbf{x}}k}^{(1)} & \mathbf{S}_{\mathbf{x}\mathbf{w}k}^{(1)} & \mathbf{S}_{\mathbf{x}\hat{\mathbf{x}}k}^{(2)} & \mathbf{S}_{\mathbf{x}\mathbf{w}k}^{(2)} \end{bmatrix} \tag{10.83}$$

For the *a posteriori* estimate, the measurement has to first be predicted in the following manner

$$\begin{aligned}\hat{\mathbf{z}}_k =& \frac{h^2 - n_{\mathbf{x}} - n_{\mathbf{w}}}{h^2}\mathbf{g}\left(\hat{\mathbf{x}}_k^-, \hat{\mathbf{w}}_k\right) \\ &+ \frac{1}{2h^2}\sum_{i=1}^{n_{\mathbf{x}}} \mathbf{g}\left(\hat{\mathbf{x}}_k^- + h\mathbf{S}_{\mathbf{x}k}^-(i), \hat{\mathbf{v}}_k\right) + \mathbf{g}\left(\hat{\mathbf{x}}_k^- - h\mathbf{S}_{\mathbf{x}k}^-(i), \hat{\mathbf{v}}_k\right) \\ &+ \frac{1}{2h^2}\sum_{i=1}^{n_{\mathbf{v}}} \mathbf{g}\left(\hat{\mathbf{x}}_k^-, \hat{\mathbf{v}}_k + h\mathbf{S}_{\mathbf{v}k}(i)\right) + \mathbf{g}\left(\hat{\mathbf{x}}_k^-, \hat{\mathbf{v}}_k - h\mathbf{S}_{\mathbf{v}k}(i)\right)\end{aligned} \tag{10.84}$$

In Eq. (10.84), $n_{\mathbf{v}}$ is the size of the measurement noise vector. The square root of the covariance can be found by the Householder triangularization of

$$\mathbf{S}_{\mathbf{z}k} = \begin{bmatrix} \mathbf{S}_{\mathbf{z}\hat{\mathbf{x}}^-k}^{(1)} & \mathbf{S}_{\mathbf{z}\mathbf{v}k}^{(1)} & \mathbf{S}_{\mathbf{z}\hat{\mathbf{x}}^-k}^{(2)} & \mathbf{S}_{\mathbf{z}\mathbf{v}k}^{(2)} \end{bmatrix} \tag{10.85}$$

The cross-covariance of measurement and state is the same as for the DD1.

$$\mathbf{P}_{\mathbf{x}\mathbf{z}k} = \mathbf{S}_{\mathbf{x}k}^- \left(\mathbf{S}_{\mathbf{z}\hat{\mathbf{x}}^-k}^{(1)}\right)^T \tag{10.86}$$

The gain is

$$\mathbf{K}_k = \mathbf{P}_{\mathbf{x}\mathbf{z}k}\left(\mathbf{S}_{\mathbf{z}k}\mathbf{S}_{\mathbf{z}k}^T\right)^{-1} \tag{10.87}$$

The state is updated in the following manner

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k\left(\mathbf{z}_k - \hat{\mathbf{z}}_k\right) \tag{10.88}$$

Finally, the *a posteriori* state covariance can be found by using the gain as

$$\begin{aligned}\mathbf{P}_k =& \left(\mathbf{S}_{\mathbf{x}k}^- - \mathbf{K}_k\mathbf{S}_{\mathbf{z}\mathbf{x}k}^{(1)}\right)\left(\mathbf{S}_{\mathbf{x}k}^- - \mathbf{K}_k\mathbf{S}_{\mathbf{z}\mathbf{x}k}^{(1)}\right)^T + \mathbf{K}_k\mathbf{S}_{\mathbf{z}\mathbf{v}k}^{(1)}\left(\mathbf{K}_k\mathbf{S}_{\mathbf{z}\mathbf{v}k}^{(1)}\right)^T \\ &+ \mathbf{K}_k\mathbf{S}_{\mathbf{z}\mathbf{x}k}^{(2)}\left(\mathbf{K}_k\mathbf{S}_{\mathbf{z}\mathbf{x}k}^{(2)}\right)^T + \mathbf{K}_k\mathbf{S}_{\mathbf{z}\mathbf{v}k}^{(2)}\left(\mathbf{K}_k\mathbf{S}_{\mathbf{z}\mathbf{v}k}^{(2)}\right)^T\end{aligned} \tag{10.89}$$

The square root of the *a posteriori* covariance, $\mathbf{S}_{\mathbf{x}k}$, can either be found from the Cholesky decomposition of Eq. (10.89) or the Householder triangularization of

$$\mathbf{S}_{\mathbf{x}k} = \begin{bmatrix} \mathbf{S}_{\mathbf{x}k}^- - \mathbf{K}_k\mathbf{S}_{\mathbf{z}\mathbf{x}k}^{(1)} & \mathbf{K}_k\mathbf{S}_{\mathbf{z}\mathbf{v}k}^{(1)} & \mathbf{K}_k\mathbf{S}_{\mathbf{z}\mathbf{x}k}^{(2)} & \mathbf{K}_k\mathbf{S}_{\mathbf{z}\mathbf{v}k}^{(2)} \end{bmatrix} \tag{10.90}$$

10.3 Summary

The performance of the various nonlinear filters is discussed here. The linear filter is not dealt with here, because the systems that the filters will be used upon will not be linear. The various nonlinear filters are first summarized and then compared with each other.

The EKF uses a 1$^{\text{st}}$-order Taylor approximation to linearize the nonlinear functions. It is the fastest of the filters, however, it can be the hardest to implement. The Jacobians have to be derived beforehand, which saves computation time. The problem is that some functions can be incredibly complex and thus, the derivation process can be very difficult. It is possible to obviate the need to derive the Jacobians analytically by using numerical differentiation, but this causes more inaccuracies. Also, the approximation assumes that the errors are small. If this is not the case, then the filter can diverge.

The UKF is very easy to implement but this results in a high computationol load. The accuracy is, however, higher than the EKF. There are a few scaling parameters, which can result in more tuning work.

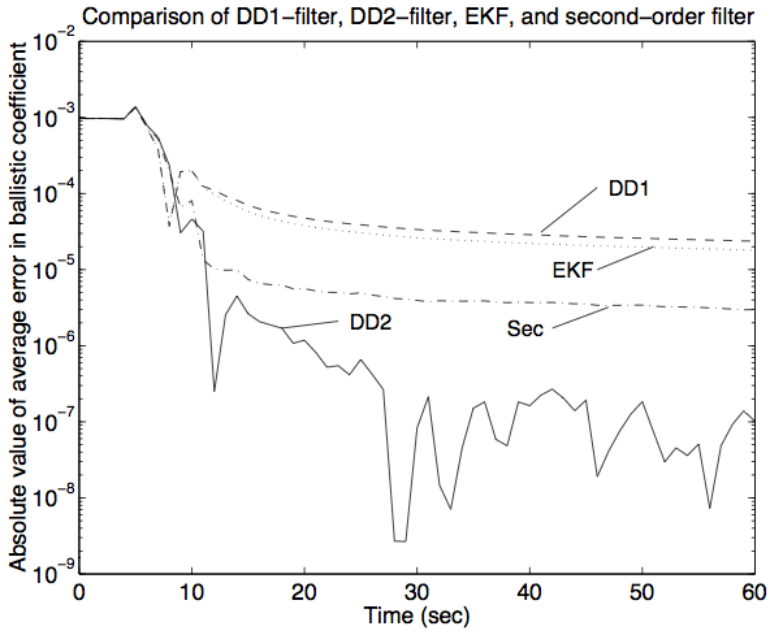


Figure 10.5: Comparison of EKF and the DDFs [Nøgaard et al., 2004]

Both DD1, and DD2 have very good numerical stability because of their implementation of matrix square roots. The DD1 has an accuracy comparable to the EKF, but with a greater computational load. It is, however, much easier to implement and less prone to diverging. The DD2 is comparable to the UKF, but is also easier to implement. This is because both the DDFs have only one tuning parameter. DD2 even has the same *a priori* state estimate as the UKF. The UKF, DD1, and DD2 are collectively referred to as sigma point filters.

[Nøgaard et al., 2004] have carried out a test comparing DD1 and DD2 with the EKF. The result of which, can be seen in Fig. 10.5. It is concluded in [Nøgaard et al., 2004] that the state estimate using the DD2 is better than the state estimates using the EKF or the DD1. The state estimates of the DD1 and the EKF have a similar accuracy, with the EKF being marginally more accurate. From the literature, the various characteristics of the nonlinear filters dealt with in this section can be summarized as seen Table 10.1.

The EKF and DD1 score the same on the accuracy because they are both 1^{st}-order filters. UKF and DD2 are both 2^{nd}-order filters and thus, are more accurate. The EKF, has a very low computational load because the Jacobians are pre-computed. In case the Jacobians are

Table 10.1: A comparison of the various nonlinear filters

	Accuracy	Computational Load	Stability	Ease of Implementation
EKF				
DD1				
UKF				
DD2				

Good Medium Bad

computed by numerical differentiation, the computational load would increase. The DD1 uses half the number of sigma points compared to DD2 and UKF and thus, has a lower computational load. The computational load is, however, still larger than that of the EKF. UKF and DD2 have almost the same computational load, which is higher than that of DD1 and the EKF. UKF and DD2, being 2^{nd}-order filters, have a higher stability than the first order filters and comparable to each other. DD1 scores better than EKF because DD1 is supposed to be less prone to divergence than the EKF according to [Nøgaard et al., 2004]. The ease of implementation factor includes the overhead required in the implementation. Jacobians have to be derived beforehand to implement the EKF in the traditional manner. This makes it the hardest filter to implement. If, however, the Jacobians are computed using numerical differentiation, the EKF would be as easy to implement as DD1, DD2, and UKF. The UKF, DD1, and DD2 have the same ease of implementation because they can be adapted to any state transition and measurement equations.

For this thesis study, all four nonlinear filters presented in this chapter are used and therefore, no tradeoff is made in Table 10.1. They all have their pros and cons, and the best way to compare the performance is to actually implement them. Thus, the following chapter implements all four filters to estimate the state of spacecraft using measurements from a ground station.

Chapter 11

Ground Station Tracking

It was shown in Chapter 8 that the performance of the USM in numerical integration is superior to that of Cartesian coordinates for most cases. For real astronautics missions, however, it is not enough to simply be able to integrate various spacecraft trajectories. It is also necessary to estimate the position and velocity of a spacecraft. The goal of this chapter is to simulate a scenario where a ground station tracks a satellite in orbit. The initial state at the start of the tracking process is simply an estimate with some error in it. Moreover, a measurement from a ground station has some noise in it and therefore, filtering techniques from Chapter 10 have to be used to successfully estimate the state. Comparison of navigation using ground station tracking has been carried out in [Chodas, 1981] where it was concluded that better estimates of the spacecraft state can be made using Cartesian coordinates than with the USM. The goal of this chapter is to recreate the tests and validate the results.

The filtering techniques are very sensitive to the tuning parameters. Tuning the various filters is a very time consuming and tedious process for the engineer. Therefore, an optimizer is used to tune the various filters for the various models. It is important to note that it is not the optimizer, but the filtering itself that is of importance here. The optimizer takes many computer hours, but in these modern times, computer hours are orders of magnitude cheaper than man hours. To start the tuning process, the tuning parameters for the various filters have to be identified.

11.1 Ground Station Measurement

Ground stations can not only be used for communicating with satellites, but also to help in the navigation of the satellite. There is a variety of ways in which satellites can be tracked using ground stations. In this section, only 2-way measurements using one clock are treated. This is because most satellites do not possess a clock that is accurate enough. A ground station works by sending an electromagnetic pulse that is reflected by the satellite. Since the speed of light is known, the time of flight is a measure of the distance to the satellite from the ground station. An example of ground based tracking is the Satellite Laser Ranging (SLR) method. It should be noted, however, that the time of flight has to be measured very accurately. Clock errors can give huge position errors, and this is why a minimum of four GPS satellites are required to estimate the state of a user. Using four satellites, the three position variables and the time can be successfully estimated.

The ground based tracking measurements take place in the local horizontal coordinate system of the ground station. Thus, the ground station measures the vector from it to the satellite. This vector is not provided in Cartesian coordinates, but in spherical coordinates, with the angles *Azimuth* and *Elevation*. It is assumed that the position and velocity of the satellite are expressed in the ECI frame. These angles can be seen in Fig. 11.2. Let the range from the ground station to the satellite be ρ. To find ρ, the position of both the ground station and the satellite have

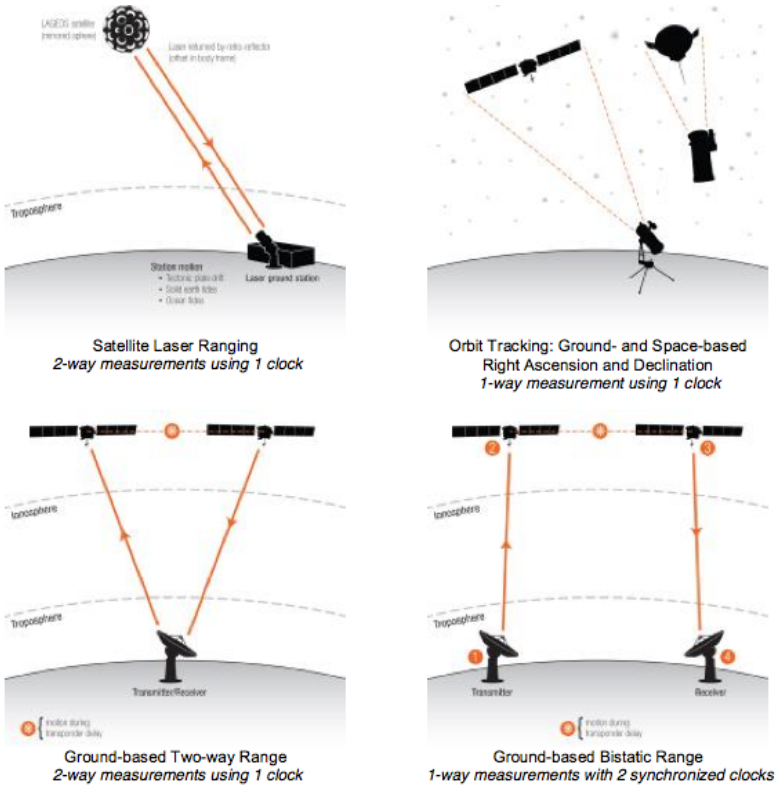


Figure 11.1: Various ways of ground based satellite tracking [Hujsak et al., 2007]

to be known in the same reference frame. The location of a ground station is usually given in the form of a height, and geodetic latitude and longitude. Moreover, the *Azimuth* and *Elevation* need to be computed in the local horizontal frame. Therefore, it is best to convert the position of the satellite to the ECEF frame from the ECI frame. The DCM $C_{ECEF,ECI}$ can be found using Eq. (4.33) found on page 31. The ECEF position of the ground station can be found using Eq. (4.37) found on page 33. The range, ρ, is simply the magnitude of the vector from the ground station to the spacecraft. This vector is the difference between the radial vector of the spacecraft and the radial vector of the ground station. The radial vector needs to be converted to the local tangent plane reference frame using the DCM found in Eq. (4.43) on page 34.

The conversions between the cartesian and spherical coordinates according to [Montenbruck and Gill, 2005; Chodas, 1981] are

$$\begin{bmatrix} \rho_E \\ \rho_N \\ \rho_U \end{bmatrix} = \rho \begin{bmatrix} \sin A \cos E \\ \cos A \cos E \\ \sin E \end{bmatrix} \tag{11.1}$$

$$A = \arctan\left(\frac{\rho_E}{\rho_N}\right) \tag{11.2a}$$

$$E = \arctan\left(\frac{\rho_U}{\sqrt{\rho_E^2 + \rho_N^2}}\right) \tag{11.2b}$$

The measurement from a ground station is in the form (ρ,A,E). These measurements are not perfect because of some sources of error. The reader is referred to [Montenbruck and Gill, 2005] for more information about these errors. The first error is *light time*, which is caused by the velocity of light. If it takes δt seconds for the laser to travel back from the satellite to the ground station,

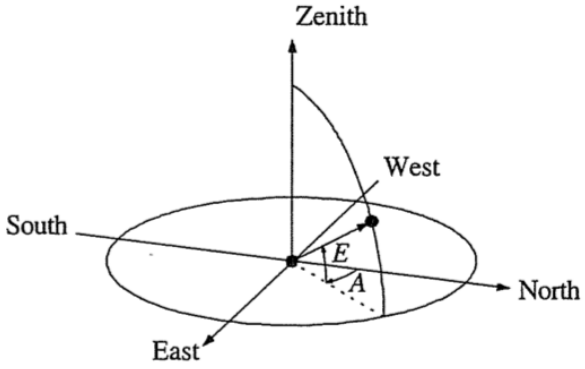


Figure 11.2: Definition of azimuth and elevation [Montenbruck and Gill, 2005]

the satellite must have a certain translation in this time. This is corrected by iteration by the ground station. According to [Montenbruck and Gill, 2005], this time is 0.01 s for LEO satellites, which corresponds to an angular correction of $7''$. Another error is *light aberration*, which is caused by relativistic effects. This is because the light path is slightly different in the rotating ground station fixed reference frame when compared to an inertial frame. This aberration also requires an angular correction, which is $0.3''$ for GEO satellites and $0.6''$ for LEO satellites.

If instead of an SLR, Ground-based Bistatic ranging from Fig. 11.1 is used, there is also a *transponder delay*, which is the time delay between the retransmitting of a signal by the satellite. According to [Montenbruck and Gill, 2005], this is 3000 ns for typical applications and a few nanoseconds for high precision hardware. There are also delays due to the different alignment of the transmitting and receiving antennae of the ground station, the internal electronics, and the weather conditions.

Ground based tracking can also be used to find the range rate. There are two possibilities to find this range rate. One way is to use two-way range rates, which takes the average of two ranges and uses signals transmitted from the ground station, and signals transmitted back from the satellite. This average range is divided by the counting time to find the range rate. There is also a one-way range rate measurement technique, which only uses the signals transmitted from the ground station. The two-way range rate technique can be seen in Fig. 11.3 , where the subscript u stands for signals going up from the ground station to the satellite and the subscript d stands for signals going down from the satellite to the ground station. For the one-way range rate technique, the signals with the subscript d are discarded.

For implementation in this study, the only difference between one-way and two-way range rates is that there will be more noise for one-way range rates. The measurement equation to find the range rate, from [Chodas, 1981] is

$$\dot{\rho} = (1/\rho)\left[\rho_x\left(v_x + \omega_E y\right) + \rho_y\left(v_y - \omega_E x\right) + \rho_z v_z\right] \tag{11.3}$$

In Eq. (11.3), the quantities are

- $\rho_{x,y,z}$ are the components of $\boldsymbol{\rho}$ in ECI
- x, y, z are the ECI components of the spacecraft radial vector
- $v_{x,y,x}$ are the ECI components of the spacecraft velocity vector
- ω_E is the rotation rate of the Earth found in Tab. 4.1 on pg. 32

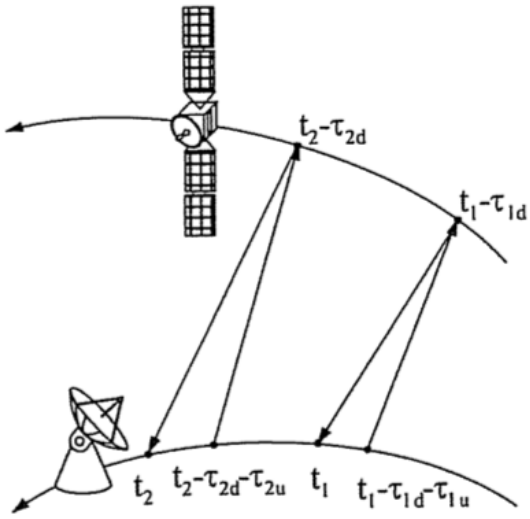


Figure 11.3: Motion of the ground station and the satellite during a 2 way range rate measurement [Montenbruck and Gill, 2005]

11.2 Implementation

Using the measurement equations, a simple ground station tracking test is set up similar to the one found in [Chodas, 1981]. For this test, the orbit of SARSAT, with the Kepler elements found in Table 8.1, is tracked for a 12 hour period. For the time-frame in [Chodas, 1981], the two ground stations that were used for tracking can be found in Table 11.1 (CLA and OTT). However, the time frame used for the simulation now differs from that of [Chodas, 1981] and it is therefore not possible to use the same ground stations. Based on the ground track of the satellite, more ground stations have been added to the pool to make a better selection. The ground track of SARSAT and the positions of the various ground station from Table 11.1 can be seen in Fig. 11.4.

11.2.1 Estimation Setup

[Chodas, 1981] assumes that all the process and measurement noises are simply additive. He also uses only two ground stations. To be consistent, the same is carried out in this study. However, the ground stations to be used differ from the original Ottawa (OTT) and Cold Lake (CLA). [Chodas, 1981] assumes a fully modeled system and the perturbations used are fourth order zonal and tesseral geopotential harmonics. In this study, however, all the perturbations found in Appendix B are used to generate the reference trajectory. Also, [Chodas, 1981] only utilizes an EKF with analytical Jacobians. In this study, the EKF is implemented with numerically computed Jacobians using the theory found in section C.5 on page 205. Along with the EKF,

Table 11.1: Location of the ground stations used for orbital tracking

Location	Code	Latitude	Longitude
Ottawa, Canada	OTT	45° N	−75° E
Cold Lake, Canada	CLA	55° N	−110° E
Alaska, USA	ASF	65° N	−147° E
Malindi, Kenya	MAL	−3° N	40° E
Hawaii, USA	UHM	30° N	−158° E

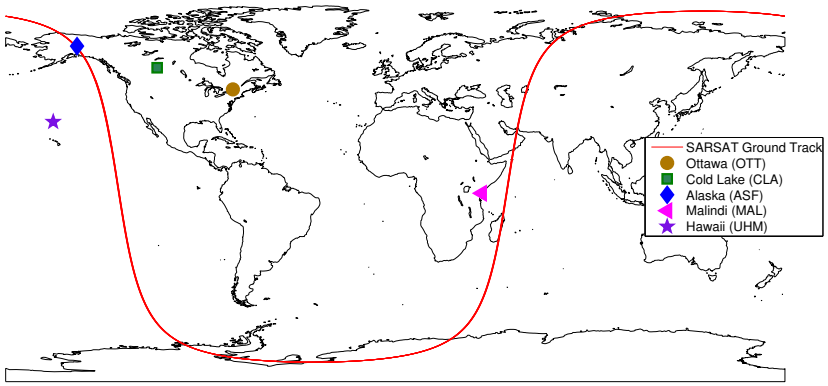


Figure 11.4: Ground track of SARSAT and the locations of various ground stations

Table 11.2: True and estimated initial states for the orbit tracking simulations

	a [km]	e [-]	i [deg]	Ω [deg]	ω [deg]	ν [deg]
Initial True State	7213	0.01	98.9	269	205	174
Initial Estimated State	7359.18	0.0078	99.01	268.21	338.07	40.95

the other nonlinear filters from Chapter 10 are also implemented (i.e., UKF, DD1, and DD2). According to [Chodas, 1981], for a first-order trajectory simulation, the perturbations can be ignored. Since the EKF is a first-order filter, this translates into the fact that the perturbations do not have to be taken into account during the propagation of the state. Since DD1 is also a first-order filtering technique, the same philosophy can be applied here. The ground station tracking is carried out with Cartesian coordinates, USM7, and USM6. The true initial state and the estimated initial state for all models and all filters are kept the same as in [Chodas, 1981] and can be seen in Table 11.2. All measurements presented in section 11.1 are used.

The standard deviations for the measurement errors used for all ground stations can be found in Table 11.3.

The setup used by [Chodas, 1981] is quite simple and straightforward. However, it is not the best possible modeling method. An additive error is used, which is fine for Cartesian coordinates

Table 11.3: Standard deviations used in [Chodas, 1981] for the various measurements available from the ground stations

Measurement type	Standard deviation
Range	2 m
Azimuth	0.02°
Elevation	0.02°
Range rate	1 m/s

where the time derivative of the velocities are the perturbing accelerations themselves. For the USM however, a perturbing acceleration is not simply added to the time derivative, but is more integrated inside the dynamics. Thus, the method of implementing the process noise during the state propagation has to be changed to model the system more realistically. An initial estimate of the state for Cartesian coordinates with a diagonal covariance matrix has a physical meaning. However, this is not the case for the USM because the state elements are all interdependent. Thus, it is better to use a nonlinear covariance transformation method like the unscented transform to convert an initial estimate in Cartesian coordinates to an initial estimate in one of the USMs.

The logical way to come up with an initial guess for the covariance is to estimate how much the USM parameters would differ from the USM parameters of the actual state. This is also the method used in [Chodas, 1981]. If an estimate with this covariance is converted back to Cartesian coordinates, we do not get a point with a sphere of uncertainty like we would get if we had a Cartesian state with a diagonal covariance matrix. The result would actually be an ellipsoid and the Cartesian coordinates of the position and the velocity would be highly correlated. A more intuitive method of getting the initial USM state estimate and the covariance would be to take the initial estimate of the state and covariance in Cartesian coordinates and the use the UT to get a state estimate in terms of the USM.

To show the effect of using a diagonal covariance matrix for the USM, the initial covariance of the state in Cartesian coordinates found in [Chodas, 1981] is compared to the UT converted initial covariance of the state in USM7 found in [Chodas, 1981]. The error ellipses for the position and velocity in Cartesian coordinate space can be seen in Fig. 11.5. The covariance in Cartesian coordinates is very skewed in all cases with much correlation. Also, there is correlation present between the position variables and the velocity variables. Using an UT to get the state covariance for the USM would eliminate all of this and provide a consistent initial estimate of position and velocity for all the different parameter sets.

11.2.2 Implementation

The equations for filtering found in Chapter 10 are quite abstract and can use for any generic filtering problem. The state $\mathbf{x}$ and the state estimate $\hat{\mathbf{x}}$ in this particular problem refer to a vector containing the position and velocity for the Cartesian coordinates, the hodographic velocities and the quaternion for the USM7, and the hodographic velocities and the MRP vector for the USM6. In Chapter 10, the nonlinear system equation is shown as

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k) \tag{11.4}$$

For the implementation here, the nonlinear equation takes care of the propagation of the spacecraft state in time. It should be noted, however, that this is not the dynamic equation. Instead, it is the complete RK4 step that propagates the state from one time step to the next. There is no control applied to the spacecraft and so $\mathbf{u}$ can be disregarded. The uncertainty present in the perturbing accelerations is represented by $\mathbf{w}_k$. Inside the dynamic equation, the noise is added on to the accelerations. For the Cartesian coordinates, the acceleration is

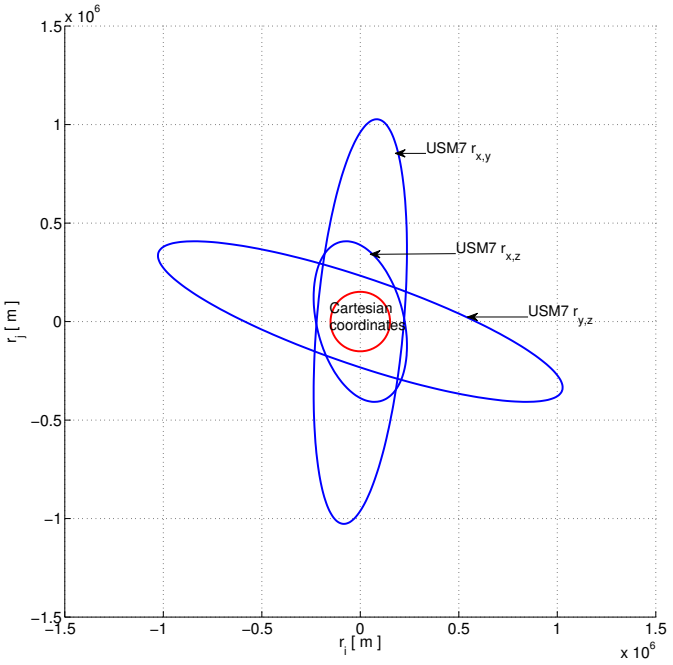
$$\mathbf{a}_{Cartesian} = \begin{bmatrix} a_x + \eta_{ax} \\ a_y + \eta_{ay} \\ a_z + \eta_{az} \end{bmatrix} \tag{11.5}$$

and $\mathbf{w}$ is

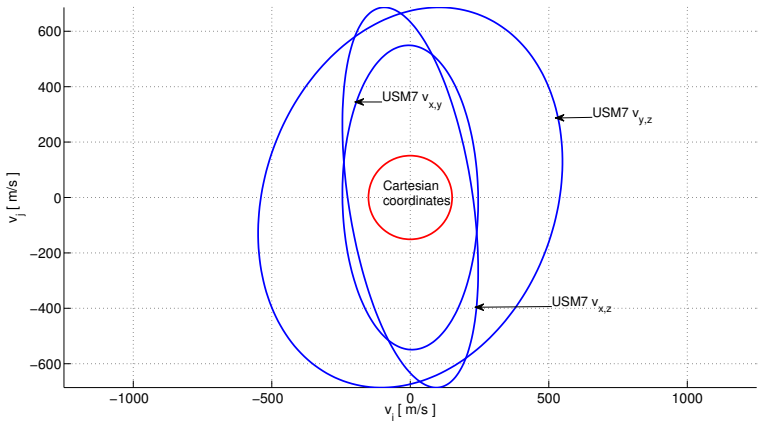
$$\mathbf{w}_{Cartesian} = \begin{bmatrix} \eta_{ax} \\ \eta_{ay} \\ \eta_{az} \end{bmatrix} \tag{11.6}$$

For the USM7 and USM6, acceleration with the noise is

$$\mathbf{a}_{USM} = \begin{bmatrix} a_{e1} + \eta_{ae1} \\ a_{e2} + \eta_{ae2} \\ a_{e3} + \eta_{ae3} \end{bmatrix} \tag{11.7}$$



(a) Error ellipses for position



(b) Error ellipses for velocity

Figure 11.5: The UT of a diagonal covariance matrix for a USM7 state to Cartesian coordinates

and $\mathbf{w}$ is

$$\mathbf{w}_{USM} = \begin{bmatrix} \eta_{ae1} \\ \eta_{ae2} \\ \eta_{ae3} \end{bmatrix} \tag{11.8}$$

For the USMs and Cartesian coordinates, all the η_a are normally distributed with zero mean. $\mathbf{Q}_k$ is the covariance matrix of these η_a.

