NTNU – Trondheim
Norwegian University of
Science and Technology

Prediction Algorithms for the NUTS Attitude Estimator and Robust Spacecraft Attitude Stabilization using Magnetorquers

Antoine Pignède

Master of Science in Cybernetics and Robotics
Submission date: May 2015
Supervisor: Jan Tommy Gravdahl, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

NTNU – Trondheim
Norwegian University of
Science and Technology

Prediction Algorithms for the NUTS Attitude Estimator and Robust Spacecraft Attitude Stabilization using Magnetorquers

Antoine Pignède

1 June 2015

MASTER'S THESIS

Department of Engineering Cybernetics

Norwegian University of Science and Technology

Supervisor: Professor Jan Tommy Gravdahl

Preface

As a part of the national student satellite program, the NUTS (NTNU Test Satellite) CubeSat project was initiated at NTNU in 2010. The projects goal is to give hands-on experience to students within different fields of satellite technology. This includes planning, specification, design, construction, launch and operation of a satellite.

The main parts of the satellite bus are:

- Power system (solar cells and batteries)
- Attitude and orbit control
- Mechanical system (structures and mechanisms)
- TT&C (telemetry, tracking and command)
- Thermal system
- On-board data handling
- Payload and experiment

The NUTS CubeSat is using the AVR32 UC3 MCU as main processor. There are two processors on board, one for the main computer and one for the communications system. The system is running the FreeRTOS operating system.

This Master's thesis is a contribution to the attitude determination and control system (ADCS) of the satellite done during the spring semester 2015. It is not the continuation of the specialization project of the fall semester 2014 even if the overall goal of developing the ADCS for NUTS still stands. This thesis has not been submitted previously and has been made by independent work. However there has been much cooperation with Henrik Rudi Haave and Marius Fløttum Westgaard, the other two members of this academic year's ADCS team.

The thesis consists of two distinct and independent parts.

The first part, *Prediction Algorithms for the Attitude Estimator*, is a direct contribution to the NTNU Test Satellite and relates the theory and implementation of three predictors that the attitude estimator needs. There is one prediction algorithm for the position of the sun with respect to the satellite, one for the position and velocity of the satellite with respect

to the earth and one for the earth's magnetic field vector at the current satellite location. The term "Predictor" is used because the position of the sun and the magnetic field vector are measured by sensors of the satellite, but this information is not used in the algorithms presented here. They just predict where the sun and how the earth's magnetic field should be before the attitude estimator merges this information with the sensor data to get the best possible attitude information.
This part is mostly self-contained and should not need much previous knowledge apart form elementary physics, mathematics and basic programming skills. The algorithms themselves are complicated, but the text can also serve as user's guide for the future developers who do not need to understand the details, but just how the software belonging to the report works.

The second part, *Robust Attitude Stabilization using Magnetorquers*, aims to prove that a known controller also works well even in the case when the inertia matrix is not known or should change. This is a theoretical issue with no direct use for the NUTS CubeSat project. Interested readers should have knowledge of nonlinear systems and attitude dynamics because not everything is repeated. The bibliography gives some hints where to start searching.

There is a reason for the presence of two such independent parts in one Master's thesis. It seemed logical for Henrik Rudi Haave and Marius Fløttum Westgaard to continue the work they began in their specialization projects, namely a focus on the determination and control parts of the ADCS respectively. There was no third large enough project related to the satellite for a whole Master's thesis especially after the successful work on the detumbling procedure in the specialization project during the fall semester 2014. So the two smaller projects were merged to one thesis.

Trondheim, 1 June 2015

Antoine Pignède

Acknowledgment

I have now spent nearly two years in Trondheim since I left Darmstadt for this Double Degree 26 July 2013. And what kind of years they were..., very busy and challenging, but in the whole one big success. And if I had questions here or in Darmstadt, people were always helpful and made it easy for me. I will now soon leave Trondheim to fulfil my Double Degree in Darmstadt, but as I always say "I will come back". But before this there are people I must mention.

Henrik Rudi Haave and Marius Fløttum Westgaard, the other two members of the ADCS team the past year. And also our office neighbors Erik and Mathias.

Our supervisor Professor Jan Tommy Gravdahl.

The whole NUTS team and the project leaders Roger Birkeland and Amund Gjersvik.

Mads and Hans Erik, Øystein.

Albana, Anna, Ela, Karolina, Marie, Clément, Dennis, Fabian and Hugo.

Andreas, JJ, Niels and Thijs.

Emma, Tiril, Anders, Fredrik and Sergio.

Angelika and Andreas, Egil.

Dedicated to the participants and organisers of the first Wings for Life World Run 4 May 2015. 35 397 runners and 62 wheelchair riders representing 162 nationalities ran on 34 race tracks in 32 countries, covered 13 time zones, on six continents for an astonishing 530 928 km.

«That makes You believe in human spirit.»

The best day of my life.

A.P.

Summary and Conclusions

Prediction Algorithms for the Attitude Estimator

The attitude estimator is an *extended Kalman filter*, a recursive algorithm to treat noisy inputs to a dynamical system. The filter is optimal to compute the system's states. One of the steps involves a model representation of the system. In the case of the attitude dynamics of a satellite that has sun sensors and a magnetometer, this requires prior knowledge of the position of the sun with respect to the satellite and the magnetic flux density vector of the earth's magnetic field at the satellite location.

These problems have all been solved already, good software is available online and no new discovery was made here. The innovation is the integration of these software parts, the harmonization of their interfaces and the legitimate simplifications done to prevent too high complexity on restricted hardware. The digital attachment contains ready to use software and test procedures for the further development of the attitude determination and control system. The thesis describes the theory and serves as user's guide for the five parts

1. **Time Conversions:** The algorithms depend on time. Different *time systems* (e.g. universal time, terrestrial time) and *time representations* (e.g. time and date, day of the year) are advantageous dependent on the application. The user deals only with the time and date representation of universal time.

2. **Frame Transformations:** As with the time, some reference frames are better suited than other for particular tasks. The user only sees vectors expressed in the Earth-Centered Inertial Frame, but internally a lot of transformations occur.

3. **Sun Vector Predictor:** A low accuracy Astronomical Almanac sun vector algorithm is used. The satellite is so close to the earth that there is no difference in the vector from the earth's centre to the sun compared to the vector from the satellite to the sun.

4. **Orbit Position and Velocity Prediction:** This is also called *orbit propagation*. The Simplified General Perturbations 4 propagator is used. It needs the Two-Line Element Set for NUTS for the orbit description and returns the satellite position needed for the next predictor. The velocity can be useful for the attitude controller.

5. **Geomagnetic Field Prediction:** There are currently two equally precise models implemented, The World Magnetic Model 2015 and The 12th Generation International Ge-

omagnetic Reference Field. Of course only one is necessary for the satellite. Hardware tests suggest to use the World Magnetic Model because it runs faster.

Robust Spacecraft Attitude Stabilization using Magnetorquers

Attitude control of a spacecraft with magnetic coils as only actuators does not suffice to attain and maintain any desired attitude. However if the aim is to stabilize the spacecraft such that one axis points to the earth, there are formal proofs that magnetorquers provide enough control action for the stabilization under certain assumptions. This chapter extends two such proofs. The need of knowledge of the spacecraft inertia is relaxed and no special property on the earth's magnetic field is required.

The first controller is a *nonlinear D controller*. A strict Lyapunov function is found with help of Matrosov's theorem. If the unknown inertia is viewed as the (small) perturbation of an estimate, the controller still stabilizes the attitude globally, uniformly and asymptotically.

The second controller is designed with the *sliding mode control* method. If the torque that the unknown inertia raises, is viewed as part of a (small) disturbance, the controller still stabilizes the attitude globally, uniformly and asymptotically.

Sammendrag og Konklusjon

Prediksjonsalgoritmer for Attitudeestimering

Attitudeestimatoren er et *extended Kalman filter*, en rekursiv algoritme som behandler de støyete inngangene til satellitens dynamiske system. Filteret er optimalt for å regne ut systemets tilstander. Et av trinnene trenger en modell av systemet. For attitudedynamikken til en satellitt som har solsensorer og et magnetometer, kreves det kunnskap om solas posisjon i forhold til satellitten, og vektoren til jordas magnetfelt der satellitten befinner seg.

Disse problemene har blitt løst allerede, god programvare er tilgjengelig på internett og ingen ny oppfinnelse ble gjort her. Hovedbidraget til denne oppgaven er sammensettelsen av de forskjellige delene, grensesnittet og forenklingene som er nødvendige slik at programvaren kan kjøre på enkel maskinvare. Det digitale vedlegget inneholder ferdig programvare med tester som kan benyttes i videre utvikling av satellittens attitudeestimering og -regulering. Denne oppgaven beskriver teorien om, og er bruksanvisningen til, de fem delene:

1. **Tidskonverteringer:** Algoritmene er tidsavhengige. Forskjellige *tidssystemer* (f.eks. universal time, terrestrial time) og *tidsformater* (f.eks. dato og tidspunkt, årets dag) er brukbare for forskjellige anvendingsområder. Brukeren jobber bare med dato og tidspunkt formatet til universal time.

2. **Rammetransformasjoner:** Som med tida er noen rammer bedre egnet enn andre for visse oppgaver. Brukeren ser bare vektorer i Earth-Centered Inertial Frame, men mange transformasjoner skjer i bakgrunnen.

3. **Solvektorprediksjon:** En algoritme fra Astronomical Almanac med lav nøyaktighet blir brukt. Satellitten er nær nok jorda til at det er neglisjerbar differanse mellom vektoren fra jorda til sola og vektoren fra satellitten til sola.

4. **Baneposisjons- og Banehastighetsprediksjon:** Også kalt *banepropagasjon*. "The Simplified General Perturbations 4" brukes. Banen til NUTS er beskrevet av et "Two-Line Element Set", og propagasjonsalgoritmen regner ut satellittens posisjon som den neste prediksjonen trenger. Hastigheten kan være nyttig for attitudereguleringen.

5. **Prediksjon av Jordas Magnetfelt:** To modeller er implementert, "The World Magnetic Model 2015" og "The 12th Generation International Geomagnetic Reference Field",

som er like gode. Selvfølgelig trengs det bare en for satellitten. Tester på maskinvaren antyder at "World Magnetic Model" bør brukes fordi den kjører raskere.

Robust Attitudestabilisering ved Bruk av Magnetspoler

Gitt kun magnetspoler vil det ikke være mulig for et romskip å nå og holde en ønsket attitude. Men hvis målet er å stabilisere romskipet slik at en akse peker på jorden, fins det formelle bevis at magnetspoler kan skape nok effekt for stabiliseringen hvis visse krav er oppfyllt. Dette kapittelet utvider to slike bevis. Det er ikke lenger nødvendig å kjenne romskipets treghet eksakt og det er ingen antagelser om egenskaper til jordas magnetfelt.

Den første regulatoren er en *ulineær D-regulator*. En "strict Lyapunov function" blir funnet ved hjelp av Matrosovs teorem. Hvis den ukjente tregheten blir sett som en (liten) perturbasjon, kan regulatoren fortsatt stabilisere attituden globalt, uniformt og asymptotisk.

Den andre regulatoren er lagt ved hjelp av *"sliding mode control"*. Hvis dreiemomentet fra grunn av den ukjente tregheten blir sett som en del av en (liten) forstyrrelse, kan regulatoren fortsatt stabilisere attituden globalt, uniformt og asymptotisk.

Contents

List of Abbreviations and Symbols

Abbreviations

Abbreviation	Description
ADCS	Attitude Determination and Control System
AU	Astronomical Unit: 1 AU = 149 597 870 700 m
BC	Before Christ
ECI	Earth-Centered Inertial, used as abbreviation for the Geocentric Equatorial Reference Frame
ECEF	Earth-Centered Earth-Fixed, used as abbreviation for the International Terrestrial Reference Frame
EKF	Extended Kalman Filter
EMM	Enhanced Magnetic Model
GMST	Greenwich Mean Sidereal Time
GNSS	Global Navigation Satellite System
HDGM	High Definition Geomagnetic Model
IAGA	International Association of Geomagnetism and Aeronomy
IAU	International Astronomical Union
IERS	International Earth Rotation and Reference Systems Service
IGRF	International Geomagnetic Reference Field
JD	Julian Date
MJD	Modified Julian Date
MOD	Mean Equator Of Date Reference Frame
NAROM	Norwegian Centre for Space-related Education (Norwegian: *Nasjonalt senter for romrelatert opplæring*)
NED	North East Down Reference Frame
NOAA/NGDC	The U.S. National Oceanic and Atmospheric Administration's National Geophysical Data Center
NTNU	Norwegian University of Science and Technology (Norwegian: *Norges teknisk-naturvitenskapelige universitet*)
NUTS	NTNU Test Satellite
SI	International System of Units (French: *Système International d'Unités*)
SGP	Simplified General Perturbations
TEME	True Equator Mean Equinox Reference Frame
TLE	Two-Line Element Set
TT	Terrestrial Time
UT	Universal Time
UTC	Coordinated Universal Time
WGS	World Geodetic System
WMM	World Magnetic Model

Symbols

Symbol	Description	Unit
$\mathbf{0}_{i\times j}$	Matrix with i rows and j columns filled with zeros	
A	Area	m^2
a	Semimajor axis of an ellipse or ellipsoid	m
$\mathbf{B}$	Magnetic flux density vector	T
b	Semiminor axis of an ellipse or ellipsoid	m
$(\cdot)^b$	Superscript for vectors expressed in the BODY frame	
$\mathbf{b}$	Vector of biases to sensors	
c	Half the distance between the foci of an ellipse or ellipsoid	m
d	Positive scalar D control gain	
E	Eccentric anomaly	°[1]
E	Energy	J
e	Eccentricity of an ellipse or ellipsoid	
$(\cdot)^e$	Superscript for vectors expressed in the ECEF frame	
$\mathbf{e}_{xx}$	Unit vector along the xx axis	
f	Flattening of an ellipse or ellipsoid	
$\mathbf{f}(t,\mathbf{x})$	State function	
g_n^m	Gaussian coefficient of degree n and order m	T
$\mathbf{g}(t,\mathbf{x})$	Perturbation term	
$\mathbf{H}$	Positive definite controller matrix	
h	Altitude, i.e. height above the ellipsoid	m
h_n^m	Gaussian coefficient of degree n and order m	T
$\mathbf{h}(t,\mathbf{x})$	Output function	
$\mathbf{I}_{i\times i}$	Identity matrix with i rows and columns	
i	Electric current intensity	A
i	Inclination of the orbit plane	°
$(\cdot)^i$	Superscript for vectors expressed in the ECI frame	
J_{xx}	Moment of inertia of a rigid body about the xx axis	kgm^2
$\mathbf{J}$	Inertia matrix of a rigid body	kgm^2
$(\cdot)_k$	Index for time discrete values or for iterations	
k	Positive scalar P control gain	
k_q, k_s, k_g	Positive scalars for sliding mode control	
$k_i(\cdot)$	Matrosov strict Lyapunov construction function number i	
M	Mean anomaly	°
$\mathbf{m}$	Magnetic dipole moment	$\frac{\mathrm{Nm}}{\mathrm{T}}$
N	Coil turn count	
$N(\phi)$	Radius of curvature in the meridian	m

[1] Note that angles in chapter 2 in general are given in degrees as function input, function output and table values. Internally the program often converts degrees to radians for the calculations.

Symbol	Description	Unit
$\mathbf{N}_{xx}$	Nutation matrix for ECI to xx frame transformation	
$\mathcal{N}_i(\cdot)$	Matrosov derivative condition function number i	
n	Mean motion	$\frac{\text{rad}}{\text{s}}$
$(\cdot)^n$	Superscript for vectors expressed in the NED frame	
$(\cdot)^o$	Superscript for vectors expressed in the ORBIT frame	
$P_n^m(\cdot)$	Schmidt quasi-normalized associated Legendre function of degree n and order m	
$\mathbf{P}$	Precession matrix	
$\mathscr{P}$	Period of one orbital revolution	s
p	Semilatus rectum (semiparameter)	m
$\mathbf{q}$	Quaternion representing the attitude	
$\mathbf{R}$	Rotation matrix for sidereal time displacement	
$\mathbf{R}_b^a$	General rotation matrix from frame A to frame B, transforms vectors from frame B to frame A	
r	Radial distance	m
r_{mean}	Mean earth radius	m
$\mathbf{r}_{xx}$	Reference vector for the EKF where xx is either the sun or the earth's magnetic field	
$\mathbf{r}^{xx}$	An arbitrary position vector in the xx frame, $\mathbf{r}^{xx} = \begin{pmatrix} x & y & z \end{pmatrix}^{\text{T}}$	m
$\dot{\mathbf{r}}^{xx}$	An arbitrary velocity vector in the xx frame	$\frac{\text{m}}{\text{s}}$
$\mathbf{S}(\cdot)$	Cross-product operator for vectors, see definition in [8]	
$\mathbf{s}$	Sliding variable	
$\mathbf{s}$	Vector pointing to the sun	m
T_{xx}	Julian century of some time point in xx time system	c.
$\mathbf{T}(\mathbf{q})$	Matrix for the quaternion differential equation $\dot{\mathbf{q}} = \mathbf{T}(\mathbf{q})\boldsymbol{\omega}$	
t	Time	s
$(\cdot)^t$	Superscript for vectors expressed in the TEME frame	
$U_i(\cdot)$	Matrosov strict Lyapunov construction function number i	
u	Argument of latitude	°
V	Lyapunov function	
V	Scalar potential for the geomagnetic field models	Tm
V_i	Auxiliary function number i	
$\mathbf{v}$	Measurement noise	
W_i	Time independent positive definite function of the state vector number i	
$\mathbf{W}$	Polar motion rotation matrix	
$\mathbf{w}$	Process noise	
x_p	Polar motion angle	°
$\mathbf{x}$	State vector	
y_p	Polar motion angle	°
z	Precession angle	°
$\mathbf{z}$	Output vector	
β	Rotation angle	rad
$\gamma_x, \gamma_y, \gamma_z$	Gains for the online parameter estimation	
ΔT	Difference of terrestrial time and universal time, $\Delta T = TT - UT$	s

Symbol	Description	Unit
$\Delta\Psi$	Longitude nutation angle	$^\circ$
δ_i^j	Kronecker delta: $\delta_i^j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$	
$\delta\Delta\epsilon_{1980}, \delta\Delta\Psi_{1980}$	Earth orientation parameter correction in obliquity nutation and in longitude nutation	$^\circ$
ϵ	Small positive number	
ϵ	True obliquity of the ecliptic	$^\circ$
$\bar{\epsilon}$	Mean obliquity of the ecliptic	$^\circ$
$\boldsymbol{\epsilon}$	Three component imaginary part of the quaternion	
ζ	Precession angle	$^\circ$
η	Real part of the quaternion	
θ	Polar angle (inclination, geocentric co-latitude)	$^\circ$
θ	Precession angle	$^\circ$
θ_{GMST}	Greenwich Mean Sidereal Time	$^\circ$
λ	Decaying constant	
λ	Longitude and azimuth	$^\circ$
$\lambda_{\min}(\mathbf{A}), \lambda_{\max}(\mathbf{A})$	The minimum and maximum eigenvalue of $\mathbf{A}$	
$\boldsymbol{\lambda}$	Rotation axis unit vector	
μ	Gravitational parameter, $\mu = Gm_\oplus$, Newton's gravitational constant times the earth's mass	$\frac{\text{m}^3}{\text{s}^2}$
ν	True anomaly	$^\circ$
ν_i	Generic bound on continuous function number i	
ρ	Density	$\frac{\text{kg}}{\text{m}^3}$
$\sigma_{\min}(\mathbf{A}), \sigma_{\max}(\mathbf{A})$	The minimum and maximum singular value of $\mathbf{A}$	
$\boldsymbol{\tau}$	Torque vector	Nm
ϕ	Latitude	$^\circ$
$\boldsymbol{\phi}(t; t_0, \mathbf{x}_0)$	Solution of a dynamical system starting at $(t_0, \mathbf{x}_0)$	
$\phi_i(\cdot)$	Matrosov derivative condition function number i	
ψ	Elevation angle (geocentric latitude)	$^\circ$
$\boldsymbol{\Omega}(\boldsymbol{\omega})$	Matrix for the quaternion differential equation $\dot{\mathbf{q}} = \boldsymbol{\Omega}(\boldsymbol{\omega})\mathbf{q}$	
ω	Angular velocity absolute value	$\frac{\text{rad}}{\text{s}}$
$\boldsymbol{\omega}_{ab}^a$	Angular velocity vector of A with respect to B expressed in frame A	$\frac{\text{rad}}{\text{s}}$
Υ	Principal direction, vernal equinox	
Ω	Ascending node	
$\mho$	Descending node	
$(\cdot)_\odot$	Subscript for the sun	
$(\cdot)_\oplus$	Subscript for the earth	
∇	Nabla or Del operator for the gradient operation	
∇^2	Laplace operator for the divergence of the gradient operation	
$\tilde{(\cdot)}$	Indicator for unknown value	
$\hat{(\cdot)}$	Indicator for estimated value	

Chapter 1

Introduction

1.1 About the NUTS Project

NUTS is a satellite building project launched in 2010 and currently under development at the Norwegian University of Science and Technology in Trondheim. The aim is to place a picosatellite in accordance with the CubeSat standard (10 cm × 10 cm × 20 cm, a so-called "Double CubeSat", **Figure 1.1**[1]) into orbit by the end of 2016. The project is the third part of the student satellite program driven by the Norwegian Center for Space-related Education (NAROM), the other two being HiNCube, built at the Narvik University College and already launched, and CubeSTAR currently under development at the University of Oslo.

The satellite's payload is an optical camera that will take pictures of the earth. But the main focus of the project is to actually go through all steps of the development of a satellite and to permit students to get in touch with state-of-the-art space technology and group project work already during their university education.

1.2 Previous Work on the NUTS ADCS

Some predecessors at the Department of Engineering Cybernetics did already a lot of work on the ADCS in their Master's theses. However most of that work concerned only theory and computer simulations, e.g. the very complete thesis [29]. It was not before the spring semester 2014 that Øyvind Rein built an ADCS prototype, that can be seen in **Figure 1.2**[2], and real testing began [26]. His work is the most interesting source since it was used as starting

[1] Image taken from [26].
[2] Image taken from [26].

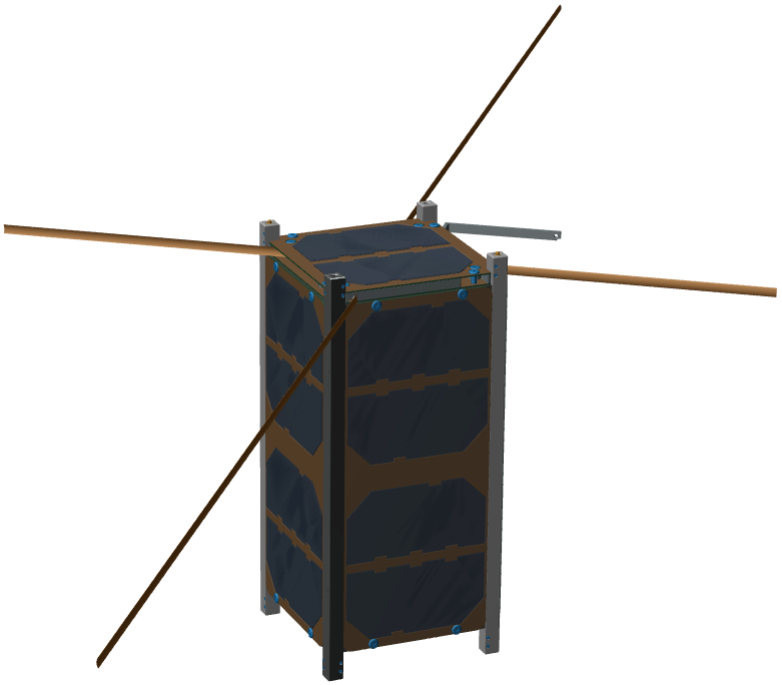


Figure 1.1: The NTNU Test Satellite

point for subsequent tasks of the academic year 2014/2015. The specialization projects of the fall semester 2014 [12], [24] and [38] refer to it to a great extent. It is therefore recommended to the reader to take a look at [26] before reading them. The same three students continued with Master's theses during the spring semester 2015, where the bounds to previous work on the NUTS ADCS are less tight. Especially the present report refers very seldom to older theses concerning NUTS directly.

Fortunately NUTS is not the first picosatellite and a lot of people faced the same problems in past years and found solutions to overcome them. There is therefore good literature accessible on this field. There is no literature review chapter in this report because the two parts concern so specific parts of spacecraft attitude determination and control, that the few interesting sources are cited directly at the relevant places.

1.3 Overall Goal of the Master's Thesis

As already stated in this introduction a lot of theoretical work and simulations have been done for the NUTS project in the past years. The time has come now to build the engineering model and for this a first fully functional ADCS is needed. This is the motivation for the first part of this thesis where a concrete problem for the NUTS ADCS is faced and software to

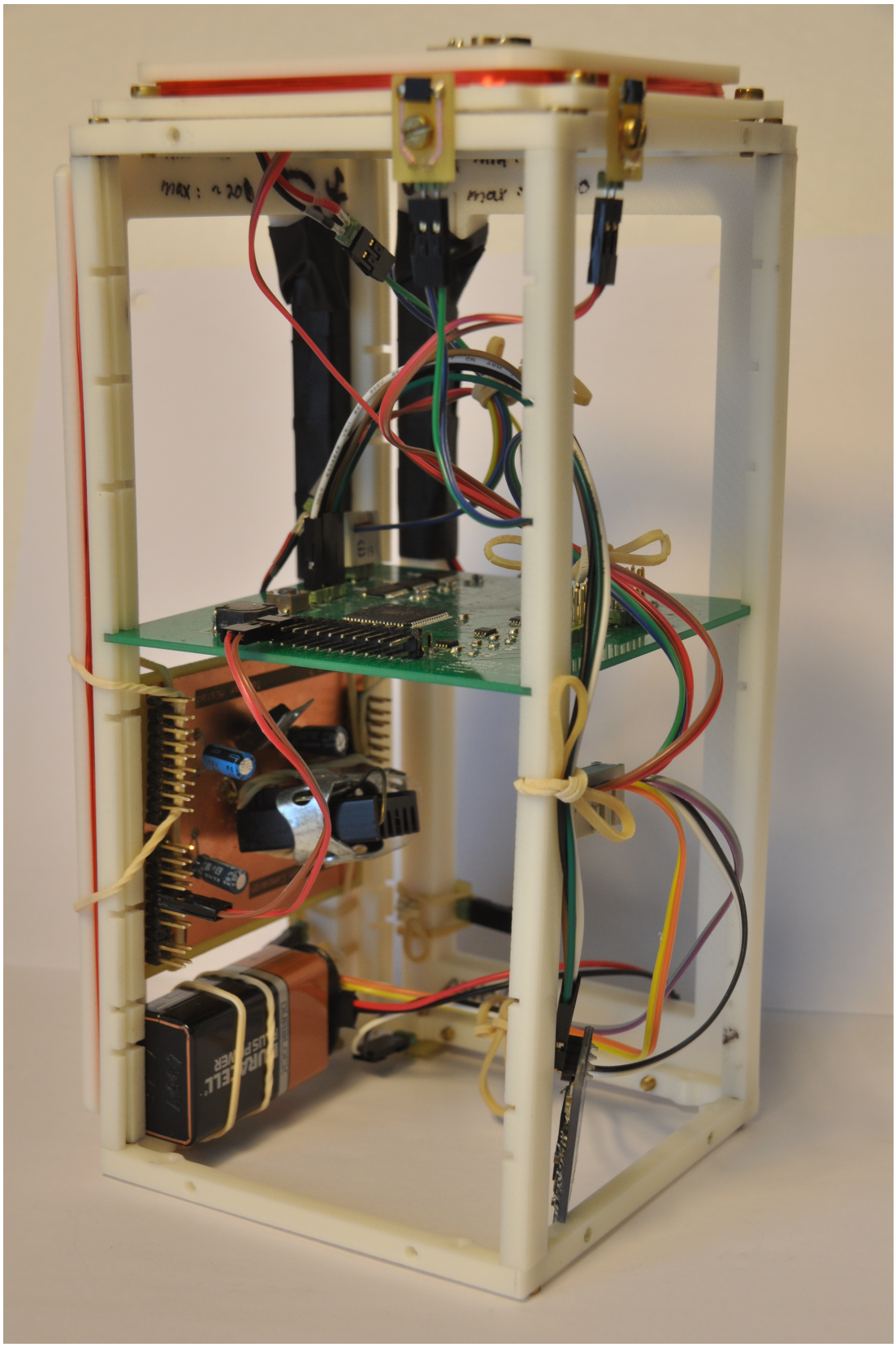


Figure 1.2: The ADCS prototype

solve it is discussed. The complete source code is in the digital attachment.

The second part also belongs to the category of attitude determination and control with magnetorquers only, but deals with a completely different problem. The bounds to NUTS are much looser, in fact this is just a theoretical inspection whose results may not be included to NUTS, but will hopefully help for the future of CubeSats and be inspiring for the interested ADCS developer. Reasons for this twofold Master's thesis are already given in the Preface and Summary of the thesis.

1.4 Structure of this Report

The structure of the thesis is actually quite simple. The two parts have each their own independent chapter. They both begin with an introduction to describe the problem and motivation before proceeding to explain the solution. Conclusions and thoughts about future work are also included for each chapter such that a global conclusion chapter is not necessary. The two chapters are ordered after their importance for the NUTS project as a whole beginning with the most important one.

The focus is on some specific ADCS parts and it is assumed that the reader has already some knowledge about the attitude determination and control system as a whole. Again [26] is a good place to start but also other Master's theses about NUTS done at NTNU's Department of Engineering Cybernetics are worth reading. For more information about the current status please also refer to the Project and Master's theses of Henrik Rudi Haave and Marius Fløttum Westgaard.

It is additionally assumed that the reader is familiar with control theory as explained in [19], rigid-body dynamics such as explained in [8], easy laws of magnetism and basics of the C programming language.

Chapter 2

Prediction Algorithms for the Attitude Estimator

2.1 Introduction

The attitude estimator for NUTS is an *extended Kalman filter*, **EKF**, which in one of its steps takes a model description of the following form in general.

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \boldsymbol{\omega}^b_{b/i,k}) + \mathbf{w}_k \tag{2.1a}$$

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{v}_k \tag{2.1b}$$

For the satellite attitude estimation the state vector $\mathbf{x}_k$, the output vector $\mathbf{z}_k$, the state equation (2.1a) and the output equation (2.1b) are [12]

$$\mathbf{x}_k = \begin{pmatrix} \mathbf{q}_k \\ \mathbf{b}^b_{s,k} \\ \mathbf{b}^b_{m,k} \end{pmatrix} \qquad \mathbf{z}_k = \begin{pmatrix} \mathbf{s}^b_{\text{norm},k} \\ \mathbf{m}^b_{\text{norm},k} \end{pmatrix} \tag{2.2a}$$

$$\mathbf{x}_{k+1} = \begin{pmatrix} \boldsymbol{\Omega}(\boldsymbol{\omega}^b_{b/i,k}) & \mathbf{0}_{3\times 6} \\ \mathbf{0}_{6\times 4} & \mathbf{0}_{6\times 6} \end{pmatrix} \mathbf{x}_k + \mathbf{w}_k \qquad \mathbf{z}_k = \begin{pmatrix} \mathbf{R}^b_i(\mathbf{q}_k) & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & \mathbf{R}^b_i(\mathbf{q}_k) \end{pmatrix} \begin{pmatrix} \mathbf{r}^i_s \\ \mathbf{r}^i_m \end{pmatrix} + \begin{pmatrix} \mathbf{b}^b_{s,k} \\ \mathbf{b}^b_{m,k} \end{pmatrix} + \mathbf{v}_k \tag{2.2b}$$

The super- and subscripts are to be read as follows:

- k denotes the discrete time.
- s stands for "sun" or "sun sensor".

- m stands for "magnetic field" or "magnetometer".
- b and i mark vectors expressed in the BODY and inertial frame, see section 2.3.