The measurement equation in Chapter 10 is shown as

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k, \mathbf{v}_k) \tag{11.9}$$

The measurement is the output of the measurement made by a ground-station and could be any combination of range, azimuth, elevation, and range-rate. The function $\mathbf{h}$ is the function that takes the spacecraft state expressed in either Cartesian coordinates, the USM7, or the USM6 and outputs the desired measurement. The noise present in the ground state measurement is represented by $\mathbf{v}$.

The navigation simulation can be split into a *real-world* part and a *onboard* part. The real-world part knows the true state of the spacecraft at all times. This part is in charge of taking the true state and then creating measurements from the ground-station. Since measurements are not perfect in reality, this measurement creator also adds a randomly generated quantity to the perfect measurement. This randomly generated number is created from a distribution that has zero mean and a standard deviation found in Table 11.3. The onboard part is in charge of carrying out the filtering with the various estimators. It is independent from the real-world part and the only communication between the two is the input of the imperfect measurement from the real-world part to the onboard part at the appropriate times.

Some additional aspects noticed or used during implementation are presented below:

The estimation process for USM7 using the first-order filtering techniques EKF and DD1 is very unstable. It results in unsuccessful estimates if the two ground stations used are ASF and UHM. Thus, the two ground stations chosen for the simulation are ASF and MAL.

Also, the Number of Diagonalizations of $\mathbf{P}_k$ (NDPK) technique is used for the first-order filters. The NDPK was utilized in [Chodas, 1981] and it reduces the state covariance matrix to its diagonal form after the first measurement update. It was shown in [Chodas, 1981] that this technique increased the accuracy of the state estimation. Unlike [Chodas, 1981], where it was found that NDPK is only beneficial for Cartesian coordinates, it was found in this thesis study that NDPK is beneficial for both Cartesian coordinates and USM7.

For the UKF, the state is not augmented. This reduces the computation load and according to [Sun et al., 2009], it does not necessarily negatively influence the results. The augmented state is shown in Eq. 10.42 on page 109. If the state is augmented, it means that the expected values of the process and the measurement noises have to be added to the state. In case all 4 measurements are used, this would increase the dimension of the state by 7. This would increase the complexity of the estimation process and the computation time.

For the USM6, switching is required between MRP and SMRP. For this purpose, UT is used for the EKF and the UKF and second-order divided differences are used for the DD1 and DD2. This ensures that there are no extra uncertainties added due to the switching of the parameters.

11.3 Optimization

Since the optimizer itself is not the goal of this thesis study, only a rudimentary overview is provided here. Since the tuning problem is nonlinear and there is no analytical function, a numerical optimizer has to be chosen. Due to wealth of experience and knowledge in optimization using Genetic Algorithms (GA) and Particle Swarm Optimization (PSO) at the Astrodynamics and Space Missions research chair, these two were the main contenders. According to [Jones, 2005], the PSO has a relatively low computation load and is easy to implement. Also according to [Jones, 2005], the PSO has difficulty in local optimization, but can quickly converge to the general area of the global optimum. This is sufficient for the problem at hand, because the same tuning parameter of a filtering process will not provide the exact same results twice due to the stochastic nature of the problem. As for the actual performance difference, some find that the PSO has better results, while others find that the GA has better results. The PSO was ultimately chosen as the optimizer to use for this tuning process since the ease of implementation was a major requirement for this thesis study.

11.3.1 Particle Swarm Optimizer

The PSO was first proposed in [Kennedy and Eberhart, 1995] and follows a simplified model of the behavior of a flock of birds. The basic philosophy is that a swarm is created out of a specified number of particles. Each particle i has a certain velocity $\mathbf{v}_{id}$ with which it flies through the search space, along with a memory. In the memory, each particle stores the most optimum of the locations that has been explored by it. There is also a sort of communication between the particles so that each particle also has a knowledge of the global best location. Based on these criteria, the velocity of a particle for the next generation can be determined. For the original PSO, there are only two tuning parameters for the update of the velocity. The first parameter c_1, is a weight based on the optimum location found by the particle, and the second parameter c_2, is a weight based on the most optimum location out of all the locations visited by the particles. Another parameter known as the inertia weight for the updating of the velocity is proposed in [Shi and Eberhart, 1998] that balances between the global and local search, and they suggest that it is best to decrease this parameter during the optimization run. When w is higher, the particles have a higher velocity and are more explorative. When w is lower, the particles have a lover velocity and thus, try to search for the local optimum. According to [Shi and Eberhart, 1998], the velocity of a particle can be found in the following way

$$\mathbf{v}_{id} = w \times \mathbf{v}_{id} + c_1 \times \text{rand}() \times (\mathbf{p}_{id} - \mathbf{x}_{id}) + c_2 \times \text{rand}() \times (\mathbf{p}_{gd} - \mathbf{x}_{id}) \tag{11.10}$$

The location in the search space of the particle in the next flight is

$$\mathbf{x}_{id} = \mathbf{x}_{id} + \mathbf{v}_{id} \tag{11.11}$$

In Eqs. (11.10) and (11.11), $\mathbf{x}_{id}$ is the present position of particle i in the search space, $\mathbf{p}_{id}$ is the best location that particle i has visited, and $\mathbf{p}_{gd}$ is the best location out of the locations that all the particles have visited.

11.3.2 Setup and Tuning Parameters

First the objective function has to be identified. The goal of the optimization process is to minimize the output of this function. For the case investigated here, the objective function is a result from the filtering process. To simplify the problem, it was decided to minimize only one parameter, which is the Mean Absolute Error (MAE) of the position estimate from the time of first measurement onwards. At each time-step, the present state estimate is converted to Cartesian coordinates. The position error vector is defined as the vector between the true position and estimated position of the spacecraft. The error is defined as the norm of this position error vector. At the end of the estimation process, the mean of the norms is taken from the time of first measurement. Since the norm is always positive, this is automatically the MAE. In case some other parameter was used as the error, which could be both positive and negative, the mean of the absolute value of this parameter should be taken.

Due to the stochastic nature of the problem, the filtering process is carried out for a specified number of times and the output is the mean of the MAEs.

Tuning Parameters for the Filters

The tuning parameters of the various filters form the search space and these are identified to be: For the EKF, the tuning parameters are:

- $\mathbf{Q}_k$ the 3×3 process noise. This is because of the accelerations in 3 dimensions and is a measure of the uncertainty in the modeling of the system. It can be predicted that if fewer perturbations are modeled, the values of $\mathbf{Q}_k$ are larger.

For the UKF, the tuning parameters are:

Table 11.4: Ranges for the tuning parameters for the various filters

Parameter	Minimum Value	Maximum Value
Q_{mag}	10^{-16}	10^{-2}
α_{UKF}	10^{-5}	1
h_{DDF}	1	$\sqrt{5}$

- $\mathbf{Q}_k$ the process noise
- α_{UKF} a scaling parameter

In reality, β_{UKF} and κ_{UKF} are also parameters that can be tuned. However, according to [Allgoewer et al., 2009], for most cases $\beta_{UKF} = 2$ and $\kappa_{UKF} = 0$. They also recommend to constrain α_{UKF} between $[10^{-5}$, 1].

For DD1 and DD2, the tuning parameters are:

- $\mathbf{Q}_k$ the process noise
- h_{DDF} the interval length

For the tuning parameters other than $\mathbf{Q}_k$, a possibility might be to have different values for the propagation and the estimation phase. This would, however, increase the number of variables to be optimized and there is no guarantee that the estimation quality would be improved. For most engineering cases, the measurement noise is known from the manufacturers and therefore, it is not considered to be a tuning parameter here. $\mathbf{Q}_k$ is modeled in the following way:

$$\mathbf{Q}_k = Q_{mag} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{11.12}$$

Thus, only the parameter Q_{mag} has to be tuned. Ultimately, the range for the tuning parameters of the filters, which form the search space for the optimizer can be found in Table 11.4.

As can be seen in Table 11.4, most of the parameters that have to be tuned vary with orders of magnitude. Therefore, it is not advisable to simply provide a search space from the lowest value to the highest value. This would result in the majority of the guesses elements used by the optimizer being in the order of magnitude of the highest value. This can be seen in Fig. 11.6, where 10,000 randomly distributed numbers were created by MATLAB between 10^{-14} and 10^{-2}. Most of the numbers are between 10^{-3} and 10^{-2}, and none of the numbers are smaller than 10^{-6}. Thus, it is better to give the power of 10 for the optimizer to vary. This will result in all particles being spread equally in the search space over all the orders of magnitude.

Tuning Parameters for the Optimizer

The tuning parameters of the PSO consist of the weights for the velocities of the particles, the number of particles, and the total number of flights. In addition to all the tuning parameters mentioned, a stopping condition has to be provided to also be provided. This stopping condition could be a given minimum value of the objective function, a time limit for the whole optimization process, or the maximum number of flights carried out without a change in the global best value found. Out of the aforementioned three extra conditions, the last one, also known as the Stall Flight Limit is implemented. As mentioned previously, the inertia weight is also implemented with a linear decrease from one flight to the next. The values of these tuning parameters can be seen in Table 11.5. To optimally use an optimizer, the tuning parameter of the optimizer have to also be tuned. The optimal values of these parameters are very problem specific. However, the standard starting values from literature are presented in Table 11.5 and used in this thesis study.

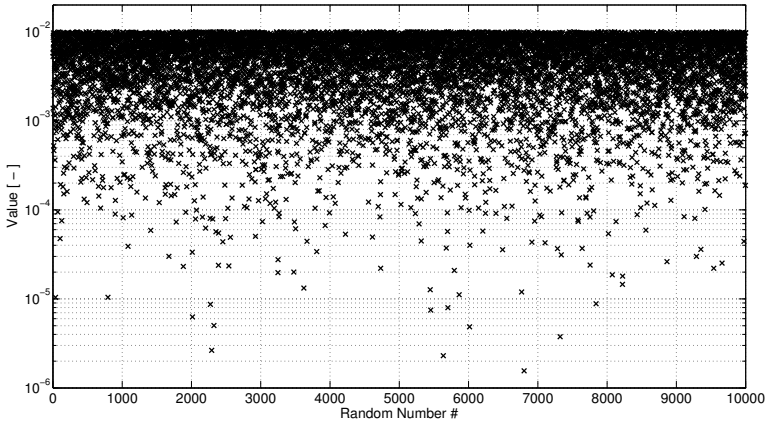


Figure 11.6: Uniformly distributed random numbers created by MATLAB between 10^{-14} and 10^{-2}

11.3.3 Implementation Issues

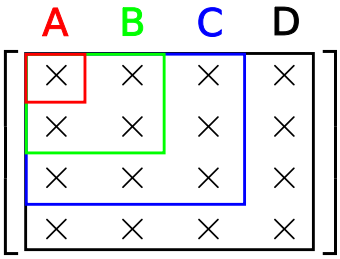


Figure 11.7: The sub-matrices of a 4×4 matrix

An estimation run using the filtering techniques does not always converge. For these cases, the singularity that occurs most often is that the covariance matrix becomes singular. Thus, there is a clause in each filter run that the present estimation procedure stops if the estimation error goes above a certain prescribed value. This automatically fills the remainder of the position errors with a high error value and thus, the optimizer will ignore these solutions. Another problem for the UKF is that the covariance matrix might loose its positive definiteness. This would again crash the simulation because the UKF involves finding square-roots of the matrix. This problem is solved by checking if the matrix is positive definite after each state propagation step and measurement update. For a matrix to be positive definite, each sub-matrix along the diagonal has to have a positive determinant. As an example, for a 4×4 matrix to be positive definite, all

Table 11.5: Ranges for the tuning parameters for the PSO

Parameter	Value
c_1	2
c_2	2
w	[1.4 , 0.5]
Particles	15
Flights	10
Stall Flight Limit	5

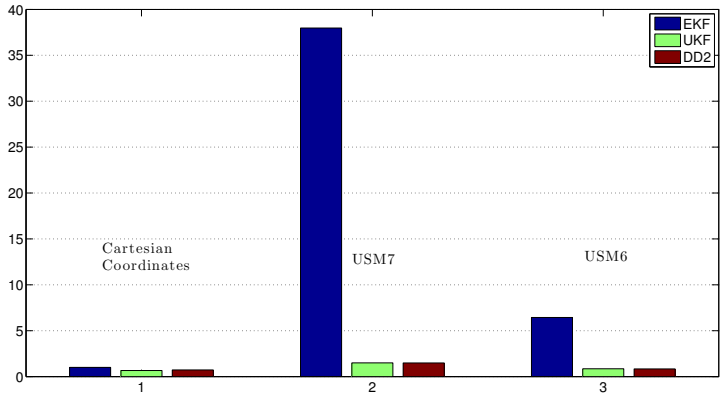


Figure 11.8: MAE of the position for the various models and filters for Case 1 scaled with respect to result of EKF with Cartesian coordinates (14.1 m)

Table 11.6: Filtering using Cartesian coordinates for Case 1

Filter	Q_{mag} [m^2 / s^4]	α_{UKF} [-]	h_{DDF} [-]	MAE Position [m]
EKF	1×10^{-16}	-	-	14.1
DD1	4.4957×10^{-16}	-	1.3275	37.1
UKF	6.2517×10^{-14}	0.1393	-	9.3
DD2	1×10^{-16}	-	$\sqrt{5}$	10.2

the sub-matrices A, B, C, and D found in Fig. 11.7 have to have a positive determinant.

11.4 Case 1

This case has the following characteristics:

- Fully modeled system, meaning that all perturbations used to create the reference trajectory are all used during the estimation.
- ASF and MAL used
- Range, Azimuth, and Elevation measured
- Simulation frequency of 0.1 Hz

The results of optimal estimation can be seen in a graphical manner in Fig. 11.8. The actual values for the MAE and the tuning parameters can be found in Tables 11.6 till 11.8. The estimation using DD1 for USM7 and USM6 is much worse than all the filtering techniques. Therefore, it is not included in the bar-plot in Fig. 11.8. This is the same for all the other cases and hence, results of estimation using DD1 are omitted from all the bar-plots. Also, DD1 results will not be included in the analysis.

All the 3 filters (excluding DD1) are able to perform the estimation of the spacecraft state successfully. It is also seen that the second order filters perform a more accurate state estimation than the EKF. This difference is marginal for Cartesian coordinates, but quite substantial for the USM. The best estimate of the state occurs when the UKF is used with Cartesian coordinates and the worst estimation occurs when the EKF is used with USM7. The estimation using EKF with

Table 11.7: Filtering using USM7 for Case 1

Filter	Q_{mag} [m^2 / s^4]	α_{UKF} [-]	h_{DDF} [-]	MAE Position [m]
EKF	1×10^{-16}	-	-	536.4
DD1	1.6205×10^{-11}	-	1.8204	3.6095×10^{4}
UKF	3.9246×10^{-14}	0.1953	-	21.1
DD2	1.4028×10^{-11}	-	2.1860	21.0

Table 11.8: Filtering using USM6 for Case 1

Filter	Q_{mag} [m^2 / s^4]	α_{UKF} [-]	h_{DDF} [-]	MAE Position [m]
EKF	1.1625×10^{-13}	-	-	91.0
DD1	1.6880×10^{-4}	-	2.1319	2.8961×10^{3}
UKF	1×10^{-16}	1	-	12.1
DD2	1×10^{-16}	-	$\sqrt{5}$	11.8

USM7 and USM6 is approximately 37 times and 6 times worse, respectively. This is because, the range, azimuth, and elevation together specify a point in 3 dimensional space that is very easy to express using Cartesian coordinates compared with the USMs. Also, the state propagation of Cartesian coordinates is simple, i.e. the time derivatives of the position elements are the velocity elements, and the time derivatives of the velocity elements are the accelerations. Thus, the measurement equation and state propagation are quite linear. This makes the Cartesian coordinate set more suitable for the EKF. The estimates using DD2 and UKF are quite similar and very accurate for both Cartesian coordinates and the USM6 with the MAE of the position ranging from 9.3 m to 12.1 m. Differences this small for a stochastic procedure like this does not really mean that one set of these is a clear winner. The state propagation using the USM is much more accurate for the SARSAT-like orbit for numerical integration. However, the time step-sizes used during this navigation scenario are very small. Thus, the bottleneck is the accuracy of the measurement update and not the error in the state propagation.

Between the USM7 and the USM6, the USM6 performs better with all three filters. The EKF error is approximately one fifth of the EKF error of the USM7, and the errors of the UKF and DD2 are approximately a half of the error of the UKF and DD2 using the USM7. This difference is caused because of the fact that the quaternion elements are not independent of each other. Therefore, this has an adverse effect on the covariance matrix.

The CPU time required for the various filters can be seen in Fig. 11.9. The fastest filter is the EKF using Cartesian coordinates, and the slowest filter is the DD2 using the USM6. For each coordinate set, the EKF is the fastest and the UKF and DD2 take approximately the same amount of CPU time. This is all to be expected as the EKF is the least complex filter, and the UKF and DD2 are of similar computational complexity. On average, the filters are the fastest for Cartesian coordinates and the slowest for the USM6. This is again to be expected as the dynamics of the USM6 and USM7 are much more complex than of the Cartesian coordinates. Even though the USM7 has one more state element than the USM6, the dynamic and kinematic equations for the USM6 are more complex. The CPU times are only shown for Case 1, and not for the other cases. For the other cases, the absolute times will differ from this case, but the ratios for the CPU time between the different filters and models will remain approximately the same.

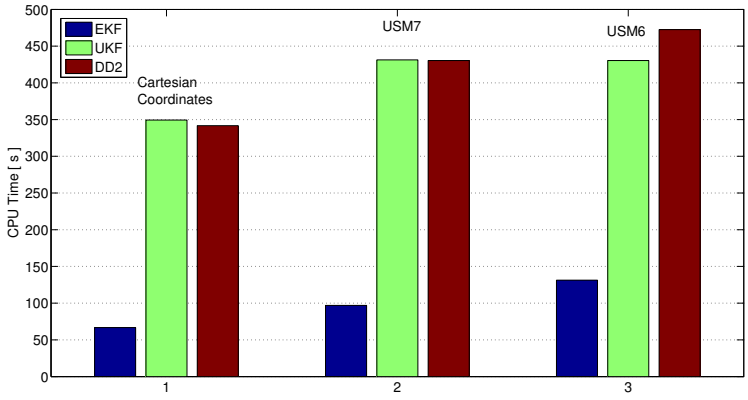


Figure 11.9: CPU time for the various filters used with Cartesian coordinates and the USMs for Case 1

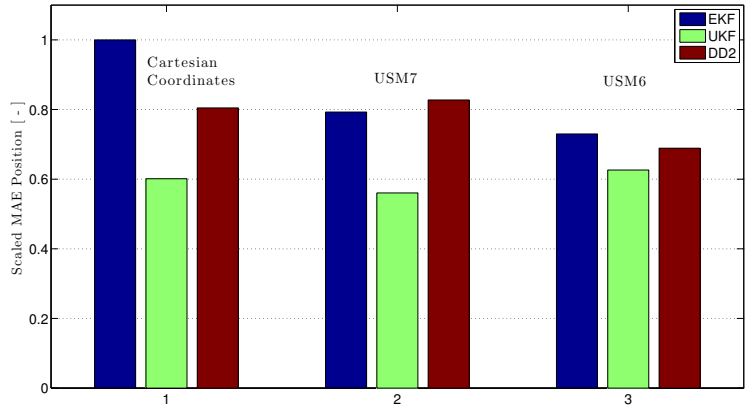


Figure 11.10: MAE of the position for the various models and filters for Case 2 scaled with respect to result of EKF with Cartesian coordinates (1120.8 m)

11.5 Case 2

For this case, it is assumed that the system is not fully modeled anymore. This simulation case would showcase the performance for the models in case the engineer does not have perfect knowledge of the process, which is often the case. This case has the following characteristics:

- Only J_2 and atmospheric drag modeled in the estimator
- ASF and MAL used
- Range, Azimuth, and Elevation measured
- Simulation frequency of 0.1 Hz

The results can be seen in a graphical manner in Fig. 11.10.

For this case, the best estimation is by using the UKF with the USM7 and the worst estimation is when the EKF is used with Cartesian coordinates. For this case, the estimates are the best

Table 11.9: Filtering using Cartesian coordinates for Case 2

Filter	Q_{mag} [m^2 / s^4]	α_{UKF} [-]	h_{DDF} [-]	MAE Position [m]
EKF	2.2060×10^{-10}	-	-	1120.8
DD1	2.7040×10^{-5}	-	$\sqrt{5}$	1,032.7
UKF	5.8939×10^{-11}	2.3972×10^{-4}	-	673.9
DD2	3.5027×10^{-7}	-	2.1588	902.0

Table 11.10: Filtering using USM7 for Case 2

Filter	Q_{mag} [m^2 / s^4]	α_{UKF} [-]	h_{DDF} [-]	MAE Position [m]
EKF	1.6025×10^{-9}	-	-	888.6
DD1	8.0612×10^{-12}	-	$\sqrt{5}$	3.701×10^4
UKF	1×10^{-16}	2.0314×10^{-4}	-	628.2
DD2	4.9136×10^{-10}	-	1.8703	927.3

when the USM7 is used, followed by the USM6. Unlike the previous scenario, all the information about the environment is not available to the estimator. As was shown in the derivation of the USM, the USM contains the assumptions of orbital motion in its dynamics. Therefore, with less information about the environment, the USMs are able to approximate the orbital motion better than Cartesian coordinates.

For Cartesian coordinates, the EKF has the worst performance and the UKF has the best performance. The MAE of the position of the EKF is approximately 66% more than that of the UKF. The MAE of the position of the DD2 is approximately 34% more than that of the UKF. The percentages are with respect to the UKF MAE in position.

For the USM7, the DD2 has the worst performance and the UKF has the best performance. The MAE of the position of the EKF is approximately 41% more than that of the UKF. The MAE of the position of the DD2 is approximately 48% more than that of the UKF. The percentages are with respect to the UKF MAE in position. The difference between the EKF and the DD2 is marginal and therefore, their accuracy can be considered to be equal.

For the USM6, the EKF has the worst performance and the UKF has the best performance. The MAE of the position of the EKF is approximately 17% more than that of the UKF. The MAE of the position of the DD2 is approximately 10% more than that of the UKF. The percentages are with respect to the UKF MAE in position. The difference between the EKF and the DD2 is marginal and therefore, their accuracy can be considered to be equal.

As mentioned previously, the UKF used with the USM7 gives the best accuracy. However, the difference between the UKF runs with the different models is marginal. The UKF with Cartesian coordinates is approximately 7% less accurate than the UKF with the USM7. The UKF with the USM6 is approximately 12% less accurate than the UKF with the USM7. The percentages

Table 11.11: Filtering using USM6 for Case 2

Filter	Q_{mag} [m^2 / s^4]	α_{UKF} [-]	h_{DDF} [-]	MAE Position [m]
EKF	1.5346×10^{-9}	-	-	818.1
DD1	1.2241×10^{-4}	-	1.9419	2,439.8
UKF	3.1857×10^{-14}	1.5332×10^{-4}	-	701.9
DD2	2.0212×10^{-9}	-	1.2252	772.1

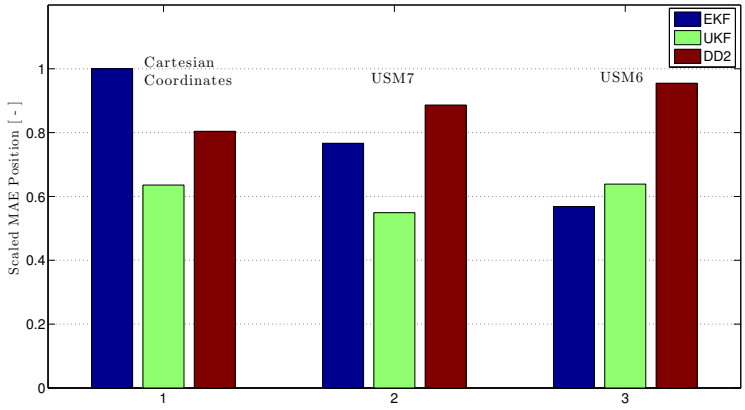


Figure 11.11: MAE of the position for the various models and filters for Case 3 scaled with respect to result of EKF with Cartesian coordinates (1120.9 m)

Table 11.12: Filtering using Cartesian coordinates for Case 3

Filter	Q_{mag} [m^2 / s^4]	α_{UKF} [-]	h_{DDF} [-]	MAE Position [m]
EKF	2.2060×10^{-10}	-	-	1120.9
DD1	2.7040×10^{-5}	-	$\sqrt{5}$	1067.5
UKF	4.9628×10^{-9}	2.6363×10^{-4}	-	712.7
DD2	3.5027×10^{-7}	-	2.1588	901.4

are with respect to the UKF MAE in position using the USM7. Since the difference is so small, it can be safely concluded that the UKF performs consistently and equally well for the various models.

11.6 Case 3

For this case, it is still assumed that the system is not fully modeled. However, the additional measurement of the range rate is also used in order to increase accuracy. This case has the following characteristics:

- Only J_2 and atmospheric drag modeled in the estimator
- ASF and MAL used
- Range, Azimuth, Elevation, and Range-rate used
- Simulation frequency of 0.1 Hz

The results can be seen in a graphical manner in Fig. 11.11 on page 134.

For this case, the best estimation is by using the UKF with the USM7 and the worst estimation is when the EKF is used with Cartesian coordinates. For this case, the estimates are the best when the USM7 is used, followed by the USM6. The reason for this is that the same information about the environment, as in Case 2 is given to the estimator. The only change in this scenario is the addition of range-rate as a measurement.

Table 11.13: Filtering using USM7 for Case 3

Filter	Q_{mag} [m^2 / s^4]	α_{UKF} [-]	h_{DDF} [-]	MAE Position [m]
EKF	1.6025×10^{-9}	-	-	859.2
DD1	1.4983×10^{-14}	-	1.3165	5,512.7
UKF	5.0119×10^{-16}	1.4727×10^{-4}	-	615.5
DD2	2.2060×10^{-10}	-	2.0496	993.7

Table 11.14: Filtering using USM6 for Case 3

Filter	Q_{mag} [m^2 / s^4]	α_{UKF} [-]	h_{DDF} [-]	MAE Position [m]
EKF	5.3040×10^{-9}	-	-	637.0
DD1	3.3037×10^{-5}	-	2.2361	1,951.4
UKF	1×10^{-16}	1.5765×10^{-4}	-	715.9
DD2	5.5539×10^{-10}	-	1.5220	1070.2

For Cartesian coordinates, the EKF has the worst performance and the UKF has the best performance. The MAE of the position of the EKF is approximately 57% more than that of the UKF. The MAE of the position of the DD2 is approximately 26% more than that of the UKF. The percentages are with respect to the UKF MAE in position. The results for Cartesian coordinates are quite similar to those of Case 2. Therefore, it can be concluded that the addition of range-rate as a measurement does not affect the accuracy of the state estimation using Cartesian coordinates.

For the USM7, the DD2 has the worst performance and the UKF has the best performance. The MAE of the position of the EKF is approximately 40% more than that of the UKF. The MAE of the position of the DD2 is approximately 61% more than that of the UKF. The percentages are with respect to the UKF MAE in position. The MAE of the position using all the filters with the USM7 again remains roughly the same as for Case 2. The accuracies of the EKF and of the UKF increase and the accuracy of the DD2 increases by a very small amount. These changes are so small that, the change can disregarded.

For the USM6, the DD2 has the worst performance and the EKF has the best performance. The MAE of the position of the UKF is approximately 12% more than that of the EKF. The MAE of the position of the DD2 is approximately 68% more than that of the EKF. The percentages are with respect to the EKF MAE in position. The difference between the EKF and the UKF is marginal and therefore, their accuracy can be considered to be equal. The addition of the range-rate as a measurement has increased the accuracy of the EKF and decreased the accuracies of the UKF and the DD2. The change in the accuracy of the UKF is very minimal and can be disregarded. However, the differences in the EKF and the DD2 are 22% and 34%, respectively. This change in the accuracy is quite substantial. Perhaps the USM6 is more sensitive to range-rante measurements, or the PSO might not have converged to the optimum solution.

As mentioned previously, the UKF used with the USM7 gives the best accuracy. However, the difference between the UKF runs with the different models is marginal. The UKF with Cartesian coordinates is approximately 16% less accurate than the EKF with the USM7. The UKF with the USM6 is also approximately 16% less accurate than the UKF with the USM7. The percentages are with respect to the UKF MAE in position using the USM7. Since the difference is so small, it can be safely concluded that the UKF performs consistently and equally well for the various models.

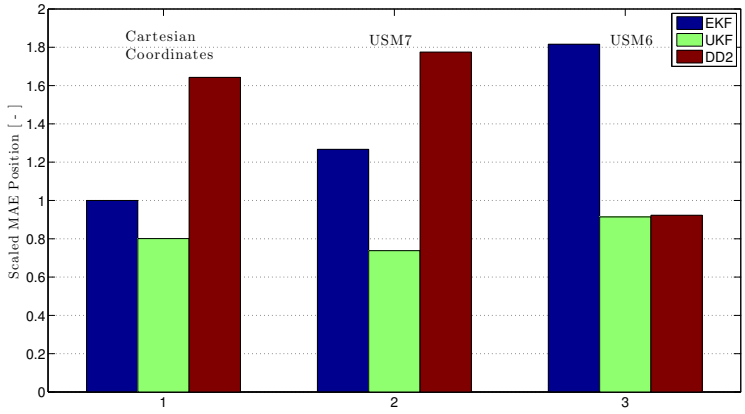


Figure 11.12: MAE of the position for the various models and filters for Case 4 scaled with respect to result of EKF with Cartesian coordinates (736.2 m)

Table 11.15: Filtering using Cartesian coordinates for Case 4

Filter	$\mathbf{Q}_k$ [m^2 / s^4]	α_{UKF} [-]	h_{DDF} [-]	MAE Position [m]
EKF	8.8471×10^{-7}	-	-	736.2111
DD1	4.0851×10^{-8}	-	1.3165	718.0730
UKF	1.2618×10^{-6}	0.6628	-	589.5825
DD2	1.9751×10^{-8}	-	2.2361	1,209.5

11.7 Case 4

For this case, it is still assumed that the system is not fully modeled. However, a third ground station is also used. This case has the following characteristics:

- Only J_2 and atmospheric drag modeled in the estimator
- ASF, MAL, and UHM used
- Range, Azimuth, Elevation, and Range-rate used
- Simulation frequency of 0.1 Hz

The results can be seen in a graphical manner in Fig. 11.12.

For this case, the best estimation is by using the UKF with the USM7 and the worst estimation is when the EKF is used with the USM6. For this case, the estimates are the best when the

Table 11.16: Filtering using USM7 for Case 4

Filter	$\mathbf{Q}_k$ [m^2 / s^4]	α_{UKF} [-]	h_{DDF} [-]	MAE Position [m]
EKF	5.3432×10^{-13}	-	-	932.7923
DD1	3.955×10^{-16}	-	2.1545	1.3724×10^4
UKF	1.3836×10^{-6}	0.6628	-	543.3478
DD2	1×10^{-16}	-	1.7759	1,306.8

Table 11.17: Filtering using USM6 for Case 4

Filter	$\mathbf{Q}_k$ [m^2 / s^4]	α_{UKF} [-]	h_{DDF} [-]	MAE Position [m]
EKF	1×10^{-16}	-	-	$1,336.8$
DD1	4.8596×10^{-5}	-	2.2224	$1,815.7$
UKF	1×10^{-16}	4.8350×10^{-5}	-	672.9
DD2	1.2618×10^{-6}	-	1.0441	679.3821

USM7 is used, followed by Cartesian coordinates. This scenario is almost the same as Case 3, but there is an additional ground-station used.

For Cartesian coordinates, the DD2 has the worst performance and the UKF has the best performance. The MAE of the position of the EKF is approximately 25% more than that of the UKF. The MAE of the position of the DD2 is approximately 105% more than that of the UKF. The percentages are with respect to the UKF MAE in position. Adding an extra ground station changes the results of the Cartesian coordinates quite drastically. The accuracies of the EKF and the UKF increase by approximately 34% and 17%, respectively. The accuracy of the DD2 decreases by approximately 25%. The effect of adding an additional ground station should increase the accuracy of the estimation. This effect is the largest for the EKF, however, the DD2 behaves in a more erratic manner with a decrease in the accuracy. This might be because of the violation of the unbiasedness assumption of the filter, or due to the convergence of the optimizer.

For the USM7, the DD2 has the worst performance and the UKF has the best performance. The MAE of the position of the EKF is approximately 42% more than that of the UKF. The MAE of the position of the DD2 is approximately 141% more than that of the UKF. The percentages are with respect to the UKF MAE in position. Due to the addition of an extra ground-station, the accuracies of the EKF and the DD2 decrease by approximately 34% and 32%, respectively. The accuracy of the UKF increases by approximately 12%. The effect of adding an additional ground station should increase the accuracy of the estimation. This effect is only seen for the UKF, however, the DD2 and the EKF behave in a more erratic manner with a decrease in the accuracy. This might be because of the violation of the unbiasedness assumption of the filter, or due to the convergence of the optimizer, or simply the stochastic nature of the problem.

For the USM6, the EKF has the worst performance and the UKF has the best performance. The MAE of the position of the EKF is approximately 99% more than that of the UKF. The MAE of the position of the DD2 is approximately 1% more than that of the UKF. The percentages are with respect to the UKF MAE in position. The difference between the DD2 and the UKF is marginal and therefore, their accuracy can be considered to be equal. The addition of an extra ground-station has increased the accuracies of the UKF and the DD2 by 6% and 37%, respectively and decreased the accuracy of the EKF by approximately 110%. The decrease in the accuracy of the EKF is probably caused by the non-convergence of the optimizer, or the stochastic nature of the problem.

As mentioned previously, the UKF used with the USM7 gives the best accuracy. However, the difference between the UKF runs with the different models is small. The UKF with Cartesian coordinates is approximately 9% less accurate than the UKF with the USM7. The UKF with the USM6 is approximately 24% less accurate than the UKF with the USM7. The percentages are with respect to the UKF MAE in position using the USM7. It can be concluded that the UKF performs consistently and equally well for the various models. However, the UKF accuracies of using Cartesian coordinates and the USM7 are almost equal, while the runs using the USM6 have a worse accuracy.

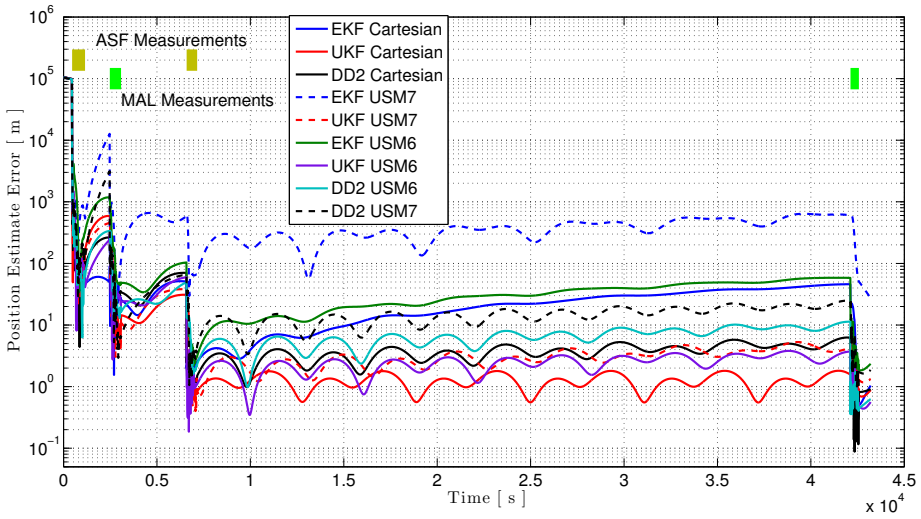


Figure 11.13: The position estimate error against time using all filters for Case 1

11.7.1 Convergence Plot

After the optimization process was complete, it was decided that the MAE of the position might not have been the best objective function for the PSO [Mooij, 2010]. Thus, the evolution of the position error of the state estimate against time for the various filters with the various models was plotted for Case 1 in Fig. 11.13.

It can be seen that for all cases, the position error reaches a converged value after approximately 6500 s. This happens when the satellite has had two batches of measurements from ASF and one batch of measurements from MAL. After the last ASF measurements, at approximately 6500 s, the satellite has this converged value of the error for almost 40,000 s. As with the MAE position bar-plots seen in Fig. 11.8 the best estimation is with the UKF with Cartesian coordinates, and the worst estimation is with the EKF with the USM7. The MAE information shows that the error with the EKF and Cartesian coordinates is almost the same as the error with the UKF and Cartesian coordinates. With the convergence plot, however, it can clearly be seen that the converged error with the Cartesian coordinates and EKF is 1-2 orders of magnitude higher than the converged error with the UKF. The MAE also shows that the Cartesian EKF is more accurate than the USM7 UKF, DD2 , and the USM6 UKF and DD2. This can clearly be seen to not be the case in Fig. 11.13. The converged errors can be found in 3, almost distinct strata. The UKF is the most accurate, followed by the DD2, and the least accurate is the EKF. For each filter, the most accurate are the Cartesian coordinates, followed by the USM6 and the least accurate is the USM7.

The difference between the MAE results and the convergence plot can be explained by the fact that the initial error is very high when compared to the low converged values. Thus, these errors influence the MAE much more than the converged error. An example can be given with the Cartesian coordinates. After the first set of ASF measurements, the UKF error increases to around 500 m before the MAL measurements start. On the contrary, the EKF error remains at around 40 m. During the time with no measurements after the MAL measurements and before the second ASF measurements, both errors behave the same way. It is only after the second ASF measurements that the UKF error converges to around 1 m. The EKF error steadily increases from around 4 m to around 40 m till the second MAL measurements occur. It can therefore be concluded here that a better way objective function would be the MAE of the position in the time between the second ASF measurements and the second MAL measurements.

11.8 Conclusions and Recommendations

The discussion of the results does not include the DD1 because of its abysmal performance for the USM6 and the USM7. There are two possible explanations for this. One is that there is a bug in the programming. This is, however, unlikely because the program was built to be as modular as possible. Thus, standard functions were made that find the required matrices for the filters. These functions take as an input, the dynamic function and the measurement function. It can be asserted that all the three functions function properly. The matrix building function performs well, because the state estimation using DD1 and Cartesian coordinates is successful. The dynamics functions for the USM6 and the USM7 perform well because numerical integration of orbital trajectories using them was successful. The measurement function also performs well because state estimation using all other filtering techniques was successful for both USM6 and USM7. Therefore, a bug in the programming can be ruled out. For a simple radar tracking test case shown in [Nøgaard et al., 2004], the DD1 already performs worse than the EKF. According to them it was because some of the assumptions the DD1 is built on were voided, and more specifically *the assumption that the state estimate is unbiased is far from satisfied here.* Since the scenario used here is similar to the radar tracking, it can be contested that the violation of the unbiasedness also occurs. The implementation in this thesis is a more complex version of the one in [Chodas, 1981], because the process noise is on the acceleration and is not simply a matrix added to the covariance matrix. In case of Cartesian coordinates, this is indeed still the case. However, the role of the acceleration inside the state dynamics for the USM is much more convoluted than is the case for Cartesian coordinates. This is suspected to cause the violation of the assumption of unbiasedness. When preliminary tests were carried out by simply adding the process noise to test the filtering, the DD1 was able to successful estimate the state for the USMs. However, this is still not the way that the dynamics of the USM works and therefore is not carried out here.

The 4 cases used here can be summarized as follows:

Case 1 Fully modeled system, 2 ground stations, no velocity measurement

Case 2 Partially modeled system, 2 ground stations, no velocity measurement

Case 3 Partially modeled system, 2 ground stations, all measurements

Case 4 Partially modeled system, 3 ground stations, all measurements

It was concluded in [Chodas, 1981] that the state estimation using ground-station tracking with Cartesian coordinates is better than with USM7. It has been shown in this thesis study that this is indeed true for the case when the system is fully modeled, which was the only case investigated in [Chodas, 1981]. However, it is found here that the EKF with both USM6 and USM7 outperform the EKF with Cartesian coordinates for Case 2 and Case 3. Case 1 and Case 4 provide more information about the environment in terms of environment modeling and measurements than Case 2 and Case 3. Thus it can be concluded that the EKF with the USMs performs better than with Cartesian coordinates when less information is available about the system and state. Out of all the Cases, the best estimation is with the UKF and USM7 as this creates the most accurate estimate for Cases 2, 3, and 4.

Among the filters, the UKF is constantly the best filtering technique. It is expected that it outperforms the EKF as the UKF is a higher order filter than the EKF. Only for Case 3 with USM6 was the MAE of the position estimate using the UKF higher than that of the EKF. This difference in the MAE was very minimal and so, it cannot be concluded that the performance of the UKF was worse. The DD2 is supposed to be similar in performance to the UKF, but this is not the case here as it is outperformed by the UKF in all cases except for one. The DD2 is built on the same assumptions as the DD1. Thus, it might be possible that the clause of unbiased estimate is being violated here. Another possible reason is that the augmented state was not used for the UKF, but the DD2 automatically augments the state for each propagation

and measurement update step. As it was stated in [Sun et al., 2009] that it is better for some cases to have not augment the state, automatic augmentation of the state by the DD2 might have decreased its performance.

The final conclusion here is that the EKF, UKF, and DD2 can all successfully estimate the state of the spacecraft for all three models. In case the system is fully modeled, it is best to use Cartesian coordinates. If it is desired to still use the USM for this case, the USM6 is recommended over the USM7. In case there is not much information available about the environment, the USM7 is recommended. However, if the EKF has to be used, the USM6 is recommended over the USM7. As for the filter type, the UKF should be the filter of choice for the most accurate results. Once the scenario set up is given, i.e. the measurement to be used, which ground stations to use, and how to model the system, the PSO is able to successfully tune all filters for all models. This is very convenient as it takes away the trial and error required to tune the filters and thus, saves much time and frustration for a filtering novice. It is probably of not much use to someone who is very experienced with filtering techniques, as they have an intimate knowledge of the problem and can therefore tune effortlessly. However, with the tuning carried out via optimization, as long as the PSO is converged, there is a guarantee that the filter will be optimally tuned for the specified problem.

This navigation work carried out here simply scratches the surface of what all can be investigated. Only the EKF, UKF, DD1, and DD2 were implemented in this study. There are, however, many different types of filters available that can be used to carry out the navigation. A few interesting ones that could be implemented in the future are the square root versions of the EKF and the UKF. Also, the augmented state version of the UKF can be investigated to check if the state estimation is improved. A modification of the UKF that could improve its speed, is the Spherical Simplex UKF [Julier, 2003], which requires fewer sigma points for the computation. Finally, the effect of using adaptive filter should also be checked.

Apart from the filtering techniques, another implementation present in this chapter is the use of a PSO for tuning the filters. Even though this is a relatively simple idea, it has not been done much in the past. Due to time constraints, not much focus has been given to the optimizer here. It was only implemented to automate the filter tuning procedure. In the future, it can be investigated if other optimizers such as Genetic Algorithms or Evolutionary optimizers can be used to see if they can tune the filters better. Only one objective, the MAE of the spacecraft position, has been minimized during the optimization process. In the future, a multi-objective optimizer should be used to minimize various aspects of the navigation. Other objectives could be the MAE of the velocity, the maximum error in position or velocity, the rate of convergence of the error, the converged error value etc. After seeing the position error in the estimate in Fig. 11.13, it is recommended to first run an optimization process with the MAE. This will produce some results that would be comparable to the true optimum. After seeing the behavior of the estimation process, another objective can be chosen to be minimized, such as the converged error. Not all the tuning parameters available to a filter have been optimized here. Also, the tuning parameters for the UKF, DD1, and DD2 were kept the same for the state propagation and the state estimation. These could be varied, along with the measurement noise that was assumed to be known. Adding more tuning parameters might increase the accuracy of the DD1 and DD2, which behaved in a very erratic manner. All these extra tuning parameters would make the problem larger, and the CPU time required for the optimization would increase. However, there might be a payoff in the accuracy of the estimation. The optimal parameters for filters using one set of coordinates should be tested out on the other set of coordinates to see the effect. An example would be to use the tuning parameters of the Cartesian UKF with the USM7 UKF. This might provide an insight on how the estimation process of the various models differs. All these additions could be easily implemented using the guidelines given in this chapter and would only require more implementation time.

Finally, other scenarios should be used in which navigation is carried out. These include among many others interplanetary flight, full 6 degrees of freedom state estimation of a spacecraft, low thrust orbit raising, formation flying etc. The state propagation frequency and the measurement frequency were kept constant for all the navigation cases in this chapter. These should be varied

to see the effect on the state estimation using Cartesian coordinates and the USMs. The measurement models for ground station tracking shown here are the conventional. In [Altman, 1975], measurement models based on velocity space are presented. These measurement models could be implemented in the future to see if there is any improvement of the navigation performance.

The next chapter focuses on Taylor Series Integration. This method, as the name implies, is a method of integration. However, there is again a dearth of information on the application of this method. Thus, the next chapter looks into this technique, and the equations required to fully use the TSI for orbit propagation using both Cartesian coordinates and the USM7 are presented.

Chapter 12

Taylor Series Integration

The previous chapter compares the navigation performance of the USMs and Cartesian coordinates. Four nonlinear filters were implemented and an optimizer was used to tune the filters. This chapter changes the area of investigation and is an exploratory look into Taylor Series Integration (TSI) and its applicability to the USM. The material is based on [Scott and Martini, 2008], which implement TSI for Cartesian coordinates. Moreover, no use of any automatic differentiation packages has been made. Therefore, this methodology can also be implemented in this work. TSI will be implemented for Cartesian coordinates and for the USM7. First, the two body problem is tested with both Cartesian coordinates and the USM7. Finally, the two body problem using low thrust is applied.

12.1 Theory

The same notation as found in [Scott and Martini, 2008] will be used here. The theory of Taylor Series (TS) states that a function can be described by an infinite series evaluated at a desired point. Let there be a state vector $\mathbf{x}$, with an initial condition $\mathbf{x}_0$. The state $\mathbf{x}$ consists of the state elements $x_n(t)$. The Taylor series expansion of $x_n(t)$ at $t = t_0$ can be written as

$$x_n(t) = \sum_{k=0}^{\infty} \left(\frac{x_n^{(k)}(t_0)}{k!}(t - t_0)^k \right) = x_n(t_0) + \sum_{k=1}^{\infty} \left(\frac{x_n^{(k)}(t_0)}{k!}(t - t_0)^k \right) \tag{12.1}$$

In Eq. (12.1),

$$x_n^k = \frac{\mathrm{d}^k x_n}{\mathrm{d}t^k} \tag{12.2}$$

To be able to use Taylor series efficiently for integration purposes, a method for finding the coefficients for the Taylor series expansion of each variable has to be used. For this purpose, it is important to define a normalized derivative of a function in the following manner

$$x^{[k]}(t) = \frac{x^{(k)}(t)}{k!} \tag{12.3}$$

Using the normalized derivative, Eq. (12.1) can be rewritten as

$$x_n(t) = \sum_{k=0}^{\infty} \left(x_n^{[k]}(t_0)(t - t_0)^k \right) \tag{12.4}$$

Using Eq. (12.3), it is clear that

$$x_n^{[0]}(t_0) = x_n(t_0) \tag{12.5}$$

For the application in this study, the state variables have to be integrated with respected to time. Thus, we define

$$x_n^{'} = u_n = \frac{\mathrm{d}x_n}{\mathrm{d}t} \tag{12.6}$$

Using Eq. (12.2), the following relations can be shown according to [Scott and Martini, 2008]

$$x_n^{(k)} = u_n^{(k-1)}, \qquad k \geq 1 \tag{12.7a}$$

$$\frac{x_n^{(k)}}{(k-1)!} = \frac{u_n^{(k-1)}}{(k-1)!}, \qquad k \geq 1 \tag{12.7b}$$

Let us define two vectors of normalized derivatives, X_n and U_n, where

$$X_n(k) = x_n^{[k]} \tag{12.8a}$$

$$U_n(k) = u_n^{[k]} \tag{12.8b}$$

Even though X_n and U_n are vectors, they are not expressed in bold. This is done in order to be consistent with the notation of [Scott and Martini, 2008]. X_n and U_n have a dimension $K \times 1$, where K is the maximum order of the TS. Finally, the following recursive relation can be shown between the elements of X_n and U_n

$$X_n(k) = \frac{U_n(k-1)}{k}, \qquad k \geq 1 \tag{12.9}$$

According to [Scott and Martini, 2008], the Taylor series solution of the original system variable $x_n(t)$ is

$$x_n(t) = \sum_{k=0}^{K} \left(X(k)(t-t_0)^k \right) + T_{n,K} \tag{12.10}$$

In Eq. (12.10), $T_{n,K}$ is the truncation error and K is the number of terms used in the expansion.

The state variables are always part of functions. The functions will undergo basic operations such as addition, division, multiplication, etc. There exist expressions that can find the normalized derivatives in a recursive manner and these normalized derivatives can be solved for using the theory found in [Jorba and Zou, 2004]. The functions are functions of time and the normalized derivatives are differentiated with respect to time. For addition and subtraction of two functions such as $c(t) = a(t) \pm b(t)$, we may rite

$$c^{[k]}(t) = a^{[k]}(t) \pm b^{[k]}(t) \tag{12.11}$$

For multiplication of two functions such as $c(t) = a(t)b(t)$:

$$c^{[k]}(t) = \sum_{j=0}^{k} \left(a^{[j]}(t) b^{[k-j]}(t) \right) \tag{12.12}$$

For division of two function such as $c(t) = a(t)/b(t)$:

$$c^{[k]}(t) = \frac{1}{b^{[0]}(t)} \left[a^{[k]}(t) - \sum_{j=1}^{k} \left(b^{[j]}(t) c^{[k-j]}(t) \right) \right] \tag{12.13}$$

For an exponent of a function such as $c(t) = a(t)^{\alpha}$:

$$c^{[k]}(t) = \frac{1}{k a^{[0]}(t)} \sum_{j=0}^{k-1} \left((\alpha(k-j) - j)\, a^{[k-j]}(t) c^{[j]}(t) \right) \tag{12.14}$$

Let two functions be defined as the sine and cosine of another function. Thus, $c(t) = \cos(a(t))$ and $b(t) = \sin(a(t))$. The normalized derivative for the cosine is

$$c^{[k]}(t) = -\frac{1}{k}\sum_{j=1}^{k}\left(jb^{[k-j]}(t)a^{[j]}(t)\right) \tag{12.15}$$

and for the sine

$$b^{[k]}(t) = \frac{1}{k}\sum_{j=1}^{k}\left(jc^{[k-j]}(t)a^{[j]}(t)\right) \tag{12.16}$$

12.2 Two-Body Problem

This setup is very similar to the one found in [Scott and Martini, 2008]. Here, however, no perturbing forces are used.

12.2.1 Cartesian Coordinates

Let the spacecraft state $\mathbf{x}$ consist of the three position and the three velocity coordinates in inertial Cartesian coordinates, and the spacecraft mass. The procedure shown here and the equations for this case are the same as shown in [Scott and Martini, 2008].

$$\mathbf{x} = \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \\ x_5(t) \\ x_6(t) \\ x_7(t) \end{bmatrix} = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \\ v_x(t) \\ v_y(t) \\ v_z(t) \\ m(t) \end{bmatrix} \tag{12.17}$$

From here on, (t) is removed for ease of notation. The first time derivative of the state elements for the two body problem are

$$\begin{aligned} x^{'} &= v_x \\ y^{'} &= v_y \\ z^{'} &= v_z \\ v_x^{'} &= -\mu\frac{x}{(x^2+y^2+z^2)^{3/2}} \\ v_y^{'} &= -\mu\frac{y}{(x^2+y^2+z^2)^{3/2}} \\ v_z^{'} &= -\mu\frac{z}{(x^2+y^2+z^2)^{3/2}} \\ m^{'} &= 0 \end{aligned} \tag{12.18}$$

This can be rewritten in terms of the state elements in the form of x_n as

$$x_1^{'} = x_4 \tag{12.19}$$

$$x_2^{'} = x_5 \tag{12.20}$$

$$x_3^{'} = x_6 \tag{12.21}$$

$$x_4^{'} = -\mu\frac{x_1}{(x_1^2+x_2^2+x_3^2)^{3/2}} \tag{12.22}$$

$$x_5^{'} = -\mu \frac{x_2}{(x_1^2 + x_2^2 + x_3^2)^{3/2}} \tag{12.23}$$

$$x_6^{'} = -\mu \frac{x_3}{(x_1^2 + x_2^2 + x_3^2)^{3/2}} \tag{12.24}$$

$$x_7^{'} = 0 \tag{12.25}$$

According to [Scott and Martini, 2008], two more elements should be added to the state.

$$x_8 = x_1^2 + x_2^2 + x_3^2 x_9 \qquad = x_8^{3/2} \tag{12.26}$$

By adding x_8 and x_9, r^2 and $r^{3/2}$ are computed by the integrator at all times. These quantities are required to compute the acceleration due to gravity. Using the definition of x_8 and x_9, the time derivatives of all the state elements are

$$x_1^{'} = x_4 \tag{12.27}$$

$$x_2^{'} = x_5 \tag{12.28}$$

$$x_3^{'} = x_6 \tag{12.29}$$

$$x_4^{'} = -\mu \frac{x_1}{x_9} \tag{12.30}$$

$$x_5^{'} = -\mu \frac{x_2}{x_9} \tag{12.31}$$

$$x_6^{'} = -\mu \frac{x_3}{x_9} \tag{12.32}$$

$$x_7^{'} = 0 \tag{12.33}$$

$$x_8^{'} = 2x_1 x_4 + 2x_2 x_5 + 2x_3 x_6 \tag{12.34}$$

$$x_9^{'} = \frac{3}{2} \frac{x_9 x_8^{'}}{x_8} \tag{12.35}$$

Using the fact that the time derivative of x_n can be expressed as u_n from Eq. (12.6), the time derivatives of the state elements can be rewritten to be

$$u_1 = x_4 \tag{12.36}$$

$$u_2 = x_5 \tag{12.37}$$

$$u_3 = x_6 \tag{12.38}$$

$$u_4 = -\mu \frac{x_1}{x_9} \tag{12.39}$$

$$u_5 = -\mu \frac{x_2}{x_9} \tag{12.40}$$

$$u_6 = -\mu \frac{x_3}{x_9} \tag{12.41}$$

$$u_7 = 0 \tag{12.42}$$

$$u_8 = 2x_1 x_4 + 2x_2 x_5 + 2x_3 x_6 \tag{12.43}$$

$$u_9 = \frac{3}{2} \frac{x_9 u_8}{x_8} \tag{12.44}$$

For the time derivatives, only simple operations on variables like addition, subtraction, multiplication and division exist. According to [Scott and Martini, 2008], these operations can be replaced by auxiliary variables on which, the recursive relations for the normalized derivative from Eqs. (12.11) till (12.16) can be applied. The auxiliary variables are

$$w_4 = \frac{x_1}{x_9} \tag{12.45}$$

$$w_5 = \frac{x_2}{x_9} \tag{12.46}$$

$$w_6 = \frac{x_3}{x_9} \tag{12.47}$$
$$w_{8,1} = x_1 x_4 \tag{12.48}$$
$$w_{8,2} = x_2 x_5 \tag{12.49}$$
$$w_{8,3} = x_3 x_6 \tag{12.50}$$
$$w_9 = \frac{x_9 u_8}{x_8} \tag{12.51}$$

From the time derivative relations, the normalized time derivatives for $k \geq 1$ are

$$U_1(k) = X_4(k) = \frac{U_4(k-1)}{k} \tag{12.52}$$
$$U_2(k) = X_5(k) = \frac{U_5(k-1)}{k} \tag{12.53}$$
$$U_3(k) = X_6(k) = \frac{U_6(k-1)}{k} \tag{12.54}$$
$$U_4(k) = -\mu W_4(k) \tag{12.55}$$
$$U_5(k) = -\mu W_5(k) \tag{12.56}$$
$$U_6(k) = -\mu W_6(k) \tag{12.57}$$
$$U_7(k) = 0 \tag{12.58}$$
$$U_8(k) = 2W_{8,1}(k) + 2W_{8,2}(k) + 2W_{8,3}(k) \tag{12.59}$$
$$U_9(k) = \frac{3}{2} W_9(k) \tag{12.60}$$

There are three generalized cases where the recursive relations have to be used. They are $w_t = x_m x_n$, $w_u = x_m / x_n$, and $w_v = x_n u_m / x_m$. The recursive relation for the generalized derivative vectors for the multiplication case is

$$W_t(k) = x_m \frac{U_n(k-1)}{k} + x_n \frac{U_m(k-1)}{k} + \sum_{j=1}^{k-1} \left(\frac{U_m(j-1)}{j} \frac{U_n(k-j-1)}{k-j} \right) \tag{12.61}$$

Eq. (12.61) can be derived by using Eq. (12.12) and Eq. (12.9). When deriving Eq. (12.61), the evaluations at $j = 0$ and $j = 1$ have to be carried out separately to avoid singularities. The steps in this derivation are shown here, then can be repeated for the other cases. Substituting the variables x_m and x_n in Eq. (12.12) gives

$$W_t(k) = \sum_{j=0}^{k} \left(X_m(j) X_n(k-j) \right) \tag{12.62}$$

For the recursive relations, the goal is to not require the knowledge of any $X_n(k)$ or $X_m(k)$. If they are required, the order of evaluating the various $X_i(k)$ would have to be fixed. This would introduce too many constraints. Thus, the goal is to use the relation in Eq. (12.9), which states that all $X_n(k)$ are equal to $U_n(k-1)/k$. However, this relation is only valid for $k \geq 1$. In Eq. (12.62), $X_m(0)$ is required when $j = 0$. Thus, the product inside the sum for when $j = 0$ can be written as

$$X_m(0) X_n(k) = x_m \frac{U_n(k-1)}{k} \tag{12.63}$$

In Eq. (12.62), $X_n(0)$ is required when $j = k$. Thus, the product inside the sum for when $j = k$ can be written as

$$X_m(k) X_n(0) = x_n \frac{U_m(k-1)}{k} \tag{12.64}$$

The remaining products inside the sum, for when $j = 1, 2, \ldots, k-1$ can be written as

$$X_m(k)X_n(k) = \frac{U_m(j-1)}{j}\frac{U_n(k-j-1)}{k-j} \tag{12.65}$$

Using Eqs. (12.63 - 12.65) with Eq. (12.62) results in Eq. (12.61). A similar method has to be used to derive the remaining recursive relations.

The recursive relation for the generalized derivative vectors for the division case is

$$W_u(k) = \frac{1}{x_n}\left[\frac{U_m(k-1)}{k} - \sum_{j=1}^{k}\left(\frac{U_n(j-1)}{j}W_u(k-j)\right)\right] \tag{12.66}$$

The recursive relation for the generalized derivative vectors for the multiplication followed by division case, which cannot be found in [Scott and Martini, 2008], is

$$W_v(k) = \frac{1}{x_m}\left[x_n U_m(k) + \sum_{j=1}^{k}\left(\frac{U_n(j-1)}{j}U_m(k-j)\right) - \sum_{j=1}^{k}\left(\frac{U_m(j-1)}{j}W_v(k-j)\right)\right] \tag{12.67}$$

For the case of Eq. (12.67), it is important to note that $U_m(k)$ has to be computed before $W_v(k)$.

12.2.2 Two Body problem with USM7

The same procedure as for the case of Cartesian coordinates is used. Again, there are no perturbing forces, so $a_{e1} = a_{e2} = a_{e3} = 0$. This fact is used heavily during the derivation of the TS functions.