The other symbols used above are

- The state $\mathbf{q}$ is the quaternion representing the attitude, see section 3.2.
- The states $\mathbf{b}$ are biases to sensors.
- The output $\mathbf{s}^b_{\text{norm},k}$ is the normed vector pointing from the satellite to the sun expressed in the BODY frame.
- The output $\mathbf{m}^b_{\text{norm},k}$ is the normed earth's magnetic field vector expressed in BODY.
- $\boldsymbol{\omega}^b_{b/i,k}$ is the angular velocity of the BODY frame with respect to the inertial frame expressed in the BODY frame (gyroscope measurement).
- $\boldsymbol{\Omega}(\boldsymbol{\omega}^b_{b/i,k})$ is the matrix for the quaternion differential equation $\dot{\mathbf{q}} = \boldsymbol{\Omega}(\boldsymbol{\omega}^b_{b/i,k})\mathbf{q}$. This is another compact notation of the attitude kinematics, equation (3.2), than $\dot{\mathbf{q}} = \mathbf{T}(\mathbf{q})\boldsymbol{\omega}^b_{b/i,k}$. In their expanded form they're exactly the same.
- $\mathbf{w}_k$ and $\mathbf{v}_k$ are the process and measurement noise.
- $\mathbf{R}^b_i(\mathbf{q}_k)$ is the rotation matrix dependent on the quaternion representing the attitude, see section 3.2. It maps vectors from the inertial to the BODY frame.
- $\mathbf{r}^i_s$ and $\mathbf{r}^i_m$ finally are the reference vectors for the sun position and the earth's magnetic field expressed in the inertial frame.

To find the reference vectors without the sensors and attitude information is the subject of this chapter. This is required because the EKF needs to linearize the output equation (2.1b) to compute the Kalman gain and the covariance matrix. More details about the Kalman filter in general can be found in [1], more details about how the predictions are used for NUTS is part of Henrik Rudi Haave's contribution to the satellite through Project [12] and Master's thesis (likewise done in the spring semester 2015).

The following sections describe the algorithms for the time and frame transformations and the predictors that the ADCS of NUTS is going to use. A lot of time and frame transformations are needed because the reference vectors are dependent on time and place of the satellite and the algorithms use different systems and representations.

The three first sections on the time conversions and frame transformations as well as the sun vector predictor are mainly based on David Vallado's "Fundamentals of Astrodynamics and Applications" [35].
The orbit position (needed for the geomagnetic vector) and orbit velocity (needed for the attitude control, see Marius Fløttum Westgaard's Project [38] and Master's thesis) predictor is based on the original [14] and revisited [36] editions of the "Spacetrack Report no. 3".
Source code implementing the algorithms of theses sources is available for free in the languages C++, Fortran, MATLAB and Pascal on the "CelesTrak" website [18].
Two different models of equal quality of the earth's magnetic field are described in the fifth section. Implementations in C [34] and MATLAB [4] respectively are also available for free on the internet.
There is then a section that lists all the global constants that are needed by several functions. They describe mainly geometrical and magnetic properties of the earth.
The tests done on the implementation are explained and justified in section 2.8. The source code is not listed in this thesis but available as digital attachment or upon request.
The very important assumption that the current time is exactly known to the ADCS board, is a strong restriction and impossible to satisfy precisely by the satellite. The last but one section discusses what happens if the on board clock differs from the high precision clocks defining the universal time on earth.
Some material of the "User's Guide" subsections addressing the duties of future NUTS ADCS users and developers are summarized in the last section of the chapter on further work needed for the prediction algorithms for the attitude estimator.

Each of the main sections about a part of the software follows the same pattern. After an introduction to explain the purpose, the theory and the algorithms are described. The most important part of each section concerns then the implementation, the code found on the internet was for the most not very well suited for the overall goal just to find the reference vectors on the satellite hardware. It was thus shorten and simplified a lot, which is in a way the red line of this chapter. The program has to run on the satellite hardware which is not as performing as a personal computer. Thus easy algorithms and as few time systems and reference frames as possible are included. A focus is set on explaining the interface and assumptions made. Most of the readers may jump to the "User's Guide" labelled subsections directly if the only interest is how to use the software.

Two software packages are in the digital attachment of this thesis. The MATLAB package was used to rapidly make the required changes and do some preliminary tests. The second one, written in C in the Eclipse Platform and compiled with the GNU C Compiler, is meant to be added to the NUTS ADCS software and was tested more thoroughly.
Every function in the source code has a longer introduction comment explaining what the function does, what inputs and outputs it has and if any other function of the software package is used. Cross references to other related or similar functions are given too.

2.2 Time Conversions

To know the current date and time as accurately as possible is necessary for several parts of the software. The position of the sun and the earth's magnetic field change with time and the transformations from inertial to non-inertial frames too. There exist a lot of different time systems and ways to represent time, several are needed in NUTS because some algorithms are easier to do with one or another.

The time conversions is the first part of the software package explained in this thesis because it does not rely on other parts, its rather that the other software parts need the time conversions.

2.2.1 Time Systems

The time is the fundamental physics dimension of the moment something occurs. A *time system* is used to accurately define this moment based on a recurring interval such as one day or one (SI) second. The predictors use the two systems *solar* time, implemented in the *universal* time system, and *terrestrial* time. In addition *sidereal* time must also be explained because the modern universal time is based on it.

The two time systems sidereal and solar time are best explained by a picture. Consider **Figure 2.1**[1] that shows the difference between a sidereal and a solar day. Note that this illustration is greatly exaggerated.

[1] "Sidereal Time en" by Francisco Javier Blanco González - May 29, 2009. Licensed under CC BY-SA 3.0 via Wikipedia - `http://en.wikipedia.org/wiki/File:Sidereal_Time_en.PNG#mediaviewer/File:Sidereal_Time_en.PNG`.

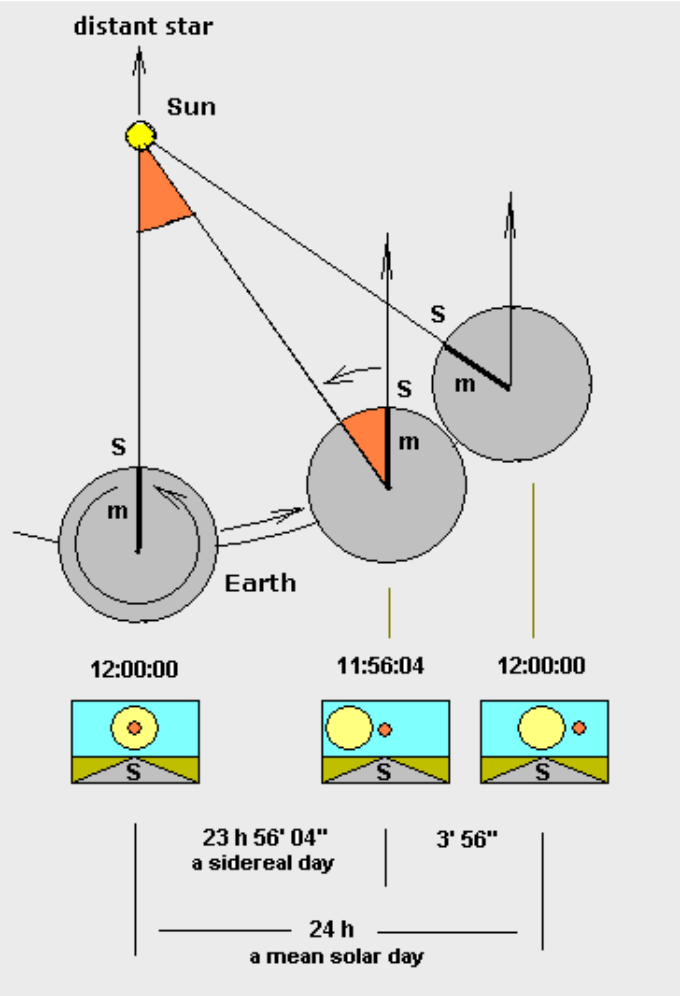


Figure 2.1: Sidereal Time and Solar Time.
Left: A distant star (the small red circle) and the sun are at culmination on the local meridian. *Center:* Only the distant star is at culmination, a sidereal day is complete. *Right:* A few minutes later the sun is on the local meridian again, a solar day is complete.

Sidereal Time

The sidereal time is based solely on the rotation of the earth about itself which takes about 236 s less than a solar day, this is the sidereal day visualized in Figure 2.1. It is usually measured as an angle from the vernal equinox, an invariable direction not rotating with the earth, see section 2.3, to the longitude of the current location positively eastward. Distant celestial objects apparently immobile with respect to the earth are used to measure this angle.

The exact vernal equinox moves, very slowly but it does, such that deviations are induced in the sidereal time. That's why one has to differ from the *mean* sidereal time with respect to a mathematically defined mean equinox and the *apparent* sidereal time that uses the true vernal equinox as reference point.

One of the frame transformations needs to take this difference of sidereal time into ac-

count. There is thus a function in the predictor software package that returns the *Greenwich mean sidereal time*, **GMST**, as angle in radians. This function though should not be needed directly.

Universal Time

The universal time is based on the sun crossing the prime meridian each day, this is the modern, precise implementation of the solar time used throughout the ages that itself is based on the solar day visualized in Figure 2.1. However as the sun's motion has irregularities, the position of distant stars and galaxies apparently immobile with respect to the earth, that is the sidereal time, is used to derive the universal time. This purest form is named **UT0**. The **UT1** form of the universal time then removes the polar motion of the earth, see section 2.3, to have a time that is independent of the location of the measurement station.

Because this universal time does not completely coincide with the generally accepted solar day of 86400 s (24 hours), the commonly used time system is *coordinated universal time*, **UTC**, where fractions of seconds are not corrected each day in the difference between UT1 and the 86400 s of a solar day. UTC is kept within ± 0.9 s of UT1 by the insertion or removal of leap seconds from time to time. This ensures small errors when UT1 is approximated with UTC. This assumption, $UT1 = UTC$, is also done for all time manipulations on the ADCS.

A part of the functions in the predictors that need time inputs, take them as coordinated universal time and all the functions that the attitude estimator calls directly and need time inputs, take them as UTC even if the underlying functions refer to another time system.

Terrestrial Time

When describing the motion of celestial objects relativistic phenomenons are important to consider. It is then often useful to have the equations of motion about the barycenter of the solar system, that is it's center of mass. The independent variable in these equations is the *barycentric dynamical time*. Describing this in terms of the motion of the earth results in the *terrestrial time*, **TT**. The correct definition is

> «the theoretical timescale of apparent geocentric ephemerides of bodies in the solar system.» [35]

It is fortunately easier than the definition suggest to include the terrestrial time system to the predictors. A conversion from UTC to TT is required because some other functions

that need time inputs e.g. for finding rotation matrices, have their coefficients dependent of terrestrial time.

The difference between both time systems is called $\Delta T = TT - UT$ where the universal time is approximated with the coordinated universal time in the NUTS software. ΔT is not constant but can be approximated with polynomials as described in [23]. For the years 2005 to 2050 it is a simple quadratic function of the year and month.

2.2.2 Time Representations

Actual time points, independent of the time system, can be represented in various ways and with different units. People are used to express time as date and time of the day but this is not very practical for computer software. The time representation for NUTS are presented in the next paragraphs. For every interaction with the users the common date and time representation is chosen for ease of use.

Date and Time

This is the common way to represent time in everyday conversation. The date has the three elements *year*, *month* and *day* and the time is depicted into *hours*, *minutes* and *seconds*. All the five first values are integers while the seconds also can contain fractional parts.

This time representation, in the coordinated universal time system, is also assumed to be available on NUTS and is thus the base time representation. Whenever the user needs to provide a time input to get one reference vector, a combined variable of five integers and one floating point number must be provided.

Instead of hours, minutes and seconds the time of the day can also be given as *angle* from a reference point, usually the vernal equinox. This value, along with the date as year, month and day, is the preferred way for sidereal time. It is just used internally at one place for the GMST and thus not important to consider for the user.

Year and Fraction, Day of the Year and Fraction

There are several disadvantages with the common representation mentioned above. The month and the day of the month do not start with 0 but with 1 and on top of that not every month has the same number of days but can have 28, 29, 30 or 31 days.

A better representation of time and one that uses less variables, namely one floating point number, is thus to give the month, day, hours and so on as fraction of the given year.

This format is for example used in the geomagnetic field predictors. The flux density coefficients are given as pairs. The first value is the coefficient at the beginning of the valid time range of the model, 1 January 2015 or 2015.0 for the moment, and the second is a slope value which is estimated to hold for the next five years. The magnetic coefficients of any date and time in between those five years can then easily be found by linear interpolation if the time is given as fractional year.

The lack of precision of this kind of representation can be improved if the fraction of the year is expressed in days. That is two values define the time point, the year and the day and fraction. This is a popular time format in space applications because it is relatively easy to read, even if one of the mentioned drawbacks is not solved by it. The unit of the second value is "day of the year" meaning that it still begins with 1 and not with 0.

Julian Date and Julian Century

As in the previous representation date and time can also just be expressed in days and fraction of days after or before an arbitrary date, called *epoch*. This is the concept of the *Julian date*, **JD**, where the time is given as the difference with 1 January 4713 BC at noon in days. Note that the Julian calendar and not the Gregorian calendar is used, meaning that each year evenly divisible by four is a leap year.

The strange epoch has its origin in the only common point for the solar cycle (28 years), the Metonic cycle (19 years) and the Roman Indication (15 years) which together create the Julian period lasting 7980 years [35]. To set the beginning at noon instead of midnight permitted astronomers to make their observations (night work) on one day. This noon however can be 12:00:00 in sidereal, universal or terrestrial time. Universal time is usually meant when the Julian date is given without explicit time system.

The Julian date tends to use very large numbers in common applications meaning a lack of precision in the milliseconds range. That's why the *modified* Julian date, **MJD**, was introduced that simply removes the two highest digits of our era and shifts the beginning of the day to midnight, i.e.

$$MJD = JD - 2\,400\,000.5$$

But a lot of functions from the source code require the full Julian date and its precision is

good enough for NUTS. The modified Julian date is just used in two short approximation algorithms in the NUTS frame transformations.

Another concept on the contrary is often used to calculate rotation matrices. They express their coefficients not in days but in *Julian centuries* (36525 days exactly) before or after 1 January 2000 at noon in terrestrial time. This epoch is commonly referred to as **J2000.0**. The conversion from the Julian date is

$$T_{TT} = \frac{JD_{TT} - 2\,451\,545}{36525} \tag{2.3}$$

where the subscript denotes that terrestrial time must be used. Note however that the same conversion can be applied for any time system.

2.2.3 Assumptions

Before moving on to detail the functions related to time manipulation in the predictors of NUTS, some important assumptions made to simplify the software must be listed.

The most important assumption is that the current coordinated universal time is known to the ADCS as year, month, day, hour, minute and seconds (with fraction). Whether it may be provided by the on-board computer or is directly accessible on the ADCS board does not matter. Should the format of this clock not be year, month, day, hour, minute and seconds, then an additional conversion function must be implemented if the software package in the digital attachment is to be used.

As already written in the presentation of the universal time concept, the precise universal time version UT1 is approximated here with the coordinated universal time. This brings in the worst case an error of 0.9 s, but is accurate enough as the sun position and the earth magnetic field do not change that fast.

The weekly published IERS Bulletin A [16] has an approximation formula for the difference $UT1 - UTC$ but this is not included in the software for NUTS as this would make even more conversions necessary. In addition the source code from [18] not always is clear when UTC or UT1 should be used.

Finally two less important assumptions, that are of no threat to NUTS, must be mentioned.

The algorithm used to find the Julian date is only valid for years between 1900 and 2100. This

exceeds the life expectation of NUTS very much.
The polynomial estimation of ΔT, the difference between universal and terrestrial time, is only valid from 2005 to 2050. NUTS will be launched in this decade and will, like every object in low earth orbit, experience rests of atmospheric drag that will slow down the speed and end the satellite's life after 25 to 30 years, so before 2050. This smaller validity period is thus not of practical meaning either. Anyway the source [23] provides further polynomials valid for other year ranges if needed.

2.2.4 Implementation Details

In total eleven functions, all having names beginning with the keyword "time", and one struct are declared and defined in the `timeConv.h` header and the `timeConv.c` source files.

The `typedef struct Time` regroups five integers and one floating point number to form a data structure for the base time representation as date and time of the day. The floating point number is of course for the seconds and fractions of seconds. The user only will have to deal with this time representation.

The first ten functions are pairs to convert time systems or representations back and forth. Not all of these actually are used in the predictors, but they were very important in the testing phase.

Time Systems Conversion

Only two time systems, terrestrial time and universal time (approximated by UTC), are part of the software package. A `time` struct expressed in one of the systems can be converted to the other with the two functions

- `Time timeUtc2tt(Time timeUTC)`
- `Time timeTt2utc(Time timeTT)`

They implement the currently valid approximation for the difference $TT - UT$ in seconds [23]

$$y = year + \frac{month - 0.5}{12}$$
$$\Delta T = 62.92 + 0.322\,17(y - 2000) + 0.005\,589(y - 2000)^2$$

and return a new `Time` struct in the other time system. The year and fractional days representation is used internally to simplify. Problems possibly arising in the beginning or end of years as well as leap years are treated.

Time Representation Conversions

The other eight functions convert between different representations. Four of them

- `double timeDatetime2days(Time time)`
- `Time timeDays2datetime(int year, double days)`
- `double timeDatetime2years(Time time)`
- `Time timeYears2datetime(double years)`

should be self-explanatory and the only difficulties are to consider leap years and that the variable "days" for the day of the year plus fraction begins with 1. They were implemented from scratch with occasional help from [35]. Note however that this source assumes the year to be between 1900 and 2100 such that every fourth year is a leap year. The functions for NUTS are universal in that case, even if the practical usage is negligible.

The two functions that handle Julian dates

- `double timeDatetime2jd(Time time)`
- `Time timeJd2datetime(double jd)`

were copied almost directly from [18], the software package that goes together with [35]. The algorithm to compute the Julian date for years between 1900 and 2100 implemented in the first of these to functions is

$$JD = 367year - \left\lfloor \frac{7\left(year + \left\lfloor \frac{month+9}{12} \right\rfloor\right)}{4} \right\rfloor + \left\lfloor \frac{275month}{9} \right\rfloor + day + 1721013.5 + \frac{\frac{sec}{60} + min}{60} + hour}{24}$$

The back conversion does not invert the function above because of the floor operator $\lfloor\cdot\rfloor$. It takes away the Julian date of 1 January 1900 from the input. It then finds the integer year and uses the easy conversion for fractional days back to `Time` listed above.

The last pair of conversion functions

- `double timeJd2jc(double jd)`

- `double timeJc2jd(double jc)`

applies equation (2.3) and its reverse. Note that even if equation (2.3) mentions terrestrial time, the same conversion is also valid for any other time system.

The last time conversion function, which also was copied form [18], is a little different and has no reverse function implemented.

- `double timeGstime(double jd)`

finds the Greenwich mean sidereal time of the Julian date (universal time) in radians with the following algorithm [35]

1. Compute the Julian century T_{UT} with the function `timeJd2jc`.
2. Compute the GMST in seconds

$$\theta_{\text{GMSTsec}} = -6.2 \cdot 10^{-6} T_{UT}^3 + 0.093\,104\, T_{UT}^2 + (876\,600 \cdot 3600 + 8\,640\,184.812\,866)\, T_{UT} + 67\,310.548\,41\,\text{s}$$

3. Compute the GMST in radians

$$\theta_{\text{GMST}} = \theta_{\text{GMSTsec}} \cdot \frac{\pi}{180} \frac{\text{rad}}{\text{deg}} \cdot \frac{1}{240} \frac{\text{deg}}{\text{s}}$$

4. Ensure that θ_{GMST} lies in the interval $[0, 2\pi]$.

There is no reverse function because it's not possible to find the date and hence the time of that date from just one angle. But that is of no concern because the GMST is only needed as angle value to compute the sidereal time rotation matrix in the frame transformations.

2.2.5 Time Conversions User's Guide

There should be no great need to use the time conversion functions directly, especially if the ADCS can provide the current time in the format of the `Time` struct. But one should know what each functions does, especially those that convert from date and time representation to another. They are often one of the first steps of the frame transformations or predictors because they need internally another time format as the user inputs. Sometimes, like for the sidereal time rotation matrix described in the coming frame transformation section, two conversion one after the other are necessary (because `timeGstime` takes a Julian date input).

2.3 Frame Transformations

Ideally every position and velocity vector would be expressed in just one reference frame regardless of the application. But it is quasi impossible to find one frame that fits all purposes. Inertial frames are often preferred for equations of motion because Newton's laws are valid just for them. The earth's magnetic field on the other hand rotates with the earth and the models make use of this by addressing the position in coordinates that rotate with the earth, rotating frames are not inertial.

In the years since the beginning of spaceflight a lot of different coordinate systems were defined depending on what effect they should emphasize and for what application they are intended. An important part of any astrodynamics software consists thus of transformation functions between several frames. This usually means computing rotation matrices, but in theory translations could also be needed if different origins such as the sun's center, the earth's center or the earth's surface at an arbitrary point make the application easy.

Note that the vectors are not changed by the transformation, they're only represented with different coordinate values according to a different set of axes.

2.3.1 Reference Frames

The number of frames in the NUTS software package was reduced to four right-handed frames from the numerous ones defined in [35] and implemented in the software package of [18]. Two of them are earth-centered, non-rotating, thus sufficiently inertial frames. The third reference frame is also earth-centered but rotates with it. In addition to Cartesian coordinates this frame also uses two other sets of coordinates, geodetic and geocentric coordinates, in the geomagnetic field prediction, see section 2.6. This predictor has as original output a magnetic vector with north-, east- and downward components at the specific location. This is the fourth frame.

Earth-Centered Inertial Frames

Earth-centered non-rotating frames are not really inertial because the earth moves around the sun and the whole solar system also moves in the galaxy (which of course moves too), but approximate an inertial frame well enough for earth orbiting satellites.

The first and most important of all the frames is the *geocentric equatorial reference frame*

which is often referred to as **ECI**, as done in this thesis, or *J2000* because the principal direction is based on the J2000.0 epoch. As its name indicates the origin is in the earth's center and the fundamental plane is the earth's equator, that means that the z-axis points to the north pole.
The principal direction, that is the x-axis, is the *vernal equinox* of the J2000.0 epoch. When an observer sees the sun crossing the intersection of the equatorial plane with the *ecliptic* (the plane of the earth's mean orbit about the sun), the vector pointing to the sun is called "equinox" because this is when day and night are equally long. This happens twice a year at the beginning of spring (around 21 March, the *vernal* equinox) and of fall (around 23 September, the *autumnal* equinox). From the northern hemisphere point of view the sun crosses the intersection upwards on the vernal equinox, that's why one also says that the sun is at its *ascending node*. Actually the principal direction of ECI is not the *true* but the *mean* equinox, a mathematically defined direction that has not the slow movement of the true equinox.
The y-axis simply is in the fundamental plane 90° east of the x-axis to complete the right-handed system.

All inputs and outputs of functions that are important for the user, will be vectors expressed in this ECI frame. Whenever another reference frame is more practical, the transformations are done internally as first and last steps.

ECI should not be confused with the *geocentric celestial reference frame* which is closely aligned with the geocentric equatorial reference frame but has varying fundamental plane and principal direction with respect to ECI to achieve an even better inertial approximation.

The reason why the orbit position and velocity estimator presented in section 2.5 is a good choice for NUTS, is explained in that section. But it has an impact on the frame transformations because the position and velocity outputs are in the *true equator mean equinox reference frame*, **TEME**, and not in the ECI reference frame. But a common base frame is wanted.

The original "Spacetrack Report no. 3" [14] published in 1980 says nothing about the reference frame for the output values and surprisingly nobody seems to exactly know how to resolve the position and velocity vectors to one of the known and well defined frames. Additionally the name is misleading because this frame uses the true equator, like ECI, as fundamental plane but not the mean equinox conventionally used today.

Likewise the time system is not explicitly named in [14].

The general belief how to deal with these uncertainties can be found in the revision of the "Spacetrack Report no. 3", [36]. With the universal time system and the given formulas to convert TEME vectors to ECI, the algorithm's outputs are equal to observations done by sensors within the measurement and numerical accuracy. A summary of this revision is contained in [35] and, as with nearly everything in this section, implementations are on [18].

Earth-Centered Earth-Fixed, International Terrestrial Reference Frame

As stated in the introduction to this section, the earth's magnetic field is modelled easiest with a frame fixed to the earth, that is a frame that rotates with the earth and is thus not inertial. The commonly used earth-centered earth-fixed reference frame is the *international terrestrial reference frame*, hereafter referred to as **ECEF** as the sources do.
Again the fundamental plane is the earth's equator, hence the z-axis points to the north pole. The principal direction points to the intersecting point of the equator and the Greenwich prime meridian. To form a right-handed system the y-axis is 90° east of the x-axis in the fundamental plane.

The latest revision of the *world geodetic system*, **WGS 84**, used for example by GNSS, does not completely coincide with ECEF but has usually deviations in the centimeters range.

In everyday speech, maps of the earth and a lot of applications ECEF positions are not given as Cartesian coordinates but as latitude ϕ, longitude λ and altitude h. Together these three values form the *geodetic coordinates* as can be seen in **Figure 2.2**[2]. Unfortunately the altitude term is ambiguous. While everyday speech most often refer to the elevation above mean sea level, altitude in this thesis means the *height above the ellipsoid* as indicated in the figure.

Even if the geodetic coordinate system is the input for the geomagnetic field calculators of the sources, they internally use another coordinate set, the *geocentric coordinates* which are the ellipsoidal equivalent of spherical coordinates. **Figure 2.3**[3] shows them as well as their relation to the geodetic coordinates. The three values are called radial distance r, az-

[2] "Geodetic coordinates" by Peter Mercator - Own work. Licensed under CC BY-SA 3.0 via Wikimedia Commons - http://commons.wikimedia.org/wiki/File:Geodetic_coordinates.svg#mediaviewer/File:Geodetic_coordinates.svg.

[3] "Geocentric coordinates" by Peter Mercator - Own work. Licensed under CC BY-SA 3.0 via Wikimedia Commons - http://commons.wikimedia.org/wiki/File:Geocentric_coordinates.svg#mediaviewer/File:Geocentric_coordinates.svg.

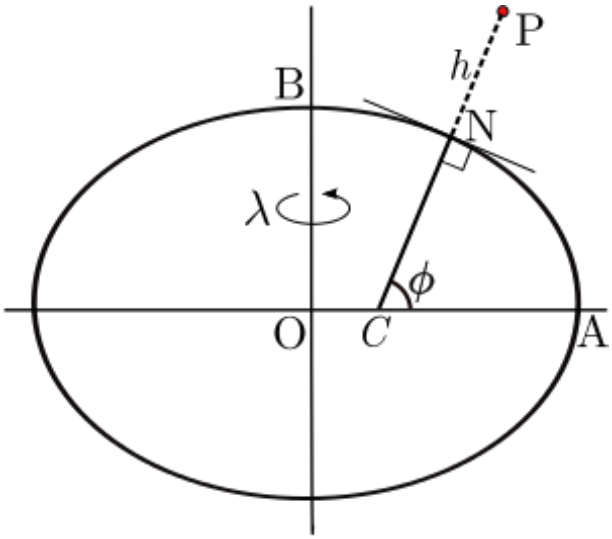


Figure 2.2: Geodetic Coordinates $P(\phi, \lambda, h)$

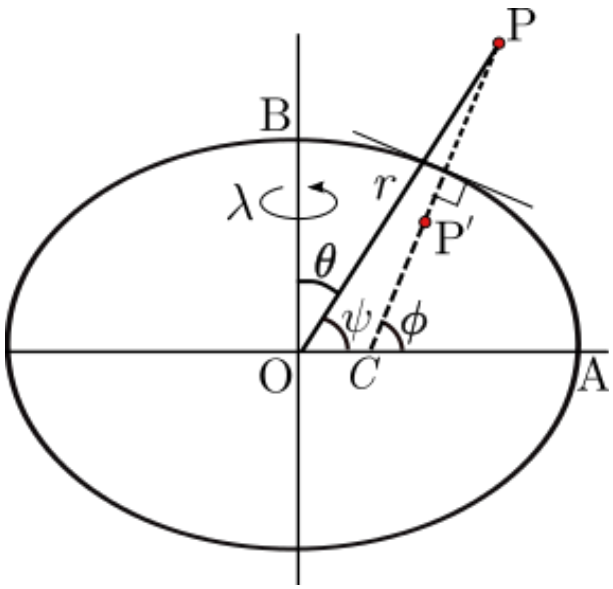


Figure 2.3: Geocentric or Spherical Polar Coordinates $P(r, \theta, \lambda)$ or $P(r, \psi, \lambda)$

imuth angle λ (it is equal to the longitude) and elevation angle ψ. The latter is often called geocentric latitude, as opposed to the geodetic latitude, or replaced by the polar angle θ, which sometimes has the names inclination or geocentric co-latitude.

The two figures greatly exaggerate the flattening of the earth. In fact the difference in (geodetic) latitude and (geocentric) elevation angle is never larger than 0.2°.

North East Down Reference Frame

The orthogonal plane that can be seen at point N in Figure 2.2 defines the *north east down reference frame*, **NED**. This is a local reference frame often used for indicating marine vessel or aircraft velocities with one component pointing to the north pole, one in eastern direction and the third perpendicularly down.

This right-handed system is usually not used in astrodynamics and is only included in this thesis because the earth's magnetic field vector is given in this frame by the geomagnetic field models. It is then transformed to ECI in the implementation for NUTS as explained in section 2.6.

One has not to account for the translation when transforming velocity vectors expressed in NED to velocity vectors expressed in ECEF. One only needs a rotation matrix that is dependent on the latitude and the longitude. Fortunately the same applies also for magnetic vectors such that the same NED to ECEF transformation as in [8] can be used.

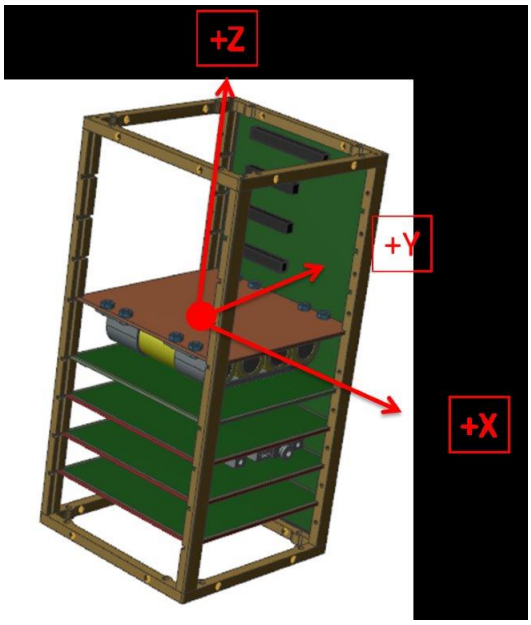


Figure 2.4: The NUTS BODY Reference Frame

BODY and ORBIT Reference Frame

Two last reference frames have to be mentioned even if they're not used in the prediction software. But it is necessary to know them, to understand the Kalman filter model (2.2) and the background, that is that the predictions are needed to perform attitude determination and control for an earth orbiting satellite. In addition they appear often in the second main chapter of this thesis.

The BODY reference frame origin is located at the satellite's center of mass and the x-, y- and z-axes are the principal axes, meaning that the inertia matrix is diagonal. **Figure 2.4**[4] shows this frame, the positive sense of the axes is through the electromagnetic coils.
The extended Kalman filter actually works in the inertial frame, but the magnetometer, the gyroscope and the sun sensors express their measurements in the BODY frame, thus rotations have to be included in the underlying model. Those however are not part of the prediction algorithms and won't be considered further because the satellite is treated as a point with negligible mass compared to the earth in this chapter. Chapter 3 and other publications on the NUTS ADCS give more information on the BODY frame and transformations.