Let the spacecraft state $\mathbf{x}$ consist of the seven USM7 elements and the spacecraft mass.

$$\mathbf{x} = \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \\ x_5(t) \\ x_6(t) \\ x_7(t) \\ x_8(t) \end{bmatrix} = \begin{bmatrix} C(t) \\ R_{f1}(t) \\ R_{f2}(t) \\ \epsilon_{O1}(t) \\ \epsilon_{O2}(t) \\ \epsilon_{O3}(t) \\ \eta_O(t) \\ m(t) \end{bmatrix} \tag{12.68}$$

The time derivative of the state elements for the unperturbed case is

$$\begin{aligned} C^{'} &= 0 \\ R_{f1}^{'} &= 0 \\ R_{f2}^{'} &= 0 \\ \epsilon_{O1}^{'} &= \frac{1}{2}\omega_3\epsilon_{O2} \\ \epsilon_{O2}^{'} &= -\frac{1}{2}\omega_3\epsilon_{O1} \\ \epsilon_{O3}^{'} &= \frac{1}{2}\omega_3\eta_O \\ \eta_O^{'} &= -\frac{1}{2}\omega_3\epsilon_{O3} \end{aligned} \tag{12.69}$$

The derivatives of the quaternion elements contains ω_3, which requires three additional parameters to be added.

$$\begin{aligned} \omega_3 &= Cv_{e2}^2/\mu \\ v_{e1} &= R_{f1}\cos\lambda + R_{f2}\sin\lambda \\ v_{e2} &= C - R_{f1}\sin\lambda + R_{f2}\cos\lambda \end{aligned} \tag{12.70}$$

For the derivatives of the new parameters, some algebraic manipulation of the original equations has to be carried out. This is so that the derivatives are simple expressions of the state elements.

$$\begin{aligned} \omega_3^{'} &= 2\frac{\omega_3 v_{e2}^{'}}{v_{e2}} \\ v_{e1}^{'} &= v_{e2}\omega_3 - C\omega_3 \\ v_{e2}^{'} &= -v_{e1}\omega_3 \end{aligned} \tag{12.71}$$

The additional elements are inserted in the following order to the state

$$x_9 = v_{e1} \tag{12.72}$$
$$x_{10} = v_{e2} \tag{12.73}$$
$$x_{11} = \omega_3 \tag{12.74}$$

The time derivatives in the form of u_n and x_n are

$$u_1 = 0 \tag{12.75}$$
$$u_2 = 0 \tag{12.76}$$
$$u_3 = 0 \tag{12.77}$$
$$u_4 = \frac{1}{2}x_5 x_{11} \tag{12.78}$$
$$u_5 = -\frac{1}{2}x_4 x_{11} \tag{12.79}$$
$$u_6 = \frac{1}{2}x_7 x_{11} \tag{12.80}$$
$$u_7 = -\frac{1}{2}x_6 x_{11} \tag{12.81}$$
$$u_8 = 0 \tag{12.82}$$
$$u_9 = x_{10}x_{11} - x_1 x_{11} \tag{12.83}$$
$$u_{10} = -x_9 x_{11} \tag{12.84}$$
$$u_{11} = 2\frac{x_{11}u_{10}}{x_{10}} \tag{12.85}$$

Some auxiliary functions need to again be defined

$$w_4 = x_5 x_{11} \tag{12.86}$$
$$w_5 = x_4 x_{11} \tag{12.87}$$
$$w_6 = x_7 x_{11} \tag{12.88}$$
$$w_7 = x_6 x_{11} \tag{12.89}$$
$$w_{9,1} = x_{10}x_{11} \tag{12.90}$$
$$w_{9,2} = x_1 x_{11} \tag{12.91}$$
$$w_{10} = x_9 x_{11} \tag{12.92}$$
$$w_{11} = \frac{x_{11}u_{10}}{x_{10}} \tag{12.93}$$

Finally, the normalized derivatives can be written as

$$U_1(k) = 0 \tag{12.94}$$
$$U_2(k) = 0 \tag{12.95}$$
$$U_3(k) = 0 \tag{12.96}$$
$$U_4(k) = \frac{1}{2}W_4(k) \tag{12.97}$$
$$U_5(k) = -\frac{1}{2}W_5(k) \tag{12.98}$$

$$U_6(k) = \frac{1}{2}W_6(k) \tag{12.99}$$
$$U_7(k) = -\frac{1}{2}W_7(k) \tag{12.100}$$
$$U_8(k) = 0 \tag{12.101}$$
$$U_9(k) = W_{9,1}(k) - W_{9,2}(k) \tag{12.102}$$
$$U_{10}(k) = -W_{10}(k) \tag{12.103}$$
$$U_{11}(k) = 2W_{11}(k) \tag{12.104}$$

12.3 Tangential Thrust

Since it was identified previously that trajectories undergoing continuous low-thrust propulsion are most suited for the USM, they will be tested for Taylor series integration. The thrust is assumed to be tangential for this scenario.

A few new recursive relations have to be derived for the normalized derivatives. For the case when $w = x_n u_m$, the relation is

$$W(k) = x_n U_m(k) + \sum_{j=1}^{k} \left(\frac{U_n(j-1)}{j} U_m(k-j) \right) \tag{12.105}$$

The equation above can again be derived by starting with the generalized recursive relation for multiplication. 12.63

$$W(k) = \sum_{j=0}^{k} (X_n(j) U_m(k-j)) \tag{12.106}$$

It is desirable to convert all the $X_n(j)$ to $U_n(j-1)/j$. There is a problem when $j = 0$, in which case the product inside the sum can be written as

$$x_n U_m(k) \tag{12.107}$$

The product for the remaining $j = 1, 2, \ldots, k$ is

$$\frac{U_n(j-1)}{j} U_m(k-j) \tag{12.108}$$

Another case is when $w = u_m/x_n$, the recursive relations for which, can be derived following the same methodology as for $w = x_n u_m$. The recursive relation for this case is

$$W(k) = \frac{1}{x_n} \left[U_m(k) - \sum_{j=1}^{k} \left(\frac{U_n(j-1)}{j} W(k-j) \right) \right] \tag{12.109}$$

12.3.1 Tangential Thrust for Cartesian Coordinates

The variables to be used for the integration are

$$x_1 = x \tag{12.110}$$
$$x_2 = y \tag{12.111}$$
$$x_3 = z \tag{12.112}$$
$$x_4 = v_x \tag{12.113}$$
$$x_5 = v_y \tag{12.114}$$

$$x_6 = v_z \tag{12.115}$$
$$x_7 = m \tag{12.116}$$
$$x_8 = r^2 = x_1^2 + x_2^2 + x_3^2 \tag{12.117}$$
$$x_9 = r^3 = x_8^{3/2} \tag{12.118}$$
$$x_{10} = v^2 = x_4^2 + x_5^2 + x_6^2 \tag{12.119}$$
$$x_{11} = v = \sqrt{x_{10}} \tag{12.120}$$
$$x_{12} = mv = x_7 x_{11} \tag{12.121}$$
$$\tag{12.122}$$

The time derivatives of the variables then are

$$u_1 = x_4 \tag{12.123}$$
$$u_2 = x_5 \tag{12.124}$$
$$u_3 = x_6 \tag{12.125}$$
$$u_4 = -\mu\frac{x_1}{x_9} + T\frac{x_4}{x_7 x_{11}} \tag{12.126}$$
$$u_5 = -\mu\frac{x_2}{x_9} + T\frac{x_5}{x_7 x_{11}} \tag{12.127}$$
$$u_6 = -\mu\frac{x_1}{x_9} + T\frac{x_6}{x_7 x_{11}} \tag{12.128}$$
$$u_7 = -\frac{T}{g_0 I_{sp}} \tag{12.129}$$
$$u_8 = 2x_1 x_4 + 2x_2 x_5 + 2x_3 x_6 \tag{12.130}$$
$$u_9 = \frac{3}{2}\frac{x_9 u_8}{x_8} \tag{12.131}$$
$$u_{10} = 2x_4 u_4 + 2x_5 u_5 + 2x_6 u_6 \tag{12.132}$$
$$u_{11} = \frac{1}{2}\frac{u_{10}}{x_{11}} \tag{12.133}$$
$$u_{12} = x_{11} u_7 + u_{11} x_7 \tag{12.134}$$

The auxiliary variables that need to be introduced are

$$w_{4,1} = \frac{x_1}{x_9} \tag{12.135}$$
$$w_{4,2} = \frac{x_4}{x_{12}} \tag{12.136}$$
$$w_{5,1} = \frac{x_2}{x_9} \tag{12.137}$$
$$w_{5,2} = \frac{x_5}{x_{12}} \tag{12.138}$$
$$w_{6,1} = \frac{x_3}{x_9} \tag{12.139}$$
$$w_{6,2} = \frac{x_6}{x_{12}} \tag{12.140}$$
$$w_{8,1} = x_1 x_4 \tag{12.141}$$
$$w_{8,2} = x_2 x_5 \tag{12.142}$$
$$w_{8,3} = x_3 x_6 \tag{12.143}$$
$$w_9 = \frac{x_9 u_8}{x_8} \tag{12.144}$$
$$w_{10,1} = x_4 u_4 \tag{12.145}$$
$$w_{10,2} = x_5 u_5 \tag{12.146}$$

$$w_{10,3} = x_6 u_6 \tag{12.147}$$
$$w_{11} = \frac{u_{10}}{x_{11}} \tag{12.148}$$
$$w_{12,1} = x_{11} u_7 \tag{12.149}$$
$$w_{12,2} = x_7 u_{11} \tag{12.150}$$

Finally, the recursive relations for finding the normalized derivatives of the time derivatives are

$$U_1(k) = X_4(k) = \frac{U_4(k-1)}{k} \tag{12.151}$$
$$U_2(k) = X_5(k) = \frac{U_5(k-1)}{k} \tag{12.152}$$
$$U_3(k) = X_6(k) = \frac{U_6(k-1)}{k} \tag{12.153}$$
$$U_4(k) = -muW_{4,1}(k) + TW_{4,2}(k) \tag{12.154}$$
$$U_5(k) = -muW_{5,1}(k) + TW_{5,2}(k) \tag{12.155}$$
$$U_6(k) = -muW_{6,1}(k) + TW_{6,2}(k) \tag{12.156}$$
$$\tag{12.157}$$
$$U_7(k) = 0 \tag{12.158}$$
$$U_8(k) = 2W_{8,1}(k) + 2W_{8,2}(k) + 2W_{8,3}(k) \tag{12.159}$$
$$U_9(k) = \frac{3}{2} W_9(k) \tag{12.160}$$
$$U_{10}(k) = 2W_{10,1}(k) + 2W_{10,2}(k) + 2W_{10,3}(k) \tag{12.161}$$
$$U_{11}(k) = \frac{1}{2} W_{11}(k) \tag{12.162}$$
$$U_{12}(k) = W_{12,1}(k) + W_{12,2}(k) \tag{12.163}$$

It should be noted that only $U_7(0) = -T/(g_0 I_{sp})$ and the remaining elements of U_7 are all 0. This is because the time derivative of the mass is a constant. Thus, the higher order derivatives should all be 0. This is also the case for any variable that varies linearly with time.

12.3.2 Tangential Thrust for USM7 Coordinates

For the case when there are no other perturbations than a continuous tangential low thrust, the variables of integration for the USM7 are

$$x_1 = C \tag{12.164}$$
$$x_2 = R_{f1} \tag{12.165}$$
$$x_3 = R_{f2} \tag{12.166}$$
$$x_4 = \epsilon_{O1} \tag{12.167}$$
$$x_5 = \epsilon_{O2} \tag{12.168}$$
$$x_6 = \epsilon_{O3} \tag{12.169}$$
$$x_7 = \eta_O \tag{12.170}$$
$$x_8 = m \tag{12.171}$$
$$x_9 = v_{e1} \tag{12.172}$$
$$x_{10} = v_{e2} \tag{12.173}$$
$$x_{11} = \omega_3 \tag{12.174}$$
$$x_{12} = v^2 \tag{12.175}$$
$$x_{13} = v \tag{12.176}$$
$$x_{14} = mv \tag{12.177}$$

Due to the complexity of the dynamics of the USM7, a new set of auxiliary variables, v_n, are defined. These variables behave like the w_n set of variables. The normalized derivatives of only their present state have to be computed, and not the normalized derivatives of their time derivative. This method is not used in [Scott and Martini, 2008] and is introduced here. This raises the important question of when an extra variable to the state, i.e., as x_n, and when the extra variable should be added as an auxiliary variable, i.e. v_n. There is no steadfast rule that can answer this question. A few guidelines are as follows: It is recommended to put the accelerations in the auxiliary set, otherwise, their time derivatives have to be implemented as well. This would unnecessarily add an extra order of complexity to the problem. Another important guideline is to check the complexity of the time derivatives compared to the actual variables. If it required fewer computations to find $U_n(k)$ than to find $X_n(k)$, it is best to add it to the list of variables, x_n, instead of v_n. Since this study only focuses a small part on TSI, a full investigation has not been carried out to optimize the calculation by moving variables from x_n to v_n and vice versa. This can be topic to be studied further in the future.

The auxiliary v_n variables are

$$v_1 = \epsilon_{O3}^2 + \eta_O^2 = x_6x_6 + x_7x_7 \tag{12.178}$$
$$v_2 = \epsilon_{O3}\eta_O = x_6x_7 \tag{12.179}$$
$$v_3 = \eta_O^2 - \epsilon_{O3}^2 = x_7x_7 - x_6x_6 \tag{12.180}$$
$$v_4 = \sin\lambda = 2v_2/v_1 \tag{12.181}$$
$$v_5 = \cos\lambda = v_3/v_1 \tag{12.182}$$
$$v_6 = p = x_1/x_{10} \tag{12.183}$$
$$v_7 = a_{e1} = Tx_9/x_{14} \tag{12.184}$$
$$v_8 = a_{e2} = Tx_{10}/x_{14} \tag{12.185}$$
$$v_9 = a_{e2}(1+p) = v_8 + v_8v_6 \tag{12.186}$$

When the v_n are used, their time derivatives are not computed. Thus, Eqs. (12.11 -12.16) can be immediately used for computations of their normalized derivatives. The time derivatives of the extended state elements are

$$u_1 = -pa_{e2} = -v_6v_8 \tag{12.187}$$
$$u_2 = a_{e1}\cos\lambda - a_{e2}(1+p)\sin\lambda = v_7v_5 - v_9v_4 \tag{12.188}$$
$$u_3 = a_{e1}\sin\lambda + a_{e2}(1+p)\cos\lambda = v_7v_4 + v_9v_5 \tag{12.189}$$
$$u_4 = \frac{1}{2}\omega_3\epsilon_{O2} = \frac{1}{2}x_{11}x_5 \tag{12.190}$$
$$u_5 = -\frac{1}{2}\omega_3\epsilon_{O1} = -\frac{1}{2}x_{11}x_4 \tag{12.191}$$
$$u_6 = \frac{1}{2}\omega_3\eta_O = \frac{1}{2}x_{11}x_7 \tag{12.192}$$
$$u_7 = -\frac{1}{2}\omega_3\epsilon_{O3} = -\frac{1}{2}x_{11}x_6 \tag{12.193}$$
$$u_8 = -T/(g_0I_{sp}) \tag{12.194}$$
$$u_9 = a_{e1} - C\omega_3 + v_{e2}\omega_3 = v_7 - x_1x_{11} + x_{10}x_{11} \tag{12.195}$$
$$u_{10} = a_{e2} - v_{e1}\omega_3 = v_8 - x_9x_{11} \tag{12.196}$$
$$u_{11} = \frac{x_{11}u_1}{x_1} + 2\frac{x_{11}u_{10}}{x_{10}} \tag{12.197}$$
$$u_{12} = 2x_9u_9 + 2x_{10}u_{10} \tag{12.198}$$
$$u_{13} = \frac{1}{2}\frac{u_{12}}{x_{13}} \tag{12.199}$$
$$u_{14} = x_8u_{13} + x_{13}u_8 \tag{12.200}$$

For all v_n and u_n, more auxiliary variables have to be created that simplify the computations

into single operations.These auxiliary variables are

$$v_{1,1} = x_6 x_6 \tag{12.201}$$
$$v_{1,2} = x_7 x_7 \tag{12.202}$$
$$v_{4,1} = v_2 / v_1 \tag{12.203}$$
$$v_{7,1} = x_9 / x_{14} \tag{12.204}$$
$$v_{8,1} = x_{10} / x_{14} \tag{12.205}$$
$$v_{9,1} = v_8 v_6 \tag{12.206}$$

$$w_1 = v_6 v_8 \tag{12.207}$$
$$w_{2,1} = v_7 v_5 \tag{12.208}$$
$$w_{2,2} = v_9 v_4 \tag{12.209}$$
$$w_{3,1} = v_7 v_4 \tag{12.210}$$
$$w_{3,2} = v_9 v_5 \tag{12.211}$$
$$w_4 = x_{11} x_5 \tag{12.212}$$
$$w_5 = x_{11} x_4 \tag{12.213}$$
$$w_6 = x_{11} x_7 \tag{12.214}$$
$$w_7 = x_{11} x_6 \tag{12.215}$$
$$w_{9,1} = x_1 x_{11} \tag{12.216}$$
$$w_{9,2} = x_{10} x_{11} \tag{12.217}$$
$$w_{10} = x_9 x_{11} \tag{12.218}$$
$$w_{11,1} = \frac{x_{11} u_1}{x_1} \tag{12.219}$$
$$w_{11,2} = \frac{x_{11} u_{10}}{x_{10}} \tag{12.220}$$
$$w_{12,1} = x_9 u_9 \tag{12.221}$$
$$w_{12,2} = x_{10} u_{10} \tag{12.222}$$
$$w_{13} = \frac{u_{12}}{x_{13}} \tag{12.223}$$
$$w_{14,1} = x_8 u_{13} \tag{12.224}$$
$$w_{14,2} = x_{13} u_8 \tag{12.225}$$

The normalized derivatives of v_i have to first be calculated, followed by the normalized derivatives of u_i. These normalized derivatives can be computed in the following way

$$V_1(k) = V_{1,1}(k) + V_{1,2}(k) \tag{12.226}$$
$$V_2(k) = V_2(k) \tag{12.227}$$
$$V_3(k) = V_{1,2}(k) - V_{1,1}(k) \tag{12.228}$$
$$V_4(k) = 2V_{4,1}(k) \tag{12.229}$$
$$v_6 = p = x_1 / x_{10} \tag{12.230}$$
$$V_5(k) = V_3(k) / V_1(k) \tag{12.231}$$
$$V_6(k) = X_1(k) / X_{10}(k) \tag{12.232}$$
$$V_7(k) = TV_{7,1}(k) \tag{12.233}$$
$$V_8(k) = TV_{8,1}(k) \tag{12.234}$$
$$V_9(k) = V_8(k) + V_{9,1}(k) \tag{12.235}$$

$$U_1(k) = -W_1(k) \tag{12.236}$$
$$U_2(k) = W_{2,1}(k) - W_{2,2}(k) \tag{12.237}$$

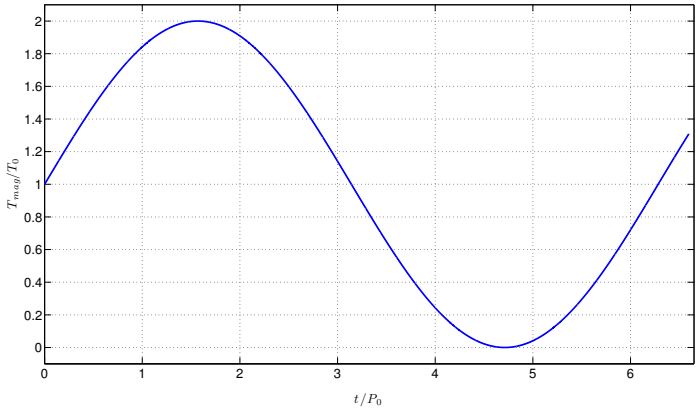


Figure 12.1: The thrusting profile

$$U_3(k) = W_{3,1}(k) + W_{3,2}(k) \tag{12.238}$$
$$U_4(k) = \frac{1}{2}W_4(k) \tag{12.239}$$
$$U_5(k) = \frac{1}{2}W_5(k) \tag{12.240}$$
$$U_6(k) = \frac{1}{2}W_6(k) \tag{12.241}$$
$$U_7(k) = \frac{1}{2}W_7(k) \tag{12.242}$$
$$U_8(k) = 0 \tag{12.243}$$
$$U_9(k) = V_7(k) - W_{9,1}(k) + W_{9,2}(k) \tag{12.244}$$
$$U_{10}(k) = V_8(k) - W_{10}(k) \tag{12.245}$$
$$U_{11}(k) = W_{11,1}(k) + 2W_{11,2}(k) \tag{12.246}$$
$$U_{12}(k) = 2W_{12,1}(k) + 2W_{12,2}(k) \tag{12.247}$$
$$U_{13}(k) = \frac{1}{2}W_{13}(k) \tag{12.248}$$
$$U_{14}(k) = W_{14,1}(k) + W_{14,2}(k) \tag{12.249}$$

It should be noted that $U_8(0) = -T/(g_0 I_{sp})$ and the remaining U_8 elements are all 0.

12.4 Varying Tangential Thrust Magnitude

In the previous section, the derived equations assumed a constant acceleration because the mass was not assumed to be varying. It is also important to check the performance when there is a certain control law. This is done by assuming that the magnitude of the thrust has a sine-like behavior over the period of initial orbit of the spacecraft, P_0. The equation for the thrust is then

$$T_{mag} = T_0 \times (\sin(t/P_0) + 1) \tag{12.250}$$

12.4.1 Cartesian Coordinates

For the varying tangential thrust magnitude, almost the same equations as found in section 12.3.1 have to be used. However, there are few additional variables that have to be added to take into

account the variation of the thrust. The additional state variables are

$$x_{13} = t \tag{12.251}$$
$$x_{14} = \sin(t/P_0) = \sin(x_{13}/P_0) \tag{12.252}$$
$$x_{15} = \cos(t/P_0) = \cos(x_{13}/P_0) \tag{12.253}$$
$$x_{16} = T_0 \sin(t/P_0) + T_0 = T_0 x_{14} + T_0 \tag{12.254}$$

The time derivatives of the additional state variables are

$$u_{13} = 1 \tag{12.255}$$
$$u_{14} = x_{15} u_{13}/P_0 \tag{12.256}$$
$$u_{15} = -x_{14} u_{13}/P_0 \tag{12.257}$$
$$u_{16} = T_0 u_{14} \tag{12.258}$$

Due to the non-constant nature of the thrust, some of the time derivatives of the original set of variables also change. More specifically, it is the time derivatives of the velocities that would change. They now become

$$\begin{aligned} u_4 &= -\mu \frac{x_1}{x_9} + \frac{x_{16} x_4}{x_{12}} \\ u_5 &= -\mu \frac{x_2}{x_9} + \frac{x_{16} x_5}{x_{12}} \\ u_6 &= -\mu \frac{x_3}{x_9} + \frac{x_{16} x_6}{x_{12}} \end{aligned} \tag{12.259}$$

Since the computation, x_{16}/x_{12}, occurs thrice, it is best to introduce a v_i variable here. Thus,

$$v_1 = \frac{x_{16}}{x_{12}} \tag{12.260}$$

This changes some of the auxiliary w_i variables to

$$w_{4,2} = v_1 x_4 \tag{12.261}$$
$$w_{5,2} = v_1 x_5 \tag{12.262}$$
$$w_{6,2} = v_1 x_6 \tag{12.263}$$

Finally,

$$U_4(k) = -\mu W_{4,1}(k) + W_{4,2}(k) \tag{12.264}$$
$$U_5(k) = -\mu W_{5,1}(k) + W_{5,2}(k) \tag{12.265}$$
$$U_6(k) = -\mu W_{6,1}(k) + W_{6,2}(k) \tag{12.266}$$
$$\tag{12.267}$$

12.4.2 USM7

For the varying tangential thrust magnitude, almost the same equations as found in section 12.3.2 have to be used. However, there are few additional variables that have to be added to take into account the varying of the thrust. The additional state variables are

$$x_{15} = t \tag{12.268}$$
$$x_{16} = \sin(t/P_0) = \sin(x_{15}/P_0) \tag{12.269}$$
$$x_{17} = \cos(t/P_0) = \cos(x_{15}/P_0) \tag{12.270}$$
$$x_{18} = T_0 \sin(t/P_0) + T_0 = T_0 x_{16} + T_0 \tag{12.271}$$

The time derivatives of the additional state variables are

$$u_{15} = 1 \tag{12.272}$$
$$u_{16} = x_{17}u_{15}/P_0 \tag{12.273}$$
$$u_{17} = -x_{16}u_{15}/P_0 \tag{12.274}$$
$$u_{18} = T_0 u_{16} \tag{12.275}$$

The computation of the auxiliary perturbing acceleration values, $v_7 = a_{e1}$ and $v_8 = a_{e2}$ will change. A new auxiliary variable $v_{7,1}$ is added

$$v_{7,1} = \frac{x_{18}}{x_{14}} \tag{12.276}$$

The accelerations then become

$$v_7 = v_{7,1}x_9 \tag{12.277}$$
$$v_8 = v_{7,1}x_{10} \tag{12.278}$$

The remaining equations are the same as the case with the constant tangential thrust. Thus, this modification is very simple for the USM7 case.

12.5 Thrust in all directions

Thrusting is not always tangential to the velocity, and also not always in the same plane. Thus, the problem has to be modified to ensure that these thrust scenarios can also be taken into account. This could happen for example when the orbital plane has to be changed using low-thrust propulsion, or if solar sailing is used. For this method, the unit vectors in the direction of the angular momentum, and in a direction laying in the orbital plane but perpendicular to the velocity has to be used. These unit vectors can be easily derived for the USM7 without adding many auxiliary variables. For the case of Cartesian coordinates, however, many auxiliary variables have to be added to compute these directions.

The arbitrarily applied thrust can be seen relative to F_e in Fig. 12.2. An angle β is defined to be the angle between the thrust vector and the orbit normal. The angle α is defined to be the angle between the projection of the thrust on the orbital plane and the velocity vector.

12.5.1 Equations for the USM7

For this case, the acceleration along the orbit normal, a_{e3} would be equal to

$$a_{e3} = \frac{T}{m}\cos\beta \tag{12.279}$$

The component of the thrust laying in the orbital plane is $T\sin\beta$. This thrust has a component laying along the velocity vector of the spacecraft $T\sin\beta\cos\alpha$, and a component perpendicular to the velocity vector but in the orbital plane $T\sin\beta\sin\alpha$. To make the computations simpler, a new reference frame $\mathcal{F}_{e'}$ is defined as seen in Fig. 12.2. This reference frame consists of the following axes: $\hat{\mathbf{e}}_3^{'}$ laying along $\hat{\mathbf{e}}_3$, $\hat{\mathbf{e}}_2^{'}$ laying along $\mathbf{v}$, and $\hat{\mathbf{e}}_1^{'}$ completing the frame. Expressed in $\mathcal{F}_e$, $\hat{\mathbf{e}}_2^{'}$ and $\hat{\mathbf{e}}_1^{'}$ are

$$\begin{aligned}\hat{\mathbf{e}}_1^{'} &= \frac{1}{v}\begin{bmatrix} v_{e2} \\ -v_{e1} \\ 0 \end{bmatrix} \\ \hat{\mathbf{e}}_2^{'} &= \frac{1}{v}\begin{bmatrix} v_{e1} \\ v_{e2} \\ 0 \end{bmatrix}\end{aligned} \tag{12.280}$$

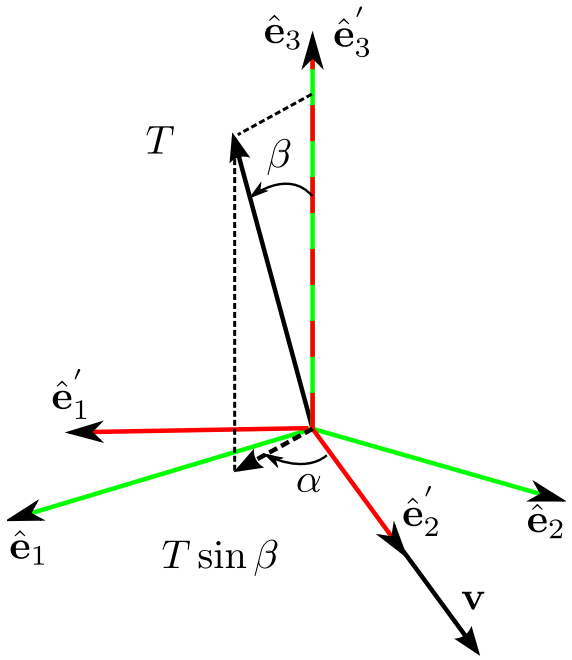


Figure 12.2: Orientation of the thrust vector

Using the fact that the thrust projected along $\hat{\mathbf{e}}_1'$ is $T\sin\beta\sin\alpha$ and the along $\hat{\mathbf{e}}_2'$ is $T\sin\beta\cos\alpha$, the remaining two accelerations can be found to be

$$\begin{aligned} a_{e1} &= \frac{T}{mv}(\sin\beta\cos\alpha v_{e1} + \sin\beta\sin\alpha v_{e2}) \\ a_{e2} &= \frac{T}{mv}(\sin\beta\cos\alpha v_{e2} - \sin\beta\sin\alpha v_{e1}) \end{aligned} \tag{12.281}$$

To make the problem more dynamically complex, the orientation of the thrust is made to vary with time. The orientation of the thrust in the orbital plane and normal to the plane is changed, i.e. α and β are varied with time. An initial circular equatorial LEO is considered, and the thrust applied continuously. The steering angle, α has a mean value of 0°, however, it has a sinusoidal behavior. The other steering angle, β starts at a certain angle and then decreases quadratically with time. The magnitude of the thrust remains constant, but the actual acceleration increases with time due to the decreasing mass. The equations to be used for this steering are

$$T(t) = T_0 \tag{12.282}$$

$$\alpha(t) = \sin(t/P_0) \tag{12.283}$$

$$\beta(t) = \frac{1}{4}\cos(t/P_0) + \beta_0 \tag{12.284}$$

Equations for the USM7 with Generic Perturbing Accelerations

To implement this scenario, it is necessary to first set up the Taylor series equations for the USM that can handle any scenario. First, the extended state vector for the USM7 is presented.

$$x_1 = C \tag{12.285}$$

$$x_2 = R_{f1} \tag{12.286}$$

$$x_3 = R_{f2} \tag{12.287}$$

$$x_4 = \epsilon_{O1} \tag{12.288}$$

$$x_5 = \epsilon_{O2} \tag{12.289}$$
$$x_6 = \epsilon_{O3} \tag{12.290}$$
$$x_7 = \eta_O \tag{12.291}$$
$$x_8 = m \tag{12.292}$$
$$x_9 = v_{e1} \tag{12.293}$$
$$x_{10} = v_{e2} \tag{12.294}$$
$$x_{11} = w_3 \tag{12.295}$$

The following v_i auxiliary variables should be used

$$v_1 = \epsilon_{O3}^2 + \eta_O^2 = x_6^2 + x_7^2 \tag{12.296}$$
$$v_2 = \epsilon_{O3}\eta_O = x_6 x_7 \tag{12.297}$$
$$v_3 = \eta_O^2 - \epsilon_{O3}^2 = x_7^2 - x_6^2 \tag{12.298}$$
$$v_4 = \epsilon_{O1}\epsilon_{O3} - \epsilon_{O2}\eta_O = x_4 x_6 - x_5 x_7 \tag{12.299}$$
$$v_5 = \sin\lambda = 2v_2/v_1 \tag{12.300}$$
$$v_6 = \cos\lambda = v_3/v_1 \tag{12.301}$$
$$v_7 = \gamma = v_4/v_1 \tag{12.302}$$
$$v_8 = p = x_1/x_{10} \tag{12.303}$$
$$v_9 = \gamma/v_{e2} = v_7/x_{10} \tag{12.304}$$
$$v_{10} = R_{f1}\gamma/v_{e2} = x_2 v_9 \tag{12.305}$$
$$v_{11} = R_{f2}\gamma/v_{e2} = x_3 v_9 \tag{12.306}$$
$$\tag{12.307}$$

The above auxiliary variables are the ones based on the state that are required for simulation of perturbed orbits. Any additional auxiliary variables that are required for the perturbing accelerations can be computed after these and before the following variables.

$$v_{12} = a_{e1} \tag{12.308}$$
$$v_{13} = a_{e2} \tag{12.309}$$
$$v_{14} = a_{e3} \tag{12.310}$$
$$v_{15} = \omega_1 = v_{14}/x_{10} \tag{12.311}$$
$$v_{16} = a_{e2}(1+p) = v_{13} + v_{13}v_8 \tag{12.312}$$
$$v_{17} = a_{e3}\gamma R_{f1}/v_{e2} = v_{14}v_{10} \tag{12.313}$$
$$v_{18} = a_{e3}\gamma R_{f2}/v_{e2} = v_{14}v_{11} \tag{12.314}$$

Finally, the time derivatives can be written as

$$u_1 = -v_8 v_{13} \tag{12.315}$$
$$u_2 = v_{12}v_6 - v_{16}v_5 - v_{18} \tag{12.316}$$
$$u_3 = v_{12}v_5 + v_{16}v_6 + v_{17} \tag{12.317}$$
$$u_4 = \frac{1}{2}x_{11}x_5 + \frac{1}{2}v_{15}x_7 \tag{12.318}$$
$$u_5 = -\frac{1}{2}x_{11}x_4 + \frac{1}{2}v_{15}x_6 \tag{12.319}$$
$$u_6 = \frac{1}{2}x_{11}x_7 - \frac{1}{2}v_{15}x_5 \tag{12.320}$$
$$u_7 = -\frac{1}{2}x_{11}x_6 - \frac{1}{2}v_{15}x_4 \tag{12.321}$$
$$u_8 = -T/(g_0 I_{sp}) \tag{12.322}$$

$$u_9 = v_{12} - x_1 x_{11} + x_{10} x_{11} \tag{12.323}$$
$$u_{10} = v_{13} - x_9 x_{11} \tag{12.324}$$
$$u_{11} = \frac{x_{11} u_1}{x_1} + 2\frac{x_{11} u_{10}}{x_{10}} \tag{12.325}$$

Equations for the USM7 for Varying Thrust Orientation

Now the equations for the USM7 for the thrusting case of this section is presented. The extended state for this thrusting scenario is

$$x_1 = C \tag{12.326}$$
$$x_2 = R_{f1} \tag{12.327}$$
$$x_3 = R_{f2} \tag{12.328}$$
$$x_4 = \epsilon_{O1} \tag{12.329}$$
$$x_5 = \epsilon_{O2} \tag{12.330}$$
$$x_6 = \epsilon_{O3} \tag{12.331}$$
$$x_7 = \eta_O \tag{12.332}$$
$$x_8 = m \tag{12.333}$$
$$x_9 = v_{e1} \tag{12.334}$$
$$x_{10} = v_{e2} \tag{12.335}$$
$$x_{11} = w_3 \tag{12.336}$$
$$x_{12} = v^2 \tag{12.337}$$
$$x_{13} = v \tag{12.338}$$
$$x_{14} = mv \tag{12.339}$$
$$x_{15} = t \tag{12.340}$$
$$x_{16} = \alpha = \sin(t/P_0) \tag{12.341}$$
$$x_{17} = \cos(t/P_0) \tag{12.342}$$
$$x_{18} = \beta = \frac{1}{4}\cos(t/P_0) + \beta_0 \tag{12.343}$$
$$x_{19} = \sin\alpha \tag{12.344}$$
$$x_{20} = \cos\alpha \tag{12.345}$$
$$x_{21} = \sin\beta \tag{12.346}$$
$$x_{22} = \cos\beta \tag{12.347}$$

The necessary auxiliary v_n variables for generic pertubed cases are

$$v_1 = \epsilon_{O3}^2 + \eta_O^2 = x_6^2 + x_7^2 \tag{12.348}$$
$$v_2 = \epsilon_{O3}\eta_O = x_6 x_7 \tag{12.349}$$
$$v_3 = \eta_O^2 - \epsilon_{O3}^2 = x_7^2 - x_6^2 \tag{12.350}$$
$$v_4 = \epsilon_{O1}\epsilon_{O3} - \epsilon_{O2}\eta_O = x_4 x_6 - x_5 x_7 \tag{12.351}$$
$$v_5 = \sin\lambda = 2v_2/v_1 \tag{12.352}$$
$$v_6 = \cos\lambda = v_3/v_1 \tag{12.353}$$
$$v_7 = \gamma = v_4/v_1 \tag{12.354}$$
$$v_8 = p = x_1/x_{10} \tag{12.355}$$
$$v_9 = \gamma/v_{e2} = v_7/x_{10} \tag{12.356}$$
$$v_{10} = R_{f1}\gamma/v_{e2} = x_2 v_9 \tag{12.357}$$
$$v_{11} = R_{f2}\gamma/v_{e2} = x_3 v_9 \tag{12.358}$$

(12.359)

The new v_n variables that are required to express the varying orientation of the thrust are

$$v_{12} = \sin\beta\cos\alpha = x_{21}x_{20} \tag{12.360}$$
$$v_{13} = \sin\beta\sin\alpha = x_{21}x_{19} \tag{12.361}$$
$$v_{14} = \frac{\sin\beta\cos\alpha}{mv} = \frac{v_{12}}{x_{14}} \tag{12.362}$$
$$v_{15} = \frac{\sin\beta\sin\alpha}{mv} = \frac{v_{13}}{x_{14}} \tag{12.363}$$

The accelerations can then be computed in the following manner

$$v_{16} = Tv_{14}x_9 + Tv_{15}x_{10} \tag{12.364}$$
$$v_{17} = Tv_{14}x_{10} - Tv_{15}x_9 \tag{12.365}$$
$$v_{18} = T\frac{x_{22}}{x_8} \tag{12.366}$$

Finally, the remaining v_n variables that are based on the perturbing accelerations are

$$v_{19} = \omega_1 = \frac{v_{18}}{x_{10}} \tag{12.367}$$
$$v_{20} = a_{e2}(1+p) = v_{17} + v_{17}v_8 \tag{12.368}$$
$$v_{21} = v_{18}v_{10} \tag{12.369}$$
$$v_{22} = v_{18}v_{11} \tag{12.370}$$

The time derivatives of the state variables are

$$u_1 = -v_8v_{17} \tag{12.371}$$
$$u_2 = v_{16}v_6 - v_{20}v_5 - v_{22} \tag{12.372}$$
$$u_3 = v_{16}v_5 + v_{20}v_6 + v_{21} \tag{12.373}$$
$$u_4 = \frac{1}{2}x_{11}x_5 + \frac{1}{2}v_{19}x_7 \tag{12.374}$$
$$u_5 = -\frac{1}{2}x_{11}x_4 + \frac{1}{2}v_{19}x_6 \tag{12.375}$$
$$u_6 = \frac{1}{2}x_{11}x_7 - \frac{1}{2}v_{19}x_5 \tag{12.376}$$
$$u_7 = -\frac{1}{2}x_{11}x_6 - \frac{1}{2}v_{19}x_4 \tag{12.377}$$
$$u_8 = -T/(g_0I_{sp}) \tag{12.378}$$
$$u_9 = v_{16} - x_1x_{11} + x_{10}x_{11} \tag{12.379}$$
$$u_{10} = v_{17} - x_9x_{11} \tag{12.380}$$
$$u_{11} = \frac{x_{11}u_1}{x_1} + 2\frac{x_{11}u_{10}}{x_{10}} \tag{12.381}$$
$$u_{12} = 2x_9u_9 + 2x_{10}u_{10} \tag{12.382}$$
$$u_{13} = \frac{1}{2}\frac{u_{12}}{x_{13}} \tag{12.383}$$
$$u_{14} = x_8u_{13} + x_{13}u_8 \tag{12.384}$$
$$u_{15} = 1 \tag{12.385}$$
$$u_{16} = x_{17}u_{15}/P_0 \tag{12.386}$$
$$u_{17} = -x_{16}u_{15}/P_0 \tag{12.387}$$
$$u_{18} = u_{17}/4 \tag{12.388}$$

$$u_{19} = x_{20}u_{16} \tag{12.389}$$
$$u_{20} = -x_{19}u_{16} \tag{12.390}$$
$$u_{21} = x_{22}u_{18} \tag{12.391}$$
$$u_{22} = -x_{21}u_{18} \tag{12.392}$$

12.5.2 Equations for Cartesian Coordinates

In this subsection, the equations for Cartesian coordinates and TSI are presented. These equations have not be implemented and validates, unlike all the previous sets of equations, and it is therefore recommended that the reader re-derive them in case implementation is the goal. Since the thrusting for this scenario is best expressed in terms of the orbital reference frame $\mathcal{F}_e$, and the velocity based reference frame $\mathcal{F}_{e'}$, the first step is to find the unit vectors of these two reference frames in terms of Cartesian coordinates. The unit vectors of $\mathcal{F}_e$ are

$$\hat{\mathbf{e}}_1 = \frac{1}{r}\begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{12.393a}$$
$$\hat{\mathbf{e}}_2 = -\frac{1}{rh}\begin{bmatrix} h_y z - h_z y \\ h_z x - h_x z \\ h_x y - h_y x \end{bmatrix} \tag{12.393b}$$
$$\hat{\mathbf{e}}_3 = \frac{1}{h}\begin{bmatrix} h_x \\ h_y \\ h_z \end{bmatrix} \tag{12.393c}$$

The unit vectors of $\mathcal{F}_{e'}$ then become

$$\hat{\mathbf{e}}_1^{'} = \frac{1}{hv}\begin{bmatrix} h_z v_y - h_y v_z \\ h_x v_z - h_z v_x \\ h_y v_x - h_x v_y \end{bmatrix} \tag{12.394a}$$
$$\hat{\mathbf{e}}_2^{'} = -\frac{1}{v}\begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} \tag{12.394b}$$
$$\hat{\mathbf{e}}_3^{'} = \frac{1}{h}\begin{bmatrix} h_x \\ h_y \\ h_z \end{bmatrix} \tag{12.394c}$$

The accelerations due to thrust expressed in Cartesian coordinates in the inertial reference frame are

$$a_{Tx} = \frac{T}{m}\left[\sin\beta\left(\cos\alpha\frac{v_x}{v} + \sin\alpha\frac{h_z v_y - h_y v_z}{hv}\right) + \cos\beta\frac{h_x}{h}\right] \tag{12.395a}$$
$$a_{Ty} = \frac{T}{m}\left[\sin\beta\left(\cos\alpha\frac{v_y}{v} + \sin\alpha\frac{h_x v_z - h_z v_x}{hv}\right) + \cos\beta\frac{h_y}{h}\right] \tag{12.395b}$$
$$a_{Tz} = \frac{T}{m}\left[\sin\beta\left(\cos\alpha\frac{v_z}{v} + \sin\alpha\frac{h_y v_x - h_x v_y}{hv}\right) + \cos\beta\frac{h_z}{h}\right] \tag{12.395c}$$

The extended state for this thrusting scenario for Cartesian coordinates is

$$x_1 = x \tag{12.396}$$
$$x_2 = y \tag{12.397}$$
$$x_3 = z \tag{12.398}$$
$$x_4 = v_x \tag{12.399}$$
$$x_5 = v_y \tag{12.400}$$

$$x_6 = v_z \tag{12.401}$$
$$x_7 = m \tag{12.402}$$
$$x_8 = r^2 = x_1^2 + x_2^2 + x_3^2 \tag{12.403}$$
$$x_9 = r^3 = x_8^{3/2} \tag{12.404}$$
$$x_{10} = v^2 = x_4^2 + x_5^2 + x_6^2 \tag{12.405}$$
$$x_{11} = v = x_{10}^{1/2} \tag{12.406}$$
$$x_{12} = t \tag{12.407}$$
$$x_{13} = r = x_8^{1/2} \tag{12.408}$$
$$x_{14} = h_x = x_2 x_6 - x_3 x_5 \tag{12.409}$$
$$x_{15} = h_y = x_3 x_4 - x_1 x_6 \tag{12.410}$$
$$x_{16} = h_z = x_1 x_5 - x_2 x_4 \tag{12.411}$$
$$x_{17} = h^2 = x_{14}^2 + x_{15}^2 + x_{16}^2 \tag{12.412}$$
$$x_{18} = h = x_{17}^{1/2} \tag{12.413}$$
$$x_{19} = \alpha = \sin(t/P_0) \tag{12.414}$$
$$x_{20} = \cos(t/P_0) \tag{12.415}$$
$$x_{21} = \beta = \frac{1}{4}\cos(t/P_0) + \beta_0 \tag{12.416}$$
$$x_{22} = \sin\alpha \tag{12.417}$$
$$x_{23} = \cos\alpha \tag{12.418}$$
$$x_{24} = \sin\beta \tag{12.419}$$
$$x_{25} = \cos\beta \tag{12.420}$$
$$\tag{12.421}$$

The auxiliary v_n variables are

$$v_1 = a_{gx} = -\mu\frac{x_1}{x_9} \tag{12.422}$$
$$v_2 = a_{gy} = -\mu\frac{x_2}{x_9} \tag{12.423}$$
$$v_3 = a_{gz} = -\mu\frac{x_3}{x_9} \tag{12.424}$$
$$v_4 = \frac{v_x}{v} = \frac{x_4}{x_{11}} \tag{12.425}$$
$$v_5 = \frac{v_y}{v} = \frac{x_5}{x_{11}} \tag{12.426}$$
$$v_6 = \frac{v_z}{v} = \frac{x_6}{x_{11}} \tag{12.427}$$
$$v_7 = \frac{h_x}{h} = \frac{x_{14}}{x_{18}} \tag{12.428}$$
$$v_8 = \frac{h_y}{h} = \frac{x_{15}}{x_{18}} \tag{12.429}$$
$$v_9 = \frac{h_z}{h} = \frac{x_{16}}{x_{18}} \tag{12.430}$$
$$v_{10} = \sin\beta\cos\alpha = x_{24}x_{23} \tag{12.431}$$
$$v_{11} = \sin\beta\sin\alpha = x_{24}x_{22} \tag{12.432}$$
$$v_{12} = \frac{h_z v_y}{hv} = v_9 v_5 \tag{12.433}$$
$$v_{13} = \frac{h_y v_z}{hv} = v_8 v_6 \tag{12.434}$$

$$v_{14} = \frac{h_z v_y - h_y v_z}{hv} = v_{12} - v_{13} \tag{12.435}$$

$$v_{15} = \frac{h_x v_z}{hv} = v_7 v_6 \tag{12.436}$$

$$v_{16} = \frac{h_z v_x}{hv} = v_9 v_4 \tag{12.437}$$

$$v_{17} = \frac{h_x v_z - h_z v_x}{hv} = v_{15} - v_{16} \tag{12.438}$$

$$v_{18} = \frac{h_y v_x}{hv} = v_8 v_4 \tag{12.439}$$

$$v_{19} = \frac{h_x v_y}{hv} = v_7 v_5 \tag{12.440}$$

$$v_{20} = \frac{h_y v_x - h_x v_y}{hv} = v_{18} - v_{19} \tag{12.441}$$

$$v_{21} = \sin\beta \cos\alpha \frac{v_x}{v} = v_{10} v_4 \tag{12.442}$$

$$v_{22} = \sin\beta \sin\alpha \frac{h_z v_y - h_y v_z}{hv} = v_{11} v_{14} \tag{12.443}$$

$$v_{23} = \cos\beta \frac{h_x}{h} = x_{25} v_7 \tag{12.444}$$

$$v_{24} = \sin\beta \cos\alpha \frac{v_y}{v} = v_{10} v_5 \tag{12.445}$$

$$v_{25} = \sin\beta \sin\alpha \frac{h_x v_z - h_z v_x}{hv} = v_{11} v_{17} \tag{12.446}$$

$$v_{26} = \cos\beta \frac{h_y}{h} = x_{25} v_8 \tag{12.447}$$

$$v_{27} = \sin\beta \cos\alpha \frac{v_z}{v} = v_{10} v_6 \tag{12.448}$$

$$v_{28} = \sin\beta \sin\alpha \frac{h_y v_x - h_x v_y}{hv} = v_{11} v_{20} \tag{12.449}$$

$$v_{29} = \cos\beta \frac{h_z}{h} = x_{25} v_9 \tag{12.450}$$

$$v_{30} = T_x = T(v_{21} + v_{22} + v_{23}) \tag{12.451}$$

$$v_{31} = a_{Tx} = v_{30}/x_7 \tag{12.452}$$

$$v_{32} = T_y = T(v_{24} + v_{25} + v_{26}) \tag{12.453}$$

$$v_{33} = a_{Ty} = v_{32}/x_7 \tag{12.454}$$

$$v_{34} = T_z = T(v_{27} + v_{28} + v_{29}) \tag{12.455}$$

$$v_{35} = a_{Tz} = v_{34}/x_7 \tag{12.456}$$

Finally, the time derivatives of the extended state variables can be computed in the following way

$$u_1 = v_x = x_4 \tag{12.457}$$

$$u_2 = v_y = x_5 \tag{12.458}$$

$$u_3 = v_z = x_6 \tag{12.459}$$

$$u_4 = a_x = v_1 + v_{31} \tag{12.460}$$

$$u_5 = a_y = v_2 + v_{33} \tag{12.461}$$

$$u_6 = a_z = v_3 + v_{35} \tag{12.462}$$

$$u_7 = \dot{m} = -T/(g_0 I_{sp}) \tag{12.463}$$

$$u_8 = 2(x_1 x_4 + x_2 x_5 + x_3 x_6) \tag{12.464}$$

$$u_9 = \frac{3}{2} \frac{x_9 u_8}{x_8} \tag{12.465}$$

$$u_{10} = 2(x_4 u_4 + x_5 u_5 x_6 u_6) \tag{12.466}$$

$$u_{11} = \frac{1}{2}\frac{u_{10}}{x_{11}} \tag{12.467}$$
$$u_{12} = \dot{t} = 1 \tag{12.468}$$
$$u_{13} = \frac{1}{2}\frac{u_8}{x_{13}} \tag{12.469}$$
$$u_{14} = \dot{h}_x = u_2x_6 + x_2u_6 - u_3x_5 - x_3u_5 \tag{12.470}$$
$$u_{15} = \dot{h}_y = u_3x_4 + x_3u_4 - u_1x_6 - x_1u_6 \tag{12.471}$$
$$u_{16} = \dot{h}_z = u_1x_5 + x_1u_5 - u_2x_4 - x_2u_4 \tag{12.472}$$
$$u_{17} = 2(x_{14}u_{14} + x_{15}u_{15} + x_{16}u_{16}) \tag{12.473}$$
$$u_{18} = \dot{h} = \frac{1}{2}\frac{u_{17}}{x_{18}} \tag{12.474}$$
$$u_{19} = x_{20}u_{12} \tag{12.475}$$
$$u_{20} = -x_{19}u_{12} \tag{12.476}$$
$$u_{21} = u_{20}/4 \tag{12.477}$$
$$u_{22} = x_{23}u_{19} \tag{12.478}$$
$$u_{23} = -x_{22}u_{19} \tag{12.479}$$
$$u_{24} = x_{25}u_{21} \tag{12.480}$$
$$u_{25} = -x_{24}u_{21} \tag{12.481}$$

All the equations and theory necessary for the implementation of the TSI method has been presented here. In the next chapter, the TSI is implemented to be able to get quantitative results.

Chapter 13

Application of the Taylor Series Integration

The theoretical background for Taylor series integration was presented in Section 12.1 and the equation setup was shown for various cases for both Cartesian coordinates and USM7. In this chapter, these equations are implemented and results compared with the traditional Runge-Kutta integration methods. The equations available in Chapter 12 are only capable of propagating the state by one time-step. To be able to fully implement the Taylor series integration method, it is essential to create an integrator, especially a variable step-size one. All results presented in this chapter are for the cases presented in Chapter 12. Therefore, the cases are repeated here.

Case 1 The 2-body problem presented in Section 12.2

Case 2 Tangential low-thrust with constant acceleration, presented in Section 12.3

Case 3 Tangential low-thrust with a varying acceleration, presented in Section 12.4

Case 4 Low-thrust with constant thrust and varying orientation, presented in Section 12.5

13.1 Integrator

A variable step-size integrator has been created with the error control method found in [Scott and Martini, 2008]. This step-size controller guarantees that the specified absolute tolerance is satisfied. For an order of integration, K, the absolute tolerance requirement, τ, can be expressed in the following relation

$$|X_n(K-1)|\, h^{K-1} + |X_n(K)|\, h^K \leq \tau \tag{13.1}$$

The suitable next time step-size, h_{i+1}, based on the current time step-size, h_i, is

$$h_{i+1} = \eta \exp\left(\frac{1}{K-1} \ln \frac{\tau}{|X_n(K-1)| + h_i\, |X_n(K)|}\right) \tag{13.2}$$

In Eq. (13.2), η is a safety factor and is chosen to be 0.9. A step-size is computed for each state variable, x_n, and the lowest time step-size is chosen.

As can be seen in the equations in Chapter 12, some extra variables have been added to the state for all the cases. For trajectory integration purposes, the user only requires a tolerance on the original state parameters, i.e. the position and velocity for Cartesian coordinates, and the quaternion and Hodographic velocities for the USM7. Thus, the integrator built during this thesis study takes as a parameter the length of the original state. Therefore, the specified tolerances are only applied to the essential, or desired parameters of the Taylor series integration state.

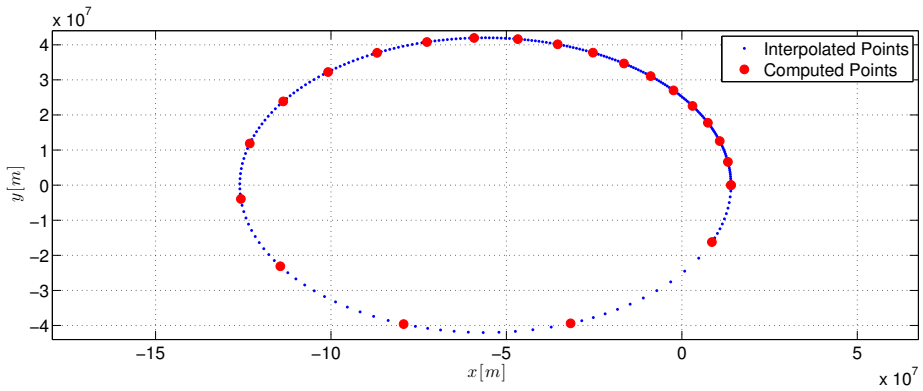


Figure 13.1: Difference between the default points used outputted by the integrator and the interpolated points

The main motivation for using Taylor series integration is that it has been shown to be very accurate and fast. Due to the possibility of using high order expansions during the integration, the time step-sizes can become very large when compared to normal Runge Kutta integration methods. Sometimes, it is necessary to have a denser output than the one created by the time step-sizes chosen by the integrator. This is especially true for trajectory visualization, or ephemeris creation. Artificially decreasing the step-size is not the most efficient way of creating the denser output. With RK type integrators, it is possible to use interpolating functions to create this. This involves additional programming and the interpolation is not the same level of accuracy as the discrete values computed by the integrator. The benefit of Taylor series integration is that the normalized derivative vectors, X_n, provide information on the trajectory in between the computed discrete points. This information is of the same order of accuracy as the computed point.

To take advantage of the approximation technique of the Taylor series, a refinement parameter can also be provided as an input to the integration. The refinement factor specifies how many points are required between each integration step. The integrator then selects the appropriate $(t - t_0)$ for each intermediate step and creates a time vector for it. The time vector, $\mathbf{t}_K$, has as its elements

$$\mathbf{t}(k) = (t - t_0)^k \tag{13.3}$$

Having the time vector also makes enables the computations to be carried out in vectorial form for the benefit of MATLAB. The state variable can be computed from its normalized derivative vector and the time vector in the following manner

$$x_n = X_n^T \mathbf{t} \tag{13.4}$$

An integration was carried out using Taylor series of an equatorial elliptical orbit with 0.8 eccentricity. The difference between the points output by the default integration and the output using interpolation using a refinement factor of 10 can be seen in Fig. 13.1.

The theory in Chapter 12 only shows the necessary computations required for the TSI to work. The procedure to implement these equations is not yet given. The first step is to define a vector with the initial conditions for the state. This has to be done for the whole extended state, and

not only for the 6 variables of the Cartesian state or the 7 variables of the USM7 state. This is then fed to the desired dynamics function to integrate. The dynamics function then initializes the various vectors of normalized derivatives. This is obviously a required step for all programming languages except for MATLAB, which can do dynamic allocation and reallocation. However, it is carried out also in MATLAB because this reduces the computation time. The vectors that need to be initialized are X_n, U_n, $W_{n,i}$, and V_n. After initialization, the first elements of all the X_n vectors are the respective state values from the input vector. All the v_n variables are only based on the state variables and not their time derivatives. Thus, the next step is to compute the v_n values from x_n. These values become the first elements of their respective V_n vectors. The $w_{n,i}$ auxiliary variables are used in the process of computing u_n and may require a previous u_{n-j}. Thus, for each n^{th} variable, the $w_{n,i}$ are computed, followed by the u_n before moving on to the $(n+1)^{\text{th}}$ variable. These are then put as first elements of their respective $W_{n,i}$ and $U_{n,i}$ vectors. Following this is a loop that computes all the normalized derivatives in a recursive manner to fill up the vectors. The procedure is the same as for initialization, but the computations like multiplication and division are replaced by the recursive relations.

13.2 Simulations

The integrator has to be tested to see how the Taylor series integration methods using Cartesian coordinates and USM7 compare. Like all integration methods, the order of the integrator also plays a role for Taylor series integration. The order of the integrator is fixed for normal RK methods, however, the order can be varied for each Taylor series integration run as it simply a parameter that the integrator accepts. The step-size controller used for integration using the RK methods varies from the step-size controller from [Scott and Martini, 2008] that has been implemented here for the Taylor series integration. The step-size controller used thus far for RK methods controls the step-size based on the absolute position and velocity error. This was to ensure that numerical simulations using both Cartesian coordinates and the USM would use a similar control strategy. Also, it was found that checking for a tolerance on each individual state element of the USM resulted in an integrator with a performance lower than ours. The reason is that the hodographic velocity parameters are usually many orders of magnitude larger than the quaternion elements. This means that a small change in the value of a quaternion results in a relatively large change in position and velocity. Thus, it is very difficult to come up with a precise tolerance value that can result in efficient integration. Since the method for TSI from [Scott and Martini, 2008] checks the tolerance for each individual state element, this method has to be made compatible with USM7. The method chosen here uses one tolerance value for all the hodographic velocities, and one tolerance value for the quaternion elements. The tolerance for the quaternion elements is three order of magnitude smaller than that of the hodographic velocities.

For the first 3 cases, a comparison will be made only between the TSI using Cartesian Coordinates and USM7. This is because initial tests show that TSI is more than an order of magnitude faster than integration using the RK5(4) method. Thus, the tests using the RK5(4) method are shown in separate plots. For Case 4, however, a comparison will be made of the number of converged digits as a test. The results for the unperturbed orbits of Case 1 will not be presented as an unperturbed orbit is not something that would be simulated using integration methods. The unperturbed orbits were used simply as a way to test the functions that compute the recursive relations for the normalized derivatives. These tests were considered to be very basic and are therefore not included in the thesis study.

13.2.1 Case 2

This is the case where low-thrust propulsion is utilized. It is assumed that the thrust is constant, laying along the direction of the velocity. Also, the change of the mass is not simulated, thus the acceleration is effectively constant. This test case is the same as the unperturbed low-thrust case used for the RK integrators. The results of the simulations are shown for orders of integration of

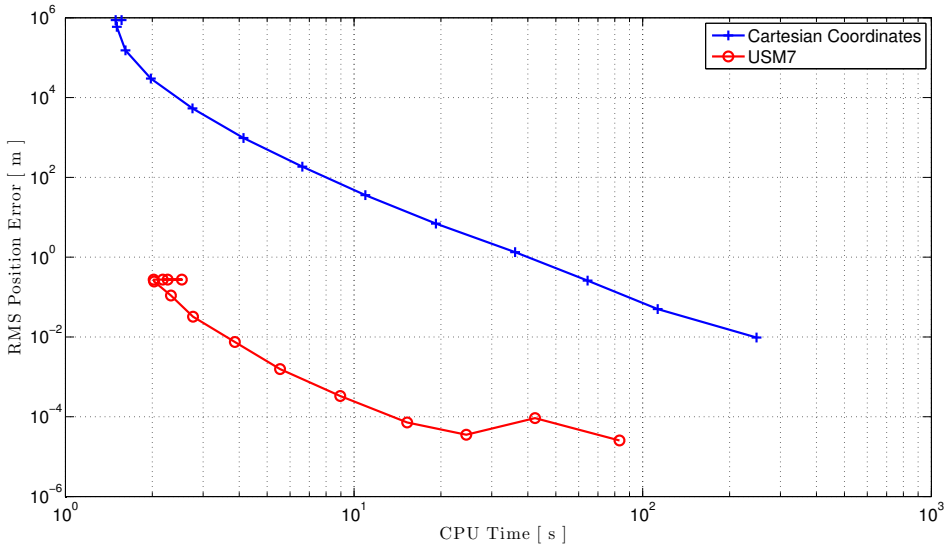


Figure 13.2: Integration using RK5(4) for Case 2

20 and 30. The ephemeris has been created using an order 40 TSI with Cartesian coordinates. Order 20 is chosen because it has been used on [Scott and Martini, 2008] and order 30 to see the effect of increasing order. The effect of using even lower or higher orders is something that can be investigated in the future. Figure 13.2 shows the results from using an RK5(4) integrator with Cartesian coordinates and the USM7. For the simulations carried out, the error RMS of the position error using the USM7 is always lower than that of the Cartesian coordinates. Also, the CPU time is lower for the USM7 for all the integrations. For both cases, the RMS position error decreases and the CPU time increases as the tolerances become more stringent. The RMS position error using the USM7 peaks at around 0.5 m, while that of the Cartesian coordinates is around 10^6 m. To have an RMS position error of approximately 0.5 m, the CPU time for the USM7 is between 2 - 3 s and for the Cartesian coordinates is around 65 s.The RMS position error of the USM7 converges to around 10^{-4} m after a CPU time of 15 s, while the RMS position error of the Cartesian coordinates is still around 10^{-2} m after a CPU time of 250 s and still does not converge.