The ORBIT frame also has its origin in the satellite's center of mass but the axes do not rotate with the satellite. The z-axis points to the earth's center, the x-axis in the velocity direction and the y-axis completes the right-handed frame. This frame rotates about the

[4] Image provided by the NUTS team.

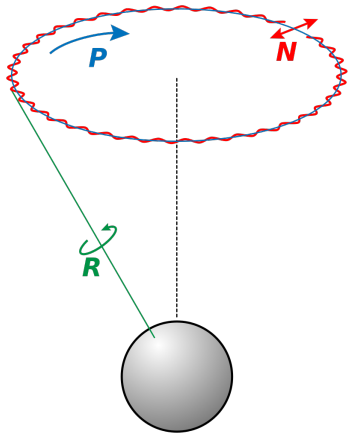


Figure 2.5: Rotation (green), Precession (blue) and Nutation in Obliquity (red) of a Planet

earth with the angular velocity ω_o.

The *attitude* of the satellite is defined as the rotation matrix or quaternion or Euler angle set describing how vectors expressed in the BODY and ORBIT frame are related to each other. To actively control this is the goal of the attitude determination and control system.

2.3.2 The Earth: Orbit, Rotation and Shape

The transformation between the earth-centered inertial frames and the earth-fixed frame is the repeated application of rotation matrices describing one variation in the earth's orbit about the sun or one variation of the earth's rotation about itself. The three first parts of this subsection will account for them.

The last paragraphs of this subsection about a model of the earth's shape are important for the three coordinate sets used in the ECEF reference frame.

Precession

Two effects are regrouped under the term *precession*. This is the blue circle shown in **Figure 2.5**[5] and can be experienced by everyone with a spinning top.

In the earth's case the blue circle is mainly caused by the gravity forces of the sun and the moon on the non-spherical earth. They induce *luni-solar precession* which results in circular motion of the earth rotation axis. The period is about 26000 years long and the half angle of

[5] "Praezession" by User Herbye (German Wikipedia). Designed by Dr. H. Sulzer - Original. Licensed under CC BY-SA 3.0 via Wikimedia Commons - `http://commons.wikimedia.org/wiki/File:Praezession.svg#mediaviewer/File:Praezession.svg`.

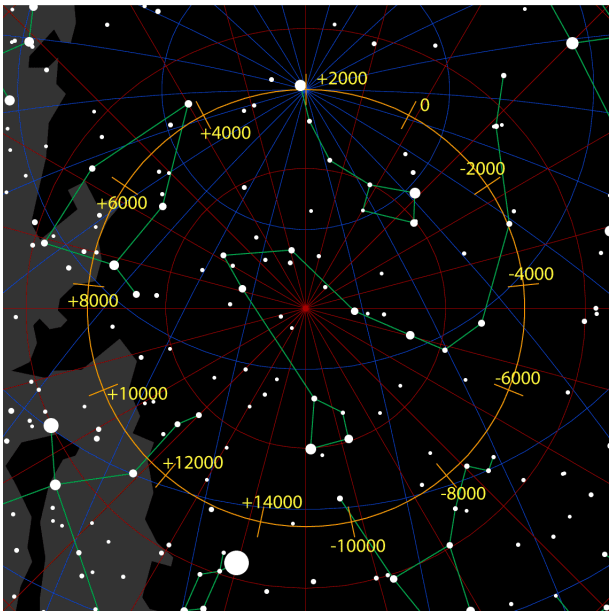


Figure 2.6: The path of the north celestial pole among the stars due to precession

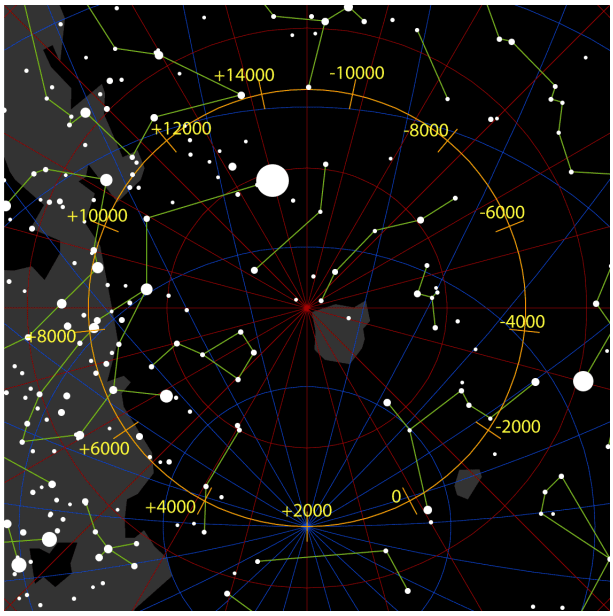


Figure 2.7: The path of the south celestial pole among the stars due to precession

the cone traced by the axis equals approximately 23.5°. **Figure 2.6**[6] shows the north circle for the current precession period. One can see that Polaris not always was so close to the north pole as it is now. Of course the same circle is drawn at the south, **Figure 2.7**[7].

In addition to this precession, there is a somewhat linearly increasing precession because the ecliptic is not fixed with respect to the distant celestial objects that create the time systems. The *obliquity*, the angle formed by ecliptic and equator, decreases very slowly and the vernal equinox moves constantly westward. This *precession of the ecliptic* is due to the gravity forces of the other planets in the solar system and is therefore still often called planetary precession even if the term is considered obsolete.

Nutation

A smaller and faster oscillation of approximately 18.6 years in the rotation axis of the earth, the red line in Figure 2.5, is induced by the moon. This *nutation* is in the range of millidegrees and has several causes. The most important is the 18.6 precession period of the moon itself, but also the orbit inclination and eccentricity of the moon's orbit about the earth as well as some solar induced perturbations result in nutation.

[6] "Precession N" by Tau'olunga - self, 4 bit GIF. Licensed under CC BY-SA 2.5 via Wikimedia Commons - `http://commons.wikimedia.org/wiki/File:Precession_N.gif#mediaviewer/File:Precession_N.gif`.

[7] "Precession S" by Tau'olunga - self, 4 bit GIF. Licensed under CC BY-SA 2.5 via Wikimedia Commons - `http://commons.wikimedia.org/wiki/File:Precession_S.gif#mediaviewer/File:Precession_S.gif`.

Because of this number of effects the nutation matrix is the most difficult rotation matrix to compute for the frame transformations. Two Euler angles are computed as weighted sums of five time varying *fundamental* or *Delaunay arguments*:

1. The mean anomaly of the moon, the angle with respect to the periapsis in the orbit about the earth.

2. The mean anomaly of the sun in the orbit about the solar system barycenter.

3. The mean argument of latitude of the moon, that is the angle between the ascending node and the current position vector.

4. The mean elongation, the angle at the moon in the sun-earth-moon triangle.

5. The right ascension of the ascending node of the moon's orbit, that is the angle in the ecliptic plane from the equinox to the ascending node.

These arguments refer to the mean moon and sun, that is they are not the exact current values for the moon or sun, but calculated with respect to mathematically defined mean motions of the moon and sun.

Polar Motion

The rotation axis of the earth is not fixed with respect to the crust. That means that the poles on a map of the earth at latitudes of $\pm 90°$ are not exactly the places were the axis pierce through the surface.

On 31 December 2014 for example the axis is shifted from the ECEF poles by $8.95° \cdot 10^{-6}$ along the prime meridian and $7.78° \cdot 10^{-5}$ along the 90° W longitude [17]. This is about 87 m from the ECEF north pole at the surface but increases of course with higher altitude, such that polar motion is more important to take into account for satellite positions.

Flattening

As already hinted in the presentation of the ECEF frame, the earth is not a sphere but approximately an oblate ellipsoid of revolution with two equal semimajor axes a in the equator plane, the equatorial radii, and one shorter polar radius, the semiminor axis b. Two semiprincipal axes are thus equal which permits a two dimensional analysis as already done

in Figure 2.2 and Figure 2.3. The mathematical description of this non ideal "roundedness" can be done with the *eccentricity e* or the *flattening f* property.

For the earth the flattening is easier to visualize. It is simply the ratio

$$f = \frac{a-b}{a}$$

where $a = \overline{\text{OA}}$ and $b = \overline{\text{OB}}$ in Figure 2.2. The value for the earth is very low, $\frac{1}{298.3}$, but due to the large dimensions the difference between the earth radii is 21 km and not negligible.

The eccentricity is easier to explain with the orbit of a light celestial object about a much heavier one located at one of the foci, e.g. the earth's orbit around the sun or the orbit of a satellite about a planet. But flattening and eccentricity describe the same phenomenon. If one denotes the half of the distance between the two focal points with c, the eccentricity is

$$e = \frac{c}{a} = \frac{\sqrt{a^2 - b^2}}{a}$$

and again very small for the earth ellipsoid, about 0.082.
The next formula gives the relation between eccentricity and flattening

$$e^2 = 2f - f^2$$

It is used very often because the flattening of the earth is easier to measure while the eccentricity often is better to use in the transformation formulas.

For even preciser analysis the ellipsoid model has to be altered for example with a latitude dependent oblateness, but the complicated math behind this won't be explained here even if the orbit position and velocity predictor uses it, see the main sources of these sections [35] and [36] for deeper insight.

2.3.3 Assumptions

Of course the assumptions for the time conversions listed longer up also count for the frame transformations as all of them, except the transformation between the Cartesian, geodetic and geocentric ECEF coordinates, have rotation angles that change with time. But additional and quite restrictive assumptions have to be made on the year in which the transformation between ECI and ECEF is valid. They do not affect the transformation between the two iner-

tial frames.

The displacement of the rotation axis with respect to the ECEF poles, the polar motion, is given by the angle x_p along the prime meridian and the angle y_p along the 90° W longitude. The weekly updated IERS Bulletin A [16] gives the daily observed values for the last week and predictions up to one year in advance in the milliarcsecond unit. But to omit computer programs to save large look-up tables an interpolation of the following form is given

$$MJD = JD - 2\,400\,000.5 \tag{2.4a}$$

$$A = 2\pi \frac{MJD - MJD_{\text{bulletin}}}{365.25}, \qquad C = 2\pi \frac{MJD - MJD_{\text{bulletin}}}{435} \tag{2.4b}$$

$$x_p = a_x + b_x \cos A + c_x \sin A + d_x \cos C + e_x \sin C \tag{2.4c}$$

$$y_p = a_y + b_y \cos A + c_y \sin A + d_y \cos C + e_y \sin C \tag{2.4d}$$

where JD is the Julian date for the time point at which one wants to get the polar displacement, MJD_{bulletin} is the modified Julian date of release of the IERS Bulletin A and $a_x, \ldots, e_y$ are coefficients given in the bulletin that slightly change each week.
The only comment on the validity of the interpolation the bulletin makes is:

> «The [...] formulas will not reproduce the predictions given below [the x_p and y_p predictions for the next 365 days], but may be used to extend the predictions beyond the end of this table.» [16]

This suggests that one can use the $a_x, \ldots, e_y$ coefficients of one bulletin longer than one year and still have meaningful values for the displacement. But whether e.g. the next five years are within this time span or not and whether dates before the release can use this approximation as well, is not known.
For the best performance it is thus recommended to update the global constants, see **Table 2.3**, used in equations (2.4) with the values from the newest bulletin close to the launch date. This should be enough for at least a few years of operation.
In this thesis the IERS Bulletin A released 12 February 2015 was used throughout the testing phase. Even if these tests involved dates that lie years before this release, reasons for the actual test dates are given in section 2.8, the approximation is in general good, at least 4 significant digits are always equal between test result and verification value.

For reasons explained in the next subsection the *IAU-76/FK5 reduction* method is used for the transformation between ECI and ECEF. Because of this choice two parameters in the

rotation matrix for nutation are needed, the earth orientation parameter corrections in longitude nutation $\delta\Delta\Psi_{1980}$ and in obliquity nutation $\delta\Delta\epsilon_{1980}$. They account for the differences between the older IAU-76/FK5 and the newer *IAU-2000 reduction* method and ensure that transformations between the same frames have the same results with both methods.
The monthly updated IERS Bulletin B [17] lists the values of the two parameters in milliarcseconds for the second but last month, that is the bulletin released on 1 February 2015 has the daily values of December 2014. It does not provide predictions, but the paper [37] that reviews the long term changes of amongst other things $\delta\Delta\Psi_{1980}$ and $\delta\Delta\epsilon_{1980}$, has linear trend lines for these two parameters (in arcseconds) using the MJD as in equations (2.4)

$$\delta\Delta\Psi_{1980} = -8\cdot10^{-6}MJD + 0.2506 \tag{2.5a}$$

$$\delta\Delta\epsilon_{1980} = -7\cdot10^{-7}MJD + 0.022 \tag{2.5b}$$

These equations yield higher values compared to [17] with approximately $-5.71^\circ\cdot10^{-5}$ for $\delta\Delta\Psi_{1980}$ and $-4.98^\circ\cdot10^{-6}$ for $\delta\Delta\epsilon_{1980}$ for 31 December 2014 (MJD 57022). The ranges observed by the IERS in the last months of 2014 oscillate between $-2.5^\circ\cdot10^{-5}$ and $2.4^\circ\cdot10^{-5}$ for $\delta\Delta\Psi_{1980}$ and between $-2.8^\circ\cdot10^{-6}$ and $2.5^\circ\cdot10^{-6}$ for $\delta\Delta\epsilon_{1980}$. This is not very good, although anyway the earth orientation parameter corrections always are small, but nonetheless the linear interpolation, equations (2.5), is implemented in the NUTS software because it was the only alternative found to constant values or to a look-up table approximated with the IERS Bulletin B [17]. In addition [37] claims that its linear trend may not be very accurate on daily basis but catches the long term variation and is based on the final values of the last 50 years.

This also is the reason why the linear trend line for x_p and y_p that [37] also provides, is not implemented in the frame transformations. The oscillations around the linear trend for $\delta\Delta\Psi_{1980}$ and $\delta\Delta\epsilon_{1980}$ have a much smaller amplitude than the oscillations of x_p and y_p. There the approximations with the weighted sine and cosine sum of equations (2.4) is closer to the truth even if the coefficients are not valid for the same long time range.

To summarize this subsection, in addition to the time assumptions the polar motion approximation (equations (2.4)) and the linear interpolation $\delta\Delta\Psi_{1980}$ and $\delta\Delta\epsilon_{1980}$ (equations 2.5) are assumed to be valid for the life span of NUTS.

2.3.4 Implementation Details

There are fourteen frame transformation functions, that all begin with the keyword "frame", and one struct in the `frameTrans.h` header and the `frameTrans.c` source files.

The `typedef struct Delaunay` regroups five floating point numbers to form a data structure for the Delaunay fundamental arguments needed to compute the nutation matrix.

The first six functions are pairs to transform position vectors from one reference frame to the other. These are the functions that are called in the predictors each time a frame change must be done.
Then come seven functions that are called during the frame transformation calculations. Five of them compute the rotation matrices that take the earth rotation descriptions into account, one calculates the fundamental arguments and one function is used recursively for a step of the ECEF Cartesian to geodetic coordinate transformation.

The last function of the fourteen provides the rotation matrix to the geomagnetic predictor to transform the north-east-down magnetic field vector to a vector expressed in the ECEF frame.

Transformations between the Geocentric Equatorial Reference Frame (ECI) and the International Terrestrial Reference Frame (ECEF)

The two functions

- `void frameEcef2eci(double recef[3], Time timeUTC, double reci[3])`
- `void frameEci2ecef(double reci[3], Time timeUTC, double recef[3])`

implement the frame transformation

$$\mathbf{r}^i = \mathbf{P}\mathbf{N}_e\mathbf{R}\mathbf{W}\mathbf{r}^e \tag{2.6a}$$

$$\mathbf{r}^e = \mathbf{W}^{\mathrm{T}}\mathbf{R}^{\mathrm{T}}\mathbf{N}_e^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}\mathbf{r}^i \tag{2.6b}$$

That is they take the values the first pointer points to, calculate the four time dependent rotation matrices at the given time point and write the transformation result where the second pointer points to. The four rotation matrices that connect vectors expressed in the ECI and ECEF together are the precession matrix $\mathbf{P}$, the nutation matrix $\mathbf{N}_e$ (there is a similar nutation

matrix for transformations between ECI and TEME, hence the subscript to distinguish), the rotation matrix for sidereal time displacement **R** and the polar motion rotation matrix **W**.

The older IAU-76/FK5 and not the newer IAU-2000 reduction method is used in the NUTS software package because the source code taken from [18] implements this method in its own `ecef2eci` and `eci2ecef` functions. There are also functions for the newer reduction but there are two different ways to implement them and they are slightly more difficult to use. With the correction parameters mentioned in the preceding subsection nearly exactly the same results come from all transformation implementations. The last argument to use the IAU-76/FK5 method, is that the transformation between ECI and TEME also is implemented. It uses the same functions for the fundamental arguments and the precession matrix and a similar nutation matrix. The choice reduces thus the number of functions needed.

The four rotation matrices are calculated by these four functions again edited from [18]

- `void framePrecess(double ttt, double prec[9])`
- `void frameNutation(double ttt, double *deltapsi, double *meaneps, double *omega, double nut[9])`, this function calls a fifth function
  - `Delaunay frameFundarg(double ttt)`
- `void frameSidereal(double jd, double deltapsi, double meaneps, double omega, double st[9])`
- `void framePolarm(double jd, double pm[9])`

`framePrecess` takes the current time as terrestrial time Julian century `ttt` and writes the matrix **P** where the `prec[9]` pointer points to. The precession is described by the three Euler angles ζ about the z-axis, θ about the y-axis and z about the z-axis. Simple third order polynomials return the angles.

`frameNutation` at first calls `frameFundarg` to interpolate the Delaunay fundamental arguments at the current time point. One correction to the original IAU-76/FK5 reduction are the newer IAU-2000 fourth order polynomials returning preciser fundamental arguments. It is legitimate to do this because this gives a higher accuracy even if this differs from the original method. The pointer `*omega` saves the fifth fundamental argument, the right ascension of the ascending node of the moon's orbit, that is needed later.

The actual algorithm to obtain the matrix $\mathbf{N}_e$ is complicated and not necessary to relate in

detail here. It involves a weighted sum of the fundamental arguments to ultimately find the true obliquity of the ecliptic ϵ for a rotation about the x-axis, the longitude nutation $\Delta\Psi$ angle of the next rotation about the z-axis (saved in the `*deltapsi` pointer) and the mean obliquity of the ecliptic $\bar{\epsilon}$ for another rotation about the x-axis, this value is also returned with the `*meaneps` pointer. This process involves the global constants IAR80 and RAR80, see Table 2.3, and the corrections $\delta\Delta\Psi_{1980}$ and $\delta\Delta\epsilon_{1980}$, that are approximated according to equations (2.5). Finally the matrix $\mathbf{N}_e$ is saved where `nut[9]` points to.

In the ECI frame the *mean* sidereal time, with respect to the *mean* vernal equinox, holds. In the ECEF frame, that moves with all movements of the earth, on the contrary the *apparent* sidereal time, measured with respect to the *true* vernal equinox, holds. The transformation must therefore have a rotation about the z-axis to account for this effect, and that's what `frameSidereal` does. The angle is a function of the Greenwich mean sidereal time θ_{GMST} (returned by a time conversion function) and the four input arguments `jd`, `deltapsi`, `meaneps` and `omega`. The rotation matrix $\mathbf{R}$ can be accessed by the pointer `st[9]`.

The final rotation matrix $\mathbf{W}$ describes the polar motion and is the result of the `framePolarm` function saved where the pointer `pm[9]` points to. Two principal rotations are performed, with the angle x_p about the y-axis and with the angle y_p about the x-axis. As difference to the original functions from [18] that simply asks for x_p and y_p to be input arguments, the algorithm (2.4) presented in the Assumptions subsection is implemented in the function and the reason why the Julian date format of the universal time `jd` is an input argument to the function.

Transformations between the Geocentric Equatorial Reference Frame (ECI) and the True Equator Mean Equinox Reference Frame (TEME)

The pair of transformation functions

- `void frameTeme2eci(double rteme[3],Time timeUTC, double reci[3])`
- `void frameEci2teme(double reci[3],Time timeUTC, double rteme[3])`

implements the back and forth transformation of ECI and TEME

$$\mathbf{r}^i = \mathbf{P}\mathbf{N}_t\mathbf{r}^t \tag{2.7a}$$

$$\mathbf{r}^t = \mathbf{N}_t^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}\mathbf{r}^i \tag{2.7b}$$

that uses the same precession matrix **P** than the transformation (2.6) but a different nutation matrix, hence called $\mathbf{N}_t$. Luckily both frames are considered inertial such that the velocity transformations between ECI and TEME, the only velocity transformations needed in the predictions, are exactly the same than the transformations for the position vectors. That is one can simply replace **r** with $\dot{\mathbf{r}}$ in equations (2.7).

The two functions work exactly like `frameEcef2eci` and `frameEci2ecef` explained in the preceding paragraphs. At first they do the necessary time conversions, then they get the time dependent matrices and finally apply the transformation (2.7).

The precession matrix is again calculated in `framePrecess` as for the transformations between ECI and ECEF. The nutation matrix $\mathbf{N}_t$ is different and the result of

- `void frameTruemean(double ttt, double nutteme[9])`

Like for the $\mathbf{N}_e$ calculation the current terrestrial time given in Julian centuries after the J2000.0 epoch `ttt` is an input argument, but here no correction terms are needed. The matrix pointed to by `nutteme[9]` is actually more than just a nutation matrix. The weighted sum of the fundamental arguments is the same than in `frameNutation` discussed in the previous section, but a kind of sidereal time matrix is multiplied afterwards.

Transformations between Cartesian and Geodetic Coordinates in the ECEF Frame

These transformation functions were not taken from [18], like the vast majority of functions in this chapter, but copied form the MATLAB IGRF source code [4]. The reason for this, is that [18] provides two slightly different transformations from Cartesian to geodetic coordinates, but no back transformation. On the other hand [4] has two functions for both transformations that are easy to read.

Nearly no change had to be made to incorporate the functions into the NUTS predictor package. Even if the help comments of the original function tell the user that the Cartesian coordinates and the height over the ellipsoid must be in meters, absolutely no change in the algorithm itself is needed if every figure is interpreted as kilometers. One must just make sure that the earth constants are in kilometers too. This is not difficult to do when the global values of Table 2.3 are used instead of locally declared constants.

Using the same naming convention as before, the transformation is implemented with

- `void frameEcef2geod(double recef[3], double rgeod[3])`

- `void frameGeod2ecef(double rgeod[3], double recef[3])`

One can see directly that these coordinate changes are easier than those described earlier, because they're independent of time, which is obvious because Cartesian and geodetic coordinates are just two different ways to express the same vector in the same reference frame. `recef[3]` points to a vector $\begin{pmatrix} x & y & z \end{pmatrix}^{\mathrm{T}}$ in Cartesian coordinates in km, while `rgeod[3]` points to a vector $\begin{pmatrix} \phi & \lambda & h \end{pmatrix}^{\mathrm{T}}$ in geodetic coordinates, see the definition in Figure 2.2. The latitude ϕ and the longitude λ is in degrees and the altitude, or height over the ellipsoid, h is in km as already said. Note that the program does not use the Greek letters but simply calls the variables `latitude`, `longitude` and `altitude`.

`frameEcef2geod` implements the coordinate transformation from Cartesian to geodetic coordinates using the following algorithm.

1. The most difficult value to find is the latitude ϕ. This is done recursively using

   - `void frameLatitudeRecur(double lat_in, double z, double rd, int iter, double *latitude, double *Nphi)`

   `lat_in` is the current best estimate for the latitude, the initial value is

   $$\phi_0 = \arcsin\left(\frac{z}{\sqrt{x^2+y^2+z^2}}\right)$$

   `z` is the z coordinate input and `rd` is $\sqrt{x^2+y^2}$, these two variables do not change during the recursive process. `iter` is the number of the current iteration. Because two output values come from the recursive function, pointers are used with `*latitude` pointing to the improved latitude estimate (k is the iteration number)

   $$\phi_{k+1} = \arctan\left(\frac{z + N_k(\phi)e^2 \sin\phi_k}{\sqrt{x^2+y^2}}\right)$$

   and `*Nphi` pointing to the *radius of curvature in the meridian*

   $$N_{k+1}(\phi) = \frac{a}{\sqrt{1 - e^2 \sin^2 \phi_{k+1}}}$$

   that is the distance from N along the line $\overline{NC}$ until it crosses the vertical axis below O in Figure 2.2.

The recursion ends either when the change in latitude is smaller than 10^{-12} or 20 iterations have been reached.

2. Now that the latitude is known the height over the ellipsoid is found according to

$$h = \sqrt{x^2 + y^2}\cos\phi + (z + e^2 N(\phi)\sin\phi)\sin\phi - N(\phi)$$

3. The longitude is simply

$$\lambda = \arctan\frac{y}{x}$$

where the angle is resolved to the interval $[-180°, 180°]$ using the `atan2` function.

The back transformation implemented in `frameGeod2ecef` is easier:

$$N(\phi) = \frac{a}{\sqrt{1 - e^2\sin^2\phi}}$$
$$x = (N(\phi) + h)\cos\phi\cos\lambda$$
$$y = (N(\phi) + h)\cos\phi\sin\lambda$$
$$z = (N(\phi)(1 - e^2) + h)\sin\phi$$

Transformations between Geodetic and Geocentric Coordinates in the ECEF Frame

The geomagnetic predictors (section 2.6 explains why two different algorithms are implemented) use both geodetic and geocentric coordinates depending on what is more convenient for the current operation. However no such transformation is part of the `frameTrans.c` source file because the IGRF uses the polar angle θ, while the WMM uses the elevation angle ψ. Thus the transformations are done directly inside the respective function.

For the sake of completeness of this section on reference frames an easy conversion for the geodetic and geocentric angles is nonetheless provided. The longitude for geodetic and azimuth for geocentric coordinates are the same, hence the common symbol λ. Elevation angle ψ and Latitude ϕ are related to each other with the known eccentricity e of the earth according to [35]

$$\psi = \arctan\left((1 - e^2)\tan\phi\right), \qquad \phi = \arctan\left(\frac{\tan\psi}{1 - e^2}\right)$$

Transformations between the International Terrestrial Reference Frame (ECEF) and the North East Down Reference Frame (NED)

The "Handbook of Marine Craft Hydrodynamics and Motion Control" [8] lists the rotation matrix $\mathbf{R}_n^e$ for converting velocity vectors expressed in the NED frame to velocity vectors expressed in ECEF. The same rotation matrix transforms any kind of vectorial quantity (forces, field strengths) and the back transformation is achieved by taking the inverse matrix, which for rotation matrices always is the transpose, that is $\mathbf{R}_e^n = (\mathbf{R}_n^e)^{-1} = (\mathbf{R}_n^e)^{\mathrm{T}}$.

This matrix is dependent on the latitude ϕ and longitude λ only:

$$\mathbf{R}_n^e = \begin{pmatrix} -\sin\phi\cos\lambda & -\sin\lambda & -\cos\phi\cos\lambda \\ -\sin\phi\sin\lambda & \cos\lambda & -\cos\phi\sin\lambda \\ \cos\phi & 0 & -\sin\phi \end{pmatrix}$$

The function

- `void frameRotationNed2ecef(double sinlat, double coslat, double sinlon, double coslon, double Ned2ecef[9])`

returns this rotation matrix at the place the pointer `Ned2ecef[9]` points to. The input arguments are not the latitude and longitude but rather their sine and cosine values. This makes the implementation slightly faster because `frameRotationNed2ecef` merely consists of some multiplications. The calling functions, the geomagnetic field predictors, already have the sine and cosine values available.

2.3.5 Frame Transformations User's Guide

Just as the time conversions, the frame transformations should seldom be needed directly, and if they are, then most probably it will be one of the functions that transform from one frame to another. Then one has first to declare a floating point array of length 3 that will be the last input parameter. The function will then write the result in that array. This is the easiest way to return arrays or multiple values in the C programming language, so even if all the functions have the return type `void`, they actually do "return" something. The same concept applies for all function returning 3×3 matrices. But the matrices are addressed as if they were 9×1 matrices, that is first the three rows of the first column then the second and finally the three rows of the third column.

An example situation where the frame transformations are addressed by the user directly, is the testing of the orbit position and velocity predictor. The test output vectors are in the ECI frame, but the verification values [36] are provided in the TEME frame. Then one has first to convert using a transformation function to see if the test was successful.

An important notice for future users is to check if the approximations still are close enough to $\delta\Delta\Psi_{1980}$ and $\delta\Delta\epsilon_{1980}$ that are published every month in the second section of the IERS Bulletin B [17]. Likewise the `MJDPOLAR`, `XPOLAR[5]` and `YPOLAR[5]` constants, see Table 2.3, should be updated with the newest IERS Bulletin A [16] before the launch.

2.4 Sun Vector Prediction

Now that the preliminaries, that is the time and reference frames, are done, the real purposes of this chapter, that is to find the reference vectors of the EKF with predictors, are addressed. The first and easiest of them is the sun vector predictor.

2.4.1 Astronomical Almanac Sun Vector Algorithm

An algorithm to find the sun vector originally published in the *Astronomical Almanac* is related in [35] and a good choice for NUTS because of its shortness. The low accuracy of 0.01° at best is no major drawback for the project. The reference vectors need not to be very precise to yield acceptable results in the attitude estimator.

This algorithm returns a vector pointing from the center of the earth to the center of the sun. The lines from the earth to the sun in the greatly exaggerated Figure 2.1 can help to visualize this vector. Because of the immense distance from the earth to the sun, one *astronomical unit* i.e. 1 AU = 149 597 870 700 m on average, it makes no difference at all that the vector points from the center of the earth and not from the accurate position of the satellite, other assumptions (next subsection) are more important.

The first step of the algorithm is to find the mean anomaly $M_\odot$ and the mean longitude $\lambda_{M_\odot}$ of the sun using linear interpolation. The latter must be adapted to the inertial frame resulting in the longitude of the ecliptic

$$\lambda_{\text{ecliptic}} = \lambda_{M_\odot} + 1.914\,666\,471^\circ \sin M_\odot + 0.019\,994\,643^\circ \sin 2M_\odot$$

which is the angle from the vernal equinox to the current sun position vector. The obliquity of the ecliptic ϵ is also interpolated linearly. The actual figures are not important to relate here and can be read in the source code or [35]. These approximations are then used to calculate the magnitude of the sun vector $r_\odot$ and finally the sun vector itself $\mathbf{r}_\odot$. The last steps are hence

$$r_\odot = 1.000\,140\,612\,\text{AU} - 0.016\,708\,617\,\text{AU}\cos M_\odot - 0.000\,139\,589\,\text{AU}\cos 2M_\odot$$

$$\mathbf{r}_\odot = \begin{pmatrix} r_\odot \cos\lambda_{\text{ecliptic}} \\ r_\odot \cos\epsilon \sin\lambda_{\text{ecliptic}} \\ r_\odot \sin\epsilon \sin\lambda_{\text{ecliptic}} \end{pmatrix}$$

2.4.2 Assumptions

The accuracy of this sun vector prediction algorithm is as said not very high, 0.01° at best [35], but that does not come from the fact that the satellite's position with respect to the earth is not taken into account. The distance from the earth to the sun is millions of km longer than the satellite's distance from the earth's center such that no significant difference occurs by the assumption that the sun vector of the satellite is the same than the sun vector starting at the center of the earth.

The following assumptions are responsible for the limited accuracy. They are still satisfied well enough by NUTS for the attitude estimation.

The polynomial interpolations of the mean anomaly $M_\odot$, the mean longitude $\lambda_{M_\odot}$ of the sun and the obliquity of the ecliptic ϵ are truncated to the first order, that is a simple linear relationship with time is assumed. This results in good approximations only for about a century. The values used in the implementation of the algorithm in [35] are taken at the J2000.0 epoch, thus the algorithm is only valid for years between 1950 and 2050. This assumption is of no practical importance for NUTS. The same assumption, thus also of no importance, holds for the truncated expressions for the longitude of the ecliptic $\lambda_{\text{ecliptic}}$ and the magnitude of the sun vector $r_\odot$ that stop after the second harmonic.