Figures 13.3 and 13.4 show the result of using TSI with $K = 30$ and $K = 20$, respectively, with Cartesian coordinates and the USM7. Finally, Fig. 13.5 shows the results of the TSI for both $K = 20$ and $K = 30$ on the same plot. For Cartesian coordinates, the tolerance on the position coordinates was varied from 10^{-5} to 10^{-15} and the on the velocity coordinates from 10^{-6} to 10^{-16}. For the USM7, the tolerance on the hodographic velocities was varied from 10^{-5} to 10^{-15} and on the quaternion elements from 10^{-8} to 10^{-18}.

One striking aspect of the results is that very accurate results can be achieved with very little CPU time. When an RK5(4) integrator is used, the Cartesian coordinates do not achieve an RMS position error of 10^{-4} m after 250 s. The integration with USM7 takes approximately 15 s to achieve this accuracy. When TSI is used with $K = 20$, both Cartesian coordinates and USM7 achieve this accuracy after a CPU time of approximately only 0.6 s. For Cartesian coordinates, this is an improvement of more than 3 orders of magnitude in CPU time. For the USM7, this is an improvement of 2 orders of magnitude in CPU time. When TSI is carried out with $K = 30$, the time required for the Cartesian coordinates to achieve an accuracy of 10^{-4} m reduces to approximately 0.38 s, and for the USM7 to approximately 0.5 s. With $K = 20$, the USM7 results converge to an accuracy of 10^{-5} m after a CPU time of 0.65 s, while the Cartesian results seem like they only converge after approximately 1.6 s. When $K = 30$, the USM7 results converge to an accuracy of 10^{-5} m after a CPU time of approximately 0.55 s, while the Cartesian results seem

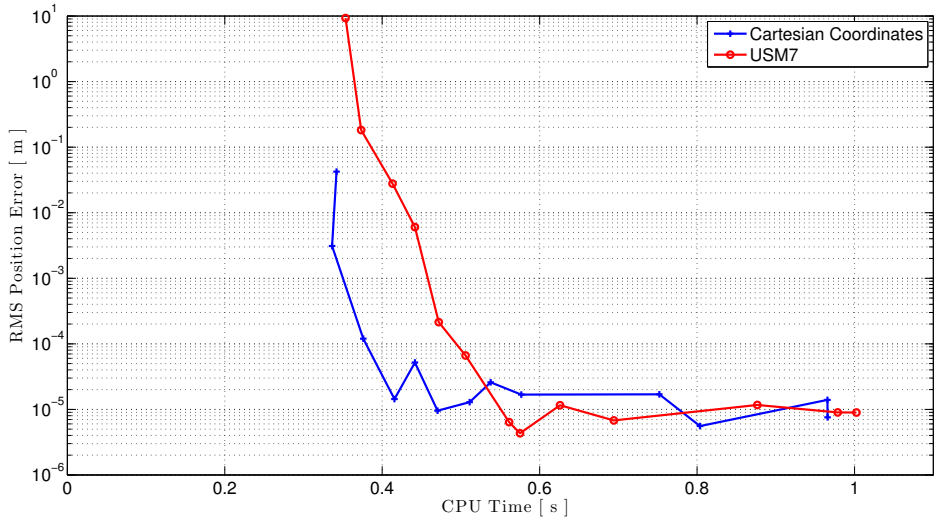


Figure 13.3: TSI of Case 2 with $K = 30$

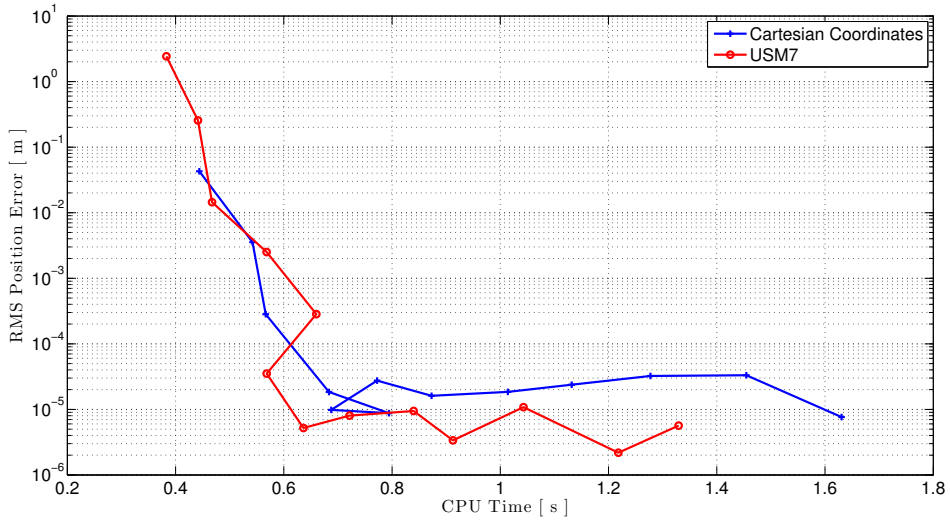


Figure 13.4: TSI of Case 2 with $K = 20$

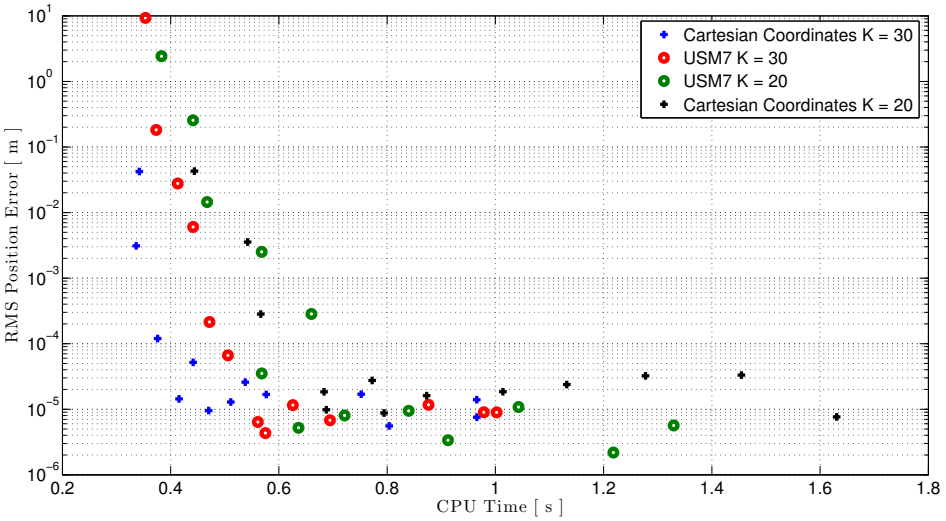


Figure 13.5: TSI of Case 2 with $K = 30$ and $K = 20$

like they only converge after approximately 0.75 s. Figure 13.5 shows that all four cases reach an accuracy of 10^{-4} after a CPU time between 0.37 s and 0.6 s. For both $K = 20$ and $K = 30$, there is some erratic behavior when the curves converge at RMS position errors of around 10^{-5} m. At this point, the solution is so accurate that the errors are caused due to numerical inaccuracies.

Increasing the K from 20 to 30 decreases the CPU time required to achieve this result for both USM7 and Cartesian coordinates. The behavior for both $K = 20$ and $K = 30$ is almost the same for the USM7. However, when K is increased from 20 to 30, the performance of Cartesian coordinates increases. The RMS position error vs. CPU time curve has the same shape and converges to the same value, but is moved to the left. It moves even to the left of the USM7 curve. This means that increasing the order really helps the integration using Cartesian coordinates. At such high orders of integration as $K = 30$, there is no real difference between the accuracy of the USM7 and Cartesian coordinates at equal time step-sizes. However, there are fewer equations for the TSI of Cartesian coordinates and therefore, the integration step takes less CPU time.

13.2.2 Case 3

This is the case where the magnitude of the thrust varies like a sine function. Even though the thrust is not constant, it still lies along the direction of the velocity. Also, the change of the mass is not simulated. The results of the simulations are shown for orders of integration of 20 and 30. The ephemeris has been created using an order 40 TSI with Cartesian coordinates. Figure 13.6 shows the results from using an RK5(4) integrator with Cartesian coordinates and the USM7. Figures 13.7 and 13.8 show the result of using TSI with $K = 30$ and $K = 20$, respectively, with Cartesian coordinates and the USM7. Finally, Fig. 13.9 shows the results of the TSI for both $K = 20$ and $K = 30$ on the same plot. For Cartesian coordinates, the tolerance on the position coordinates was varied from 10^{-5} to 10^{-15} and the on the velocity coordinates from 10^{-6} to 10^{-16}. For the USM7, the tolerance on the hodographic velocities was varied from 10^{-5} to 10^{-15} and on the quaternion elements from 10^{-8} to 10^{-18}.

For the simulations carried out, the error RMS of the position error using the USM7 is always lower than that of the Cartesian coordinates. Also, the CPU time is lower for the USM7 for all the integrations. For both cases, the RMS position error decreases and the CPU time increases as the tolerances become more stringent. The RMS position error using the USM7 peaks at around 0.7 m, while that of the Cartesian coordinates is around 10^6 m. To have an RMS position error of

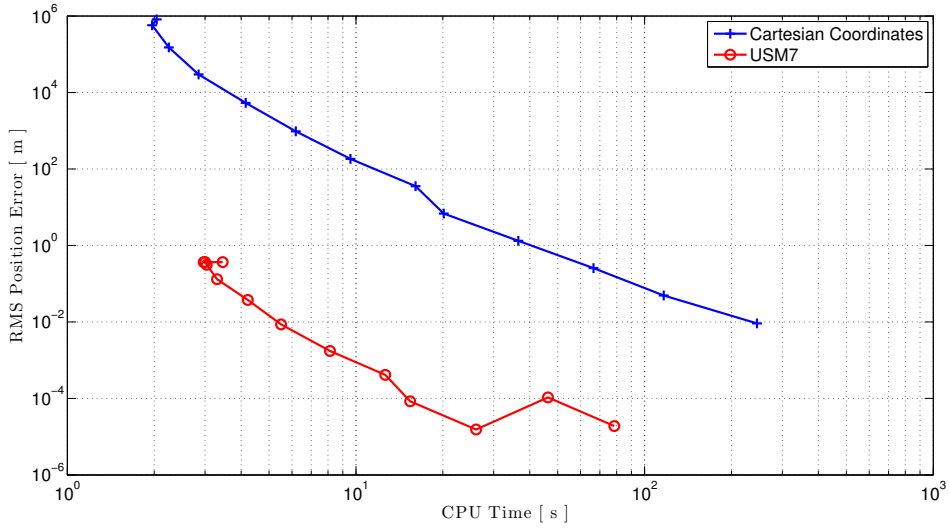


Figure 13.6: Integration using RK5(4) for Case 3

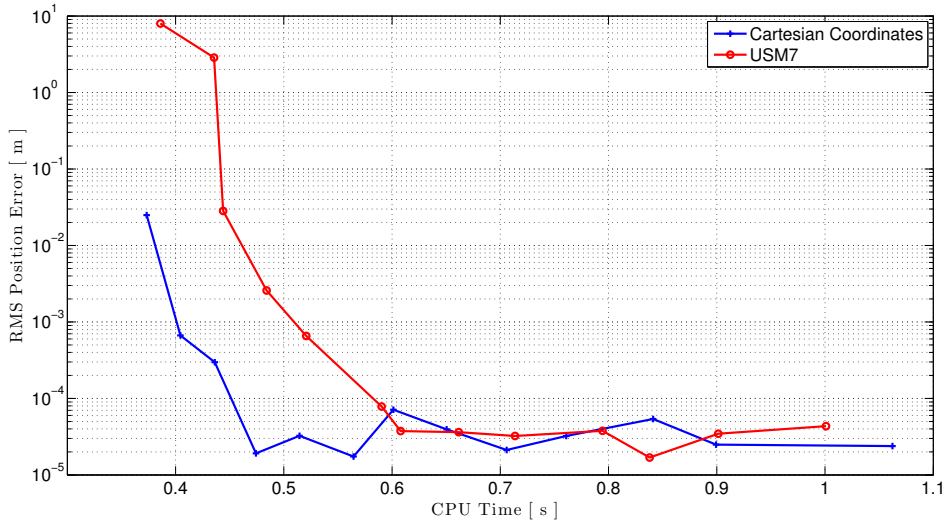


Figure 13.7: TSI of Case 3 with $K = 30$

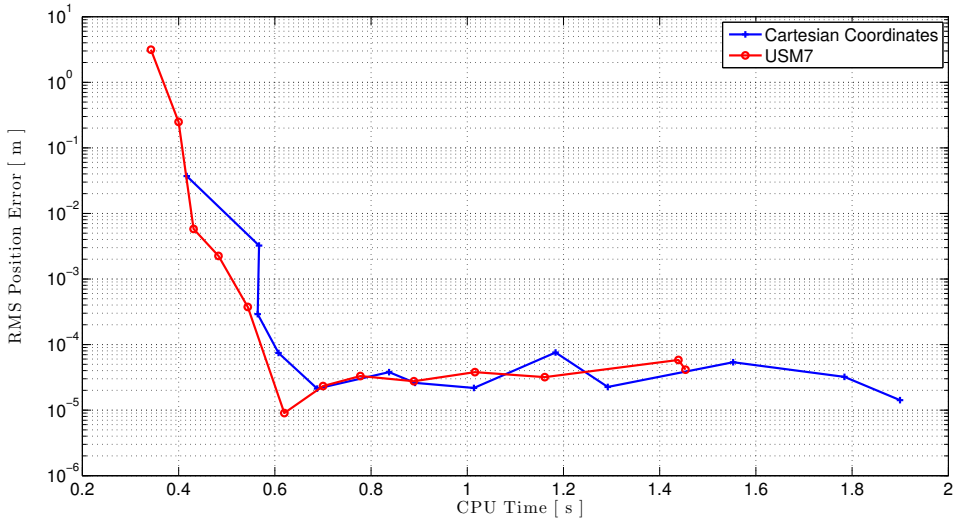


Figure 13.8: TSI of Case 3 with $K = 20$

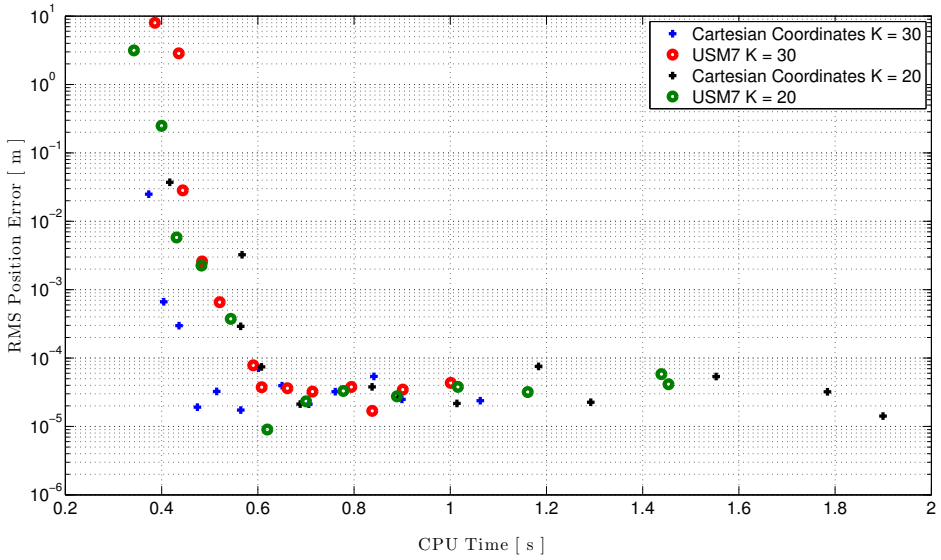


Figure 13.9: TSI of Case 3 with $K = 30$ and $K = 20$

approximately 0.7 m, the CPU time for the USM7 is around 3 s and for the Cartesian coordinates is around 60 s.The RMS position error of the USM7 converges to around 10^{-4} m after a CPU time of 16 s, while the RMS position error of the Cartesian coordinates is still around 10^{-2} m after a CPU time of 250 s and still does not converge.

When TSI is used with $K = 20$, Cartesian coordinates achieve this accuracy after a CPU time of approximately only 0.6 s and the USM7 achieves this accuracy after approximately 0.58 s. For Cartesian coordinates, this is an improvement of more than 3 orders of magnitude in CPU time. For the USM7, this is an improvement of 2 orders of magnitude in CPU time. When TSI is carried out with $K = 30$, the time required for the Cartesian coordinates to achieve an accuracy of 10^{-4} m reduces to approximately 0.45 s, and for the USM7 it remains the same. Figure 13.9 shows that all four cases reach an accuracy of 10^{-4} after a CPU time between 0.45 s and 0.6 s. Increasing the K from 20 to 30 decreases the CPU time required to achieve this result for Cartesian coordinates, but has almost no effect on the USM7. This is again like for Case 2, where the curve of Cartesian coordinates shift to the left. The solution of the USM7 has converged with respect to the order of integration K. However, since the Cartesian TSI equations have fewer variables, it is possible to decrease the CPU time by taking larger time-steps.

13.2.3 Case 4

Low-thrust propulsion is still used in this case, but the thrust is not along the velocity vector anymore. Also, the mass is assumed to not be constant anymore. For this case, the goal of the simulations is not to check the difference between the integrations using Cartesian TLI and USM7 TLI. Instead, this integration test is similar to the one carried out by [Scott and Martini, 2008] who state that *it should be noted that convergence itself does not necessarily imply accuracy. However, it does indicate that a necessary condition for accuracy is satisfied.* The simulation will be carried out using TLI and RK8(7) with USM7. A run is carried out first using the RK8(7) integrator and a very stringent tolerance. The tolerance for the position $\varepsilon_{pos} = 10^{-11}$ and the tolerance for the velocity $\varepsilon_{vel} = 10^{-12}$. The trajectory is again simulated using TLI with $K = 30$ and also a very stringent tolerance. A tolerance of 10^{-15} was used on the hodographic velocities, and a tolerance of 10^{-18} was used on the quaternion elements.These two trajectories are used as the baseline. The end state of both the baseline trajectories is stored and used to check the quality of the other simulations. The simulation is then repeated using TLI with $K = 10$, $K = 20$, and $K = 30$ of varying tolerances. The tolerance on the hodographic velocities is varied from 10^{-4} to 10^{-14} and the tolerance on the quaternion elements is varied from 10^{-7} to 10^{-17}. The end state is stored in USM7 form and converted and stored in Cartesian coordinates. The end states are compared with the end states of the baseline trajectories and the number of converged digits is the check. Instead of displaying the number of converged digits for each USM7 and Cartesian element separately, the sum of the converged digits of all the state elements is presented.

The trajectory of the spacecraft due to the varying thrust can be seen in Fig. 13.10. There is a varying thrust component both in and out of the orbital plane. The final state of the reference trajectory created by TSI and RK8(7), expressed in USM7 elements, can be seen in Table 13.1. It should be noted that the RK8(7) integration took approximately 3000 seconds of CPU time, while the TSI took 2 seconds of CPU time. Using the simulation duration and the assumption that the magnitude of the thrust is constant, the true final mass can be computed. The difference between the true final mass and the final mass using TSI is 0, which the difference between for the RK8(7) $\approx 1.92 \times 10^{-8}$. Also, the absolute value of the difference between the quaternion norm and one for the TSI is approximately 1×10^{-15}, and for the RK8(7) is approximately 7×10^{-14}. This in itself shows how the numerical inaccuracies creep in more, but yet very small, for the RK methods than for TSI.

The end values converted to Cartesian coordinates can be seen in Table 13.2.

The sum of the number of converged digits at the end of the simulation is plotted for each simulation with a different tolerance. The final state is checked with the baseline final state from both the TSI and the RK8(7). All the final states are also converted to Cartesian coordinates to

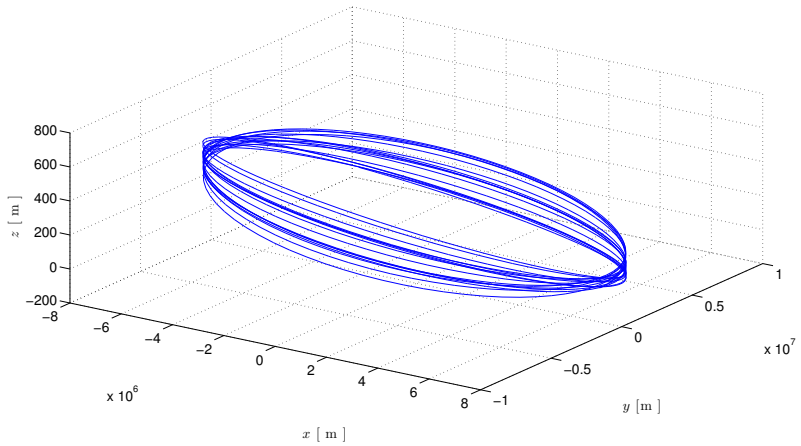


Figure 13.10: Trajectory of the spacecraft due to the varying thrust (Note that the z scale differs from the x and y scales)

Table 13.1: The end values of the USM7 elements using RK8(7) and TSI

Parameter	Value using TSI	Values using RK8(7)	Converged digits
C	$7.424822031065767 \times 10^{3}$	$7.424822031065606 \times 10^{3}$	13
R_{f1}	-0.501720582427736	-0.501720582433396	10
R_{f2}	74.206115285428879	74.206115285432148	12
ϵ_{O1}	$1.753351673616902 \times 10^{-5}$	$1.753351673667076 \times 10^{-5}$	11
ϵ_{O2}	$1.935462505092579 \times 10^{-5}$	$1.935462505112064 \times 10^{-5}$	10
ϵ_{O3}	0.921910940000300	0.921910939999448	7
η_O	0.387401881804590	0.387401881806788	11

Table 13.2: The end values of the Cartesian coordinates using RK8(7) and TSI

Parameter	Value using TSI	Values using RK8(7)	Converged digits
x	$-5.095554341339908 \times 10^{6}$	$-5.095554341316119 \times 10^{6}$	11
y	$5.200841730884816 \times 10^{6}$	$5.200841730909565 \times 10^{6}$	10
z	$1.261995483562007 \times 10^{2}$	$1.261995483610015 \times 10^{2}$	10
v_x	$-5.304052582593376 \times 10^{3}$	$-5.304052582618455 \times 10^{3}$	10
v_y	$-5.121978091786204 \times 10^{3}$	$-5.121978091761771 \times 10^{3}$	11
v_z	0.364215918098659	0.364215918104544	9

check the convergence as the output of an integration is usually required in terms of Cartesian coordinates. The convergence of the TSI with $K = 30$, $K = 20$, and $K = 10$, can be seen in Figs 13.11(a), 13.11(b), and 13.11(c), respectively on page 178.

It can be deduced that the TSI baseline is more accurate than the RK8(7) baseline integration. This is because, the TSI using all three orders has the number of digits that converge increase as the tolerances become more stringent. All three orders converge at roughly the same number of digits when compared to the RK8(7) baseline values. However, if the tolerance is made more stringent, they further converge with the TSI baseline values.

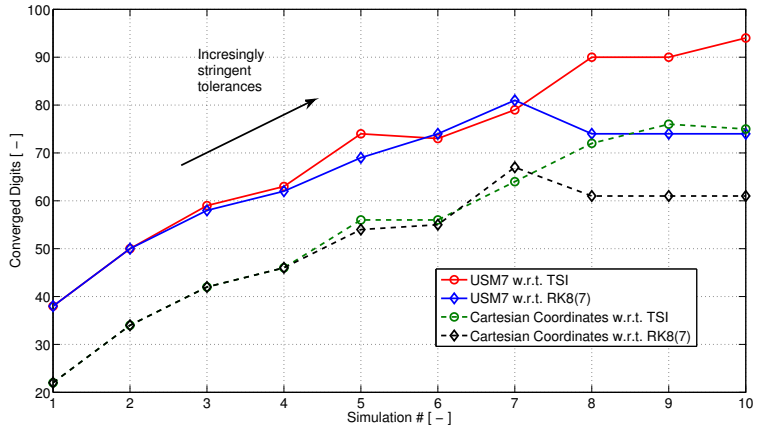
When the integration results of different orders are compared to each other, some further conclusions can be drawn. It should first be noted that each simulation number corresponds to the same tolerance value in all three integration cases. The tolerance is the least stringent for Simulation 1, and most stringent for Simulation 10. For the least stringent tolerance, the highest number of converged digits is actually for the simulation with $K = 10$ with 49 converged digits, followed by the simulation with $K = 20$ with 39 converged digits, and the least amount of converged digits is 38 for $K = 30$. By Simulation 5, the case with $K = 10$ is still the most converged with 79 converged digits, while the cases with $K = 20$ and $K = 30$ have only 74 converged digits.

All the cases converge to roughly 89 to 94 converged digits. This boundary is first passed at Simulation 6 by the case with $K = 20$. It is then passed by the case with $K = 10$ at Simulation 7, and finally at Simulation 8 by the case with $K = 30$. It can be concluded that for lower tolerances, the lower order TSI solutions have more converged digits. However, for more stringent tolerance, $K = 20$ converges more quickly than $K = 10$. The case with $K = 30$ keeps converging at the slowest rate, but steadily. It is not certain about what happens once the solutions converge. The baseline was created with $K = 30$, so as the tolerance is made more stringent, this case will continue to converge till all the digits are converged ($7 \times 15 = 105$). Once the solution with $K = 20$ and $K = 10$ converge, it might be the case that their solution becomes more accurate than the baseline because of the smaller time step-sizes. However, larger orders might also have more inaccuracies because of the increased number of computations in the recursive relations for the normalized derivatives. These are aspects that could be investigated further in the future.

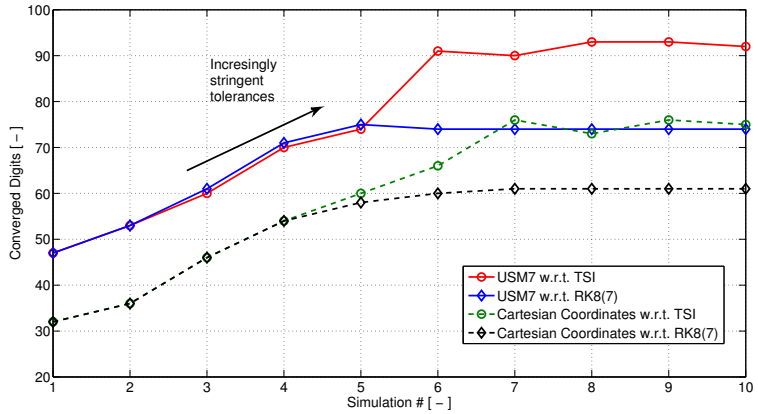
13.3 Conclusions and Recommendations

As can be seen from the results in this chapter, 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. There is a price attached to this increase in simulation speed. This price is the complexity of implementation. First the engineer or researcher has to get familiar with the theory behind TSI. Afterwards, the various recursive relations have to be derived, programmed, and tested. The testing can be carried out by using small vectors of integers. This way, the answers can be manually computed and thus, the results of the programming can be checked.

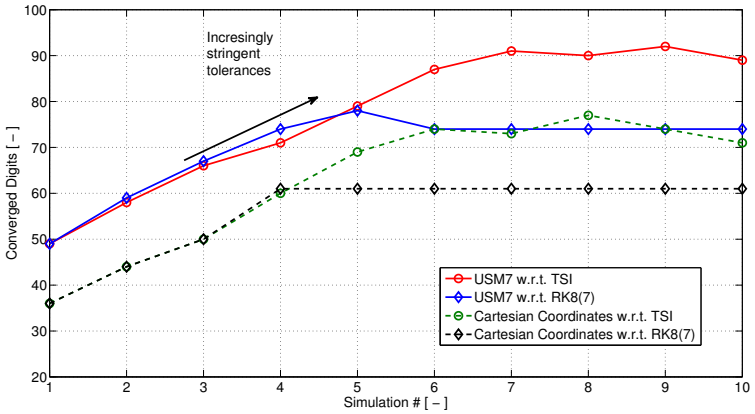
Following this, the equations of motion or any dynamic equation should be rearranged and split up so that it can be put in the correct format for the TSI. It is recommended to first start with simple problems and then slowly build up to more complex ones. For this thesis study, the author first started with the simple two body problem and Cartesian coordinates. The next case was the two body problem with USM7. The next case was the constant magnitude tangential low-thrust for Cartesian and then USM7 coordinates. The case after that was varying magnitude tangential low-thrust for Cartesian and USM7 coordinates. Due to time constraints, the next case of low-thrust with varying orientation was only carried out for USM7. The cases prior to this for the USM7 had either no perturbations, or only perturbation in the orbital plane, i.e. a_{e1} and a_{e2}. When there is also out of plane perturbation, the dynamics of the USM7 become even more complex. Thus, a generic set of equations for the TSI were made that could take a constant perturbing acceleration in any direction. Finally, the case with time varying thrust orientation and mass was derived and implemented. It is necessary to take small steps towards more complex cases and check every new case with an RK integration.



(a) Convergence of the digits using TSI with $K = 30$



(b) Convergence of the digits using TSI with $K = 20$



(c) Convergence of the digits using TSI with $K = 10$

Figure 13.11: Convergence of the digits using TSI with $K = 30$ (a), $K = 20$ (b), and $K = 10$ (c)

Since there are many variables and many relations between them present for the TSI, it is very easy to make a mistake. Once there is an error, it could be in one of 3 places. Either one of the recursive relations is not programmed properly, there is mistake in the derivation of the equations, or there is a mistake in the programming. If the recursive relations are extensively tested in the beginning, this can be ruled out. This testing is carried out by trying to integrate functions of which the analytical solution is already know. The function must be multivariate, with few variables. Most of the times, it is only possible to catch a mistake in the other two categories when the integration is checked with an RK integration. Once it is known that a mistake is present, it is only possible to identify it by perusing the equations and the program script. Unfortunately, they can take hours to days to spot.

Ultimately, the TSI is a very challenging method to implement. Something that can be implemented in a few minutes or hours using RK methods can take hours or even days to implement for TSI. However, once everything is set up properly, it is incredibly efficient in integration. It can then be used to generate accurate ephemeris very quickly, or also to optimize trajectories. In the opinion of the author, the payoff is proportional to the extra work. One important thing to note for future users of TSI is that if a perturbing acceleration is not constant, its actual function with respect to other state variables must be included in the dynamics function. This way, the TSI can approximate the perturbation as it approximates the other state variables. It is not correct to compute the perturbation externally and simply give the value. This makes the TSI assume that the value remains constant over the integration period and thus, the integrated model would not be modeling reality. This is not only true for perturbing accelerations, but also for parameters like solar sailing aiming angle that might have to be computed iteratively. This can be done, however, if it can be assumed that this angle or acceleration can be assumed to be constant even over large durations of time because the TSI usually take very large time steps compared to conventional RK integration methods.

For the cases tested here, the performance of the USM7 and Cartesian coordinates are very similar when TSI is used. For RK methods, the USM7 vastly outperforms Cartesian coordinates for low eccentricity orbits and orbits using low-thrust propulsion. However, for large orders of TSI, such as $K = 30$, the USM7 loses its advantage over Cartesian coordinates. For both the low-thrust cases tested here the USM7 performs marginally better than Cartesian coordinates for TSI with $K = 20$. When $K = 30$ is used, both USM7 and Cartesian coordinates converge to the same accuracy. At such high order of integration, the difference in accuracy between a Cartesian and USM7 step is marginal. However, the simulation time for Cartesian coordinates is smaller due to the smaller number of state and auxiliary variables that are required to propagate the state. Thus, the simulations using Cartesian coordinates converge for a smaller CPU time. This means that for lower tolerances, it takes approximately 0.1 s less of CPU time for Cartesian coordinates to have the same accuracy before convergence. This translates to a 20% faster simulation for Cartesian coordinates. Thus, if $K = 20$ is to be used, the USM7 is recommend and Cartesian coordinates are recommended if $K = 30$ is to be used. It should be noted, however, that due to the numerical inaccuracy involved in computing the recursive relations of such high order in different programming languages, it might be a better option to remain with $K = 20$ and the USM7. For example, [Scott and Martini, 2008] also used $K = 20$.

Even though much work has been carried out on TSI in this thesis study, there are many more things to be done in the future. The step-size controller proposed in [Scott and Martini, 2008] was implemented here, but the effect of using the traditional step-size controller should also be investigated. These step-size controlling methods only vary the time step-size, but TSI also possesses another parameter, the integration order K, that can be varied. Using a higher order requires a smaller time step-size, but can have a higher numerical error due to the higher number of recursive calculations. Using a lower order requires a smaller time step-size and so a higher numerical error due to additional function evaluations. Thus, it might be possible to find a new step-size controlling method that balance these two errors to find the optimal order and time step-size for each integration step.

Only unperturbed orbits and perturbations in the form of thrust have been used. Other perturbations should also be implemented such as J_2, atmospheric drag, third body perturbations

etc. This should not be too difficult in the future as a detailed implementation method for TSI has been shown here with many examples. To add more perturbations simply involves adding more variables to the already existing framework. Also, TSI should be implemented in an optimizer for low-thrust trajectories to reap them benefits of their fast integration times. It is stated in [Scott and Martini, 2008] that they added 3$^{\text{rd}}$ body perturbations by simulating the trajectory of each additional perturbing body along with the spacecraft. It is possible to get accurate JPL ephemeris of other bodies in terms of the Chebyshev polynomials. Thus, an investigation should be made to check if the Chebyshev polynomials could be converted to Taylor series that could be used within the TSI. The USM6 should also be implemented in TSI form.

Finally, a check should be made to see if the present implementations of the various cases for TSI are as efficient as possible. There are many variables required for the computation that could be either added as additional x_n variables, or as v_n variables. A comparison should always be made to see if it requires fewer computations in one case or the other. One of the variables that was added as an x_n variable for many of the TSI cases is the product of mass and velocity, mv. The time derivative of this variable, u_n, is then $\dot{m}v + m\dot{v}$. This requires two recursive multiplication function calls. Since m and v are already x_n variable, it would have been more efficient to add mv as v_n variable because this would have required only one recursive multiplication function call. Another improvement that would increase programming efficiency and not the CPU efficiency is the naming of the v_n variables. For the USM7 case with all generic perturbations, there are certain v_n variables that are always necessary like a_{e1}, a_{e2} etc. Depending on the scenario, they are computed from different v_n variables. In this thesis study, all the v_n variables are numbered serially. Thus, when the variation of the thrust orientation had to be computed, it had to be computed before the accelerations. Thus, the variable name for the acceleration had to be changed and thus all other computations that depend on this variable also had to be checked and modified. If the accelerations had been been given uniques v_n names such as $v_{a,n}$, it would have been easier to add this variation in the simulation environment.

This concludes the work on TSI and all other investigations carried out in this thesis study. The following chapter will give an overview of the important conclusions made in this study and the related topics that can be investigated in the future.

Chapter 14

Conclusions and Future Work

The work carried out during this thesis study is only the tip of the iceberg. Many conclusions and recommendations are spread throughout the report, but the major ones will be summarized here. There are many aspects that can still be investigated further. One of the major ones can be found in [Mooij et al., 2010], which is a work in progress.

14.1 Conclusions

During this thesis study, the following tasks have been carried out:

- Provide a complete derivation of the traditional USM
- Propose a modification of the USM that uses Modified Rodrigues Parameters instead of a quaternion
- Compare the numerical integration performances of the USMs and Cartesian coordinates for orbital and re-entry trajectories
- Compare the navigation performances of the USMs and Cartesian coordinates using the four different filters
- Implement the following four nonlinear filters: Extended Kalman Filter, Unscented Kalman Filter, Divided Difference Filter 1 and 2
- Implement a Particle Swarm Optimizer than can automate the tuning of filters
- Successfully implement various scenarios for the USM7 and Cartesian coordinates using Taylor Series Integration

The main conclusion of this thesis study is that the USM is an elegant theory for expressing orbital trajectories, that unfortunately did not get the recognition that it deserved.

The USM is heavily based on characteristics of an orbit. The first three parameters give information about the hodograph for both the USM6 and the USM7. The remaining element describe the orientation of the orbital frame and the angular location of the spacecraft within it, with respect to an inertial reference frame. The last 4 elements of the USM7 are elements of a quaternion and the last 3 elements of the USM6 are elements of a Modified Rodrigues Parameters vector. The USM is not capable of handling linear trajectories and pure retrograde trajectories, i.e. $i = 180°$.

For numerical integration of orbital trajectories using the RK methods:

- The USM is better than Cartesian coordinates for orbits with low eccentricities and orbits under the influence of low thrust propulsion. Especially for orbits with low thrust propulsion, the USM is an order of magnitude faster than Cartesian coordinates.
- Cartesian coordinates are more suitable for highly eccentric orbits. The USM is based on the assumption of rotational motion and for highly eccentric orbits, i.e.,$e > 0.9$, much of the orbit is spent in an almost linear motion.
- Cartesian coordinates are more suitable for re-entry. This is because the motion of a vehicle is very *un-orbit-like*. Since, the USM specifically designed on the assumptions of and for orbital motion, it is logical that the USM does not perform as well as Cartesian coordinates.
- Between the USM7 and USM6, it was found that the USM6 had better performance for most of the cases. The only scenario where the USM7 outperforms USM6 is for trajectories with constant tangential low-thrust and no other perturbations.

For more information, the reader is referred to Chapters 6 to 9.

For navigation, four nonlinear filers were implemented for the two USMs and for Cartesian coordinates. The two first-order filters were the EKF and the DD1, and the two second-order filters were the UKF and the DD2. To obviate the tedious tasks of manually tuning the various filters for the various model for the various testing scenarios, a PSO was built that could automatically tune the filters. Even though, this is computationally intensive, it can save many engineer man-hours. The main conclusions from the navigation part of the thesis study were:

- The performance of the DD1 was abysmal compared to the other filters for the USM7 and the USM6. It is suspected that this is because the assumption of unbiased state estimate was violated.
- The UKF was implemented without augmenting this state and this turned out to be the best filter. Augmenting the state means that the spacecraft state is extended by adding the parameters of the measurement and process noise. Thus, the filter tries to estimate these along with the original state elements.
- For three of the four cases investigated, the best state estimate occurred when the UKF was used with the USM7. For the other case, the UKF with Cartesian coordinates gave the best result.
- When the system is fully modeled, which is an unrealistic case, the Cartesian coordinates give the best navigation performance. For all the other cases, however, it is the USM7.
- The USM6 performs better than the USM7 when the system is fully modeled, but worse for the other cases.
- The reasons that the DD2 did not perform as well as the Cartesian coordinates are suspected to be the violation of the unbiased state estimation assumption and the augmentation of the various covariance matrices.
- All the above conclusion have been drawn using the MAE of the position estimate as the objective to be minimized. A better objective would have been to the converged MAE of the position estimate.

For a more detailed conclusion, the reader is referred to Section 11.8 on page 139.

The final part of the thesis study was to investigate Taylor Series Integration. This turned out to be quite challenging since there is not much literature available on how to implement this method of integration. However, this problem was slowly tackled and the TSI equations were derived for both the USM7 and Cartesian coordinates for the simple two body problem and some low-thrust propulsion scenarios. The main conclusions regarding TSI are:

- Taylor series was found to be much harder than the RK methods to implement. However, the CPU time for the simulations is at least an order of magnitude lower than for RK methods. Thus, it was found to be worth the extra effort.
- Only the USM7 and Cartesian coordinates were implemented, and not the USM6. The dynamics of the USM6 are more complex than the dynamics of the USM7, which are more complex than the dynamics of Cartesian coordinates. Thus it would have taken longer to implement the equations of motion for the USM6.
- Between the USM7 and Cartesian coordinates, it was found that integration using USM7 is still faster up till an integration order of 20. When the integration order is increased to 30, the accuracy of an integration step using Cartesian coordinates and the USM7 is similar. However, for the scenarios tested in this thesis study, many more variables are required during each integration step. Thus, the TSI using Cartesian coordinates requires less CPU time.

For a more detailed conclusion, the reader is referred to Section 13.3 on page 177.

14.2 Future work

One never notices what has been done; one can only see what remains to be done.

- Marie Curie

After investigating all the various topics present in this thesis study, many questions have been answered. However, there are also many questions that have been raised. Some of these aspects could be investigated in the future through other theses. Each individual topic that has been treated has its own set of specific subtopics that should be investigated further. For a detailed list, the reader is referred to Section 8.5 on page 91 for future tasks regarding integration using the RK methods. The reader is referred to Section 11.8 on page 139 for future tasks regarding navigation. Finally, the reader is referred to Section 13.3 on page 177 for future tasks regarding TSI. A few of the important tasks are repeated here.

For the integration of the USMs using the RK methods:

- If it is decided to fully implement the USM6 or the USM7, the perturbation models specific to these models should be derived and used.
- The step-size controlling should be investigated and a method that obviates the need to switch to Cartesian coordinates should be found.
- Some application oriented work, such as finding an optimal trajectory to a planet or an asteroid should be carried out. This is already being done now in the form of a solar sailing mission for [Mooij et al., 2010].

There are many areas that can be investigated further regarding navigation.

- The measurement techniques in velocity space from [Altman, 1975] could be implemented.
- Navigation in other scenarios should be carried out e.g. deep space orbits, low-thrust orbit raising, orbit around other celestial bodies etc.
- Different filters than the ones chosen in this study could be implemented. Also, adaptive filtering techniques could be used.
- Different optimization techniques could be used to compare the result of the tuning process. Especially, it might be important to use a multi-objective optimizer.

- The filter tuning problem was simplified to use as few variables in the search space for the PSO, as possible. This decreases the number of filter tuning parameters. However, the optimization should be repeated so that many more filter tuning parameters could be optimized. This would give even more adaptability to the filters.
- Standard values from literature were used for the PSO tuning parameters. These should be tuned to allow the optimizer to work more efficiently.
- The objective of the optimizer was the MAE of the position estimate from the time of first measurement. It was already found that the MAE of the converged position estimate is a better objective to minimize. Thus, various objectives should be tried for the PSO to be able to pick the most desirable one.
- The effect of using different measurement types and measurement frequencies should be investigated.

For the TSI part:

- The next step would be to implement the TSI inside an optimizer to find out if such a large CPU time gain is still present for a complete mission design problem.
- This work has been carried out in MATLAB, so it should be converted to C++ or Fortran.
- The TSI equations for the USM6 should be derived and implemented to see what its performance is.
- In this study, only the thrust is included as a perturbation. In this regard, the next step would be to use the same methodology to find the equations for other perturbations.
- Investigation should also be carried out regarding the best type of step-size controller to use. Also, some method of step-size control that can predict step-size and also the order of integration could be very useful.
- Apply the TSI to an atmospheric re-entry problem.

References

Allgoewer, F., Boehm, C., Olaru, S., Findeisen, R., Alamo, T., Lazar, M., Goodwin, G. C., Camacho, E. F., Limon, D., and Heemeis, W. P. M. H. (2009). *Nonlinear Model Predictive Control*, volume 384/2009 of *Lecture Notes in Control and Information Sciences*. Springer Berlin / Heidelberg.

Altman, S. P. (1952). Equations of motion of the f-80 aileron boost. Memorandum Report WADC-52-43, U. S. Air Force Wright Air Development Center.

Altman, S. P. (1965). *Orbital Hodograph Analysis*. AAS science and technology series 3. North Hollywood Western Periodicals.

Altman, S. P. (1967a). Acceleration hodograph analysis techniques for powered orbital trajectories. Nasa-cr-61616, NASA.

Altman, S. P. (1967b). Hodographic treatment of the restricted three body problem. Technical Report NASA-CR-61615, NASA.

Altman, S. P. (1972). A unified state model of orbital trajectory and attitude dynamics. *Celestial Mechanics*, 6:425–446.

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

Altman, S. P. and Pistiner, J. S. (1968). A new generating principle for two-fixed-center orbits - conic solutions. Technical Report IAF PAPER AD-6, NASA.

Chodas, P. (1981). Application of the extended kalman filter to several formulations of orbit determination. UTIAS Technical Note 224, University of Toronto Institute for Aerospace Studies.

Crassidis, J. L. and Junkins, J. L. (2004). *Optimal Estimation of Dynamic Systems*. Chapman and Hall.

Curtis, H. (2005). *Orbital Mechanics for Engineering Students*. Elsevier.

Eades, J. B. (1968). Orbit information derived from its hodograph. Technical Report TM X-63301, NASA.

Grallert, H. (1988). Executive summary on re-entry guidance & control study. Technical report, MBB.

Grewal, M. S., Weill, L. R., and Andrews, A. P. (2001). *Global Positioning Systems, Inertial Navigation, and Integration*. John Wiley & Sons, Inc.

Howison, M. (2007). Stereographic projection.

Hughes, P. C. (1986). *Spacecraft Attitude Dynamics*. John Wiley & Sons, Inc.

Hughes, S. (2009). General mission analysis tool (gmat). mathematical specifications. draft. Technical report, NASA Goddard Space Flight Center.

Hujsak, R. S., Woodburn, J. W., and Seago, J. H. (2007). The orbit determination tool kit (odtk) - version 5. White Paper AAS 07-125, Analytical Graphics, Inc.

Jones, K. O. (2005). Comparison of genetic algorithm and particle swarm optimisation. In *International Conference on Computer Systems and Technologies*.

Jorba, A. and Zou, M. (2004). *A software package for the numerical integration of A Software Package for the Numerical Integration of ODEs by Means of High-Order Taylor Methods*. Universitat de Barcelona and The University of Texas at Austin.

Julier, S. J. (2003). Spherical simplex unscented transformation. In *Proceedings of the American Control Conference*.

Julier, S. J. and Uhlmann, J. (1997). A new extension of the kalman filter to nonlinear systems. In *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls, Orlando, F.*

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *In Proceedings of IEEE International Conference on Neural Networks*, pages 1942–1948. IEEE, IEEE Press.

Martins, J. R. R. A., Kroo, I. M., and Alonso, J. J. (2000). An automated method for sensitivity analysis using complex variables. In *AIAA paper 2000-0689, 38th Aerospace Sciences Meeting*, pages 2000–0689.

Michalewicz, Z. (1996). *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, thrid edition.

Montenbruck, O. and Gill, E. (2005). *Satellite Orbits Models Methods Applications*. Springer.

Mooij, E. (1997). The motion of a vehicle in a planetary atmosphere. Technical report, Delft University of Technology.

Mooij, E. (1998). Aerospace-plane flight dynamics analysis of guidance and control concepts.

Mooij, E. (2010). personal communication.

Mooij, E. and Chu, Q. P. (2002). Tightly-coupled imu/gps re-entry navigation system. In *AIAA Guidance, Navigation, and Control Conference*.

Mooij, E., Noomen, R., and Candy, S. (2006). Evolutionary optimization for a solar sailing polar mission. In *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*.

Mooij, E., Vittaldev, V., and Naeije, M. (2010). Application of the unified state model to optimization of a solar sailing mission (tentative title). In *AIAA/AAS Astrodynamics Specialist Conference*.

Nøgaard, M., Poulsen, N. K., and Ravn, O. (2004). Advances in derivatice-free state estimation for nonlinear systems. Technical Report IMM-REP-1998-15, Technical University of Denmark.

Petropoulos, A. E. and Longuski, J. M. (2004). Shape-based algorithm for automatd design of low-thrust, gravity-assist trajectories. *Journal of Spacecraft and Rockets*, 41(5):787–790.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1994). *Numerical Recipes in C The Art of Scientific Computing*. Cambridge University Press, second edition.

Raol, J. R. and Sinha, N. K. (1985). On the orbit determination problem. In *IEEE Transactions on Aerospace and Electronic Systems*, volume AES-21, pages 274–291. IEEE.

Rose, D. (1998). International space station familiarization. Technical Report TD 9702, NASA.

Schaub, H. and Junkins, J. L. (2002). *Analytical Mechanics of Aerospace Systems*. AIAA Education Series. AIAA.

Schlyter, P. (1979). How to compute planetary positions.

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*.

Shi, Y. and Eberhart, R. (1998). A modified particle swarm optimizer. In *In Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 69 – 73. IEEE Press.

Shiraishi, F., Furuta, S., Ishimatsu, T., and Akhter, J. (2007). A simple and highly accurate numerical differentiation method for sensitivity analysis of large-scale metabolic reaction systems. *Mathematical Biosciences*, 208(2):590 – 606.

Stanley, W. S. (1978). Quaternion from rotation matrix. *AIAA Journal of Guidance and Control*, I(3):223–224.

Sun, F. (1965). Hodograph analysis of the free-flight trajectories between two arbitrary terminal points. Contractor Report CR-153, NASA.

Sun, F., Li, G., and Wang, J. (2009). Unscented kalman filter using augmented state in the presence of additive noise. *Computer-Aided Software Engineering, International Workshop on*, 0:379–382.

Tewari, A. (2007). *Atmospheric and Space Flight Dynamics*. Modelling and Simulation in Science, Engineering and Technology. Birkhäuser.

Török, J. S. (2000). *Analytical Mechanics with an Introduction to Dynamical Systems*. John Wiley & Sons, Inc., 1 edition.

Trefethen, L. N. and Bau III, D. (1997). *Numerical Linear Algebra*. Siam.

Wakker, K. F. (2007a). Astrodynamics i; ae4-874. pt. 1. Lecture note, Delft University of Technology.

Wakker, K. F. (2007b). Astrodynamics ii; ae4-874. pt. ii. Lecture note, Delft University of Technology.

Wall, B. J. and Conway, B. A. (2009). Shape-based approach to low-thrust trajectory design. *Journal of Guidance Control and Dynamics*, 32(1):95 – 101.

Wan, E. A. and Van der Merwe, R. (2000). The unscented kalman filter for nonlinear estimation. In *Proceedings of IEEE Symposium on Adaptive Systems for Signal Processing, Communications and Control*.

Welch, G. and Bishop, G. (2001). An introduction to the kalman filter. Technical Report SIGGRAPH 2001, University of North Carolina at Chapel Hill.

Wertz, J. R. and Larson, W. J., editors (2003). *Space Mission Analysis and Design*. Space Technology Series. Microcosm Press and Kluwer Academic Publishers.

Appendix A

Trigonometric Identities

A few important trigonometric identities will be shown here.

A.1 Angle Sums

If there are two angles, A and B, the sine of the sum of the angles can be expressed as

$$\sin(A+B) = \sin A \cos B + \cos A \sin B \tag{A.1}$$
$$\sin(A-B) = \sin A \cos B - \cos A \sin B \tag{A.2}$$

The cosine of the sum of the angles can be expressed as

$$\cos(A+B) = \cos A \cos B - \sin A \sin B \tag{A.3}$$
$$\cos(A-B) = \cos A \cos B + \sin A \sin B \tag{A.4}$$

A.2 Half-Angle

The equations presented here follow from the equations above.

$$\cos A = \cos^2 \frac{A}{2} - \sin^2 \frac{A}{2} = 2\cos^2 \frac{A}{2} - 1 = 1 - 2\sin^2 \frac{A}{2} \tag{A.5}$$

$$\sin A = 2\sin \frac{A}{2} \cos \frac{A}{2} \tag{A.6}$$

$$\tan A = \frac{2\tan \frac{A}{2}}{1 - \tan^2 \frac{A}{2}} \tag{A.7}$$

$$\cos^2 \frac{A}{2} = \frac{1}{2}(1 + \cos A) \tag{A.8}$$

$$\sin^2 \frac{A}{2} = \frac{1}{2}(1 - \cos A) \tag{A.9}$$

Appendix B

Space Environment and Perturbations

An orbiting body assumed to a be perfect point mass, moving in a pure central gravity field is a good first-order approximation of orbital motion. However, there are many perturbing forces that cause the motion of the satellite to differ from the pure Keplerian motion. Only Earth orbiting satellites are considered in this chapter.

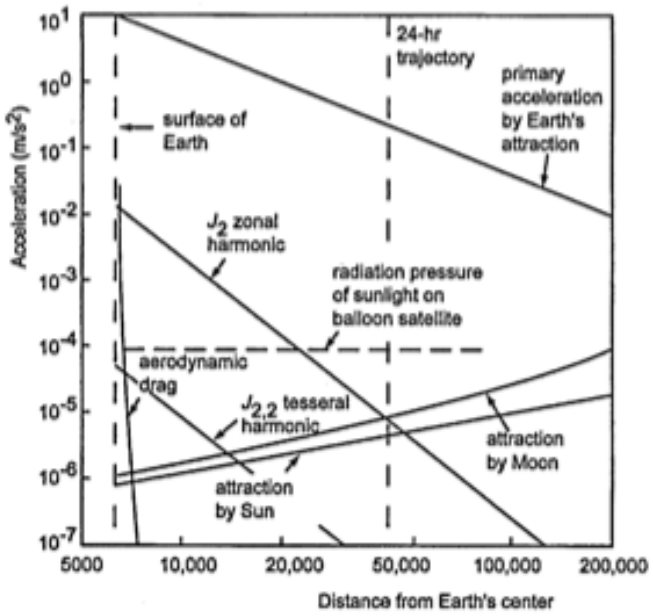


Figure B.1: The various perturbations and their magnitudes for Earth orbiting satellites [Wakker, 2007b]

As can be seen in Fig. B.1, the Earth's central gravity field is always at least three orders of magnitude larger than the perturbing accelerations. These perturbations are still very important to account for, since space applications these days require the knowledge of the satellite's state to a very high accuracy. It is possible to split the perturbations into four distinct categories that are used during the thesis study. They are

- Aerodynamic forces
- Third-body attraction

- Deviation of Earth from point mass
- Solar radiation pressure

There are other perturbing forces, such as the Earth's electromagnetic field, but the magnitudes of these forces are even smaller than the ones listed above.

B.1 Earth Gravity Model

The Earth is not a perfect sphere with radially symmetric mass distribution and thus, is not a point mass. A different, more complex gravity model for the Earth needs to be used for accurate simulation.

The gravity potential for the Earth, or any celestial body, can be written in terms of spherical harmonics according to [Montenbruck and Gill, 2005] as

$$U = \frac{\mu}{r}\sum_{n=0}^{\infty}\sum_{m=0}^{n}\left(\frac{R_E}{r}\right)^n P_{n,m}(\sin\phi_{gc})\left(C_{nm}\cos(m\lambda) + S_{n,m}\sin(m\lambda)\right) \tag{B.1}$$

In Eq. (B.1)

$P_{n,m}(\sin\phi_{gc})$ are the associate Legendre polynomials of degree n and order m

ϕ_{gc} is the geocentric latitude in ECEF

λ is the geocentric longitude in ECEF

$C_{n,m}$ and $S_{n,m}$ are geopotential coefficients

The acceleration can then be found by differentiating the potential.

$$\ddot{\mathbf{r}} = \nabla\frac{\mu}{r}\sum_{n=0}^{\infty}\sum_{m=0}^{n}\left(\frac{R_E}{r}\right)^n P_{n,m}(\sin\phi_{gc})\left(C_{nm}\cos(m\lambda) + S_{n,m}\sin(m\lambda)\right) \tag{B.2}$$

The geopotential coefficients are published and can be looked up in tables. To define the associated Legendre polynomials, it is necessary to first define Legendre polynomials of degree n. The Legendre polynomial of a function x can be written as

$$P_n(x) = \frac{1}{(-2)^n n!}\frac{d^n}{dx^n}(1-x^2)^n \tag{B.3}$$

The associated Legendre polynomial is a function of the Legendre polynomial and is written as

$$P_{n,m}(x) = (1-x^2)^{\frac{m}{2}}\frac{d^m P_n(x)}{dx^m} \tag{B.4}$$

All the coefficients can be grouped into the following 3 categories

$m = 0$ Zonal Coefficients

$m < n$ Tesseral Coefficients

$m = n$ Sectorial Coefficients

For zonal coefficients, $S_{n,m} = 0$, and it is possible to define Jeffrey's constants, J_n, as

$$J_n = -C_{n,0} \tag{B.5}$$

Jeffrey's constants describe the mass distribution, which is axially symmetric around the spin axis. The effect of this on a spherical body can be seen in Fig. B.2.

As can be seen in Fig. B.1, the major perturbing influences are due to $(n, m) = (2, 0)$, known as the J_2 effect, and $(n, m) = (2, 2)$, known as the $J_{2,2}$ effect. The associated Legendre polynomials and the geopotential coefficients for these two cases, according to [Montenbruck and Gill, 2005], can be seen in Table B.1.

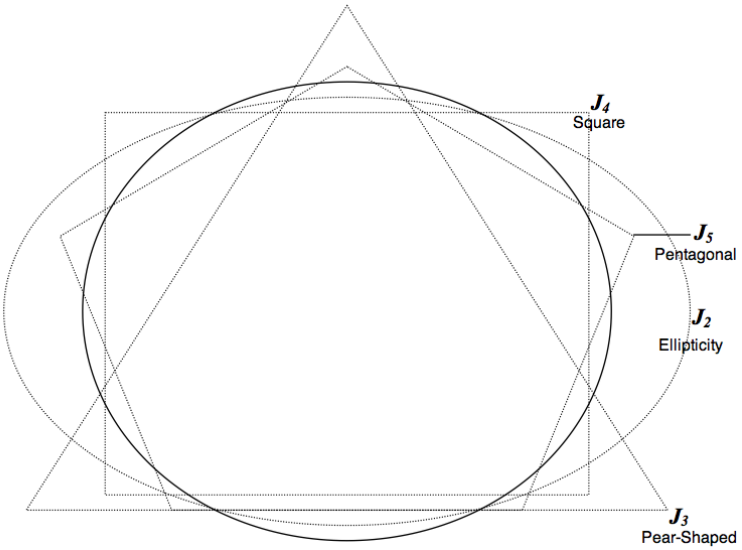


Figure B.2: The effect of zonal coefficients, modified from [Tewari, 2007]

B.2 Third Body Attraction

There are many celestial bodies in the solar system, and all of them affect each other and also any satellite that is orbiting the Earth. These perturbations are all very small, but the two main ones are due to the Moon and the Sun. The equation of motion for the orbiting body in a many body can be seen in Eq. (4.7). For the case of an Earth-orbiting satellite, the equation of motion can be simplified. It is assumed that an ECI frame is used and that the mass of the satellite is much smaller than the mass of the Earth. The equation of motion with the Sun as the perturbing force is

$$\ddot{\mathbf{r}}_{Sat} = -\frac{\mu_{Earth}}{r_{Sat}^3}\mathbf{r}_{Sat} + \mu_{Sun}\left(\frac{\mathbf{r}_{Sun}-\mathbf{r}_{Sat}}{r_{Sun\rightarrow Sat}^3} - \frac{\mathbf{r}_{Sun}}{r_{Sun}^3}\right) \tag{B.6}$$

The equation of motion with the Moon as the perturbing force is

$$\ddot{\mathbf{r}}_{Sat} = -\frac{\mu_{Earth}}{r_{Sat}^3}\mathbf{r}_{Sat} + \mu_{Moon}\left(\frac{\mathbf{r}_{Moon}-\mathbf{r}_{Sat}}{r_{Moon\rightarrow Sat}^3} - \frac{\mathbf{r}_{Moon}}{r_{Moon}^3}\right) \tag{B.7}$$

It can be seen that the equation of motion consists of the central gravity field and an additional acceleration caused by the perturbing third body. This perturbing acceleration is the difference between the acceleration felt by the orbiting body and the acceleration felt by the central body. The perturbations can just be added to each other if the effect of both is required. The equation of motion now becomes

$$\ddot{\mathbf{r}}_{Sat} = -\frac{\mu_{Earth}}{r_{Sat}^3}\mathbf{r}_{Sat} + \mu_{Sun}\left(\frac{\mathbf{r}_{Sun}-\mathbf{r}_{Sat}}{r_{Sun\rightarrow Sat}^3} - \frac{\mathbf{r}_{Sun}}{r_{Sun}^3}\right) + \mu_{Moon}\left(\frac{\mathbf{r}_{Moon}-\mathbf{r}_{Sat}}{r_{Moon\rightarrow Sat}^3} - \frac{\mathbf{r}_{Moon}}{r_{Moon}^3}\right) \tag{B.8}$$

Table B.1: Legendre polynomials and geopotential coefficients for largest perturbations [Montenbruck and Gill, 2005]

n	m	$P_{n,m}(\sin\phi_{gc})$	$C_{n,m}$	$S_{n,m}$
2	0	$\frac{1}{2}(3\sin^2\phi_{gc}-1)$	-1.08×10^{-3}	0
2	2	$3\cos^2\phi_{gc}$	1.57×10^{-6}	-9.03×10^{-7}

This theory can be expanded to include as many perturbing bodies as necessary. Also, the central body can be any celestial body and not just the Earth.