When describing the motion of planets and other celestial bodies in the solar system, it's best to use the barycentric dynamical time. This time system is the independent variable in the equations of motion. It can without problem be approximated with the terrestrial time because the difference is in the ms range [35].

Finally as before the universal time is implemented by the coordinated universal time UTC only. This also is well within the needed accuracy for NUTS as the sun vector won't change significantly even in the worst case when UTC differs by 0.9 s from UT1.

Even if this subsection might seem long, actually there is no real restrictive assumption that harms the usage of this easy algorithm for the NUTS ADCS.

2.4.3 Implementation Details

A short and easy to use function returning the sun vector after the algorithm presented here is available at the main code source [18]. Some minor changes were done to remain consistent with the defined interface.

- `void sun(Time timeUTC, double rsun[3])`

now writes the sun vector at the desired universal time, representing by the `timeUTC` struct (see section 2.2), in the array `rsun` is pointing to.

The Astronomical Almanac uses the *mean equator of date*, **MOD**, inertial frame which is not the ECI frame used throughout this thesis as the inertial reference frame. Thus a final step

$$\mathbf{r}_{\odot}^{i} = \mathbf{P}\mathbf{r}_{\odot}$$

must be added to the algorithm which transforms the sun vector form MOD to ECI, hence the i superscript on the left hand side, by finding the precession matrix $\mathbf{P}$ with the `framePrecess` function discussed earlier.

The output unit is AU which is somewhat the mean distance of the earth from the sun. But the magnitude changes a little periodically with the slightly elliptic orbit of the earth about the sun. To make the processing in the EKF easier the sun vector is always normed to a magnitude of 1 before it is returned.

The last difference between the original code from [18] and the implementation for NUTS concerns the time systems. The barycentric dynamical time, that actually should be used in the linear interpolations, is approximated with universal time in the original function. However a far better approximation is available in the NUTS software, the terrestrial time which differs from the barycentric dynamical time only by a few ms at most. The difference between terrestrial (thus also the difference between barycentric dynamical) time and universal time however is about 69 s for January 2015 according to the polynomial approximation

Listing 2.1: Sun Vector Prediction Example

```
...
#include "sunPred.h"
...
int main(){
    // Excerpt of ex5_1.m from Vallado (2013) Fundamentals of
       Astrodynamics and Applications
    Time time;
    time.year = 2006;
    time.mon = 4;
    time.day = 2;
    time.hr = 0;
    time.min = 0;
    time.sec = 0;
    double rsun[3];
    sun(time, rsun);
    printf("rsun: %f %f %f\n", rsun[0], rsun[1], rsun[2]);
    return 0;
}
```

[23] and is expected to continue to increase. The better approximation i.e. terrestrial time was added to the sun function too.

2.4.4 Sun Vector Prediction User's Guide

The sun vector predictor is most likely the first presented function to be of great importance for the user, because its output is directly fed into the extended Kalman filter algorithm through the estimator model. It is the reference vector $\mathbf{r}_s^i$ in equation (2.2). Therefore it is important to know how to use the predictor. Fortunately this one is very simple.

Once the `sunPred.h` header file is included, the `sun` function is available to get the sun vector at an arbitrary time. This function has two input arguments and at first sight no output because the return type is `void`. However this is not completely true as the second input argument is a pointer where the sun vector is written by the function. Thus the user must first declare a floating point array, preferably of double precision type, of length three and provide the pointer name as second input argument. The first argument is the time point when the sun vector shall be calculated. The format is the `Time` struct presented in section 2.2 and must use the universal time system.

Note that the sun vector is normed to magnitude 1 and expressed in the geocentric equatorial reference frame (ECI) which is not the same output as the Astronomical Almanac sun

vector algorithm provides (MOD). The ECI frame is rotated with respect to the MOD frame with the precession matrix.

The usage is best explained by a short example. See **Listing 2.1** which is taken from example 5-1 in [35]. The output of the NUTS implementation is

```
«rsun:  0.978049 0.191181 0.082883 »
```

which is exactly equal to the result of example 5-1 in [35] rotated to the ECI frame and normed.

2.5 Orbit Position and Velocity Prediction

The EKF needs a reference vector $\mathbf{r}_m^i$ for the geomagnetic field in the model equation (2.2). Of course the earth's magnetic field depends primarily on the current position of the satellite with respect to the earth. There is however no possibility to get a good estimate of the position from the sensors on NUTS. The position has thus to be estimated directly without a correction method based on sensors. The alternative would be to have a GNSS receiver but this has not been under consideration so far.

Processes that calculate positions of space objects based on orbital elements only, are called *orbit propagators*, the name *orbit predictor* is chosen for the scope of this thesis to show the connection with the sun vector and geomagnetic field predictors that are the main parts of the software package. Theses two are called predictors because the sensors on NUTS will measure a sun vector and the earth's magnetic field with sensors and both prediction and measurement will be used to estimate the attitude.

Orbit propagators have in general also the ability to give an estimate of the satellite velocity. This is the second reason for the inclusion of such a function in the ADCS.
The gyroscope on board measures the angular velocity of the BODY frame with respect to the inertial frame expressed in BODY $\boldsymbol{\omega}_{ib}^b$. However the angular velocity with respect to the ORBIT frame $\boldsymbol{\omega}_{ob}^b$ is needed in the attitude controller and the equation relating both together is [38]

$$\boldsymbol{\omega}_{ib}^b = \boldsymbol{\omega}_{ob}^b + \boldsymbol{\omega}_{io}^b = \boldsymbol{\omega}_{ob}^b + \mathbf{R}_o^b \boldsymbol{\omega}_{io}^o \tag{2.8}$$

where $\boldsymbol{\omega}_{io}^o = \begin{pmatrix} 0 & -\omega_o & 0 \end{pmatrix}$. ω_o is the orbital angular velocity and there are two ways to calculate it. The first equation has been used so far in simulations of the attitude controller. The

gravity and the centrifugal forces are in balance, thus $\omega_o = \sqrt{\frac{\mu}{a+h}}$ using the earth's gravitational parameter μ, its semimajor axis a and the satellite's altitude h assumed to be constant. However with the predictor the easier formula without square root $\omega_o = \frac{\dot{r}}{r}$ involving just the orbit position and velocity magnitudes is possible to use instead. Note that this formula is exact just for circular orbits, however the other formula also assumes a circular orbit and in addition a constant altitude.
For the moment this newer equation is not implemented but both approaches will be considered in the future development. See Marius Fløttum Westgaard's contribution to the satellite through Project [38] and Master's thesis (likewise done in the spring semester 2015).

In theory if one knows the initial position and velocity of an object and integrates all forces and torques acting on this object, one can give the position and velocity at any time in the future with great accuracy. But there is a drawback to this approach, numerical integration. One cannot just specify the desired time for the prediction and directly get the result, rather a lot of small steps in between are needed, where each time the position and the velocity have to be computed, to ensure a small error. As space surveillance tracks thousands of objects at the same time, this approach is not practical and analytical solutions the better choice. Even if the NUTS software only will track one satellite, itself, and the time steps between two requests will be small, two or three tenths of a second, such analytical solutions are easier to implement and to use. One has namely not to tune the step size to balance the accuracy and computation speed, the accuracy of the model is known and readjustments of parameters are easy to perform.

For NUTS the *simplified general perturbations 4*, **SGP4**, model originally presented in the "Spacetrack Report no. 3" [14] was chosen for several reasons. It was also used in the successful UWE-3 CubeSat mission [9] and source code in C++, Fortran, Java, MATLAB and Pascal is available for free on the "CelesTrak" website [18] along with explanations. SGP4 is very popular because of the good compromise between accuracy and efficiency and the extremely large amount of object data, the *two-line element set*, **TLE**, published for all satellites on the "CelesTrak" website too.

2.5.1 Orbital Elements

Just like three position and three velocity components fully define an object's state, six orbital elements are required in order to fully define an object's orbit and thus to use an analytical

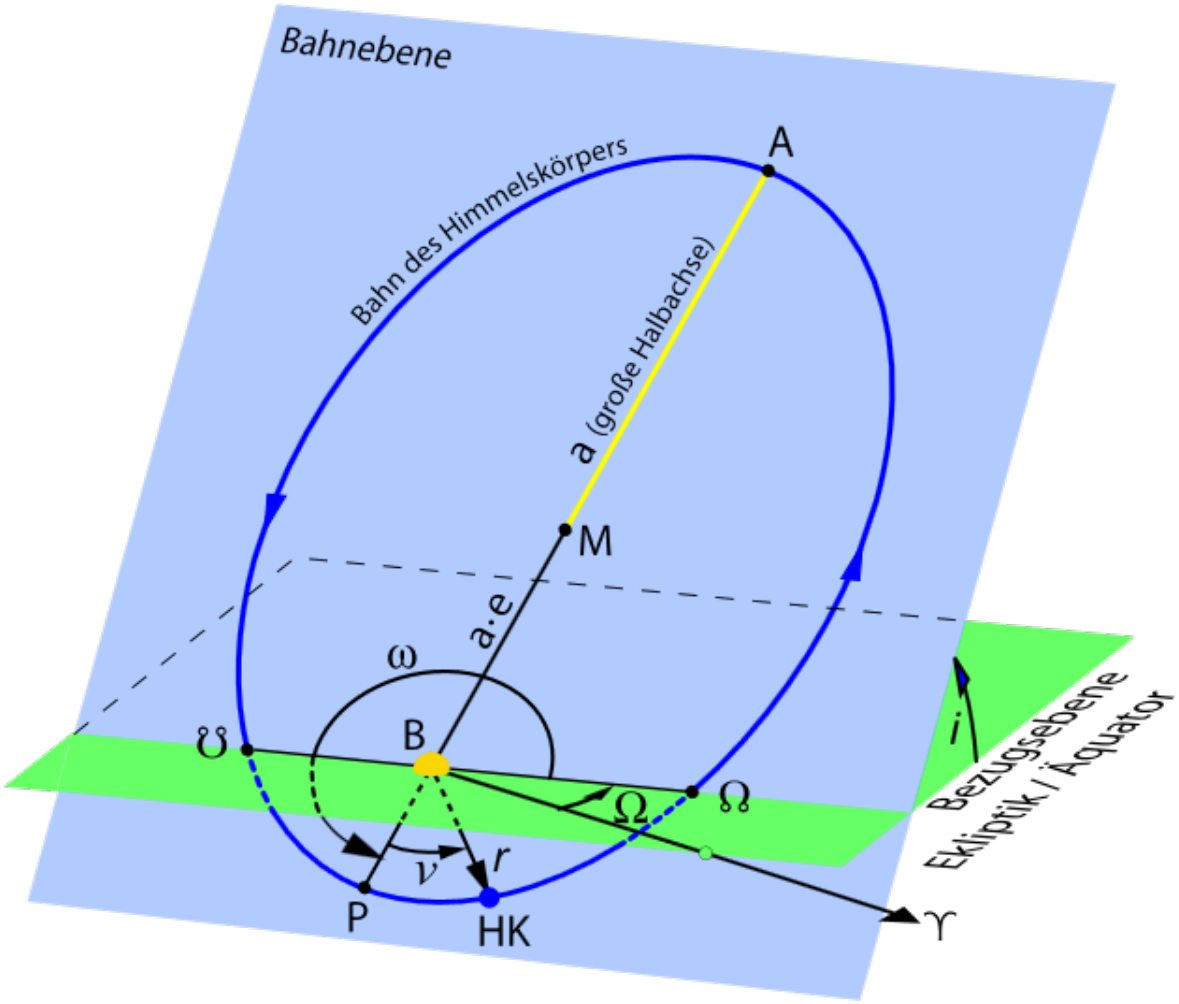


Figure 2.8: Geometry of an Orbit.
Green Plane: Fundamental Plane, *Blue Plane:* Orbit Plane, *Blue Line:* Orbit of Celestial Body, *A*: Apoapsis, *B*: Primary Focus, *HK*: Celestial Body, *M*: Center of the Ellipse, *P*: Periapsis, a: Semimajor Axis, $a \cdot e$: Half the Distance between Foci (Semimajor Axis multiplied with Eccentricity e), i: Inclination, r: Distance from Primary Focus to Celestial Body, ν: True Anomaly, Ω: Right Ascension of the Ascending Node, ω: Argument of Periapsis, ♈: Principal Direction, ☊: Ascending Node, ☋: Descending Node

orbit propagator.

The classical orbital elements, also known as *Keplerian elements*, are in general the set of choice for orbit definitions. **Figure 2.8**[8] shows the six elements as well as some other important points and parameters of elliptical orbits. The six classical orbital elements are

- The semimajor axis a, half the distance between apoapsis and periapsis (the furthest and nearest point of the orbit to the primary focus, the earth for earth orbiting objects).

[8] „BahnelementeEllipse“ von Modalanalytiker - Eigenes Werk. Lizenziert unter CC BY-SA 3.0 über Wikimedia Commons - `http://commons.wikimedia.org/wiki/File:BahnelementeEllipse.svg#mediaviewer/File:BahnelementeEllipse.svg`.

- The eccentricity e of the orbit.

$$e = \frac{c}{a} = \frac{\sqrt{a^2 - b^2}}{a} = \sqrt{2f - f^2}, \qquad 0 \leq e < 1$$

- The inclination i of the orbit plane to the fundamental plane, $0° \leq i < 180°$.
- The right ascension of the ascending node Ω, that is the angle from the principal direction to the ascending node, where the orbit crosses the fundamental plane from "below" (from south to north referring to the equator), $0° \leq \Omega < 360°$.
- The argument of periapsis ω, from the ascending node to the periapsis , $0° \leq \omega < 360°$.
- The true anomaly ν, that is the object's current angle from the periapsis, $0° \leq \nu < 360°$.

With these six elements a solution to *Kepler's problem*, the position change of an object in orbit after a certain amount of time, is possible in an ideal world with just the body at the primary focus, the celestial body in orbit and gravity as only force.

However there always are other objects like the sun, the moon and the planets as well as drag due to the outer layers of the atmosphere or solar winds. These perturbations must be taken into account and lead to models like the Simplified General Perturbations.

Two-Line Element Set

The SGP4 implementation does not take the classical elements as input but several quantities regrouped to the two-line element set. They differ slightly from the classical orbital elements and there are additional elements. It is very important to use only these elements when tracking satellites with simplified general perturbations models (and only with these models), the results will be erroneous otherwise. This is because the elements are generated with SGP models and have, like any other program, their own way to represent data and model the world.

The values are not updated regularly but rather just when needed. This can happen some times a day for manoeuvrable objects like the International Space Station, once or twice per week for low earth orbit satellite that don't manoeuvre like NUTS and more seldom for example for geostationary satellites that experience less atmospheric drag. Thousands of TLEs are published on the "CelesTrak" website [18]. Any new object is added as soon as its TLE is established and remains there for 30 days. Upon request this span is prolonged.

Two-line element sets are issued in a specific format consisting of two lines of 69 characters. The format is best explained with an example set. Consider the following TLE [36] (third and fourth line) and see **Table 2.1**[9] for the short explanation. The first two lines help to enumerate the 69 characters of each line.

```
0        1         2         3         4         5         6
12345678901234567890123456789012345678901234567890123456789012345678901
1 00005U 58002B    00179.78495062  .00000023  00000-0  28098-4 0  4753
2 00005  34.2682 348.7242 1859667 331.7664  19.3264 10.82419157413667
```

Some of the fields, like the majority of the fields in the second line, are self explanatory, others need additional comments [18].

1.2 & 2.2: Of course the satellite numbers of both lines must be the same.

1.4 - 1.6: These three fields together form the international designator, a unique name to the object assigned by the World Data Center-A for Rockets and Satellites.

1.7 & 1.8: This is the universal time epoch at which the orbital elements are referenced.

1.9, 1.10 & 2.8: Instead of the semimajor axis a, the TLE uses the mean motion n, the average angular rate of the object,

$$n = \sqrt{\frac{\mu}{a^3}} = \sqrt{\frac{Gm_\oplus}{a^3}} = \sqrt{\frac{6.673 \cdot 10^{-20} \frac{\mathrm{km}^3}{\mathrm{kgs}^2} \cdot 5.973\,332 \cdot 10^{24}\,\mathrm{kg}}{a^3}}$$

where μ is the gravitational parameter of the primary focus, the earth. The values in fields 1.9 and 1.10 are precisely $\frac{\dot{n}}{2}$ in $\frac{\text{revolutions}}{\text{day}^2}$ and $\frac{\ddot{n}}{6}$ in $\frac{\text{revolutions}}{\text{day}^3}$. However these two values are not used by the SGP4 model.

1.10 & 1.11: `+NNNNN-N` has to be read like $\pm$`0.NNNNN`$\cdot 10^{\pm\mathtt{N}}$, i.e. `28098-4` is $0.28098 \cdot 10^{-4}$.

1.11: The drag term is defined as $B^* = \frac{1}{2} \cdot \frac{C_D A}{m} \rho_0 R_\oplus$ and has the unit $\frac{1}{\text{earthradius}}$. C_D is the drag coefficient of the object, A its cross-sectional area and m its mass. ρ_0 is the reference atmospheric density and $R_\oplus$ the equatorial earth radius.

[9] '`N`': a figure or a space, '`A`': a letter or a space, '+': a plus sign, a minus sign or a space, '-': a plus sign or a minus sign, '`C`': '`U`' for unclassified data or '`S`' for secret data (of course, only unclassified data are publicly available).

Table 2.1: Two-Line Element Set Format Definition [18]

Field	Column	Description	Pattern[9]	Example
1.1	01	Line number	1	1
1.2	03-07	Satellite number	NNNNN	00005
1.3	08	Classification	C	U
1.4	10-11	International designator (last two digits of launch year)	NN	58
1.5	12-14	International designator (launch number of the year)	NNN	002
1.6	15-17	International designator (piece of the launch)	AAA	B
1.7	19-20	Epoch (last two digits of year)	NN	00
1.8	21-32	Epoch (day of the year and fractional day)	NNN.NNNNNNNN	179.78495062
1.9	34-43	First time derivative of the mean motion	+.NNNNNNNN	.00000023
1.10	45-52	Second time derivative of the mean motion	+NNNNN-N	00000-0
1.11	54-61	B^* drag term	+NNNNN-N	28098-4
1.12	63	Ephemeris type	N	0
1.13	65-68	Element number	NNNN	475
1.14	69	Checksum (modulo 10) (letters, blanks, periods, '+' = 0, '–' = 1)	N	3
2.1	01	Line number	2	2
2.2	03-07	Satellite number	NNNNN	00005
2.3	09-16	Inclination in degrees	NNN.NNNN	34.2682
2.4	18-25	Right ascension of the ascending node in degrees	NNN.NNNN	348.7242
2.5	27-33	Eccentricity (decimal point assumed)	NNNNNNN	1859667
2.6	35-42	Argument of perigee in degrees	NNN.NNNN	331.7664
2.7	44-51	Mean anomaly in degrees	NNN.NNNN	19.3264
2.8	53-63	Mean motion in $\frac{\text{revolutions}}{\text{day}}$	NN.NNNNNNNN	10.82419157
2.9	64-68	Revolution number at epoch	NNNNN	41366
2.10	69	Checksum (modulo 10)	N	7

1.12: The ephemeris type used to designate which model generated the TLE. Today only the SGP4 model generates TLEs and field 1.12 is always 0.

1.13: This number should correspond to the set number of the object and increase with one at each update. Unfortunately [18] points out that this is not always the case.

1.14 & 2.10: To calculate the checksum of a line, simply add all but the last digits together, ignore letters, blanks, periods and plus signs and finally add 1 for each minus sign. The last digit of the checksum should correspond to field 1.14 and 2.10 respectively.

2.7: The simplified general perturbations use the mean anomaly M instead of the true anomaly ν, see **Figure 2.9**[10]. While the true anomaly is the angle from the periapsis to the current position along the orbit, the mean anomaly is the angle around a circular orbit with radius a if the object travels at the constant mean motion speed, i.e. $M(t) = n(t - T)$ where T is the epoch.

2.9: This is the revolution the object is performing at the epoch. Revolution 1 begins the first time the object crosses the ascending node.

2.5.2 Simplified General Perturbations Models

As already said several times the simplified general perturbations models were originally presented in the "Spacetrack Report no. 3" [14] in 1980. The term "perturbation" in this context means that the two-body equation, that is the mathematical description of a satellite with negligible mass orbiting a far greater body (the earth or the sun) disregarding any other object or force beside gravity, is enlarged with a model of how the real world differs from this idealization. Like every model of the world, this also makes assumptions and simplifications but in general the results are reliable at least for some shorter time in the future.

The first simplified general perturbation model, **SGP**, has its origins in the 1960s and by this time only low earth orbiting objects were considered. Following some development time four newer, quite different and assumed more accurate models were released in the mentioned paper. Two of them, **SGP4** and **SGP8**, again only consider low earth orbits, but both of them have an extension model for deep-space named **SDP4** and **SDP8** respectively.

[10] "Kepler's equation scheme German" von Kepler's_equation_scheme.svg: AndrewBuckderivative work: René Schwarz (talk) - Kepler's_equation_scheme.svg. Lizenziert unter CC BY-SA 3.0 über Wikimedia Commons - http://commons.wikimedia.org/wiki/File:Kepler%27s_equation_scheme_German.svg#mediaviewer/File:Kepler%27s_equation_scheme_German.svg.

The term "deep-space" may be misleading today because it is actually just modelling further perturbations form medium earth to geosynchronous orbits and not meant for interplanetary missions. The SDP4 or SDP8 model should be used for orbit periods that are longer than 225 min, which is about 6000 km altitude. There the atmospheric drag becomes smaller than the perturbations due to the inhomogeneous gravity fields of the earth, the moon and the sun. The actual threshold value however is merely historically and empirically defined [18].

The description of the models in [14] is very short. The equations are listed with sparse comments and a Fortran translation is provided. Neither assumptions nor derivations are given. That's also the reason why the actual algorithm won't be repeated here. The interested reader is referred to the bibliography or the source code.

Of the five models only the SGP4 and SDP4 survived. The SGP was clearly inferior while the SGP8 and SDP8, although claimed better, especially when the tracked object attains special cases, apparently never came to use [36].

SGP4

The SGP4 and SDP4 models were both revised several times, the last time in "Revisiting Spacetrack Report #3: Rev 2" [36] issued in 2006. This paper merges both models together by applying the deep-space corrections only if needed (the period is known anyway, so it's not difficult to include or skip some steps) and publishes programs in several languages of which the MATLAB and C++ versions [18] are the basis for the NUTS implementation. This paper again reminds the important fact for consistency reasons to use the SGP4/SDP4 routines with the official two-line element sets only. Issues concerning reference frames, time formats and computer code are well retraced but again the actual algorithm is just provided as source code with sparse comments. A lot of parameters and equations are just declared and it's left to the user to try and understand what happens.

The central equation is *Kepler's equation*

$$\sqrt{\frac{a^3}{\mu}} = \frac{t-T}{E-e\sin E}$$

expressing Kepler's third law (t is the independent time variable, T the TLE epoch)

> «The square of the period of a planet is proportional to the cube of its mean distance to the sun.» [35]

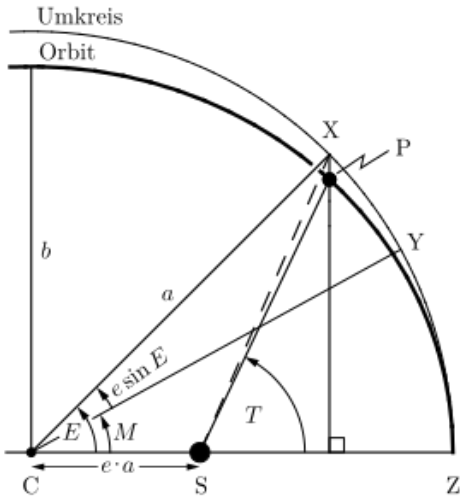


Figure 2.9: Geometry for Kepler's Equation.
Umkreis: Auxiliary Circle, C: Center of the Ellipse, E: Eccentric Anomaly, M: Mean Anomaly, P: Celestial Body, S: Primary Focus, T: True Anomaly, X and Y: Fictive Positions, Z: Periapsis, a: Semimajor Axis, $a \cdot e$: Half the Distance between Foci (Semimajor Axis multiplied with Eccentricity e), b: Semiminor Axis

in mathematical terms. This also explains why the TLE use the mean motion $n = \sqrt{\frac{\mu}{a^3}}$ and the mean anomaly $M = E - e \sin E$ to describe the orbit. The eccentric anomaly E is the angle at the center of the ellipse to an object on the auxiliary circle of this ellipse having the same perpendicular base on the semimajor axis. Figure 2.9 should make this clear.

"Solving" Kepler's equation is the usually iterative calculation of the time needed to travel between two points and is a central part of Kepler's problem (i.e. orbit propagation) which also involves Kepler's second law

> «The line joining the planet to the sun sweeps out equal areas in equal times.»
> [35]

expressed mathematically as

$$\frac{t - T}{A} = \frac{\mathscr{P}}{\pi a b}$$

with the period for one revolution $\mathscr{P}$ and the area A enclosed by the ellipse and the three points S, Z, and P in Figure 2.9.

This is basically what the SGP4 algorithm does. It only in addition includes several steps for the most important perturbations. With these calculations it finds the magnitude and direction of the position and velocity vectors of the satellite for any time difference prior or posterior to the TLE epoch.

Defining the position magnitude r_k, its first derivative with respect to time $\dot{r}_k$, a short pe-

riod periodic $r\dot{f}_k$[11], the right ascension of the ascending node Ω_k, the inclination i_k and the argument of latitude u_k (angle between ascending node and satellite position), that all have been found through several calculations involving the satellite parameters, the last steps of the SGP4 algorithm are [14]

$$\mathbf{u} = \begin{pmatrix} -\sin\Omega_k \cos i_k \\ \cos\Omega_k \cos i_k \\ \sin i_k \end{pmatrix} \sin u_k + \begin{pmatrix} \cos\Omega_k \\ \sin\Omega_k \\ 0 \end{pmatrix} \cos u_k$$

$$\mathbf{v} = \begin{pmatrix} -\sin\Omega_k \cos i_k \\ \cos\Omega_k \cos i_k \\ \sin i_k \end{pmatrix} \cos u_k - \begin{pmatrix} \cos\Omega_k \\ \sin\Omega_k \\ 0 \end{pmatrix} \sin u_k$$

$$\mathbf{r} = r_k \mathbf{u}$$

$$\dot{\mathbf{r}} = \dot{r}_k \mathbf{u} + r\dot{f}_k \mathbf{v}$$

where the k subscript denotes that the values changed from the TLE epoch (subscript 0) to the time for the prediction.

2.5.3 Assumptions

The NUTS implementation assumes in addition to the stated assumptions of the preceding sections, that a two-line element set for NUTS is available. TLEs for all satellites launched at longest 30 days ago are in the data base of the "CelesTrak" website [18] and the duration it keeps them, can be prolonged upon request. The difficulty with this assumption will be to send the TLE to NUTS in orbit. Before this happens the detumbling part of the attitude control will work, but the estimator will deliver erroneous values to the attitude controller, that thus will not be able to work properly. Remember that no orbit position prediction equals no geomagnetic reference vector and a wrong EKF model.

As long as no updated TLE is published one can assume the current values to be correct.

All deep-space related parameters and calculations have been removed from the source code. Therefore only positions and velocities for satellites with less than 225 min revolution time can accurately be predicted. NUTS is constructed to have a low earth orbit, between

[11] This combined variable is unfortunately not nearer defined in the original source [14]. While r obviously is the position magnitude, f and its first time derivative $\dot{f}$ remain unclear. The revised code [36] calls this short period periodic variable `rvdot` ($r\dot{v}_k$) instead which could mean velocity magnitude v.

160 and 2000 km altitude or orbital periods of 90 to 130 min, for which the SGP4 model is intended. There is thus no constraint from this side.

2.5.4 Implementation Details

The starting point for the orbit position and velocity predictor for NUTS is the SGP4/SDP4 implementation published in [36] and available for free in several languages on the internet [18]. This implementation merges the basic SGP4 and the deep-space extension model SDP4 together and is intended to be used to track a lot of objects at the same time.

After removing all unnecessary parts of the code, that is

- All parameters and routines for deep-space objects.
- Unofficial additions to the TLE to specify time points for the desired positions.
- The ability to track several satellites at the same time using one data structure.
- Input and output interaction with the user.

and adjusting the implementation to the interface (ECI frame, UTC time), three functions, beginning with the "orbit" keyword, declared and defined in the files `orbitPred.h` and `orbitPred.c`, remain. The header file also defines the `typedef struct Satrec` that saves all the relevant satellite information to perform the prediction. In detail it contains

- One char array `error[128]` where errors and warning messages are written.
- One char, one char array, five integers and ten double precision floating point numbers that are a direct translation of the two-line element set.
- One integer and 26 double precision floating point numbers that are additional parameters derived from the TLE data.

The two functions

- `Satrec orbitTwoline2rv(char tle[139])`
- `void orbitSgp4init(Satrec *satrec)`

do the initialization work. `orbitTwoline2rv` converts as its name indicate the two-line element set to a data structure containing the same information easier to treat on a computer, this is the place where the `Satrec` struct is instantiated.
The two-line element set has to be provided as input char array *not* formatted as two lines like in the example TLE of Table 2.1. The `char tle[139]` must contain just all 138 characters and the string ending sign, make sure that there is no newline or return sign between the last character of the first line and the first character of the second line.
The first thing the functions does is then to check that the length of the char array is long enough for the processing. If not a message is written in the `error` field of the struct that is returned and the function stops. This is the only place where an error causes the function to stop. All other errors are written in the `error` field too, but don't produce a C error that would cause the function to crash, thus the calculations continue, but the outputs will probably be wrong. So it is recommended to check for error messages each time a new TLE is processed.
If the input array is long enough, the function translates it to the mentioned fields, to be precise:

char: Classification

char array: International designator

five integers: Satellite number, Year of epoch, Ephemeris type, Element number and Revolution number

ten floating point numbers: Day of epoch, First derivative of the mean motion (converted to $\frac{\text{rad}}{\text{min}^2}$), Second derivative of the mean motion (converted to $\frac{\text{rad}}{\text{min}^3}$), B^* drag term, Inclination (converted to rad), Right ascension of the ascending node (converted to rad), Eccentricity, Argument of perigee (converted to rad), Mean anomaly (converted to rad) and Mean motion (converted to $\frac{\text{rad}}{\text{min}}$)

Before the function continues with the initialization of additional satellite values, six simple tests on the TLE are done to detect possible errors.

1. Assert that first line number is '1'.
2. Assert that the second line number is '2'.
3. Assert that the classification is 'U'.

4. Assert that both satellite numbers (field 1.2 and 2.2) are the same.

5. Calculate the first line checksum and compare with field 1.14.

6. Calculate the second line checksum and compare with field 2.10.

Even if one or more of the tests fail, the function will not stop, only a message will be written to the `error` field of the `satrec` struct. This is because the algorithm will not crash, but a fail is a strong indication that the orbit prediction is going to have errors.

The function `orbitTwoline2rv` gives a pointer `*satrec` to the `orbitSgp4init` function as last step. This function fills the other 26 floating point variables of the struct with values, for example the squared eccentricity, sine and cosine values of the inclination and coefficients derived from B^*. The epoch as Julian date and the time at which the SGP4 algorithm shall compute the velocity and position in minutes, are two fields too. Of course the latter is 0 min during the initialization function and will first be used in the actual predictor.
`orbitSgp4init` also computes the perigee of the satellite. If it is less than 220 km the integer `isimp` of `Satrec`, which actually is a boolean variable, is set from the default 0 (False) to 1 (True). This done, not all fields are initialized and some steps of the algorithm will be skipped because some minor perturbations are irrelevant at such low altitudes.
When this function is finished the calling function `orbitTwoline2rv` returns the new `Satrec` struct which now is ready for the orbit propagation.

The actual orbit position and velocity prediction is done by the function

- `void orbitSgp4(Satrec *satrec, Time timeUTC, double reci[3], double veci[3])`

that returns the position and the velocity vectors `reci[3]` and `veci[3]` of the object `*satrec` points to for the UTC time point `timeUTC`. Just a few, small changes in the actual predictor steps were done to the original source code, nearly all of them concern the deep-space extension (tests on the conditions for deep-space and subsequent further calculations) that simply was removed.
As for a lot of functions in the NUTS package, the most important changes concern the interface. Instead of providing the time difference in minutes, positive or negative, from the TLE epoch saved in `*satrec`, the user gives the UTC time as `Time` struct (see section 2.2) and the function computes the time difference. Likewise the position and velocity vectors are

transformed from the TEME to the ECI frame before they're returned (see `frameTeme2eci` in section 2.3).

The other important change is to provide a pointer to the `Satrec` struct that contains the satellite parameters instead of the struct itself. The reason is to have the same possibility of error messages in the `error` char array of the struct than in the initialization functions. At four places conditions that do not result in meaningful output when fulfilled, are checked. These are legacies of the source code copied from [18] that is more intended to track several satellites at once, where input errors can be more frequent. In normal operation and assuming that the TLE was successfully transferred to NUTS, they should never yield positive responses. If so the position and velocity arrays will be filled with zeros in the first three cases.

1. Soon after the beginning the mean motion is read out of the `*satrec` pointer. If it happened to be negative a wrong TLE must have been sent to the satellite, because $n = \sqrt{\frac{\mu}{a^3}}$ cannot be negative.
   It is very unlikely that this ever will happen, as such an error most probably also would yield to a wrong checksum in `orbitTwoline2rv`.

2. Directly after this, the second condition looks on the eccentricity (from the TLE) and semimajor axis (derived). The condition states that the eccentricity has to be between -0.001 and 1 and the semimajor axis larger than 0.95 times the equatorial earth radius, `REQU` in Table 2.3, to be in the valid region of the SGP4 model.
   Again this is unlikely to happen especially for the eccentricity (NUTS is intended to have a nearly circular orbit) if the TLE is transmitted correctly.

3. One of the derived quantities more in the middle of the program is the semilatus rectum $p = \frac{b^2}{a}$, also known as semiparameter. This is the distance from the primary focus to the orbit perpendicular to the semimajor axis, in other words the line $\overline{B\wp}$ in Figure 2.8. This test is to ensure its positiveness.
   A fail of this condition is also an obvious case of wrong data, but the error reason would be more difficult to find, as its involves a lot of steps and parameters.

4. The final test is actually done after the whole algorithm, so even if it occurs `reci[3]` and `veci[3]` can be different from zero.
   The position vector is calculated as magnitude and direction before it is expressed in

the TEME reference frame (and later rotated to the ECI frame). A magnitude smaller than one earth radius means obviously that the satellite has decayed, and this would be fatal for the implementation presented here, because it will run on the satellite. However if a calculation or an algorithmic error results in a smaller magnitude than the earth radius, while the satellite still is in orbit, it is better not to return zero because the direction of the position vector might still be right and at least one piece of information useful.

The SGP4 algorithm is surprisingly short in terms of lines of code, 650 for the three functions including the long introductory comments of each function, and running time because there is no iteration as a numerical integrator would require. The program assignments are for the most additions and multiplications and occasionally trigonometric function calls. It is always taken care to save the trigonometric function results to minimize the number of times these longer calculations need to be done.

2.5.5 Orbit Position and Velocity Prediction User's Guide

The orbit position is not needed directly in the attitude estimator and attitude controller for NUTS, but the estimator needs the reference vector of the earth's magnetic field which of course has to be predicted at the satellite's position. The orbit position and velocity can be used for the gyroscope measurement adjustment of equation (2.8) in the controller instead of the calculation based on the balance of gravity and centrifugal force. Therefore the right usage of the predictor is important and will, as the sun vector predictor from section 2.4, be explained with an example. It is taken from [35] on page 234 (no example number) and executed as shown in **Listing 2.2**. The NUTS software prints

```
«satrec.error after initialization:  ''
satrec.error after SGP4:  ''
reci:  -9059.921859 4659.702830 814.126503 km
veci:     -2.233430   -4.110207   -3.157239 km/s»
```

which is very close to the values in the book

$$\mathbf{r}^i = \begin{pmatrix} -9059.941\,3786 \\ 4659.697\,2000 \\ 813.958\,8875 \end{pmatrix} \text{km}, \qquad \dot{\mathbf{r}}^i = \begin{pmatrix} -2.233\,348\,094 \\ -4.110\,136\,162 \\ -3.157\,394\,074 \end{pmatrix} \frac{\text{km}}{\text{s}}$$

Listing 2.2: Orbit Position and Velocity Prediction Example

```
...
#include "orbitPred.h"
...
int main(){
    // TEME example from Vallado (2013) Fundamentals of Astrodynamics
       and Applications p. 234
    char tle[139] = "1 00005U 58002B   00179.78495062  .00000023
       00000-0  28098-4 0  47532 00005  34.2682 348.7242 1859667
       331.7664  19.3264 10.82419157413667";
    Satrec satrec = orbitTwoline2rv(tle);
    printf("satrec.error after initialization: '%s'\n", satrec.error);
    Time time;
    time.year = 2000;
    time.mon = 6;
    time.day = 30;
    time.hr = 18;
    time.min = 50;
    time.sec = 19.733571;
    double reci[3];
    double veci[3];
    orbitSgp4(&satrec, time, reci, veci);
    printf("satrec.error after SGP4: '%s'\n", satrec.error);
    printf("reci: %f %f %f km\n", reci[0], reci[1], reci[2]);
    printf("veci: %f %f %f km/s\n", veci[0], veci[1], veci[2]);
    return 0;
}
```

The reason for the small differences is found in the time and frame transformations. It is e.g. assumed that the coordinated universal time is the only universal time implementation and some other functions actually are not valid for the time of the example that already lies 15 years in the past.

The listing shows the most important things to consider. Even if it's not easy to see, the char array that is saved in `tle[139]` has no line breaks, just the 138 TLE characters enclosed by quotation marks. The `satrec` struct is then returned by calling the `orbitTwoline2rv` function to translate the TLE. The `orbitSgp4init` function is called internally and should not be needed explicitly by the user. It is recommended to check for error messages each time a new TLE is translated and each time a new prediction is computed, the first check being especially important. The usage regarding the time input and the two output vectors should be obvious from Listing 2.2 and is in accordance to the interface used throughout.

A very important task for the future users will be to provide the two-line element set to the satellite in orbit via the radio uplink. The actual place to save the char array has not been

defined so long, for the tests local variable arrays have been used. The "CelesTrak" website [18] will list the TLEs for NUTS automatically upon release some time after the launch. To maintain the updates online a request not to omit them after 30 days must be sent.

As final comment on this section, one could suggest to use the orbit position vector to improve the sun vector prediction, that as said in section 2.4 returns the vector from the center of the earth to the sun. A simple vector addition will return the vector from the satellite position to the sun. This is not implemented in the predictor software for NUTS for the moment because after a unit conversion (1 AU = 149 597 870.7 km) the magnitude of the sun vector is so much higher, that no significant change occurs.

2.6 Geomagnetic Field Prediction

Now that the position of the satellite is known, the magnetic flux density at this point can be predicted to have the second reference vector of the EKF model, $\mathbf{r}_m^i$ in equation (2.2). Again the term "prediction" is used to mark the difference with the magnetic vector returned by the magnetometer.

In contrast to the other predictions not one but two different predictors are part of the NUTS software for the moment. This is because there exist two models of the earth's magnetic field, the *international geomagnetic reference field*, **IGRF**, and the *world magnetic model*, **WMM**, that according to the online calculator for both models [33] and its "Frequently Asked Questions" [32] are of comparable quality. They also have similar numbers of lines of code, coefficients to save and running time (at least on a modern personal computer).

Just one of them will finally be part of the NUTS software package but both have been implemented in this thesis to find out which one suits the project's needs best. The main decision criterion being the algorithm running time on a microcontroller. The last subsection on the geomagnetic field prediction gives arguments why the WMM is the better choice.

2.6.1 The Earth's Magnetic Field

The earth's magnetic field, also called the *geomagnetic field*, surrounds and penetrates the earth and protects it from the solar winds. Its emergence is still not completely understood, but the generally accepted theory for the main source is that of the *geodynamo* or *dynamo mechanism of the earth* [10]. The electrical conductivity of the solid inner core of

earth, mainly composed of iron, cannot solely be the source of the field because of its very high temperature (iron above 700 °C cannot be magnetized permanently). The mechanism is rather located in the liquid outer core, that is colder and made out of iron too. There is a constant flow in the iron due to convection currents, that is rising and falling of fluid at different temperatures and densities. But the earth itself is rotating and therefore the flow is not straight but helical. As **Figure 2.10**[12] depicts, circulating electrical currents arise because of this and, like every electric current, generate a magnetic field. The complexity of the convection flow causes the magnetic field to change both with time and place. It has even changed its polarity several times, the last time about 750 000 to 780 000 years ago [32]. In addition to this *core field* other contributions to the *inner field* are due to magnetic materials in the mantel and crust. This is the *crustal field* that does nearly not change with time.
The *external field* varies a lot faster but is in general weak, although high flux densities may occur from time to time. It comes from the charged particles moving chaotically at high speeds in the atmosphere [3].

On the surface of the earth the magnetic flux density is between 25 000 and 65 000 nT and is not aligned with the earth-centered earth-fixed reference frame. It is easiest to give a picture of the surface field with maps showing the total intensity, **Figure 2.12**[13], the declination, that is the angle between a compass needle and the ECEF north pole, **Figure 2.13**[14], and the inclination, that is if the compass needle points upwards or downwards, **Figure 2.14**[15]. Note that these maps are only a model, the latest world magnetic model to be exact, of the slowly changing core field which is responsible for more than 90 % of the total field. Likewise maps for other dates differ from the maps shown here, of course more the larger the time gap with January 2015 is.

On a cosmical scale the *magnetosphere* is formed as shown in **Figure 2.11**[16]. The left side of the figure shows the side that is turned towards the sun, there the magnetosphere extends to approximately ten earth radii. On the other side the field can be measured up

[12] "Outer core convection rolls" by United States Geological Survey - http://geomag.usgs.gov/images/faq/Q6.jpg. Licensed under Public Domain via Wikimedia Commons - `http://commons.wikimedia.org/wiki/File:Outer_core_convection_rolls.jpg#mediaviewer/File:Outer_core_convection_rolls.jpg`.

[13] Maps taken from [34].

[14] Maps taken from [34].

[15] Maps taken from [34].

[16] "Magnetosphere Levels" by Magnetosphere_Levels.jpg: Dennis Gallagher derivative work: Frédéric MICHEL - Magnetosphere_Levels.jpg. Licensed under Public Domain via Wikimedia Commons - `http://commons.wikimedia.org/wiki/File:Magnetosphere_Levels.svg#mediaviewer/File:Magnetosphere_Levels.svg`.

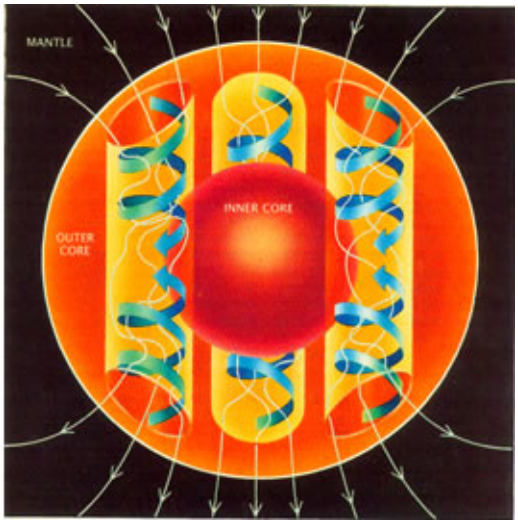


Figure 2.10: Illustration of the dynamo mechanism that creates the earth's magnetic field. Convection currents of magma in the earth's outer core, driven by heat flow from the inner core, organized into rolls by the Coriolis force, create circulating electric currents, which generate the magnetic field.

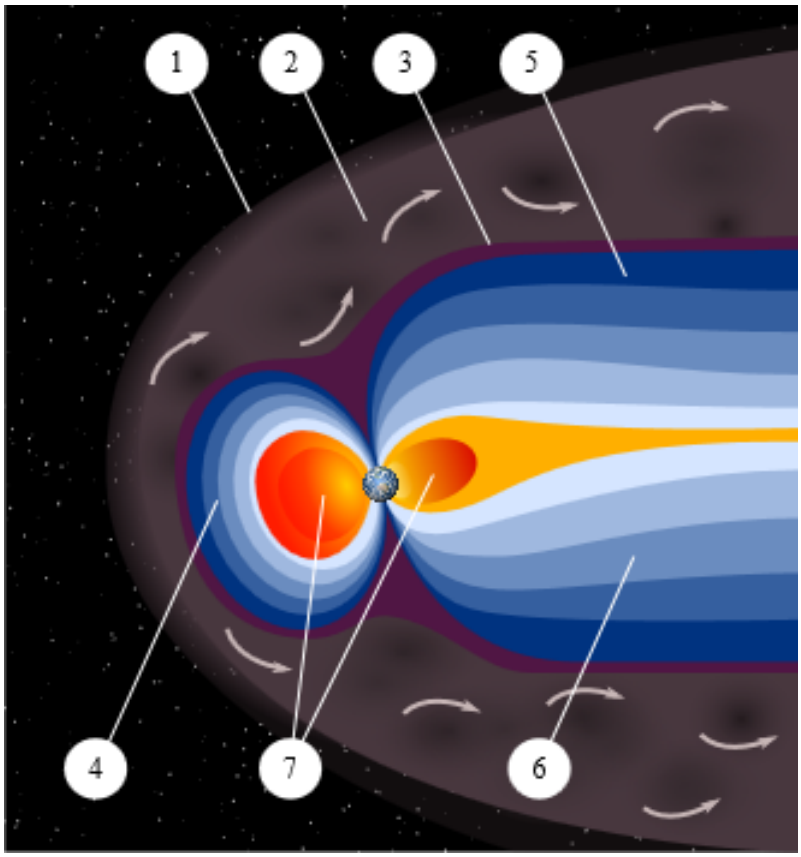
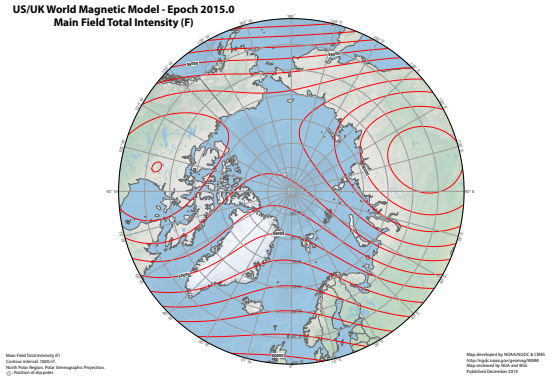
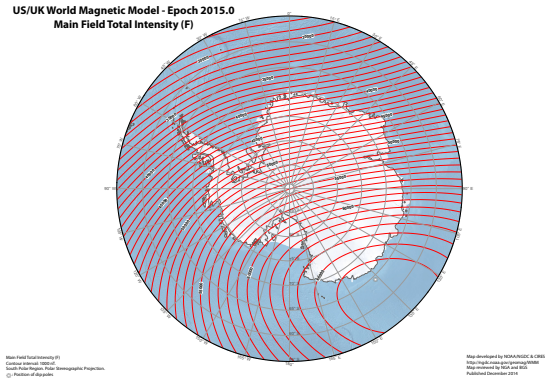


Figure 2.11: An artist's rendering of the structure of a magnetosphere. *1:* Bow Shock, *2:* Magnetosheath, *3:* Magnetopause, *4:* Magnetosphere, *5:* Northern Tail Lobe, *6:* Southern Tail Lobe, *7:* Plasmasphere





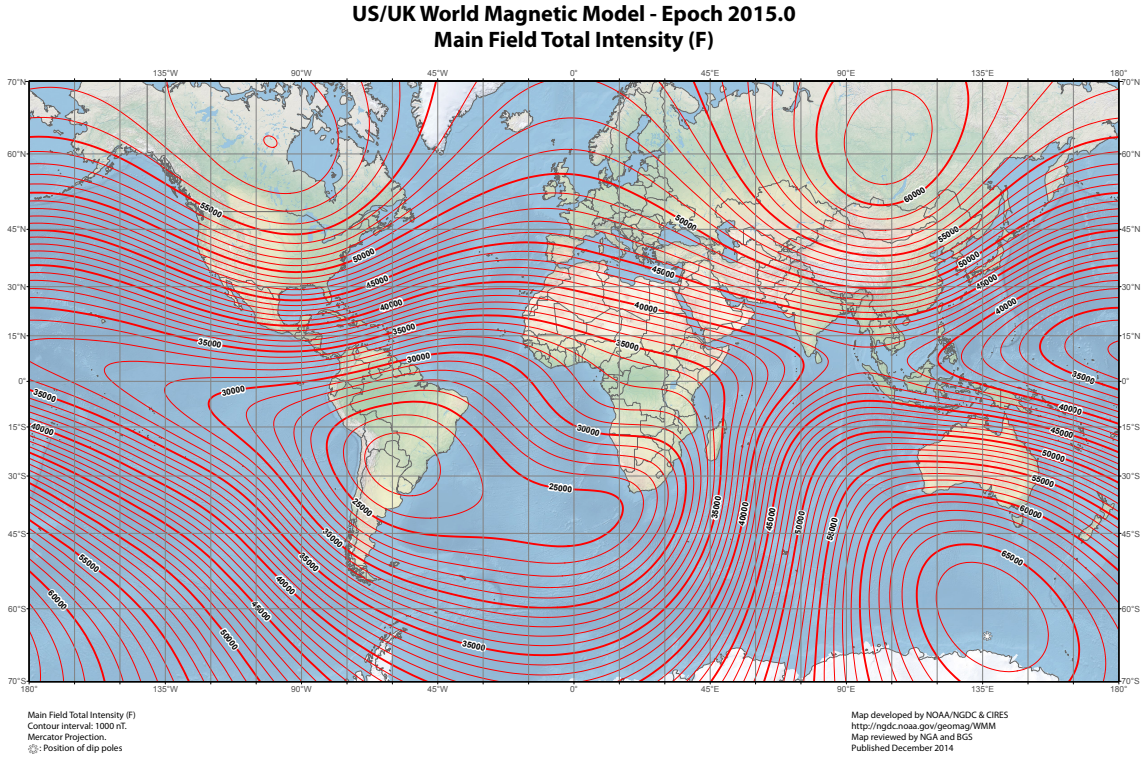
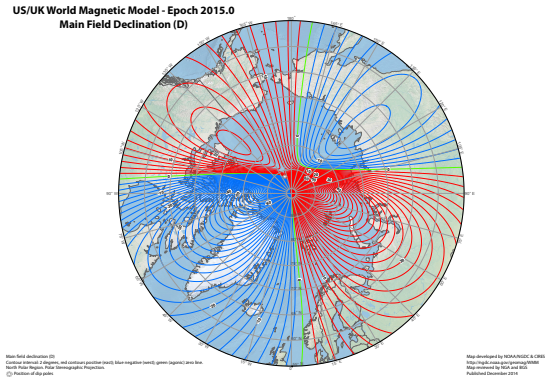
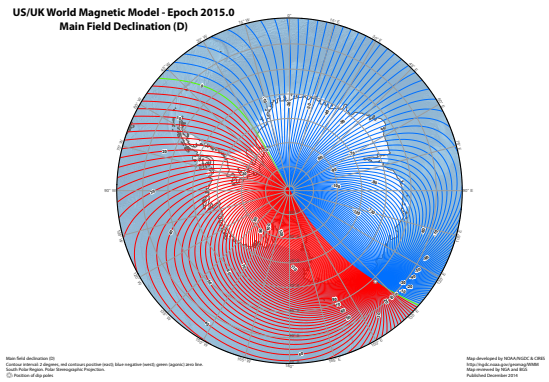


Figure 2.12: Main Magnetic Field Total Intensity modelled by the World Magnetic Model 2015 for the time and date 01/01/2015 00:00:00.
The strongest intensity is between the Antarctic and Australia as well as over central Russia and Canada. The lowest intensity is over South America and the South Atlantic Ocean.





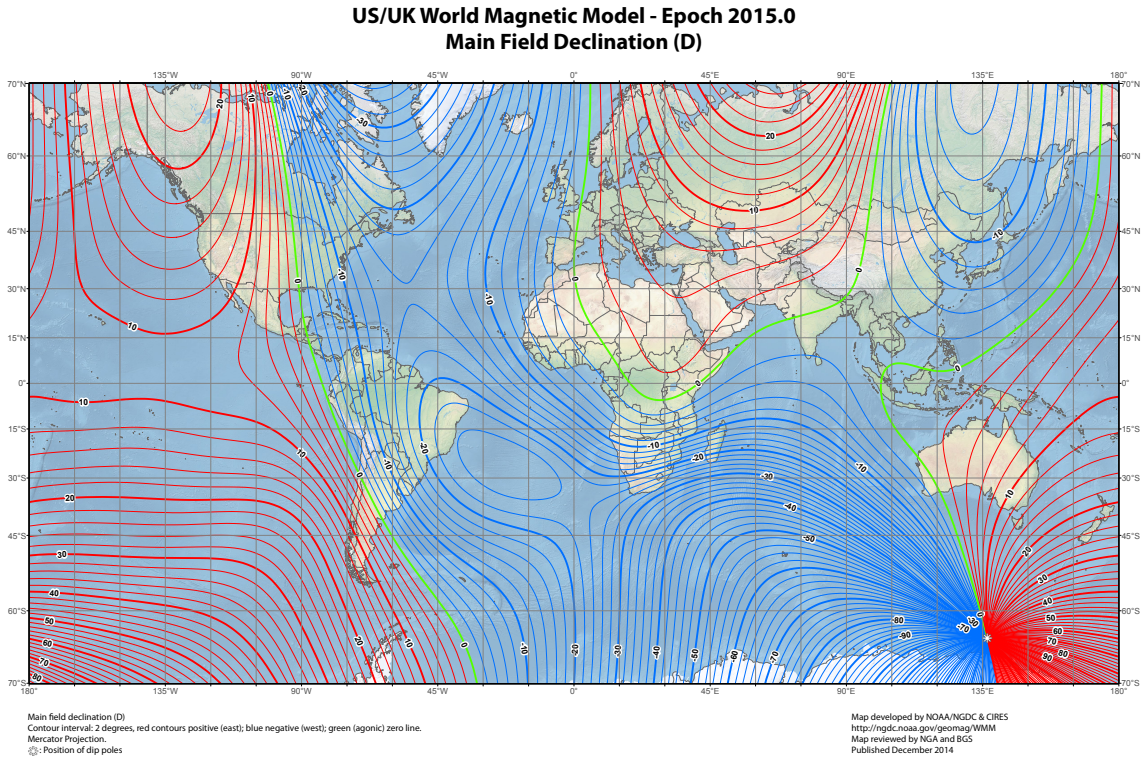
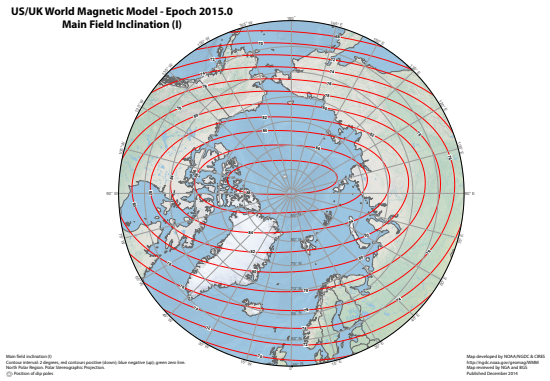
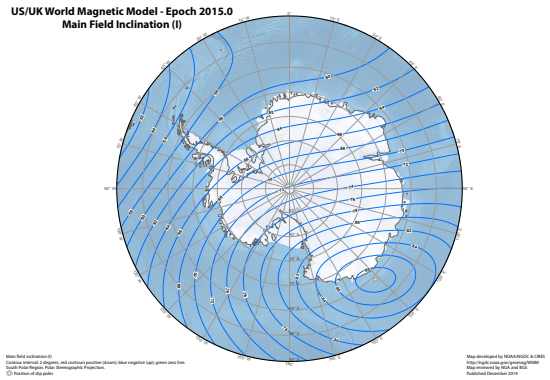


Figure 2.13: Main Magnetic Field Declination modelled by the World Magnetic Model 2015 for the time and date 01/01/2015 00:00:00.
Red means positive (east) and blue negative (west) declination. The declination is zero on the green lines.





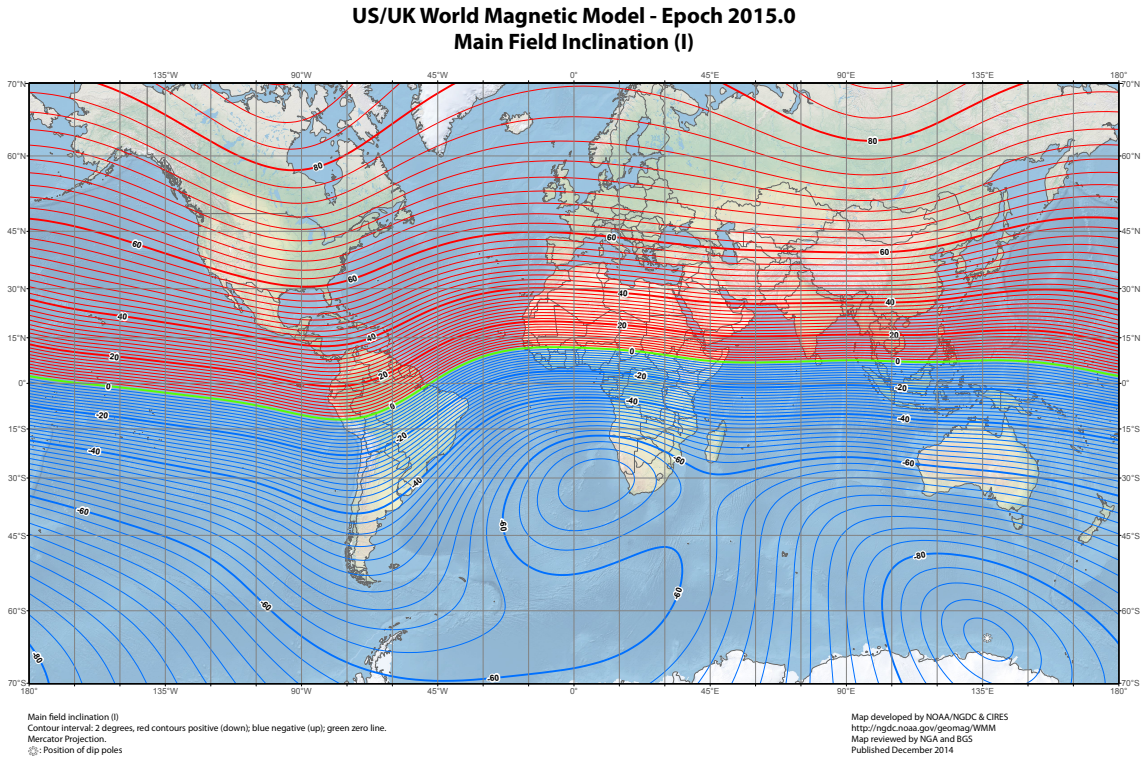


Figure 2.14: Main Magnetic Field Inclination modelled by the World Magnetic Model 2015 for the time and date 01/01/2015 00:00:00.
The green line is the magnetic equator (no inclination), while red means positive (down) and blue negative (up) inclination.

to more than one hundred earth radii [32]. Luckily NUTS is a low earth orbit satellite and thus so close to the surface that the asymmetry of the magnetosphere due to the sun has no measurable impact. This allows to use easier models of the magnetic field that just depend on the earth-fixed position and time.

2.6.2 Modelling the Earth's Magnetic Field

There is a need for models of the earth's magnetic field as one cannot simply place magnetometers everywhere. Depending on the application different model assumptions and degrees (number of harmonic wavelengths considered) lead to several geomagnetic field models. The most popular ones are the IGRF and WMM that will be explained with more details soon. They only account for the core field and have a low degree, so if higher precision is demanded, higher degree models should be used.

The natural extension to the WMM is the *enhanced magnetic model*, **EMM** [30], which models the crustal field on top of the core field. It extends the number of spherical harmonics from 12 to 720. The ability to run the program on a microcontroller in reasonable time is therefore surely limited. The *high definition geomagnetic model*, **HDGM** [31], adds predictions for the external field to the EMM and is consequently even more complicated. It is e.g. intended to help high precision drilling. This model completely overgrows the needs for NUTS and has other drawbacks like the yearly updates (instead of every five years for the WMM and EMM) and the price.

Beside these *predictive* models that attempt to foresee the earth's magnetic field, there are *historic* models relating the field direction and strength of the past. Each new IGRF generation for example includes a new *definitive* or DGRF model for the past five years that is added to the field database. This database begins with estimates for the early 20th century and has then the definitive models from 1945 to the present. Nonetheless even these models will not equal the output of magnetometers done at the same time and place, DGRF and consorts remain models of the true magnetic field. Likewise should the predictions obviously be treated with care especially by the end of the validity period.

The International Geomagnetic Reference Field

The international geomagnetic reference field model is issued by the International Association of Geomagnetism and Aeronomy, **IAGA**, and updated every five years. The current

model is the 12th generation IGRF [15] that contains estimates for the years 1900 to 1945 (IGRF1900, IGRF1905, etc. to IGRF1940), the definitive model for the years 1945 to 2015 (DGRF1945, DGRF1950, etc. to DGRF2010) and the predictive model that is valid for 2015 to 2020 (IGRF2015 and secular variation for 2015 to 2020), the part of the 12th generation IGRF that is interesting for NUTS. However the model structure and equations do not change from the historic to the predictive models, just the Gaussian coefficients are not the same for each five year range.

The magnetic flux density **B** modelled by the IGRF accounts only for the core field which contributes to about 90 to 95 % of the total field [32]. It is calculated as the negative gradient of a scalar potential V in geocentric coordinates (r,θ,λ), see Figure 2.3,

$$\mathbf{B} = -\nabla V(r,\theta,\lambda,t) = -\frac{\partial V}{\partial r}\mathbf{e}_r - \frac{1}{r}\frac{\partial V}{\partial \theta}\mathbf{e}_\theta - \frac{1}{r\sin\theta}\frac{\partial V}{\partial \lambda}\mathbf{e}_\lambda \tag{2.9}$$

The scalar potential V is given by the series [7]

$$V(r,\theta,\lambda,t) = r_{\text{mean}} \sum_{n=1}^{N} \left(\frac{r_{\text{mean}}}{r}\right)^{n+1} \sum_{m=0}^{n} (g_n^m(t)\cos m\lambda + h_n^m(t)\sin m\lambda) P_n^m(\cos\theta) \tag{2.10}$$

where n is the *degree* (maximum N) and m is the *order* for the time-varying Gaussian coefficients $g_n^m(t)$ and $h_n^m(t)$ as well as the Schmidt quasi-normalized associated Legendre functions $P_n^m(\cos\theta)$. r_{mean} is the mean earth radius, see Table 2.3, and should not be confounded with the semimajor axis or equatorial radius a when reading [7].

Function (2.10) is a spherical harmonic representation of the magnetic potential. Spherical harmonics are the natural extension of Fourier series in 3 dimensions. In the same manner that Fourier series are used to represent functions on a circle, spherical harmonics are defined on the surface of a sphere and are an instrument to solve differential equations, this because they're homogeneous solutions to Laplace's equation $\nabla^2 V = 0$. **Figure 2.15**[17] shows the three first degrees of spherical harmonic functions.

Inserting equation (2.10) in equation (2.9) yields the magnetic flux density components

[17] "Spherical Harmonics" by Inigo.quilez - Own work. Licensed under CC BY-SA 3.0 via Wikimedia Commons - `http://commons.wikimedia.org/wiki/File:Spherical_Harmonics.png#mediaviewer/File:Spherical_Harmonics.png`.