The notion of a sphere of influence is very important in the case of perturbing bodies. If a satellite is orbiting the Earth and the Sun in considered to a perturbing body, there should be a certain distance from the Earth where the Earth becomes the perturbing body. This distance is known as the sphere of influence, r_{SOI}, and can be calculated, according to [Wakker, 2007a] in the following manner

$$r_{SOI} = a_{Planet} \left(\frac{m_{Planet}}{m_{Sun}} \right)^{\frac{2}{5}} \tag{B.9}$$

The sphere of influence depends on the semi-major axis of the orbit of the planet, and the ratio of the masses of the planet and the Sun. This analogy can also be extended to a planet and its Moon. The sphere of influence is useful for perturbations and also interplanetary trajectories, where a method called patched-conic can be used. In this method, each planet has a sphere of influence where the satellite considers that planet as the primary attracting body. The Sun is considered to be the primary attracting body when the satellite is not inside any other sphere of influence.

To be able to compute the influence of the Sun and Moon on the orbit, the position of these two bodies needs to be computed. One method is to assume that the celestial bodies are in orbit and then use their mean orbital elements to compute the position. The implementation of this method along with the orbital elements of the celestial bodies can be found in [Schlyter, 1979]. The orbital elements are provided with respect to the J2000 frame. Even though the Earth orbits the Sun, it is assumed that the Sun is orbiting the Earth. This makes things simple for the case when the satellite is Earth-orbiting and only the Sun and Moon are considered to be perturbations. If other planets are considered, the position of the Sun has to be computed because this actually represents the position of the Earth. This is not a problem as usually the perturbation of Sun will be taken into account before the perturbation of any other planets. The orbital elements of the Sun and the Moon are presented in Table B.2. The orbital elements for the rest of the planets can be found in [Schlyter, 1979]. In Table B.2, d is the day from January 2000 0:00 UT. It can be computed in the following manner

$$d = 365 \times year - \frac{7}{4}\left(year + \frac{month + 9}{12} \right) + \frac{275}{9} month + day - 730530 + \frac{UT}{24} \tag{B.10}$$

In Eq. B.10, UT is the Universal Time expressed in hours. Once the day is calculated, the mean orbital elements from Table B.2 can be used to calculate the position of the Sun and the Moon. In case very accurate value for the position of the Moon, other terms that express the effect of other celestial bodies on the Moon have to be calculated. This position obtained from the mean orbital elements is expressed in the ecliptic, and not the equatorial plane. The ecliptic plane has an inclination, i_{EC}, with respect to the equatorial plane that is constantly changing. This

Table B.2: Mean orbital elements of the Sun and the Moon [Schlyter, 1979]

	Sun	Moon
Ω [deg]	0	$125.1228 - 0.0529538083 \times d$
i [deg]	0	5.1454
ω [deg]	$282.9404 + 4.70935 \times 10^{-5} \times d$	$318.0634 + 0.1643573223 \times d$
a [km]	149.60×10^6	383958.509
e [-]	$0.016709 - 1.151E - 9 \times d$	0.054900
M [deg]	$356.0470 + 0.9856002585 \times d$	$115.3654 + 13.0649929509 \times d$

inclination, in degrees, can be calculated in the following manner

$$i_{EC} = 23.4393 - 3.563 \times 10^{-7} \times d \tag{B.11}$$

The rotation matrix to convert from the ecliptic to the equatorial frames is

$$C_{EQ,EC} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos i_{EC} & -\sin i_{EC} \\ 0 & \sin_{EC} & \cos i_{EC} \end{bmatrix} \tag{B.12}$$

B.2.1 Atmospheric Forces

The perturbation due to atmospheric forces is very important, especially for LEO satellites. The atmosphere causes drag, which acts in a opposite sense to the velocity. The acceleration caused by drag, according to [Montenbruck and Gill, 2005], is

$$\ddot{\mathbf{r}} = -\frac{1}{2} C_D \frac{A}{m} \rho v_r^2 \hat{\mathbf{e}}_{vr} \tag{B.13}$$

In Eq. B.13

C_D is the coefficient of drag
A is the cross-sectional area coming into contact with the atmosphere
v_r is relative velocity of the spacecraft with respect to the atmosphere
$\hat{\mathbf{e}}_{vr}$ is the unit vector in the direction of the relative velocity
m is the mass of the satellite

The coefficient of drag varies from 2 to 3 according to [Wakker, 2007b], and between 2 to 2.3 according to [Montenbruck and Gill, 2005]. It is assumed that the atmosphere rotates along with the Earth and thus, the angular velocity of the atmosphere is the same as the Earth, $\omega_E = 0.7292 \times 10^{-4}$ rad/s. A satellite in an orbit has no out of plane velocity, but there will still be an out of plane drag component due to the atmospheric rotation.

Normally on Earth, the atmosphere is considered to be a continuum. In space the density is so low that free molecular flow regime is encountered. This means that the mean distance that atmospheric constituent molecules travel without collisions is greater than the dimensions of the satellite. Only when the satellite is flying very low, does it encounter other regimes such as the transition and continuum flow. Those cases are more applicable for re-entry and therefore, will not be dealt with in this chapter. Thus, only the atmospheric models for the upper atmosphere are presented in this section. Since the accuracy of the perturbation models is not the main focus of this thesis study, the mean atmospheric density values at various altitudes was used.

To calculate the acceleration due to drag from Eq. (B.13), it is required to calculate the density. The atmospheric scale height and the mean density for various altitudes is found in Table B.3. Between the altitudes found in the table, the atmosphere is assumed to be exponential. According to [Montenbruck and Gill, 2005] the density can then be approximated by

$$\rho = \rho_0 \mathrm{e}^{-h/H_0} \tag{B.14}$$

In Eq. (B.14)

h is the altitude

ρ_0 is the reference density corresponding to the largest height entry lower than h in Table B.3

H_0 is the atmospheric scale height corresponding to the largest height entry lower than h in Table B.3

B.3 Solar Radiation Pressure

The final perturbation that is treated in this chapter is the solar radiation pressure. The perturbation from solar radiation pressure is caused due to the exchange of momentum between the incoming photons from the Sun and the satellite. The solar radiation pressure depends on the solar radiation flux, Φ, and the speed of light, c. The pressure is directly proportional to the flux, which has a value of approximately 1367 Wm^{-2}. The solar radiation pressure can be written as

$$P = \frac{\Phi}{c} \tag{B.15}$$

The constant, $P_E \approx 4.56 \times 10^{-6}$ Nm^{-2}, is defined as the value of the solar radiation pressure at the Earth. The actual perturbing force that the satellite experiences depends on 3 main factors

- The orientation with respect to the Sun
- The area, A, in contact with the photons
- The reflectivity of the surface, expressed by the reflectivity coefficient ε

The reflectivity coefficient, ε, is a measure of the amount of the percentage of incoming photons reflected by the surface. If $\varepsilon = 0$, the solar radiation perturbation acceleration will be directed radially away from the Sun. If $\varepsilon = 1$, the solar radiation perturbation acceleration will be directed in an opposite direction to the surface normal. If $\hat{\mathbf{e}}_S$ is the unit vector pointing from the spacecraft to the Sun and $\mathbf{n}$ is the surface normal unit vector, the perturbing acceleration can be expressed, according to [Montenbruck and Gill, 2005], as

$$\ddot{\mathbf{r}} = -P\frac{C_R A}{m r_{Sun}^2}\hat{\mathbf{e}}_S \tag{B.16}$$

Table B.3: Atmospheric table with densities and scale heights for various altitudes [Wertz and Larson, 2003]

Altitude [km]	Atmospheric Scale Height [km]	Mean Atmospheric Density [$\mathrm{kg/m}^3$]
0	8.4	1.2
100	5.9	4.79×10^{-7}
150	25.5	1.81×10^{-9}
200	37.5	2.53×10^{-10}
250	44.8	6.24×10^{-11}
300	50.3	1.95×10^{-11}
350	54.8	6.98×10^{-12}
400	58.2	2.72×10^{-12}
450	61.3	1.13×10^{-12}
500	64.5	4.89×10^{-13}
550	68.7	2.21×10^{-13}
600	74.8	1.04×10^{-13}
650	84.4	5.15×10^{-14}
700	99.3	2.72×10^{-14}
750	121	1.55×10^{-14}
800	151	9.63×10^{-15}
850	188	6.47×10^{-15}
900	226	4.66×10^{-15}
950	263	3.54×10^{-15}
1,000	296	2.79×10^{-15}
1,250	408	1.11×10^{-15}
1,500	516	5.21×10^{-16}

In Eq. B.16, C_R is the solar radiation pressure coefficient, and is defined as

$$C_R = 1 + \varepsilon \tag{B.17}$$

Appendix C

Numerical Methods

There are many computations carried out in during the thesis work. This chapter will also include any numerical methods that are necessary in implementing the theory. They are: Numerical Integrators, Euler parameters normalization, matrix Cholesky decomposition, Householder triangularization, and Numerical Differentiation.

C.1 Numerical Integration

Numerical integration is used to calculate the trajectory of a satellite by integrating the second order differential equation of the central gravity field along with any perturbations. An excellent overview of many of the numerical integration methods can be found in [Montenbruck and Gill, 2005]. There are many different numerical integrators available that are all suitable for various purposes. Thus, only a qualitative description of the various integrators is presented in this section. To see the actual algorithms, the reader is referred to [Montenbruck and Gill, 2005] and [Press et al., 1994].

The basic theory behind all the numerical integration methods is the *Euler step*. The Euler step is a first order taylor approximation of the time derivative multiplied by the time step. The approximation that the time derivative is constant during the time step is made. This is a fair approximation when the time step is very small, but will cause the integrated solution to diverge from the true solution sooner if the time step is large. In the long run, the solution will always diverge. The downside of having a small time step is that more calculations are necessary to cover the same range and thus, the numerical inaccuracies and roundoff errors will have a higher impact. It should be noted that the numerical integration is normally carried out on first order differential equations. Thus, the second order differential equation for orbital motion has to be split into first order differential equations.

To be able to use larger time steps, higher order integrators have to be used. The higher order integrators, such as the famous Runge-Kutta 4 (RK4), approximate the function better during the time step interval. These integrators are designed in a such a way that the function does not have to be analytically manipulated, so that all the calculations can be carried out numerically. This is done by using various pre-computed coefficients, the derivations of which are beyond the scope of this work. The higher the order of the integrator, the larger the time step can be to achieve the desired accuracy. The time to carry out one integration step by a higher order integrator is higher than that of a lower order one, but the lower order integrator requires smaller time steps and thus, more integrations. Thus, the integrator to be used, along with the time step is dependent on the application. The main characteristic of the single step method is that the integration is only dependent on information of the present result. The most important single step integrators of various orders are: Dormand & Prince (DOP), Runge-Kutta-Fehlberg (RKF), and Runge-Kutta-Nyström (RKN). Of the aforementioned integrators, the RKN is more suited

for integration of second order and so the orbital motion can be directly integrated according to [Montenbruck and Gill, 2005].

The time step can be arbitrarily chosen and remain constant through out the integration procedure. It is, however, possible to let the time step vary and adapt to the integration process. The solution to the integration process is not always equally dynamic. The goal of having a variable step size is that the error due to integration over any given interval should be the same as error over any other interval, while trying to reduce computation time. Over a very dynamic interval, it is required to use a relatively small time step. If the same small time step is used over a less dynamic interval, then the accumulated roundoff error might be more than the error accumulated by using a larger time step. This can be analogized with orbital motion because the velocity of the satellite is higher near the pericenter than the apocenter. Thus, it would be a wiser use of resources to use a smaller time step near the pericenter. The local truncation error is the difference between the solution from an integrator of a specified order and the solution of an integrator that is one order lower. A tolerance value can be specified, which serves as the upper limit for the local truncation error. If the local truncation error is greater than the tolerance, the time step is decreased, otherwise the time step is increased.

C.1.1 Runge-Kutta Fourth-Order

For first approximations, the work horse in engineering applications is the Runge-Kutta 4^{th} order (RK4) integrator. For the RK4, a function $\mathbf{y}$ is taken whose derivative is a function of $\mathbf{y}$ and time, t.

$$\dot{\mathbf{y}} = \mathbf{f}\left(t, \mathbf{y}\right) \tag{C.1}$$

If the time step is chosen to be h, the value of $\mathbf{y}$ after h units of time is

$$\mathbf{y}_{k+1} = \mathbf{y}_k + \frac{h}{6}\left(\mathbf{f}_{k1} + 2\mathbf{f}_{k2} + 2\mathbf{f}_{k3} + \mathbf{f}_{k4}\right) \tag{C.2}$$

with

$$\mathbf{f}_{k1} = \mathbf{f}\left(t_k, \mathbf{y}_k\right) \tag{C.3a}$$

$$\mathbf{f}_{k2} = \mathbf{f}\left(t_k + \frac{h}{2}, \mathbf{y}_k + \mathbf{f}_{k1}\frac{h}{2}\right) \tag{C.3b}$$

$$\mathbf{f}_{k3} = \mathbf{f}\left(t_k + \frac{h}{2}, \mathbf{y}_k + \mathbf{f}_{k2}\frac{h}{2}\right) \tag{C.3c}$$

$$\mathbf{f}_{k4} = \mathbf{f}\left(t_k + h, \mathbf{y}_k + \mathbf{f}_{k3}h\right) \tag{C.3d}$$

If only the Euler step is used, the value of $\mathbf{y}$ after h units of time is

$$\mathbf{y}_{k+1} = \mathbf{y}_k + h\mathbf{f}\left(t_k, \mathbf{y}_k\right) \tag{C.4}$$

The conceptual locations of the intermediate points of the RK4 integrator can be seen in Fig. C.1.

C.1.2 Runge-Kutta Fifth-Order using Cash-Karp Coefficients

The other numerical integrator used during the thesis study is the embedded RK5(4) with Cash-Karp coefficients. This is a fifth-order method that uses the embedded fourth-order formula as a way to check the truncation error. The coefficients for this integrator can be found in Table C.1. Using the coefficients in Table C.1, the fifth-order Runge-Kutta formula is

$$\mathbf{y}_{k+1} = \mathbf{y}_k + c_1\mathbf{k}_1 + c_2\mathbf{k}_2 + c_3\mathbf{k}_3 + c_4\mathbf{k}_4 + c_5\mathbf{k}_5 + c_6\mathbf{k}_6 \tag{C.5}$$

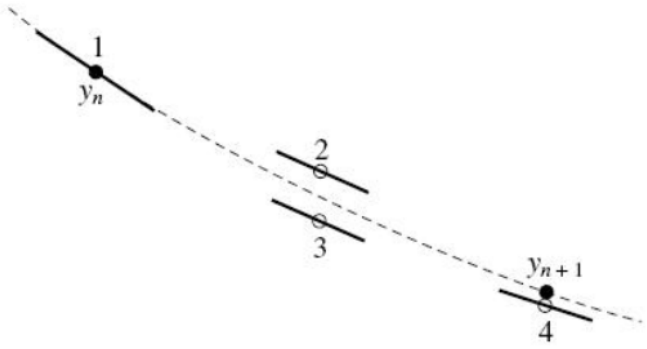


Figure C.1: All the points calculated during a fourth order Runge-Kutta integration step [Press et al., 1994]

The fourth-order formula is

$$\mathbf{y}^*_{k+1} = \mathbf{y}_k + c^*_1\mathbf{k}_1 + c^*_2\mathbf{k}_2 + c^*_3\mathbf{k}_3 + c^*_4\mathbf{k}_4 + c^*_5\mathbf{k}_5 + c^*_6\mathbf{k}_6 \tag{C.6}$$

In Eqs. (C.5) and (C.6), the various $\mathbf{k}_i$ can be found using the following formula

$$\mathbf{k}_i = h * \mathbf{f}\left(t_k + a_i h, \mathbf{y}_k + \sum_{j=1}^{i-1} b_{ij}\mathbf{k}_j\right) \tag{C.7}$$

The error estimated is the difference between the fifth-order formula and the embedded fourth-order formula.

$$\Delta \equiv \mathbf{y}_{k+1} - \mathbf{y}^*_{k+1} \tag{C.8}$$

C.1.3 Runge-Kutta Eighth-Order

To generate the ephemeris, a very high-order numerical integrator is used. This the eighth-order Runge-Kutta integrator with a seventh-order error check RK8(7). The coefficients for this method

Table C.1: The coefficients used for numerical integration with Cash-Karp coefficients [Press et al., 1994]

Cash-Karp Parameters for Embedded Runga-Kutta Method								
i	a_i	b_{ij}					c_i	c^*_i
1							$\frac{37}{378}$	$\frac{2825}{27648}$
2	$\frac{1}{5}$	$\frac{1}{5}$					0	0
3	$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$				$\frac{250}{621}$	$\frac{18575}{48384}$
4	$\frac{3}{5}$	$\frac{3}{10}$	$-\frac{9}{10}$	$\frac{6}{5}$			$\frac{125}{594}$	$\frac{13525}{55296}$
5	1	$-\frac{11}{54}$	$\frac{5}{2}$	$-\frac{70}{27}$	$\frac{35}{27}$		0	$\frac{277}{14336}$
6	$\frac{7}{8}$	$\frac{1631}{55296}$	$\frac{175}{512}$	$\frac{575}{13824}$	$\frac{44275}{110592}$	$\frac{253}{4096}$	$\frac{512}{1771}$	$\frac{1}{4}$
$j =$		1	2	3	4	5		

were derived by Prince & Dormand and can be found in Table C.2.

The eighth-order Runge-Kutta formula is

$$\mathbf{y}_{k+1} = \mathbf{y}_k + \sum_{i=1}^{13} \hat{b}_i \mathbf{k}_i \tag{C.9}$$

The embedded seventh-order Runge-Kutta formula is

$$\mathbf{y}_{k+1} = \mathbf{y}_k + \sum_{i=1}^{12} b_i \mathbf{k}_i \tag{C.10}$$

$$\mathbf{k}_i = h * \mathbf{f}\left(t_k + c_i h, \mathbf{y}_k + h \sum_{j=1}^{i-1} a_{ij} \mathbf{k}_j\right) \tag{C.11}$$

C.2 Cholesky Decomposition

Cholesky decomposition is useful in finding the square root matrix required for the DD1 and DD2 filtering methods. Cholesky decomposition works only on positive definite symmetric matrices, such as covariance matrices. This decomposition is used to find an upper triangular matrix, $\mathbf{U}$, or a lower triangular matrix, $\mathbf{L}$, from matrix, $\mathbf{A}$, that satisfies the following relation

$$\mathbf{A} = \mathbf{L}\mathbf{L}^T = \mathbf{U}^T\mathbf{U} \tag{C.12a}$$

$$\mathbf{U} = \mathbf{L}^T \tag{C.12b}$$

In MATLAB, $\mathbf{U}$ can be found simply by using $chol(\mathbf{A})$ and $\mathbf{L}$ can be found by using $chol(\mathbf{A}$,'lower'). To show how this calculation is carried out, assume that a_{ij}, l_{ij}, and u_{ij} are the components of the i^{th} row and j^{th} column of $\mathbf{A}$, $\mathbf{L}$, and $\mathbf{U}$ respectively. Since $\mathbf{U} = \mathbf{L}^T$, the diagonal entries of both these matrices are equal and can be calculated using

$$u_{ii} = l_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} u_{ki}^2} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} l_{ik}^2} \tag{C.13}$$

The off-diagonal terms of $\mathbf{U}$ can be calculated using

$$u_{ij} = \frac{a_{ij} - \sum_{k=1}^{i-1} u_{ki} u_{kj}}{u_{ii}} \quad j = i+1, \ldots, n \tag{C.14}$$

The off-diagonal terms of $\mathbf{L}$ can be calculated using

$$l_{ji} = \frac{a_{ij} - \sum_{k=1}^{i-1} l_{ik} l_{jk}}{l_{ii}} \quad j = i+1, \ldots, n \tag{C.15}$$

For UKF, DD1, and DD2, the lower diagonal Cholesky decomposition, $\mathbf{L}$, is required.

C.3 Householder Triangularization

Householder triangularization is a way of creating upper triangular matrices. The lower triangular matrices that are necessary for DD1 and DD2 can then be found from the transpose of the upper triangular matrix. The upper diagonal matrix $\mathbf{U}$ is created from $\mathbf{A}$ by

$$\mathbf{Q}_n \ldots \mathbf{Q}_2 \mathbf{Q}_1 \mathbf{A} = \mathbf{U} \tag{C.16}$$

Table C.2: The coefficients used for numerical integration with RK8(7) [Montenbruck and Gill, 2005]

c_i	a_{ij}													
0														
$\frac{1}{18}$	$\frac{1}{18}$													
$\frac{1}{12}$	$\frac{1}{48}$	$\frac{1}{16}$												
$\frac{1}{8}$	$\frac{1}{32}$	0	$\frac{3}{32}$											
$\frac{5}{16}$	$\frac{5}{16}$	0	$-\frac{75}{64}$	$\frac{75}{64}$										
$\frac{3}{8}$	$\frac{3}{80}$	0	0	$\frac{3}{16}$	$\frac{3}{20}$									
$\frac{59}{400}$	$\frac{29443841}{614563906}$	0	0	$\frac{77736538}{692538347}$	$-\frac{28693883}{1125000000}$	$\frac{23124283}{1800000000}$								
$\frac{93}{200}$	$\frac{16016141}{946692911}$	0	0	$\frac{61564180}{158732637}$	$\frac{22789713}{633445777}$	$\frac{545815736}{2771057229}$	$-\frac{180193667}{1043307555}$							
$\frac{5490023248}{9719169821}$	$\frac{39632708}{573591083}$	0	0	$-\frac{433636366}{683701615}$	$\frac{421739975}{26162923001}$	$\frac{100302831}{723423059}$	$\frac{790204164}{839813087}$	$\frac{800635310}{3783071287}$						
$\frac{13}{20}$	$\frac{246121993}{1340847787}$	0	0	$-\frac{37695042795}{15268766246}$	$\frac{309121744}{1061227803}$	$\frac{12992083}{490766935}$	$\frac{6005943493}{2108947869}$	$\frac{3930062217}{1396673457}$	$\frac{123872331}{1001029789}$					
$\frac{1201146811}{1299019798}$	$-\frac{1028468189}{846180014}$	0	0	$\frac{8478235783}{508512852}$	$\frac{1311729495}{1432422823}$	$-\frac{10304129995}{1701304382}$	$\frac{48777925059}{3047939560}$	$\frac{153367262248}{1032824649}$	$-\frac{45442868181}{3398467696}$	$\frac{3065993473}{597172653}$				
1	$\frac{185892177}{718116043}$	0	0	$-\frac{3185094517}{667107341}$	$\frac{477755414}{1098053517}$	$-\frac{703635378}{230739211}$	$\frac{5731566787}{1027545527}$	$\frac{5232866602}{850066563}$	$-\frac{4093664535}{808688257}$	$\frac{3692137247}{1805957418}$	$\frac{65686358}{487910083}$			
1	$\frac{403863854}{491063109}$	0	0	$-\frac{5068492393}{434740067}$	$\frac{411421997}{543043805}$	$\frac{652783627}{914296604}$	$\frac{11173962825}{925320556}$	$-\frac{13158990841}{6184727034}$	$\frac{3936647629}{1978049680}$	$-\frac{160528059}{685178525}$	$\frac{248638103}{1413531060}$	0		
	$\hat{b}_i$	$\frac{14005451}{335480064}$	0	0	0	0	$-\frac{59238493}{1068277825}$	$\frac{181606767}{758867731}$	$\frac{561292985}{797845732}$	$-\frac{1041891430}{1371343529}$	$\frac{760417239}{1151165299}$	$\frac{118820643}{751138087}$	$-\frac{528747749}{2220607170}$	$\frac{1}{4}$
	b_i	$\frac{13451932}{455176623}$	0	0	0	0	$-\frac{808719846}{976000145}$	$\frac{1757004468}{5645159321}$	$\frac{656045339}{265891186}$	$-\frac{3867574721}{1518517206}$	$\frac{465885868}{322736535}$	$\frac{53011238}{667516719}$	$\frac{2}{45}$	

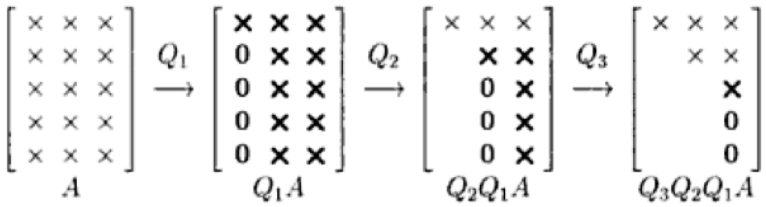


Figure C.2: Methodology of the Householder triangularization [Trefethen and Bau III, 1997]

where the products of the $\mathbf{Q}$ matrices is

$$\mathbf{Q}^T = \mathbf{Q}_n \dots \mathbf{Q}_2 \mathbf{Q}_1 \tag{C.17}$$

Finding the matrix $\mathbf{Q}$, which is unitary, is not essential for the filters and thus, will not be dealt with in this section. Also, the aim here is not to present a theoretical overview, but simply the application process. A more theoretical approach can be found in [Trefethen and Bau III, 1997].

The goal of each successive $\mathbf{Q}_i$ is to change the i^{th} column of a modified $\mathbf{A}$ so that there are only zeroes beneath the diagonal which, can be seen in Figure C.2. In Fig. C.2, the blank elements are zeros and the bold elements are the ones that are modified during each operation. The total number of $\mathbf{Q}_i$ matrices necessary, n, is either equal to smallest of the number of rows or columns of $\mathbf{A}$. $\mathbf{U}$ is found in an iterative manner starting from $i = 1$ till $i = n$. For example, after $\mathbf{Q}_1$ is used there is a new transformed $\mathbf{A}^{(1)}$. $\mathbf{Q}_2$ is then used on the 2^{nd} column of $\mathbf{A}^{(1)}$ to give $\mathbf{A}^{(2)}$. The structure of $\mathbf{Q}_i$ is as follows

$$\mathbf{Q}_i = \begin{bmatrix} \mathbf{I}_{i-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{F} \end{bmatrix} \tag{C.18}$$

In Eq. C.18, $\mathbf{F}$ is a unitary square matrix of dimension $(n - i + 1)$ and can be found using

$$\mathbf{F} = \mathbf{I} - 2\frac{\mathbf{v}\mathbf{v}^T}{\mathbf{v}^T\mathbf{v}} \tag{C.19}$$

with $\mathbf{v}$ being

$$\mathbf{v} = -\text{sign}(x_1)\,|\mathbf{x}|\,\mathbf{e} + \mathbf{x} \tag{C.20}$$

In Eq. C.20, $\mathbf{x}$ is the i^{th} column of $\mathbf{A}$. Note that the superscript of $\mathbf{A}$ has been dropped, but the suitable modified $\mathbf{A}$ has to be used and not the original one. x_1 is the first entry of $\mathbf{x}$ and $\mathbf{e}$ is a vector of the same length as $\mathbf{x}$. The ith element of $\mathbf{e}$ is 1, and the rest of the elements are zeros. Finally, the desired lower diagonal matrix, $\mathbf{L}$, is found by taking the transpose of $\mathbf{U}$.

C.4 Quaternion Normalization

Quaternions, as mentioned previously, should be of unit magnitude. Numerical integration, as shown previously, do not have an accuracy of 100%. When Euler parameters are integrated, their magnitude will tend to diverge from 1 due to these numerical inaccuracies. The fact that the unit magnitude property is also used for rotation matrices makes it essential that this property is maintained. The norm of the Euler parameters can also deviate from 1 whenever an update using one of the filters is used. The way to renormalize the the parameters is to divide by the magnitude.

$$\boldsymbol{\epsilon}_{normalized} = \frac{\boldsymbol{\epsilon}}{\sqrt{\epsilon_1^2 + \epsilon_2^2 + \epsilon_3^2 + \eta^2}} \tag{C.21a}$$

$$\eta_{normalized} = \frac{\eta}{\sqrt{\epsilon_1^2 + \epsilon_2^2 + \epsilon_3^2 + \eta^2}} \tag{C.21b}$$

Another way is to only consider the vectorial part of the Euler parameters and then consider the fourth parameter, η, as dependent term. This is not good practice because it takes away one of the possibilities to check the accuracy of the integration or estimation.

C.5 Numerical Differentiation

Jacobians are necessary for the implementation of the EKF and are time consuming to derive analytically. Thus, it is more convenient to be able to numerically differentiate a function. Only the first derivative is discussed here. A first-order approximation of the derivative of a function, $f(x)$, can be found using the forward-difference formula with h being the step size.

$$f'(x) \approx \frac{f(x+h) - f(x)}{h} \tag{C.22}$$

The central-difference formula gives the derivative accurate to the second-order.

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h} \tag{C.23}$$

Increasing the order or decreasing the step-size both increase the accuracy of the derivative. There are other methods such as complex-step differentiation [Martins et al., 2000] that can also provide accurate derivatives. However, they can cause problems when the norms of vectors are used. Thus, a new form of derivative has been used here [Shiraishi et al., 2007].

The differentiation for a univariate case is shown here, but can easily be extended to multiple variables. It is assumed that the goal is to find the derivative $df(x)/dx$ at $x = x_0$. To find this first derivative, various orders can be used from Table C.3. The order corresponds to the number of central differences to be computed. This central difference is written as

$$y_n = f\left(x_0 + n\Delta x\right) - f\left(x_0 - n\Delta x\right) \qquad (n = 1, \ldots, N) \tag{C.24}$$

The derivative can then be estimated as

$$\left.\frac{df(x)}{dx}\right|_{x=x_0} \approx \frac{\sum_{n=1}^{N} A_n y_n}{2\Delta x} \tag{C.25}$$

In Eqs. (C.24) and (C.25), it is most desirable to provide Δx in the form of

$$\Delta x = \varepsilon x_0 \tag{C.26}$$

Table C.3: Coefficients for numerical differentiation [Shiraishi et al., 2007]

N	A_1	A_2	A_3	A_4	A_5	A_6	A_7
1	1						
2	4/3	-1/6					
3	3/2	-3/10	1/30				
4	8/5	-2/5	8/105	-1/140			
5	5/3	-10/21	5/42	-5/252	1/630		
6	12/7	-15/28	10/63	-1/28	2/385	-1/2772	
7	7/4	-7/12	7/36	-7/132	7/660	-7/5148	1/12010