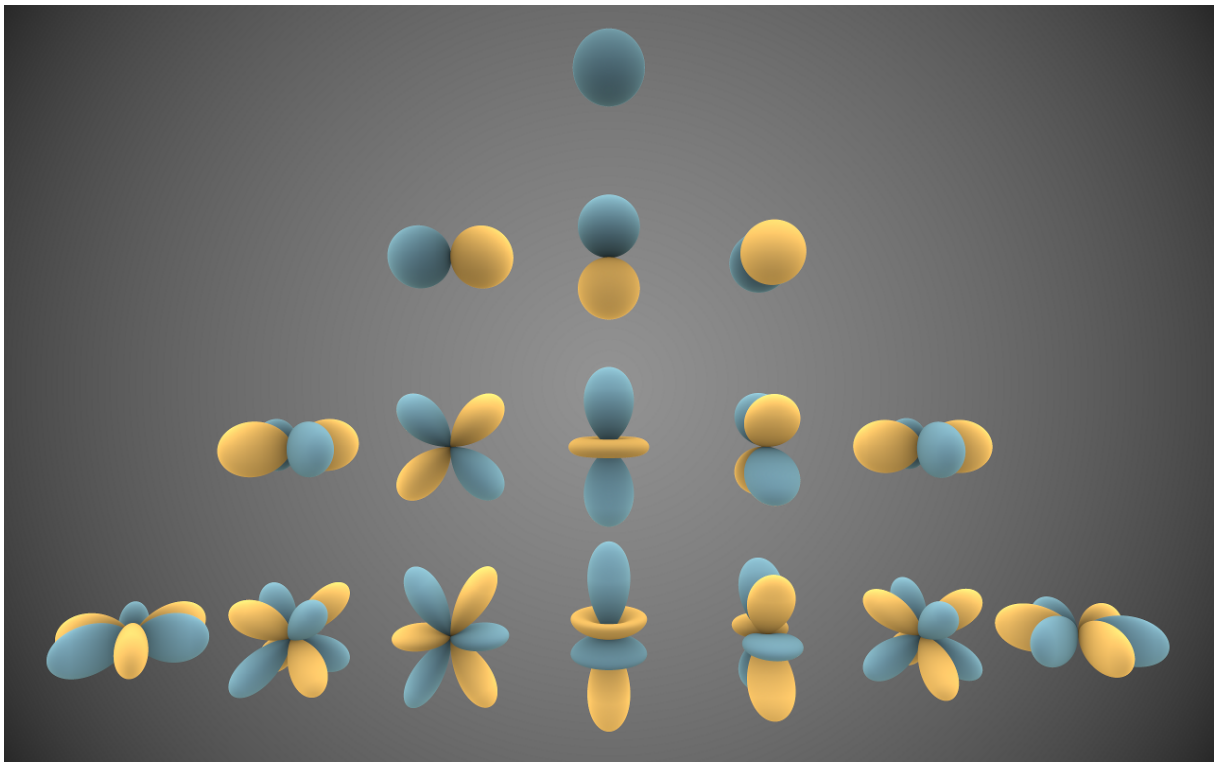


Figure 2.15: Visual representations of the first few real spherical harmonics.
Blue portions represent regions where the function is positive, *yellow* portions represent where it is negative. The *distance* of the surface from the origin indicates the magnitude of the spherical harmonic.

in geocentric coordinates

$$B_r = \sum_{n=1}^{N} \left(\frac{r_{\text{mean}}}{r}\right)^{n+2} (n+1) \sum_{m=0}^{n} (g_n^m(t)\cos m\lambda + h_n^m(t)\sin m\lambda) P_n^m(\cos\theta) \tag{2.11a}$$

$$B_\theta = -\sum_{n=1}^{N} \left(\frac{r_{\text{mean}}}{r}\right)^{n+2} \sum_{m=0}^{n} (g_n^m(t)\cos m\lambda + h_n^m(t)\sin m\lambda) \frac{\partial P_n^m}{\partial\theta}(\cos\theta) \tag{2.11b}$$

$$B_\lambda = \frac{-1}{\sin\theta} \sum_{n=1}^{N} \left(\frac{r_{\text{mean}}}{r}\right)^{n+2} \sum_{m=0}^{n} m(-g_n^m(t)\sin m\lambda + h_n^m(t)\cos m\lambda) P_n^m(\cos\theta) \tag{2.11c}$$

As one can see, the components are only dependent on the geocentric coordinates. When looking at Figure 2.11 it is obvious that the sun's position plays a major role for the shape of the magnetosphere and thus of the magnetic flux density at a specific point and time, especially when the place is far away from the surface as it is the case for satellites. Unfortunately there seems not to be an altitude validity range for the IGRF. However due to the similarities with the WMM the same upper validity limit of 850 km should hold. This complies with the specification of low earth orbits and the IGRF can be used as predictor for NUTS.

Legendre functions are a generalization of *Legendre polynomials* $P_n(x)$ to non integer de-

gree. *Associated* Legendre polynomials $P_{n,m}(x)$ are related to them with the following equation [7]

$$P_{n,m}(x) = (1-x^2)^{\frac{m}{2}} \frac{\mathrm{d}^m P_n}{\mathrm{d}x^m}(x)$$

The Schmidt quasi-normalized associated Legendre polynomials $P_n^m(x)$ are then obtained by applying the Schmidt quasi-normalization

$$P_n^m(x) = \sqrt{\frac{2(n-m)!}{(n+m)!}} P_{n,m}(x)$$

However it is computationally more efficient to use the Gaussian normalization

$$P^{n,m}(x) = \frac{2^n!(n-m)!}{(2n)!} P_{n,m}(x)$$

because the Gaussian normalized associated Legendre functions and their derivatives for a sine or cosine input are easy to compute recursively [4]

$$P^{0,0}(\cos\theta) = 1\,, \qquad P^{1,1}(\cos\theta) = \sin\theta \tag{2.12a}$$

$$P^{n,n}(\cos\theta) = \sqrt{1-\frac{1}{2n}}\sin\theta P^{n-1,n-1}(\cos\theta) \tag{2.12b}$$

$$P^{n,m}(\cos\theta) = \frac{2n-1}{\sqrt{n^2-m^2}}\cos\theta P^{n-1,m}(\cos\theta) - \sqrt{\frac{(n-1)^2-m^2}{n^2-m^2}} P^{n-2,m}(\cos\theta) \tag{2.12c}$$

$$\frac{\partial P^{0,0}}{\partial\theta}(\cos\theta) = 0\,, \qquad \frac{\partial P^{1,1}}{\partial\theta}(\cos\theta) = \cos\theta \tag{2.12d}$$

$$\frac{\partial P^{n,n}}{\partial\theta}(\cos\theta) = \sqrt{1-\frac{1}{2n}}\left(\sin\theta\frac{\partial P^{n-1,n-1}}{\partial\theta}(\cos\theta) + \cos\theta P^{n-1,n-1}(\cos\theta)\right) \tag{2.12e}$$

$$\begin{aligned}\frac{\partial P^{n,m}}{\partial\theta}(\cos\theta) = {} & \frac{2n-1}{\sqrt{n^2-m^2}}\left(\cos\theta\frac{\partial P^{n-1,m}}{\partial\theta}(\cos\theta) - \sin\theta P^{n-1,n-1}(\cos\theta)\right) \\ & - \sqrt{\frac{(n-1)^2-m^2}{n^2-m^2}}\frac{\partial P^{n-2,m}}{\partial\theta}(\cos\theta)\end{aligned} \tag{2.12f}$$

The Gaussian coefficients g_n^m and h_n^m of the IGRF model are actually *Schmidt quasi-normalized* coefficients to take care of the normalization change in the Legendre function explained above. Then one can just use the Gaussian normalized associated Legendre functions (2.12) with the given Schmidt quasi-normalized coefficients in equations (2.11) to obtain the three geocentric components of the geomagnetic field.

g_n^m and h_n^m are time dependent and must be linearly interpolated between the according time points, that is e.g. with the DGRF2005 and DGRF2010 coefficients for dates between 1

January 2005 and 31 December 2009. For doing this the year plus fraction time representation (section 2.2) is useful. The maximum degree N is 10 until DGRF1995 and 13 beginning with DGRF2000. There are no predictions for the 2020 epoch, i.e. no IGRF2020 coefficients, nonetheless the current IGRF is also valid for years between 2015 and 2020. The coefficients can be interpolated linearly with the IGRF2015 values and the *secular variation* coefficients that give a slope assumption for the 8 first degrees for the years 2015 to 2020. Higher degrees are very small and vary slowly, they need no interpolation, that anyway would result in no significant change.

The World Magnetic Model

> «Sponsored by the U.S. National Geospatial-Intelligence Agency (NGA) and the U.K. Defence Geographic Centre (DGC), the World Magnetic Model (WMM) is produced by the U.S. National Oceanographic and Atmospheric Administration's National Geophysical Data Center (NOAA/NGDC) and the British Geological Survey (BGS). It is the standard model used by the U.S. Department of Defense (DoD), the U.K. Ministry of Defence, the North Atlantic Treaty Organization (NATO) and the International Hydrographic Organization (IHO), for navigation, attitude and heading referencing systems using the geomagnetic field. It is also used widely in civilian navigation and heading systems.» [3]

The world magnetic model [34] works much like the IGRF, but thanks to its background has more and better official documentation, e.g. the WMM website [34] that also has a free to use C program, the technical report [3], an online calculator [33] (that also has IGRF values) and an FAQ page [32]. It also gives precise information about what type of field is included (just the core field, neither the crustal nor the external field) and that the WMM only is valid

> «from 1 km below the Earth's surface to 850 km above the surface.» [3]

Fortunately NUTS's orbit will be inside this range where the effects of solar winds are negligible and just the position with respect to the surface matters.

The WMM is issued every five years where again just the coefficients and not the model structure changes. Despite its military background it can be used by particulars just like the more international, cooperative and research-orientated IGRF. In contrast to it no definitive model for past dates exists, there is no need for historical data for the official users of the

WMM. The current issue, the world magnetic model 2015, just has the coefficients for the beginning of the validity range, 1 January 2015, and the secular variation slope values for interpolation until 31 December 2019. This is no drawback for NUT as the satellite also won't need past values but only the current field.

As soon as one has the geocentric position and time for the prediction, the calculation follows the steps for the IGRF. The magnetic flux density vector **B** is again given as gradient field, however the WMM defines geocentric coordinates not with the polar angle θ but with the elevation angle ψ, Figure 2.3. That's why the argument of the Schmidt quasi-normalized associated Legendre functions is $\sin\psi$ instead of $\cos\theta$. Likewise one must replace every $\sin\theta$ with $\cos\psi$. Apart from this the WMM uses equations (2.9), (2.10) and (2.11) too [3].
The equations (2.12) (with $\sin\psi$ instead of $\cos\theta$ and $\cos\psi$ instead of $\sin\theta$) are also used and Gaussian normalized associated Legendre functions calculated. g_n^m and h_n^m for the WMM however are given as Gaussian coefficients, that's why the Schmidt quasi-normalized Legendre function have first to be calculated form the Gaussian normalized associated Legendre functions before using the equations (2.11). The step is a simple multiplication

$$P_n^m(x) = S_{n,m}P^{n,m}(x), \qquad \frac{\partial P_n^m}{\partial x}(x) = S_{n,m}\frac{\partial P^{n,m}}{\partial x}(x)$$

with the recursive algorithm for the factors $S_{n,m}$ [7]

$$\begin{aligned} S_{0,0} &= 1 \\ S_{n,0} &= \frac{2n-1}{n}S_{n-1,0} \\ S_{n,m} &= \sqrt{\frac{(n-m+1)(\delta_m^1+1)}{n+m}}S_{n,m-1} \end{aligned}$$

The Kronecker delta δ_m^1 is defined in the List of Abbreviations and Symbols and x of course is $\sin\psi$ in the WMM algorithm.

2.6.3 Assumptions

Both the 12th generation IGRF and the WMM of 2015 are in the predictors of NUTS and before giving some details about the implementation, the obvious but nonetheless important assumption that *the geomagnetic prediction represents the true earth magnetic field for the years 2015 to and including 2019* must be stated.

In the following paragraphs this assumption will be tested against the known weaknesses of the models. But they also give arguments why the assumption is valid.

The magnetic models of low degree, i.e. IGRF and WMM, only model the geomagnetic core field. All other contributions to the total magnetic field, that is the crustal and the external field, are considered to be perturbations. [15] states the model error due to this to be in the order of 10 nT with additional 10 nT errors in the coefficients. [34] has a figure to visualize these two kinds of errors. **Figure 2.16**[18] shows the assumed differences between the output of the calculation and the actual compass declination around the globe. One part of the error is due to the finite degree of the model, the other error kind is of numerical nature because only a finite precision is possible.

In the huge majority of times when NUTS will predict the magnetic field vector, the assumption that the model truly represents the world will not harm the operation. Declination errors in the order of magnitude of a few degrees are within the noisy measurements of the magnetometers. Cases where the disturbance field greatly exceeds the mentioned bounds will likely occur, but they usually are just quick peaks. The satellite attitude dynamics are slow and won't respond significantly to peaks and theses cases can be ignored.

This however counts only inside the specified validity range of the WMM up to 850 km above the earth's surface, which is also assumed to hold for the IGRF. The earth's surface is very difficult to model but in general not very far away from the height over the ellipsoid, such that the demand can be formulated that NUTS shall not orbit the earth at altitudes higher than 850 km. This is well inside the assumed altitude NUTS finally will have.
This ensures also that the geomagnetic field is not influenced by solar winds, see Figure 2.11, thus the model in fact can just express the position in the ECEF frame in geocentric coordinates as it is done in the equations (2.9) and (2.10).

Finally the 12th generation international geomagnetic reference field and the 2015 version of the world magnetic model are standards for the years 2015 to 2020 with obviously higher accuracy at the beginning of this interval. **Figure 2.17**[19] shows the expected accuracy loss in declination over the five years. While it is still very small on a majority of locations on the earth, the declination in neighborhood of the magnetic poles returned by the WMM will be of poorer quality. The IGRF on its side states about 20 $\frac{\text{nT}}{\text{year}}$ loss in accuracy.
This should not be of greater importance as long as the year 2020 did not begin. Then how-

[18] Image taken from [34].
[19] Image taken from [34].

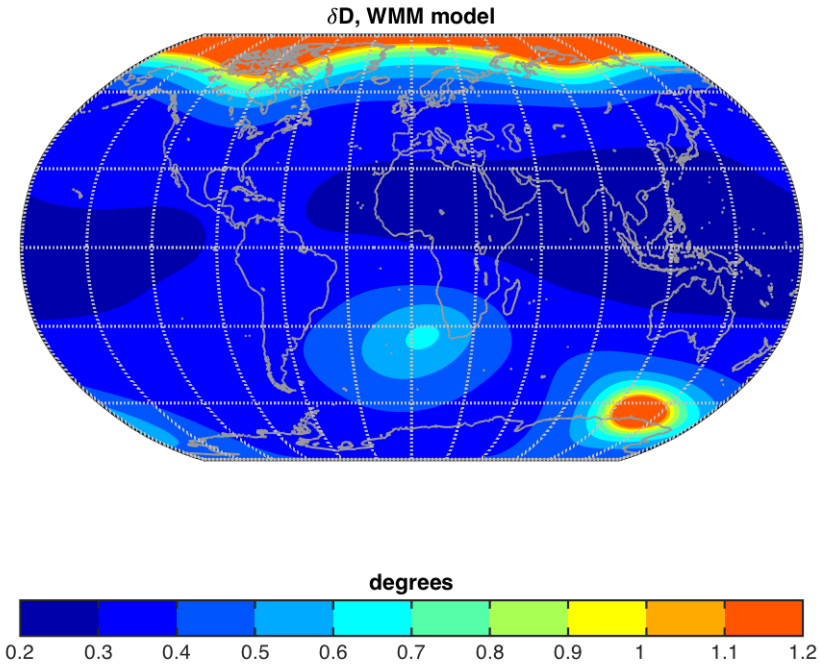


Figure 2.16: Global distribution of the declination error provided by the WMM2015 error model

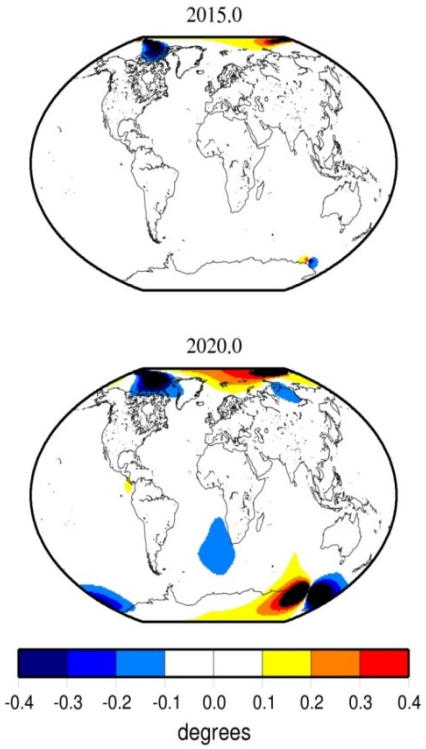


Figure 2.17: Estimated WMM2015 declination inaccuracy without considering crustal and disturbance field contributions

ever with the advent of a new IGRF generation and the WMM2020, an update of the coefficients is recommended. Because of reasons explained in the next subsection a coefficient change is relatively easy to perform but not an increase of degree. That is the highest degree will remain 13 for the IGRF and 12 for the WMM implementations on NUTS unless someone changes the C code at several places.

2.6.4 Implementation Details

IAGA has a Fortran program for the IGRF algorithm, but it was not used in this thesis. It was a lot easier to adapt the MATLAB file exchange implementation [4] to C source code.
All program lines of code relating to the historic models have been removed and the coefficient loading and interpolation merged into the main function. This means that just one function for returning the IGRF prediction remains

- `void magIgrf(double reci[3], Time timeUTC, double Beci[3])`

Again the interface just makes use of vectors expressed in the inertial reference frame (position input vector `reci`, magnetic flux density output vector `Beci`) and coordinated universal time points (time input struct `Time`). The algorithm itself follows the steps presented in the preceding subsections along with the model. But some points are worth noting.

A lot of arrays are defined in the function, they all have fixed (hard coded) sizes to avoid dynamical allocation which would use too much time and place on the ADCS microcontroller. This is also the reason why a higher degree of IGRF would come along with a lot of small changes in the function whenever arrays are declared and involved in loops, e.g. for the Legendre functions and derivatives.

The Schmidt quasi-normalized spherical harmonic main and secular variation coefficients are defined in two arrays of size 196 in exactly the same way then the official source [15], that is g_n^m and h_n^m together in one array to save place for the coefficients which by definition are 0. The first array has all the main field coefficients, the second all secular variation coefficients. MATLAB begins array indexation at 1, C begins at 0. Place holders are at the front of the two arrays in the C code to effectively start the indexation at 1 too. The year plus fraction representation of the time is advantageously used for the linear interpolation.

The potential function is expressed in the geocentric coordinates radial distance r, polar angle θ and azimuth λ, the `frameEci2ecef` function of section 2.3 however returns Cartesian

coordinates and `frameEcef2geod` the geodetic coordinates latitude ϕ, longitude λ (equals azimuth) and altitude h. On top of that the magnetic flux density is actually calculated as north east down values and needs to be expressed in ECI before the extended Kalman filter can use it. For these transformations [4] uses the Fortran method of the IGRF source

$$\begin{aligned}
\rho &= \sqrt{(a\sin(90^\circ-\phi))^2+(b\cos(90^\circ-\phi))^2} \\
r &= \sqrt{h^2+2h\rho+a^4\sin^2(90^\circ-\phi)+b^4\frac{\cos^2(90^\circ-\phi)}{\rho^2}} \\
c_d &= \frac{h+\rho}{r} \\
s_d &= \frac{(a^2-b^2)\cos(90^\circ-\phi)\sin(90^\circ-\phi)}{r\rho} \\
\cos\theta &= c_d\cos(90^\circ-\phi)-s_d\sin(90^\circ-\phi) \\
\sin\theta &= c_d\sin(90^\circ-\phi)+s_d\cos(90^\circ-\phi)
\end{aligned}$$

where ρ is a help value and a, b are the semimajor and semiminor axes of the earth. The polar angle θ is not computed explicitly but just its sine and cosine values that are needed in the algorithm. c_d, s_d are help values called like this because of their replacement of sine and cosine value in the next three equations.

$$\begin{aligned}
B_N &= -B_\theta c_d - B_r s_d \\
B_E &= B_\lambda \\
B_D &= B_\theta s_d - B_r c_d
\end{aligned}$$

Finally $\mathbf{B}^n = \begin{pmatrix} B_N & B_E & B_D \end{pmatrix}$ is transformed to $\mathbf{B}^i$ with the rotation matrix returned by the function `frameRotationNed2ecef` (see section 2.3).

The U.S. National Oceanic and Atmospheric Administration's National Geophysical Data Center (NOAA/NGDC) publishes on its website [34] an official C implementation of the WMM that private persons are allowed to use and edit for free. Of course this was the starting point for the NUTS implementation. Much of the work was again removing unnecessary parts and compliance to the interface such that the only remaining function is

- `void magWmm(double reci[3], Time timeUTC, double Beci[3])`

with the obvious parallels to `magIgrf`.

As for the IGRF function the arrays have fixed lengths and place holders are used to begin array indices with 1 (this is also the case in the source). Likewise the time is converted form the `Time` struct to a year plus fraction representation to facilitate the linear interpolation of the Gauss coefficients g_n^m and h_n^m. They are separated here such that four arrays of length 91 are defined, two for the main field coefficients and two for the secular variation.

The main part of the algorithm is still unchanged, but, as usual, steps in the beginning and end have been added. The potential function is expressed in the geocentric coordinates radial distance r, elevation angle ψ and azimuth λ. [34] uses a more classic transformation

$$r = \sqrt{x^2 + z^2}, \qquad \psi = \arcsin\left(\frac{z}{r}\right)$$

The conversion to a magnetic flux density expressed in the North East Down frame is then also the classic approach

$$\begin{aligned} B_N &= B_\psi \cos(\psi - \phi) - B_r \sin(\psi - \phi) \\ B_E &= B_\lambda \\ B_D &= B_\psi \sin(\psi - \phi) + B_r \cos(\psi - \phi) \end{aligned}$$

Again the result transformed to the ECI frame is written in the array `Beci` points at.

It should be noted that both models have singularities near the (geographic) poles, that is for very high elevation or very low polar angle respectively. The implementations take care of this with special routines in the respective cases

- $|\sin\theta| < 10^{-6}$ in `magIgrf`
- $|\cos\psi| < 10^{-10}$ in `magWmm`

2.6.5 Geomagnetic Field Prediction User's Guide

The way to use the geomagnetic predictors corresponds to the way to use the other functions described in this chapter. Note that again the usual user only needs to know the coordinated universal time and the ECI place because all time and frame manipulations happen inside the functions. The header file `magPred.h` and the source file `magPred.c` contain the two earth's magnetic field predictors.

An example will show the necessary steps. To verify own implementations, the WMM report has test values, one of them is used in **Listing 2.3**. The IGRF test value for the same time and place comes from the official online field calculator for IGRF and WMM [33].
Neither the location of the test value nor the magnetic flux density itself can directly be compared between the sources and the NUTS implementation. The location is given in geodetic coordinates and the magnetic flux density as north east down values in [3] and [33]. The NUTS software interacts only in the ECI reference frame with the user. Thus the latitude 80° N, longitude 0° E and altitude of 0 km above the ellipsoid are first converted to an ECI vector with the functions of section 2.3. In another small program not displayed here the North East Down source values are transformed to the inertial reference frame from

$$\mathbf{B}^n_{\mathrm{WMM}} = \begin{pmatrix} 6627.1 \\ -445.9 \\ 54\,432.3 \end{pmatrix} \mathrm{nT}, \qquad \mathbf{B}^n_{\mathrm{IGRF}} = \begin{pmatrix} 6630.9 \\ -447.2 \\ 54\,434.5 \end{pmatrix} \mathrm{nT}$$

to

$$\mathbf{B}^i_{\mathrm{WMM}} = \begin{pmatrix} 3174.479\,998 \\ -15\,647.973\,476 \\ -52\,460.043\,531 \end{pmatrix} \mathrm{nT}, \qquad \mathbf{B}^i_{\mathrm{IGRF}} = \begin{pmatrix} 3176.483\,417 \\ -15\,651.804\,494 \\ -52\,461.553\,370 \end{pmatrix} \mathrm{nT}$$

where one also sees that both models yield comparable flux densities.

The output following the execution of Listing 2.3 is

```
«WMM: 3174.465260 -15647.939589 -52460.008194 nT
IGRF: 3176.537955 -15651.713869 -52461.553057 nT »
```

and clearly very near to the official values. It's not sure where the small differences arise, but each manipulation of floating point numbers and the extensive usage of trigonometric functions induces rounding errors. Likewise different summing and multiplication orders may result in slightly different results. Finally the global constants for the earth's shape or π are not guaranteed to be exactly equal.
Anyway the result of the NUTS implementation is very good and so it is with all other official test values. One of the two predictor results can thus be used as reference vector $\mathbf{r}^i_m$ in equation (2.2).

Future users will also have the duty to update the model coefficients every five years, the first time in 2020. The satellite should be in orbit then and maybe still operational. Of course

Listing 2.3: Geomagnetic Field Prediction Example

```
...
#include "magPred.h"
...
int main(){
    // Official WMM 2015 test value 1
    Time time;
    time.year=2015;
    time.mon = 1;
    time.day = 1;
    time.hr =  0;
    time.min= 0;
    time.sec= 0.0;
    double rgeod[3] = {80.0, 0.0, 0.0};
    double recef[3];
    frameGeod2ecef(rgeod, recef);
    double reci[3];
    frameEcef2eci(recef, time, reci);

    double Beci[3];
    magWmm(reci, time, Beci);
    printf("WMM:  %f %f %f nT\n", Beci[0], Beci[1], Beci[2]);

    // Official WMM 2015 test value 1, IGRF result
    magIgrf(reci, time, Beci);
    printf("IGRF: %f %f %f nT\n", Beci[0], Beci[1], Beci[2]);
    return 0;
}
```

the software won't fail totally if this is not done directly because the linear interpolation will carry on working, but the accuracy decays little by little as shown in Figure 2.17. As already said, even if newer IGRF and WMM updates have higher degrees of precision, they must be cut down to 13 and 12 respectively if one wants to avoid to rewrite some parts of the code.

2.6.6 Recommendations on the Geomagnetic Field Model Choice

So there are two geomagnetic field models as part of the predictor package of NUTS for the moment. This redundancy is of course not required in the satellite flight model so one of the two models must be chosen.

All sources agree that no model is known to yield better results, that is nearer to the true geomagnetic field, so arguments for the IGRF or the WMM have to be found at other places.

No previous Master's thesis or paper on NUTS looked at the WMM and the book [35] e.g. also just mentions the IGRF. This is because of its international and scientific community

origin (it also accounts for historic values), while the WMM has a more military background. On the other hand the WMM has more and better documentation as well as a very good official implementation with test values to verify.

But the decision for one or the other model should be mainly based on performance of the simplified implementation on the NUTS hardware.

The IGRF operates with 13 degrees of spherical harmonics while the WMM has one less. Therefore it needs less coefficients, but due to a clever way of the IGRF to save them, that is leaving out all zeros by definition, the difference is small. There are $4 \cdot 91 = 364$ WMM and $2 \cdot 196 = 392$ IGRF coefficients of double precision among the global constants. This is a weak argument for the WMM.

The IGRF has slightly less lines of code because the spherical harmonic coefficients are Schmitt quasi-normalized instead of the Gauss normalized coefficients of the WMM. This avoids one normalization step and should also result in a faster function.

The running time is the most important comparison criterion because hardware on the satellite is restricted and the predictor functions are used often. Four running time tests answer this question (the decision for the time point and place to evaluate is explained in section 2.8). The same C source code written in the Eclipse Platform 3.8.1 was used on these two platforms

- Samsung P580 computer with Intel® Core™ i5 CPU M 430 @ 2.27 GHz ×4 running the Ubuntu 14.04LTS 64-bit Linux operating system, code compiled with the GNU C Compiler 4.8.2.

- Atmel UC3-A3 Xplained evaluation kit with the Atmel AVR AT32UC3A3256 microcontroller running FreeRTOS for AVR UC3 version 8.0.0, code compiled and written on the microcontroller with Atmel Studio 6.2.

  The Atmel AVR AT32UC3A3256 is actually not the microcontroller that will be the ADCS controller on the NUTS flight model, an Atmel AVR AT32UC3C2256C that has a floating point unit was chosen in the spring semester 2015. However it was not available for testing purposes, such that a comparable evaluation board was used. This also explains why the differences between the PC and the microcontroller are so high. There are a lot of floating point operations to do and trigonometric functions to evaluate, processes that can take very long without floating point unit. But these differences are not important, what counts is which model is the fastest on the respective platform.

Table 2.2: Running Time of IGRF and WMM on Computer and Microcontroller in Clock Ticks

Model	Date and Time	Personal Computer	Microcontroller
WMM	01.01.2015 00:00:00	68	2224000
IGRF	01.01.2015 00:00:00	63	2885500
WMM	15.10.2018 12:34:56.789	68	2224200
IGRF	15.10.2018 12:34:56.789	60	2893300

The `clock()` function of the C standard library is triggered just before and after the predictor call and the time difference is the running time of the predictor. The unit in which the results are expressed in **Table 2.2**, is *clock ticks*. This unit varies from system to system, the macro `CLOCKS_PER_SEC` returns how many ticks are in one second for the specific system, for the personal computer and the microcontroller it is both times 1 000 000.

The IGRF prediction needs always only about 90 % of the time of the WMM prediction on the tests on a personal computer because it has one normalization step less to do. The absolute time differences are however very small, both functions run quasi in an instant. This is interesting but again of minor importance for the suggestion which geomagnetic field model to use for the satellite flight model.

The best way to decide is to look on the function running times on the microcontroller the ADCS is going to use. One should expect a similar result than on the personal computer but in fact the WMM took only about 77 %, see Table 2.2, of the time of the IGRF, most likely because it has one degree of precision less. That is why this thesis suggests that *the world magnetic model should be used as geomagnetic field predictor on the NTNU Test Satellite*.

2.7 Libraries and Global Constants

All these predictors and transformations need a lot of coefficients and global constants. Likewise some special functions to treat strings or for mathematical operations are used a lot. That's why not just the usual `stdlib.h` and `stdio.h` C header files must be included in order to compile the software. One needs also the C standard libraries

- `string.h`, which helps a lot when handling the two-line element set, and
- `math.h`. It provides a macro `M_PI` for π and the functions for one floating point number `sqrt` for the square root as well as the trigonometric functions. Functions for two floating point numbers such as `fmod` for the remainder of a division and `pow` to raise to

Table 2.3: Global Constants for the Prediction Algorithms

Name	**Description, Unit, Source**	**Value** (Row 1-13) **Dimension** (Row 14-23)
DEG2RAD	Conversion factor from deg to rad, [35]	`M_PI / 180.0`
RAD2DEG	Conversion factor from rad to deg, [35]	`180.0 / M_PI`
TWOPI	Circumference of the unit circle, [35]	`2.0 * M_PI`
REQU	Earth semimajor axis, equatorial radius, km, [35]	`6378.137`
RPOL	Earth semiminor axis, polar radius, km, [35]; semimajor axis reduced with earth flattening	`6378.137 * (1 - (1.0/298.257223563))`
RMEAN	Mean earth radius, km, [4]	`6371.2`
ECCSQRD	Earth eccentricity squared, [35]; two times earth flattening minus earth flattening squared	`2.0*(1.0/298.257223563)- pow(1.0/298.257223563,2)`
J2	J2 earth oblateness coefficient, [36]	`0.00108262998905`
J4	J4 earth oblateness coefficient, [36]	`-0.00000161098761`
J3OJ2	J3 earth oblateness coefficient divided by J2 earth oblateness coefficient, [36]	`-0.00000253215306 / 0.00108262998905`
VELKMPS	Orbit velocity at 0 km altitude, $\frac{\text{km}}{\text{s}}$, [35]; square root of gravitational parameter divided by semimajor axis	`sqrt(398600.4418 / 6378.137)`
XKE	Reciprocal of the time unit (time to travel 1 rad in orbit), $\frac{1}{\text{min}}$, [36]	`1.0 / 13.4468520637`
MJDPOLAR	Modified Julian date of the Bulletin for polar motion estimation coefficients, [16]	`57065`
XPOLAR	Polar motion estimation coefficients, [16]	`double[5]`
YPOLAR	Polar motion estimation coefficients, [16]	`double[5]`
IAR80	Integer IAU-1980 nutation coefficients, [35]	`int[530]`
RAR80	Real number IAU-1980 nutation coefficients, [35]	`double[424]`
IGRFGH	12th gen. IGRF Schmidt quasi-normalized spherical harmonic main coefficients, nT, [15]	`double[196]`
IGRFGHSV	12th gen. IGRF Schmidt quasi-normalized secular variation coefficients, $\frac{\text{nT}}{\text{year}}$, [15]	`double[196]`
WMMG	2015 WMM main Gauss G coefficients, nT, [34]	`double[91]`
WMMGSV	2015 WMM secular variation G coefficients, $\frac{\text{nT}}{\text{year}}$, [34]	`double[91]`
WMMH	2015 WMM main Gauss H coefficients, nT, [34]	`double[91]`
WMMHSV	2015 WMM secular variation H coefficients, $\frac{\text{nT}}{\text{year}}$, [34]	`double[91]`

power are used often too.

`time.h` must be included for the time tests. This header is not required by the predictors.

All global constants were exported into an own header file named `constants.h`. and declared as `static const double` scalars and arrays. Table 2.3 shows all constants with their name, description, value (scalars) or dimension (arrays, see the digital attachment for the values) as well as source. The earth constants are those of the world geodetic system 1984.

2.8 Verifying the NUTS Prediction Software

The last source file of the predictor package, `testC.c`, has a method with at least one test for all the functions of the package. The values returned from the test are verified against known values taken from [35], [36] and [33]. If the absolute and relative difference is small, the main function just prints the tested function's name and `"OK!"`, the returned value and the expected value are printed next to each other otherwise. In total 39 tests are done and 29 of them return the expected value. For the 10 others the differences are slightly higher, but a closer inspection shows that they're still very acceptable.

The time conversion and frame transformation functions are just tested once and their correctness verified with examples from [35]. This may seem too little testing especially as small errors are returned by some frame functions, but the predictor functions use nearly all of them such that the correct implementation of the time and frame functions is proven indirectly by the successful verification of the predictors.

The only test of the sun predictor is Listing 2.1 [35] which shows that the algorithm works as awaited.

In addition to Listing 2.2, the orbit propagator is tested with another example of [36] which uses a different TLE, to be precise the first TLE ever used to verify the SGP4 algorithm published in [14]. The tests return nearly exactly the expected position and velocity.

Four of the official WMM2015 test values and their four IGRF counterparts [33] are compared to the NUTS implementations of the earth's magnetic field models. Because the test values possibly are special cases, Listing 2.3, which is one them, for example has 0° longitude and does not need the secular variation because it asks for the field on 1 January 2015 at midnight, a ninth and tenth test are implemented. They ask for the field vectors 5 km

above Trondheim (63° north and 10° east) on 15 October 2018 at 12:34:56.789.

Six of the ten tests are "OK!", two tests of the IGRF and two tests of the WMM have little higher but still very small differences in the magnetic flux density compared to the online calculator [33], so also this predictor works well enough for the purposes of NUTS.
The first and last tests of the WMM and IGRF are timed to have an objective argument which model is better for the satellite, see the last subsection on the geomagnetic field predictors. The running times on a personal computer and microcontroller are found in Table 2.2.

2.9 Relaxing the Clock Assumption

The central assumption in section 2.2 on the time conversions, which thus also affects all other predictor parts, is that the ADCS knows precisely the current coordinated universal time. Clearly this is almost impossible to realize on a CubeSat which does not have a highly precise atomic clock on board.

There are of course clock functions on the microcontroller and in FreeRTOS, but the time on the ADCS board will drift compared to the accurate clocks for universal time on earth. The NUTS team will need to readjust the clock from time to time, depending on the clock drift and the comments of this section that looks how an inexact clock affects the predictors.

2.9.1 Verification Values

In order to relate the impact of the clock on the predictors some other tests than those of section 2.8 are needed because here they must satisfy all the other assumptions of the chapter. To maintain this section related to what the user will need, only the three predictors for sun, orbit and geomagnetic field are inspected. It's not needed to look at the time conversions, as they're mostly just to change the representation of the same time point. Some of the frame transformations of course are affected by a drifting clock, but this impact will show itself through the predictors.

The exact time that the clock on board should have for the tests throughout this section, has to be *between 2005 and 2050* to satisfy the additional time conversion assumptions, in the *neighborhood of the IERS Bulletin A or B release date* from which the coefficients are taken for the frame transformations, *between 1950 and 2050* for the sun vector assumptions, in the *neighborhood of the TLE release epoch* to satisfy the orbit position and velocity assumptions

and *between January 2015 and December 2019* because of the geomagnetic field prediction assumptions. The easiest way to tackle everything at the same time, is to take the epoch of a recent Two-Line Element Set for a low earth orbit satellite.

The already mentioned UWE-3 (e.g. [9]) has orbit specifications that are very similar with those of NUTS (in addition it is a CubeSat too). The orbit has a very low eccentricity (circular obit), a short period of 97 min (low earth orbit) and the inclination is 97.7° (nearly polar, in fact sun-synchronous), thus the following recent TLE was taken from [18]

```
1 39446U 13066AG  15091.16814487  .00002750  00000-0  38274-3 0  9998
2 39446  97.7351 154.4636 0072683  33.0976 327.4752 14.76760372 71880
```

and the epoch 15091.16814487 set as basis for the tests of this section. This corresponds to the date 1 April 2015 and the time 04:02:08. In the following the time was perturbed by adding 10 s, 30 s, 1 min, 2 min and 3 min to the epoch. It is assumed that the performance when setting the clock back instead of forth is the same. Before commenting the performance with inexact clock the verification values must be calculated from the sources.

Sun Vector Verification Value

The original MATLAB implementation copied from [18] of the sun vector algorithm [35] was used to get the sun vector in the MOD reference frame. It was then transformed to the ECI frame using the original MATLAB function for the precession matrix. To ensure the highest possible precision, the appropriate IERS Bulletin B [17] was downloaded and the ΔT value for 1 April 2015 (modified Julian date 57113) used for the time conversions. The sun vector verification value (normed to one) in the ECI reference frame is thus

$$\mathbf{r}_{\odot}^{i} = \begin{pmatrix} 0.981\,949 \\ 0.173\,541 \\ 0.075\,229 \end{pmatrix}$$

Orbit Position and Velocity Verification Value

Similarly to the preceding paragraph the original MATLAB implementation copied from [18] of the SGP4 algorithm [36] was used to get the orbit position and velocity vectors of the mentioned TLE at the epoch time in the TEME reference frame. Likewise the resulting vectors were transformed to the ECI frame using the functions of [35], here again the highest possible

precision was used in the transformation options (highest order 106 and complete nutation matrix) and for the time conversions. The orbit position and velocity verification vectors in the ECI reference frame are

$$\mathbf{r}^i = \begin{pmatrix} -6285.864\,466 \\ 3029.483\,180 \\ 9.366\,419 \end{pmatrix} \text{km}, \qquad \dot{\mathbf{r}}^i = \begin{pmatrix} 0.488\,089 \\ 0.902\,297 \\ 7.513\,168 \end{pmatrix} \frac{\text{km}}{\text{s}}$$

Geomagnetic Field Verification Value

The official online calculator for the earth's magnetic field [33] returns the magnetic flux density for both models. The TLE epoch is converted to 2015.245 334 years and the location is $\mathbf{r}$ from the preceding paragraph. This has first to be converted to the ECEF reference frame and then expressed in geodetic coordinates. Here again [18] provides all the necessary functions and the appropriate IERS Bulletin B [17] the missing constants $UT1-UTC$, ΔT, x_p, y_p, $\delta\Delta\Psi_{1980}$ and $\delta\Delta\epsilon_{1980}$ to have the magnetic flux densities

$$\mathbf{B}^n_{\text{IGRF}} = \begin{pmatrix} 21\,722.7 \\ 1934.9 \\ 7375.4 \end{pmatrix} \text{nT}, \qquad \mathbf{B}^n_{\text{WMM}} = \begin{pmatrix} 21\,720.4 \\ 1936.5 \\ 7373.3 \end{pmatrix} \text{nT}$$

that expressed in the ECI frame become

$$\mathbf{B}^i_{\text{IGRF}} = \begin{pmatrix} 5835.947\,136 \\ -4945.923\,954 \\ 21\,713.949\,796 \end{pmatrix} \text{nT}, \qquad \mathbf{B}^i_{\text{WMM}} = \begin{pmatrix} 5833.357\,342 \\ -4946.453\,473 \\ 21\,711.653\,587 \end{pmatrix} \text{nT}$$

2.9.2 Impact of an inexact Clock on the Predictions

The results of all test runs to describe the impact of an inexact clock on the predictions are summarized in **Table 2.4**, they were essentially achieved using the approach already shown in the listings of the respective sections.

Sun Vector Prediction

Comparing at first the exact time result of the NUTS implementation with the verification value shows once again that there are no mistakes in the implementation of this predictor.

Table 2.4: Impact of an inexact Clock on the Predictions

Function	Date and Time	Result
sun	01.04.2015 04:02:08	rsun:0.981949 0.173541 0.075229
sun	01.04.2015 04:02:18	rsun:0.981949 0.173543 0.075230
sun	01.04.2015 04:02:38	rsun:0.981948 0.173546 0.075231
sun	01.04.2015 04:03:08	rsun:0.981947 0.173552 0.075233
sun	01.04.2015 04:04:08	rsun:0.981945 0.173563 0.075238
sun	01.04.2015 04:05:08	rsun:0.981942 0.173573 0.075243
orbitSgp4	01.04.2015 04:02:08	reci:-6285.864160 3029.483033 9.366418
orbitSgp4	01.04.2015 04:02:18	reci:-6280.614041 3038.328117 84.496584
orbitSgp4	01.04.2015 04:02:38	reci:-6267.900441 3054.946258 234.718525
orbitSgp4	01.04.2015 04:03:08	reci:-6243.306855 3077.178132 459.821743
orbitSgp4	01.04.2015 04:04:08	reci:-6174.329672 3111.851524 908.325985
orbitSgp4	01.04.2015 04:05:08	reci:-6079.203873 3133.346060 1352.973031
orbitSgp4	01.04.2015 04:02:08	veci:0.488089 0.902297 7.513167
orbitSgp4	01.04.2015 04:02:18	veci:0.561911 0.866651 7.512615
orbitSgp4	01.04.2015 04:02:38	veci:0.709385 0.795041 7.508854
orbitSgp4	01.04.2015 04:03:08	veci:0.930031 0.686896 7.496573
orbitSgp4	01.04.2015 04:04:08	veci:1.368412 0.468403 7.448144
orbitSgp4	01.04.2015 04:05:08	veci:1.801315 0.247768 7.368041
magIgrf	01.04.2015 04:02:08	IGRF: 5834.848416 -4945.775712 21714.086846
magIgrf	01.04.2015 04:02:18	IGRF: 6473.450124 -5247.516299 21653.679747
magIgrf	01.04.2015 04:02:38	IGRF: 7750.654918 -5859.634172 21483.475284
magIgrf	01.04.2015 04:03:08	IGRF: 9660.398950 -6796.687715 21103.388941
magIgrf	01.04.2015 04:04:08	IGRF:13417.026973 -8720.128474 19889.058864
magIgrf	01.04.2015 04:05:08	IGRF:17013.342942 -10673.677670 18070.269757
magWmm	01.04.2015 04:02:08	WMM: 5832.282720 -4946.270106 21711.751311
magWmm	01.04.2015 04:02:18	WMM: 6470.811765 -5247.914865 21651.532915
magWmm	01.04.2015 04:02:38	WMM: 7747.975451 -5859.839359 21481.722613
magWmm	01.04.2015 04:03:08	WMM: 9657.916464 -6796.628674 21102.212427
magWmm	01.04.2015 04:04:08	WMM:13415.662558 -8719.786141 19888.615585
magWmm	01.04.2015 04:05:08	WMM:17013.310283 -10673.458899 18069.589674

Clearly the sun vector prediction is not affected by an inexact clock, the first four significant digits in all three vector components remain the same throughout the tests. This is no surprise as the earth is not moving far in just three minutes. Therefore suggestions on what is an acceptable time shift, are not based on the sun vector predictor. A test with an even higher clock error of 10 min still yielded an acceptable sun vector.

Orbit Position and Velocity Prediction

The verification value and the exact time calculation with the NUTS implementation again coincide very well, so the inexact time results can directly be compared to the exact time results in Table 2.4 without having to look back to the verification value.

The differences are of course higher than for the sun vector, but it's difficult to judge if this is acceptable for NUTS, based on these values only because they're not used directly in the attitude estimator. The position will just be the input of the geomagnetic field predictor.

It was suggested in the beginning of section 2.5 that $\omega_0 = \frac{\dot{r}}{r}$ (the quotient of the magnitudes of velocity and position) should be used in equation (2.8). The magnitudes of `reci` and `veci` in Table 2.4 do not change very much such that this equation still is valid even with 2 min and 3 min clock error.

Geomagnetic Field Prediction

This predictor is a bit different then the other two because it depends on the time and position. The time dependence of the earth's magnetic field is rather low, but as an inexact clock results in a different position (see preceding paragraph) the impact is doubled. In fact the decision how much clock error is acceptable for NUTS should be based on the geomagnetic field predictor. Which model, the WMM or IGRF, is used is insignificant for the present discussion as can be seen in the tabled values. They also show one more time that the implementation is working if one compares the output for the exact time and the verification values longer above.

To help to understand Table 2.4 the result vector of the world magnetic model are drawn in **Figure 2.18**[20]. A plot of the IGRF values is not included because they essentially are equal to the WMM vectors, neither is the verification value plotted because no difference can be seen with the vector labelled "exact".

[20] Own production using MATLAB's `quiver3` function.

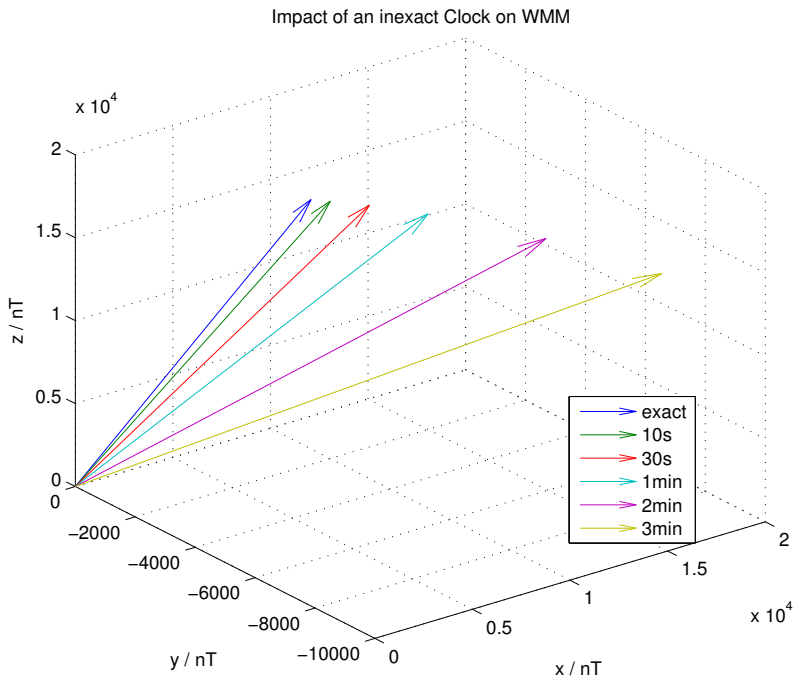


Figure 2.18: Impact of an inexact Clock on the World Magnetic Model

Unfortunately the differences with a slightly inexact clock already are somewhat large. One minute is still OK, but two minutes or more is obviously unacceptable. In addition the variation in the earth's magnetic field is higher near the (geographic and magnetic) poles as well as above the South Atlantic Anomaly, than in the example (equatorial plane, because the `reci` z value nearly is 0 for the exact time).

To summarize this section it is of course desirable to have the best possible clock but should it drift too much, the geomagnetic field prediction will be the first affected. This mainly because of a wrong position computed by the orbit propagator. In any case one should aim to *maintain the clock error under one minute*.

But to finish on a positive note, the implementation of the predictors for the attitude estimator of NUTS was once again verified to yield the expected results. If all assumptions are satisfied, they match within a very small error with the highly reliable sources.

2.10 Conclusion and Further Work

The outcome of this chapter is very positive as the faced problems were solved. A fully functional and easy to use prediction software package is now available for NUTS and all future developers. In addition the theory behind e.g. the frame transformations, the orbit propaga-

tion and the earth's magnetic field are described with much more details than in comparable works such that hopefully the reader has now a better understanding in this exciting field.

Not much further work concerning the prediction algorithms is needed for NUTS. All the necessary functions have been written and tested several times successfully. However the development was done on a personal computer and not on a microcontroller. The reasons for this are the easier handling and faster debugging as well as the fact that no exemplar of the new microcontroller for the ADCS board was available during most of the time of the semester. The remaining work to do concerns thus to port and test the code on the microcontroller. This should not be a problem as the functions run on a very similar microcontroller without error.

A serious issue could be the long running time of some functions, especially the geomagnetic models and the two functions `frameNutation` and `frameTruemean` that return the nutation matrices $\mathbf{N}_e$ and $\mathbf{N}_t$, see section 2.3.
The order of the WMM and IGRF can be reduced from 12 and 13 respectively down to 8 or 9 without losing much precision in a shorter computation time. Note that changes must be done at several places (marked in the comments) because fixed array lengths are used.
The most time taking part to calculate the nutation matrices is a weighted sum for two parameters with 106 terms. The coefficients are ordered by size such that 106 can easily be reduced in the `for` loop of the functions (down to 20 or 40 for example) to speed up the computation. The result will still be accurate enough for NUTS.

Finally as said at several places, the global constants `MJDPOLAR`, `XPOLAR` and `YPOLAR` must be adjusted with the newest values from the IERS Bulletin A [16] close to the launch. They provide a good interpolation for at least one or two years, occasional cross checking between the interpolation and the IERS Bulletin A can be interesting to see when the coefficients must be replaced. Likewise it's recommended to change the coefficients of the IGRF and WMM should the satellite operate beyond 2020. And of course the two-line element set from [18] must be transferred to the satellite as soon as possible after its publication.

Chapter 3

Robust Spacecraft Attitude Stabilization using Magnetorquers

3.1 Introduction

Spacecraft attitude control using magnetorquers has successfully been done for decades especially for low earth orbit missions and small satellites. However mathematical proofs for the theoretical global feasibility (or infeasibility) of magnetic control are still missing despite being the object of active research. There are numerous papers facing the problem and equally numerous solutions to specific parts of the problem (e.g. detumbling procedure, see [24], or just the orbit stabilization as in this chapter) or solutions under certain assumptions (e.g. the satellite's orbit faces a periodic geomagnetic field).

Indeed magnetorquers have proven themselves in the field and their obvious advantages are appreciated by spacecraft engineers. There are no moving parts, there is potentially an unlimited energy supply as no tank can go empty and solar panels recharge the batteries, magnetorquers are cheap, easy to operate and reliable. The main disadvantage is that it is not possible to provide a torque in every direction at all times but only torques *perpendicular* to the surrounding magnetic field of the earth. One can overcome this by adding a reaction wheel to the actuator set, but this is not the subject of this chapter. The other drawback of magnetic control, namely the relatively weak torques, is irrelevant for theoretical analyses.

The genesis of the present work begins with two papers demonstrating asymptotic attitude stability of a small satellite under different circumstances using magnetorquers only. In short [11] assumes a perfectly known satellite inertia and nothing about the geomagnetic

field and proves asymptotic stability for nadir pointing using purely nonlinear analysis. The second and newer paper [2] on the other side assumes a periodic earth's magnetic field but an unknown inertia matrix and proves local exponential stability for alignment of the principal satellite axes with the orbit considering a linear average approximation of the attitude dynamics. The aim of this chapter is to combine both papers and try to prove stability for an unknown satellite inertia while not assuming anything special for of the geomagnetic field.

For the whole chapter it is assumed that the reader is familiar with nonlinear system analysis and Lyapunov stability as for example explained in [19]. It is also highly recommended to read [11] and [2].

3.2 Problem Description and Preliminaries

3.2.1 Spacecraft Attitude Kinematics and Kinetics

The notation for this chapter follows for the most the notation of [11] but is harmonized with the preceding chapter. Many ideas are also taken from [8]. For this problem three reference frames are relevant the *Earth-Centered Inertial*, **ECI**, the *ORBIT* and *BODY* reference frame. As in chapter 2 vectors expressed in the different frames are marked with $(\cdot)^i$, $(\cdot)^o$ and $(\cdot)^b$ superscripts respectively.

The rotation matrices are $\mathbf{R}_i^o$, $\mathbf{R}_i^b$ etc. are of minor importance apart from the rotation matrix between ORBIT and BODY that describes the satellite's attitude. There are several possibilities to parametrize it, [11] uses three (column) vectors of *direction cosines*, that is projections of the ORBIT axes onto the BODY frame axes

$$\mathbf{R}_o^b = \begin{pmatrix} \mathbf{c}_1^b & \mathbf{c}_2^b & \mathbf{c}_3^b \end{pmatrix} = \begin{pmatrix} c_{11}^b & c_{12}^b & c_{13}^b \\ c_{21}^b & c_{22}^b & c_{23}^b \\ c_{31}^b & c_{32}^b & c_{33}^b \end{pmatrix}$$

[2] describes the rotation like in the extended Kalman filter of chapter 2 with a unit quaternion q, a computationally efficient way to regroup the rotation axis unit vector $\boldsymbol{\lambda}$ and angle β

$$\mathbf{q} = \begin{pmatrix} \boldsymbol{\epsilon} \\ \eta \end{pmatrix} = \begin{pmatrix} \boldsymbol{\lambda} \sin\frac{\beta}{2} \\ \cos\frac{\beta}{2} \end{pmatrix}, \qquad \epsilon_1^2 + \epsilon_2^2 + \epsilon_3^2 + \eta^2 = 1$$

Note that the order of $\boldsymbol{\epsilon}$ and η often is the reversed in $\mathbf{q}$ like e.g. in [8]. The rotation matrix for the attitude is then

$$\mathbf{R}_o^b = \begin{pmatrix} 1-2(\epsilon_2^2+\epsilon_3^2) & 2(\epsilon_1\epsilon_2-\epsilon_3\eta) & 2(\epsilon_1\epsilon_3+\epsilon_2\eta) \\ 2(\epsilon_1\epsilon_2+\epsilon_3\eta) & 1-2(\epsilon_1^2+\epsilon_3^2) & 2(\epsilon_2\epsilon_3-\epsilon_1\eta) \\ 2(\epsilon_1\epsilon_3-\epsilon_2\eta) & 2(\epsilon_2\epsilon_3+\epsilon_1\eta) & 1-2(\epsilon_1^2+\epsilon_2^2) \end{pmatrix}$$

However these rotation matrices are not very important for the kinematics and dynamics of the satellite attitude as one is not interested in position and linear velocity but in the attitude and angular velocity. The *rotation matrix differential equation*

$$\dot{\mathbf{R}}_o^b = \mathbf{R}_o^b \mathbf{S}(\boldsymbol{\omega}_{ob}^b) = \mathbf{R}_o^b \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix}$$

using the cross product operator $\mathbf{S}(\cdot)$ not only provides a way to define the angular velocity vector $\boldsymbol{\omega}$ but also the attitude differential equations for the direction cosine vectors as used in [11]

$$\dot{\mathbf{c}}_1^b = \mathbf{S}(\mathbf{c}_1^b)\boldsymbol{\omega}_{ob}^b, \qquad \dot{\mathbf{c}}_2^b = \mathbf{S}(\mathbf{c}_2^b)\boldsymbol{\omega}_{ob}^b, \qquad \dot{\mathbf{c}}_3^b = \mathbf{S}(\mathbf{c}_3^b)\boldsymbol{\omega}_{ob}^b \tag{3.1}$$

When using unit quaternions as in [2] the attitude differential equation is

$$\dot{\mathbf{q}} = \mathbf{T}(\mathbf{q})\boldsymbol{\omega} = \frac{1}{2}\begin{pmatrix} \eta\mathbf{I}_{3\times 3} + \mathbf{S}(\boldsymbol{\epsilon}) \\ -\boldsymbol{\epsilon}^{\mathrm{T}} \end{pmatrix}\boldsymbol{\omega} \tag{3.2}$$

Equations (3.1) and (3.2) are the attitude *kinematics*. They form the *dynamics* or *equations of motion* together with the attitude *kinetics*

$$\mathbf{J}\dot{\boldsymbol{\omega}}_{ib}^b = -\mathbf{S}(\boldsymbol{\omega}_{ib}^b)\left(\mathbf{J}\boldsymbol{\omega}_{ib}^b\right) + \boldsymbol{\tau}^b \tag{3.3}$$

with the vector $\boldsymbol{\tau}^b$ regrouping all the torques acting on the satellite and its inertia matrix $\mathbf{J}$. The latter is a diagonal matrix since equation (3.3) is expressed in the BODY frame (center of gravity, principal axes). However one is most often not interested in reducing the angular velocity of the spacecraft with respect to the inertial frame, but rather with respect to the

ORBIT frame. Simplifying equation (2.8) to $\boldsymbol{\omega}_{ib}^b = \boldsymbol{\omega}_{ob}^b + \omega_0 \mathbf{c}_1^b$ yields the kinetics

$$\mathbf{J}\dot{\boldsymbol{\omega}}_{ob}^b = -\omega_0 \mathbf{J}\mathbf{S}(\mathbf{c}_1^b)\boldsymbol{\omega}_{ob}^b - \mathbf{S}(\boldsymbol{\omega}_{ob}^b)\left(\mathbf{J}\boldsymbol{\omega}_{ob}^b\right) - \omega_0 \mathbf{S}(\boldsymbol{\omega}_{ob}^b)\left(\mathbf{J}\mathbf{c}_1^b\right) - \omega_0 \mathbf{S}(\mathbf{c}_1^b)\left(\mathbf{J}\boldsymbol{\omega}_{ob}^b\right) - \omega_0^2 \mathbf{S}(\mathbf{c}_1^b)\left(\mathbf{J}\mathbf{c}_1^b\right) + \boldsymbol{\tau}^b \tag{3.4}$$

3.2.2 Attitude Stabilization

For the scope of this thesis *attitude stabilization* aims to bring the satellite attitude to a certain equilibrium point and to hold it there. This is a difference to full attitude control where the goal is to turn the satellite to any desired attitude.

For [11] this objective is *nadir pointing* where the first and third axes of ORBIT and BODY are parallel. This is mathematically expressed as

$$\boldsymbol{\omega}_{ob}^b = \mathbf{0}_{3\times 1}, \qquad c_{21}^b = 0, \quad c_{31}^b = 0, \quad c_{13}^b = 0, \quad c_{23}^b = 0 \tag{3.5}$$

or $\mathbf{x} = \mathbf{0}_{7\times 1}$ in short, where the state vector $\mathbf{x}$ regroups the seven variables. There are thus four such points depending if the BODY x-axis, respectively z-axis, points in the same ($c_{11}^b = 1$, respectively $c_{33}^b = 1$) or opposite direction ($c_{11}^b = -1$, respectively $c_{33}^b = -1$) as the ORBIT x-axis, respectively z-axis. Remember that the three $\mathbf{c}^b$ are unit vectors. The alignment of the x-axes is not relevant for NUTS (i.e. c_{11}^b can be ± 1), but the alignment of z-axes is because the camera on board is intended to take pictures of the earth (i.e. $c_{33}^b = 1$ must hold). There is no need to specify a condition for the BODY y-axis. Because of the coupling of the direction cosines, parallel x- and z-axes result in parallel y-axes of BODY and ORBIT.

The equilibrium point defined by [2] is much stronger, namely to align the satellite principal axes with the axes of the inertial frame ECI i.e.

$$\boldsymbol{\omega}_{ib}^b = \mathbf{0}_{3\times 1}, \qquad \mathbf{q} = \begin{pmatrix} \mathbf{0}_{3\times 1} \\ \pm 1 \end{pmatrix} \tag{3.6}$$

where $\mathbf{q}$ is the unit quaternion parametrizing the $\mathbf{R}_i^b$ rotation matrix. $\eta = \pm 1$ does *not* mean that there are two equilibrium points, but results from the definition of the unit quaternion and the symmetry of the sine and cosine functions. Each attitude has two quaternion representations, $\mathbf{q}$ and $-\mathbf{q}$.

3.2.3 Magnetorquers

Figure 1.2 shows the NUTS ADCS prototype. On the two larger sides not facing the camera and on the top square side are coils through which a current can flow. This is the usual way to do magnetic attitude control. The current raises a magnetic field which interacts with the earth's magnetic field through the well known fact that the magnetic field lines tend to align to each other. Here it also becomes clear, and equation (3.7) confirms this qualitative statement, that only torques that are perpendicular to the surrounding magnetic field can be generated with magnetorquers. Of course the earth's magnetic field does not change but the satellite that raises the magnetic field, can move freely and the torque trying to align the field lines will affect the coils an eventually the satellite body rigidly attached to them.

This effect is easily described with the concept of *magnetic moment*. The magnetic dipole moment of a coil with N turns, covering the area A is $m = NAi$ when a current i is flowing. This quantity links the resulting torque $\boldsymbol{\tau}_m^b$ experienced by a magnet (permanent, electromagnet or charged particle) to the magnetic flux density $\mathbf{B}^b$ that surrounds it following the law

$$\boldsymbol{\tau}_m^b = \mathbf{S}(\mathbf{m}^b)\mathbf{B}^b \tag{3.7}$$

where $\mathbf{B}^b$ is the flux density vector of the earth's magnetic field in the orbit and $\mathbf{m}^b$ is the magnetic dipole moment vector and in case of electromagnets the control variable. All the vectors are expressed in the BODY frame, but this fundamental relationship is global.

Of course anyone can design a control law for $\mathbf{m}^b$ at free will, but it is wise to include a cross product with the surrounding magnetic field such that in any case $\mathbf{m}^b$ is orthogonal to $\mathbf{B}^b$. This requires the presence of magnetometer which for this theoretical analysis is no restriction.

3.2.4 Robust Attitude Stabilization

The last term in the chapter's title to define is "robust". Throughout the following this is associated with unknowns, uncertainties or changing parameters in the satellite model. In short a robust control algorithm works well not just in the idealized world of the model used for the design, but also in the real environment with all its disturbances.

Considering equation (3.3) (or (3.4)) there are two obvious and hence central uncertainties, the inertia matrix $\mathbf{J}$ and the total torque $\boldsymbol{\tau}^b$ acting on the spacecraft.

The inertia can be measured to some extent but not as precisely as e.g. the coil parameters area and turn count. That's why the control variable $\mathbf{m}^b$ won't be perturbed in the following. The analysis in the next section however will assume an unknown inertia matrix $\tilde{\mathbf{J}}$. This is actually quite realistic as a lot of factors can alter the inertia for example during the launch phase or because of space debris. Satellites with fuel tanks will experience even greater change of inertia.
$\tilde{\mathbf{J}}$ is not completely unknown because every inertia matrix is positive definite, it is also still diagonal when expressed with reference to the principal axes, these can however change. In addition a lower and upper bound may be known

$$\mathbf{0}_{3\times3} < \lambda_{\min}(\tilde{\mathbf{J}})\mathbf{I}_{3\times3} \leq \tilde{\mathbf{J}} \leq \lambda_{\max}(\tilde{\mathbf{J}})\mathbf{I}_{3\times3} \tag{3.8}$$

Beside the control torque, equation (3.7), applied to the satellite, other environmental torques will affect the attitude. It's difficult to chose which of these are large enough such that they're relevant to consider and even more difficult to model the chosen disturbance torques.
[2] does not include any other torque than the magnetic torque, but [11] includes the *gravity gradient torque*

$$\boldsymbol{\tau}_{\mathrm{g}}^b = 3\omega_0^2\mathbf{S}(\mathbf{c}_3^b)\left(\mathbf{J}\mathbf{c}_3^b\right) \tag{3.9}$$

which is quite popular to add to the analysis because it usually has stabilizing effect and is in the order of magnitude of the torque the coils can generate.

Another robustness aspect is of importance specially for spacecraft with magnetorquers. Some stability proofs require the geomagnetic field to be periodic along the satellite orbit, that is the same flux density is present after the satellite did one revolution about the earth. This is e.g. necessary if the attitude dynamics are linearized and a linear quadratic regulator implemented. [25] shows that when the earth's magnetic field is approximated by a dipole, the PD controller relying on this design technique is robust with respect to small perturbations in the inertia matrix but that larger perturbations can destabilize the system. While this assumption is not far from reality for a shorter time period, it is well known that the magnetic field changes with time in a way difficult to predict, see section 2.6. Clearly controllers not assuming periodicity of the earth's magnetic field can be qualified as more robust than the others.

3.2.5 Preliminary Results

Results of [11]

[11] shows that the equilibrium point (3.5) is globally uniformly asymptotically stable when the gravity gradient torque is considered and the magnetic torque is controlled by

$$\mathbf{m}^b = \mathbf{H}\left(\mathbf{S}(\boldsymbol{\omega}_{ob}^b)\mathbf{B}^b\right) \tag{3.10}$$

where $\mathbf{H}$ is a positive definite controller matrix. It is assumed that the inertia matrix is perfectly known.

The beginning of the argumentation and the control law (3.10) are repetitions of the analysis of [40] with the correction proposal [5]. Global uniform stability is shown with the Lyapunov function

$$\begin{aligned} V = E_{\text{kin}} + E_{\text{g}} + E_{\text{gyro}} &= \frac{1}{2}\left(\boldsymbol{\omega}_{ob}^b\right)^{\mathrm{T}}\mathbf{J}\boldsymbol{\omega}_{ob}^b + \frac{3}{2}\left(\omega_0^2\left(\mathbf{c}_3^b\right)^{\mathrm{T}}\mathbf{J}\mathbf{c}_3^b - J_z\right) + \frac{1}{2}\omega_0^2\left(J_x - \left(\mathbf{c}_1^b\right)^{\mathrm{T}}\mathbf{J}\mathbf{c}_1^b\right) \\ &= \frac{1}{2}\left(\boldsymbol{\omega}_{ob}^b\right)^{\mathrm{T}}\mathbf{J}\boldsymbol{\omega}_{ob}^b + \frac{3}{2}\omega_0^2\left((J_x - J_z)c_{13}^2 + (J_y - J_z)c_{23}^2\right) + \frac{1}{2}\omega_0^2\left((J_x - J_y)c_{21}^2 + (J_x - J_z)c_{31}^2\right) \end{aligned} \tag{3.11}$$

which has the nice property of describing the most important energy parts of the satellite in orbit. The first term E_{kin} is the kinetic energy of the rotation of the satellite with respect to the ORBIT frame, the second summand E_{g} takes the energy due to the gravity gradient and the third term E_{gyro} finally is the potential energy of a satellite orbiting the earth. The Lyapunov function is positive definite for the state vector $\mathbf{x}$ containing the seven variables of equation (3.5) if $J_x > J_y > J_z$ in the diagonal inertia matrix $\mathbf{J} = \text{diag}(J_x, J_y, J_z)$. This condition eventually means that the gravity gradient has a stabilizing effect.

The first time derivative of the Lyapunov function is found to be

$$\begin{aligned} \dot{V} = \frac{\partial V}{\partial \mathbf{x}}\mathbf{f}(t,\mathbf{x}) &= \left(\boldsymbol{\omega}_{ob}^b\right)^{\mathrm{T}}\boldsymbol{\tau}_m^b \\ &= \left(\boldsymbol{\omega}_{ob}^b\right)^{\mathrm{T}}\mathbf{S}\left(\mathbf{H}\left(\mathbf{S}(\boldsymbol{\omega}_{ob}^b)\mathbf{B}^b\right)\right)\mathbf{B}^b = -\left(\boldsymbol{\omega}_{ob}^b\right)^{\mathrm{T}}\mathbf{S}^{\mathrm{T}}(\mathbf{B}^b)\mathbf{H}\mathbf{S}(\mathbf{B}^b)\boldsymbol{\omega}_{ob}^b \end{aligned} \tag{3.12}$$

after some reduction steps using the kinematic and kinetic equations, the gravity gradient torque and the controller. This is a negative semidefinite function and the equations (3.11) and (3.12) show that the equilibrium (3.5) of the satellite dynamics (3.1) and (3.4) is globally uniformly stable with the controller (3.10).

Without making $\dot{V}$ negative definite [11] shows nonetheless that $\mathbf{x} = \mathbf{0}_{7\times1}$ is globally uniformly *asymptotically* stable by applying Matrosov's theorem as presented and proven by [20]. The difference with [40] (and the correction proposal [5]), that uses LaSalle's invariance principle to prove local uniform asymptotic stability, is that Matrosov's theorem does not need to assume that the earth's magnetic field is periodic along the satellite orbit.

Results of [2]

[2] takes a different way in his proof on stability of the equilibrium (3.6) with the controller

$$\mathbf{m}^b = -\mathbf{S}(\mathbf{B}^b)(\epsilon^2 k_1 \boldsymbol{\epsilon} + \epsilon k_2 \boldsymbol{\omega}_{ib}^b) \tag{3.13}$$

where k_1 and k_2 are two positive control gains and ϵ is a sufficiently small positive number (this should not be confounded with the three quaternion components $\boldsymbol{\epsilon}$).

Only local results come out of the analysis as the satellite dynamics are at first linearized, however this at first permits to relax the knowledge of the inertia matrix because the difference between $\mathbf{J}$ and $\tilde{\mathbf{J}}$ acts just as a perturbation of the averaged linearized system. Additionally the stability of a linear system is exponential and not just asymptotic. However the condition

$$\lim_{T\to\infty} \frac{1}{T} \int_0^T \mathbf{S}(\mathbf{B}^i)\mathbf{S}^{\mathrm{T}}(\mathbf{B}^i)\mathrm{d}t > \mathbf{0}_{3\times3} \tag{3.14}$$

must hold for the average system to be exponentially stable. [2] argues that this is equivalent to a periodic geomagnetic field along the orbit.

After several steps the averaged linearized system is proven to be (exponentially) stable which by nature of exponential stability means that it's robust to small parameter perturbations. Local exponential stability of (3.6) for the satellite dynamics (3.2) and (3.3) follows immediately.

In a second relaxation step of the model [2] even proves that his controller (3.13) remains stable without the measurements of the angular velocity. They can be replaced by a dynamic filter of the attitude quaternion, that eventually does not change much in the stability proof. In the same sense saturation in the magnetorquers, i.e. a limit in the capacity of generating a magnetic dipole moment, does not alter the robust stabilization achieved by the control law.

At first sight equation (3.14) seems to restrict the analysis to a very special case because the multiplication of a skew-symmetric matrix with its transpose always is singular. However

[21] gives arguments that show that the requirement in fact is not restrictive. The variation in the surrounding magnetic field is mainly dependent on the orbital velocity about the earth ω_0. If the absolute value of the angular velocity of the satellite about the earth $\|\boldsymbol{\omega}_{ib}^b\|$ is sufficiently smaller than ω_0, the magnetic flux density seen by the satellite $\mathbf{B}^b$ cannot change fast enough to attain a point where equation (3.14) does not hold longer. $\|\boldsymbol{\omega}_{ib}^b\|$ almost always is small when using magnetic attitude control because of the detumbling procedure [24] and the small torques raised by the coils. The analysis of [2], that extends [21] for an unknown inertia, is therefore often near the reality in space. This effect applies even more for orbits with high inclination.

3.2.6 Ideas for Extension

The task is now to extend the presented stability analyses by removing assumptions to build a more realistic environment. The question is if it's still possible to prove stability of the equilibrium points.

It is difficult to remove the periodicity of the geomagnetic field from [2]. It is a central assumption ([40] and [5] assume it too) and without the condition (3.14) there is no point in considering the average system because stability cannot be shown longer (indefinite matrix). This thesis will thus disregard linear analysis and concentrate on extending [11] by introducing an unknown inertia matrix (with known bounds). Two main approaches with several methods to find an answer to the question above are distinguished.

- Keep the controller (3.10) and
  - prove exponential stability, because exponential stability is robust to vanishing perturbations.
  - find another Lyapunov function than (3.11) for the system with *known* inertia that has a negative definite time derivative, because a proof of asymptotic stability even for perturbations needs a Lyapunov function with negative definite time derivative.
  - find another Lyapunov function than (3.11) for the system with *unknown* inertia that is positive definite and has a negative (semi-) definite time derivative, because $\tilde{J}_x > \tilde{J}_y > \tilde{J}_z$ is no longer guaranteed then. In other words the Lyapunov

function is no longer positive definite because the gravity gradient torque may destabilize the attitude dynamics.

This is the subject of the next section.

- Study the robustness properties of other stabilizing controllers that
  - include the attitude term explicitly.
  - use sliding mode control.
  - use more advanced nonlinear control design techniques.

  This is done in section 3.4.

An unknown inertia matrix with known bounds seems to be a classical application area for adaptive control, where the unknown parameter is estimated online. However such adaptive control laws won't be considered with the controllers of [11] and [2] because they do not have gains or other values that are directly dependent on the satellite inertia. And making the control matrix **H** of the controller (3.10) dependent on the estimated inertia matrix does not remove the definiteness problems of the Lyapunov functions. Therefore the estimation of the actual inertia matrix does not help much. This is actually the case for a lot more of control laws in the literature on this particularly field, but online parameter estimation can be a good extension for sliding mode control (second subsection of section 3.4).

A limitation however is whether the system input is sufficiently rich, i.e. whether the *persistence of excitation* condition is satisfied, to estimate the three unknown moments of inertia $\tilde{J}_x$, $\tilde{J}_y$ and $\tilde{J}_z$ (assuming a diagonal inertia matrix).

3.3 Advanced Stability Analysis for the Control Law (3.10)

There are numerous control laws for spacecraft using magnetorquers so there in the beginning seems to be no need to design a new one. In addition the controller equation (3.10) was successfully demonstrated to work in simulations [11] and in the field with for example the Ørsted Satellite [40] that was launched in 1999 and is still operational 16 years later.

The following discussion of this section is based on the introduction and first section of chapter nine of Hassan Khalil's "Nonlinear Systems" [19].

The satellite attitude system with unknown inertia can be rewritten in the standard form $\dot{\mathbf{x}} = \tilde{\mathbf{f}}(t,\mathbf{x})$ with the general torque replaced with the gravity gradient (3.9) and the magnetic torque generated by the magnetorquers. The inertia matrix for a satellite will always be positive definite and thus invertible.

$$\begin{aligned}
\dot{\boldsymbol{\omega}}_{ob}^{b} &= \tilde{\mathbf{J}}^{-1}\left(-\omega_0\tilde{\mathbf{J}}\mathbf{S}(\mathbf{c}_1^b)\boldsymbol{\omega}_{ob}^{b} - \mathbf{S}(\boldsymbol{\omega}_{ob}^{b})\left(\tilde{\mathbf{J}}\boldsymbol{\omega}_{ob}^{b}\right) - \omega_0\mathbf{S}(\boldsymbol{\omega}_{ob}^{b})\left(\tilde{\mathbf{J}}\mathbf{c}_1^b\right) - \omega_0\mathbf{S}(\mathbf{c}_1^b)\left(\tilde{\mathbf{J}}\boldsymbol{\omega}_{ob}^{b}\right) - \omega_0^2\mathbf{S}(\mathbf{c}_1^b)\left(\tilde{\mathbf{J}}\mathbf{c}_1^b\right)\right.\\
&\quad \left. +3\omega_0^2\mathbf{S}(\mathbf{c}_3^b)\left(\tilde{\mathbf{J}}\mathbf{c}_3^b\right) + \mathbf{S}^{\mathrm{T}}(\mathbf{B}^b)\mathbf{H}\mathbf{S}(\mathbf{B}^b)\boldsymbol{\omega}_{ob}^{b}\right)\\
\dot{\mathbf{c}}_1^b &= \mathbf{S}(\mathbf{c}_1^b)\boldsymbol{\omega}_{ob}^{b}\\
\dot{\mathbf{c}}_3^b &= \mathbf{S}(\mathbf{c}_3^b)\boldsymbol{\omega}_{ob}^{b}
\end{aligned}$$

This can be viewed as a perturbation of the *nominal system* (3.1), (3.4) in standard form $\dot{\mathbf{x}} = \mathbf{f}(t,\mathbf{x})$ in the following way

$$\dot{\mathbf{x}} = \tilde{\mathbf{f}}(t,\mathbf{x}) = \mathbf{f}(t,\mathbf{x}) + \left(\tilde{\mathbf{f}}(t,\mathbf{x}) - \mathbf{f}(t,\mathbf{x})\right) = \mathbf{f}(t,\mathbf{x}) + \mathbf{g}(t,\mathbf{x}) \tag{3.15}$$

which is advantageous for the further analysis because the Lyapunov function (3.11) requires $J_x > J_y > J_z$ to be positive definite. This can of course not be guaranteed for an unknown inertia matrix $\tilde{\mathbf{J}}$. However perturbations in the form (3.15) permit statements on the perturbed system based on an analysis of the nominal system. In addition the perturbed system has an equilibrium point at the origin just as the nominal system, i.e.

$$\mathbf{f}(t,\mathbf{0}) = \mathbf{0}\,, \quad \tilde{\mathbf{f}}(t,\mathbf{0}) = \mathbf{0}\,, \qquad \mathbf{g}(t,\mathbf{0}) = \mathbf{0}$$

The last term is known as *vanishing perturbation* and makes the following steps easier.

3.3.1 Attempts to prove exponential Stability

The following lemma, taken from [19], suggests to try at first to prove exponential stability of the origin of the nominal system, that would imply exponential stability for the perturbed system.

Lemma 3.1 (Lemma 9.1 in [19]) *Let $x = 0$ be an exponentially stable equilibrium point of the nominal system*

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

Let $V(t,x)$ be a Lyapunov function of the nominal system that satisfies

$$c_1\|x\|^2 \leq V(t,x) \leq c_2\|x\|^2$$
$$\frac{\partial V}{\partial t} + \frac{\partial V}{\partial x} f(t,x) \leq -c_3\|x\|^2$$
$$\left\| \frac{\partial V}{\partial x} \right\| \leq c_4\|x\|$$

in $[0,\infty) \times D$. Suppose the perturbation term $g(t,x)$ satisfies

$$\|g(t,x)\| \leq \gamma\|x\|, \qquad \forall\, t \geq 0, \quad \forall\, x \in D$$
$$\gamma \leq \frac{c_3}{c_4}$$

Then, the origin is an exponentially stable equilibrium point of the perturbed system

$$\dot{x} = f(t,x) + g(t,x)$$

Moreover, if all the assumptions hold globally, then the origin is globally exponentially stable.

The first set of conditions is simply the direct Lyapunov method to prove exponential stability of the origin. The conditions on the perturbation term state that the vanishing perturbation must grow linearly in the norm of the state vector.

Lemma 3.1 in its pure form requires the knowledge of an appropriate Lyapunov function. But even in the case that it should be unknown, the qualitative statement of the lemma, that exponentially stable equilibrium points are robust to vanishing perturbations that grow linearly, always holds. The knowledge of the Lyapunov function (guaranteed to exist for exponentially stable equilibria) just adds a bound on the growth rate to this qualitative statement. For the question of this chapter a qualitative statement would already be a good answer.

In short the difference between *asymptotic* and *exponential* stability of an equilibrium point is that the Lyapunov function is positive definite and its time derivative negative definite in the *same order of magnitude* in all states, e.g. all states appear in positive quadratic terms in the Lyapunov function and negative quadratic terms in the time derivative.

The seven states are all present with positive quadratic terms in equation (3.11) but the time derivative (3.12) only has the angular velocity as negative quadratic terms, the other four states vanish in the analysis and exponential stability seems impossible to show. But

there are possibilities to show exponential stability even without a negative definite Lyapunov function, e.g. directly with the definition or using the following theorem, also taken from [19].

Theorem 3.1 (Theorem 8.5 in [19]) *Let $D \subset \mathbb{R}^n$ be a domain containing $x = 0$ and suppose $f(t,x)$ is piecewise continuous in t and locally Lipschitz in x for all $t \geq 0$ and $x \in D$. Let $x = 0$ be an equilibrium point for $\dot{x} = f(t,x)$ at $t = O$. Let $V : [0,\infty) \times D \to \mathbb{R}$ be a continuously differentiable function such that*

$$W_1(x) \leq V(t,x) \leq W_2(x)$$
$$\dot{V}(t,x) = \frac{\partial V}{\partial t} + \frac{\partial V}{\partial x} f(t,x) \leq 0$$
$$V(t+\delta, \phi(t+\delta;t,x)) - V(t,x) \leq -\lambda V(t,x), \qquad 0 < \lambda < 1^1$$

$\forall\, t \geq 0$, $x \in D$, for some $\delta > 0$, where $W_1(x)$ and $W_2(x)$ are continuous positive definite functions on D and $\phi(\tau;t,x)$ is the solution of the system that starts at (t,x). Then, the origin is uniformly asymptotically stable. If all the assumptions hold globally and $W_1(x)$ is radially unbounded, then the origin is globally uniformly asymptotically stable. If

$$W_1(x) \geq k_1 \|x\|^c, \quad W_2(x) \leq k_2 \|x\|^c, \qquad k_1 > 0, \quad k_2 > 0, \quad c > 0$$

then the origin is exponentially stable.

The difficulty here is to find the solution function which is often impossible to express in closed form. Nonetheless this theorem gives a good hint whether the attempts to prove exponential stability should be pursued or not.
As V defined by equation (3.11) is bounded from above and below by positive definite, radially unbounded and quadratic functions and because $\dot{V}$ (equation (3.12)) is negative semidefinite, the only condition to show global uniform exponential stability of the origin with **Theorem 3.1** is

$$V(t+\delta, \boldsymbol{\phi}(t+\delta;t,\mathbf{x})) - V(t,\mathbf{x}) \leq -\lambda V(t,\mathbf{x}), \qquad 0 < \lambda < 1$$

[1] There is no loss of generality in assuming that $\lambda < 1$, for if the inequality is satisfied with $\lambda_1 \geq 1$, then it is satisfied for any positive $\lambda < 1$, since $-\lambda_1 V \leq -\lambda V$. Notice, however, that this inequality could not be satisfied with $\lambda > 1$, since $V(t,x) > 0$, $\forall\, x \neq O$.

No formal proof for a against this condition was found, but intuitively it is not satisfied. Assume that $\boldsymbol{\omega}_{ob}^{b} \neq \mathbf{0}_{3\times 1}$ is constant. The left hand side simplifies to

$$\frac{3}{2}\omega_0^2\left(\left(\mathbf{c}_3^b(t+\delta)\right)^{\mathrm{T}}\mathbf{J}\mathbf{c}_3^b(t+\delta)-\left(\mathbf{c}_3^b(t)\right)^{\mathrm{T}}\mathbf{J}\mathbf{c}_3^b(t)\right)+\frac{1}{2}\omega_0^2\left(\left(\mathbf{c}_1^b(t)\right)^{\mathrm{T}}\mathbf{J}\mathbf{c}_1^b(t)-\left(\mathbf{c}_1^b(t+\delta)\right)^{\mathrm{T}}\mathbf{J}\mathbf{c}_1^b(t+\delta)\right)$$

that is the quadratic term $\frac{1}{2}\left(\boldsymbol{\omega}_{ob}^{b}\right)^{\mathrm{T}}\mathbf{J}\boldsymbol{\omega}_{ob}^{b}$ vanishes. The right hand side of the condition on the other hand still has this term and no matter how small λ is chosen, the condition most likely cannot be satisfied.

So in conclusion it is improbable that the controller makes the origin exponentially stable and thus robust to perturbations, Lemma 3.1 can't be applied and another approach must be found.

3.3.2 Searching another Lyapunov Function

The uniformly asymptotically stable origin of the nominal system does not almost automatically guarantee stability for the perturbed system like exponential stability of the origin does. The condition to show now is [19]

$$\left\|\frac{\partial V}{\partial \mathbf{x}}\mathbf{g}(t,\mathbf{x})\right\| < W_3(\mathbf{x}) \tag{3.16}$$

where $W_3(\mathbf{x})$ is a positive definite, time independent function that is the right hand side in $\dot{V} \leq -W_3$ for the nominal system. In other words the time derivative of the Lyapunov function along the trajectories of the perturbation must have a smaller magnitude than the time derivative of the Lyapunov function along the trajectories of the nominal system.

Here a problem arises in the present analysis because (3.12) is just negative *semidefinite* such that there is no positive definite function W_3 to test equation (3.16) with. However the converse Lyapunov theorems state that there is a function for the nominal system (3.1), (3.4), (3.9) and (3.10) with negative definite time derivative because the origin is uniformly asymptotically stable and the Jacobian matrix is bounded (the system is smooth, so there are no singularities) [19]. It remains just to find it.

[22] provides a systematic way in form of the following theorem to achieve this, when one has proven the asymptotic stability with Matrosov's theorem as done in [11]. The assumptions may seem long and difficult but in most of the cases when Matrosov's theorem can

successfully be applied, they're satisfied almost instantly. This is fortunately the case with the application of Matrosov's theorem done in [11].

Assumption 3.1 (Assumption 3.1 in [22]) *There exist an integer $j \geq 2$; known functions*

$$V_i : \mathscr{X} \to \mathbb{R},$$
$$\mathscr{N}_i : \mathscr{X} \to [0,\infty), \text{ and}$$
$$\phi_i : [0,\infty) \to [0,\infty);$$

and real numbers $a_i \in (0,1]$ such that $V_i(0) = 0$ and $\mathscr{N}_i(0) = 0$ for all i;

$$\nabla V_1(x) f(x) \leq -\mathscr{N}_1(x) \qquad \forall x \in \mathscr{X}; \text{ and}$$
$$\nabla V_i(x) f(x) \leq -\mathscr{N}_i(x) + \phi_i(V_1(x)) \sum_{l=1}^{i-1} \mathscr{N}_l^{a_i}(x) V_1^{1-a_i}(x)$$

for $i = 2, \ldots, j$ and all $x \in \mathscr{X}$. The function V_1 is also assumed to be positive definite on $\mathscr{X}$.

Assumption 3.2 (Assumption 3.2 in [22]) *The following conditions hold:*

1. *there exists a function $\rho : [0,\infty) \to (0,\infty)$ such that*

$$\sum_{l=1}^{j} \mathscr{N}_l(x) \geq \rho(V_1(x)) V_1(x) \qquad \forall x \in \mathscr{X}; \text{ and}$$

2. *there exist functions $p_2, \ldots, p_j : [0,\infty) \to [0,\infty)$ and a positive definite function $\bar{p}$ such that for each $i \in \{2, \ldots, j\}$, the following hold:*

   (a) *If V_i is positive definite, then*

$$p_i(r) = 0 \qquad \text{and} \qquad |V_i(x)| \leq \bar{p}(V_1(x))$$

   for all $r \geq 0$ and $x \in \mathscr{X}$.

   (b) *If V_i is not positive definite, then*

$$|V_i(x)| \leq p_i(V_1(x)) V_1(x)$$

   holds for all $x \in \mathscr{X}$.

Theorem 3.2 (Theorem 3.1 in [22]) *Let Assumptions 3.1 and 3.2 be satisfied. Then one can explicitly determine C^1 functions $k_l, \Omega_l \in \mathcal{K}_\infty$ such that the function*

$$S(x) = \sum_{l=1}^{j} \Omega_l \left(k_l(V_1(x)) + V_l(x)\right)$$

satisfies

$$S(x) \geq V_1(x)$$

and

$$\nabla S(x) f(x) \leq -\frac{1}{4}\rho(V_1(x)) V_1(x)$$

for all $x \in \mathcal{X}$.

The assumptions are satisfied with the following set of functions using the naming convention of [22] and the results of [11]

$$V_1(\mathbf{x}) = \frac{1}{2}\left(\boldsymbol{\omega}_{ob}^b\right)^{\mathrm{T}} \mathbf{J}\boldsymbol{\omega}_{ob}^b + \frac{3}{2}\left(\omega_0^2\left(\mathbf{c}_3^b\right)^{\mathrm{T}} \mathbf{J}\mathbf{c}_3^b - J_z\right) + \frac{1}{2}\omega_0^2\left(J_x - \left(\mathbf{c}_1^b\right)^{\mathrm{T}} \mathbf{J}\mathbf{c}_1^b\right) \tag{3.17a}$$

$$V_2(\mathbf{x}) = -\left(\mathbf{c}_3^b\right)^{\mathrm{T}} \mathbf{J}^{\mathrm{T}}\mathbf{S}^{\mathrm{T}}(\mathbf{c}_3^b)\left(\mathbf{J}\boldsymbol{\omega}_{ob}^b\right) \tag{3.17b}$$

$$V_3(\mathbf{x}) = \left(\mathbf{c}_1^b\right)^{\mathrm{T}} \mathbf{J}^{\mathrm{T}}\mathbf{S}^{\mathrm{T}}(\mathbf{c}_1^b)\left(\mathbf{J}\boldsymbol{\omega}_{ob}^b\right) \tag{3.17c}$$

$$\mathcal{N}_1(\mathbf{x}) = \left(\boldsymbol{\omega}_{ob}^b\right)^{\mathrm{T}} \mathbf{S}^{\mathrm{T}}(\mathbf{B}^b)\mathbf{H}\mathbf{S}(\mathbf{B}^b)\boldsymbol{\omega}_{ob}^b \tag{3.17d}$$

$$\mathcal{N}_2(\mathbf{x}) = 3\omega_0^2\left(\mathbf{c}_3^b\right)^{\mathrm{T}} \mathbf{J}^{\mathrm{T}}\mathbf{S}^{\mathrm{T}}(\mathbf{c}_3^b)\mathbf{S}(\mathbf{c}_3^b)\left(\mathbf{J}\mathbf{c}_3^b\right) \tag{3.17e}$$

$$\mathcal{N}_3(\mathbf{x}) = \omega_0^2\left(\mathbf{c}_1^b\right)^{\mathrm{T}} \mathbf{J}^{\mathrm{T}}\mathbf{S}^{\mathrm{T}}(\mathbf{c}_1^b)\mathbf{S}(\mathbf{c}_1^b)\left(\mathbf{J}\mathbf{c}_1^b\right) \tag{3.17f}$$

V_1 is positive definite and [11]

$$\dot{V}_1(\mathbf{x}) = -\mathcal{N}_1(\mathbf{x}) \tag{3.18a}$$

$$\begin{aligned}\dot{V}_2(\mathbf{x}) &= -\left(\dot{\mathbf{c}}_3^b\right)^{\mathrm{T}} \mathbf{J}^{\mathrm{T}}\mathbf{S}^{\mathrm{T}}(\mathbf{c}_3^b)\left(\mathbf{J}\boldsymbol{\omega}_{ob}^b\right) - \left(\mathbf{c}_3^b\right)^{\mathrm{T}} \mathbf{J}^{\mathrm{T}}\mathbf{S}^{\mathrm{T}}(\dot{\mathbf{c}}_3^b)\left(\mathbf{J}\boldsymbol{\omega}_{ob}^b\right) - \left(\mathbf{c}_3^b\right)^{\mathrm{T}} \mathbf{J}^{\mathrm{T}}\mathbf{S}^{\mathrm{T}}(\mathbf{c}_3^b)\left(\mathbf{J}\dot{\boldsymbol{\omega}}_{ob}^b\right) \\ &\leq -\mathcal{N}_2(\mathbf{x}) + \nu_2\left(\boldsymbol{\omega}_{ob}^b\right)^{\mathrm{T}} \boldsymbol{\omega}_{ob}^b\end{aligned} \tag{3.18b}$$

$$\begin{aligned}\dot{V}_3(\mathbf{x}) &= \left(\dot{\mathbf{c}}_1^b\right)^{\mathrm{T}} \mathbf{J}^{\mathrm{T}}\mathbf{S}^{\mathrm{T}}(\mathbf{c}_1^b)\left(\mathbf{J}\boldsymbol{\omega}_{ob}^b\right) + \left(\mathbf{c}_1^b\right)^{\mathrm{T}} \mathbf{J}^{\mathrm{T}}\mathbf{S}^{\mathrm{T}}(\dot{\mathbf{c}}_1^b)\left(\mathbf{J}\boldsymbol{\omega}_{ob}^b\right) + \left(\mathbf{c}_1^b\right)^{\mathrm{T}} \mathbf{J}^{\mathrm{T}}\mathbf{S}^{\mathrm{T}}(\mathbf{c}_1^b)\left(\mathbf{J}\dot{\boldsymbol{\omega}}_{ob}^b\right) \\ &\leq -\mathcal{N}_3(\mathbf{x}) + \nu_4\left(\boldsymbol{\omega}_{ob}^b\right)^{\mathrm{T}} \boldsymbol{\omega}_{ob}^b + \nu_5\left(\mathbf{c}_3^b\right)^{\mathrm{T}} \mathbf{c}_3^b\end{aligned} \tag{3.18c}$$

The last two equations look a bit different from [11] because the bound is quadratic and not linear. That however is no mistake because if one replaces the $\dot{\mathbf{c}}_3^b$ terms in $\dot{V}_2$ with the

kinematic equation (3.1) and does the same in $\dot{V}_3$, the angular velocity appears in quadratic terms. This is easier to comply with the last condition of **Assumption 3.1**. $a_{1,2} = 1$ and the positive constants ϕ_2 and ϕ_3 adjust the coefficients of the quadratic terms such that $\phi_2 \mathscr{N}_1$, $\phi_3 \mathscr{N}_1$ and $\phi_3 \mathscr{N}_2$ are compatible with $\dot{V}_2$ and $\dot{V}_3$ and the last two right hand sides of the equation set (3.18).

The first condition of **Assumption 3.2** can be satisfied with $\rho = 1$, i.e.

$$\mathscr{N}_1(\mathbf{x}) + \mathscr{N}_2(\mathbf{x}) + \mathscr{N}_3(\mathbf{x}) \geq V_1(\mathbf{x}) \tag{3.19}$$

if the positive definite controller matrix $\mathbf{H}$ is chosen sufficiently large, the other terms follow directly. The second conditions are equally easy to satisfy with sufficiently large positive constants p_2 and p_3, i.e.

$$|V_2(\mathbf{x})| \leq p_2 V_1(\mathbf{x}), \qquad |V_3(\mathbf{x})| \leq p_3 V_1(\mathbf{x})$$

such that the conclusion of **Theorem 3.2** can be applied. There is a Lyapunov function V for which

$$V(\mathbf{x}) \geq V_1(\mathbf{x}), \qquad \dot{V}(\mathbf{x}) \leq -\frac{1}{4} V_1(\mathbf{x}) = -W_3(\mathbf{x})$$

holds globally. This means that the time derivative of the Lyapunov function is bounded from above by a negative definite function of the same order of magnitude than V itself. Note that the notation from the source [22] was harmonized with the notation of this thesis.

So if one compares, in the sense of the condition (3.16), that the time derivative of the Lyapunov function along the trajectories of the perturbation must have a smaller magnitude than the time derivative of the Lyapunov function along the trajectories of the nominal system, this equals

$$\left\| \frac{\partial V}{\partial \mathbf{x}} \mathbf{g}(t,\mathbf{x}) \right\| < W_3(\mathbf{x}) \tag{3.20a}$$

$$\left\| \frac{\partial V}{\partial \mathbf{x}} \left(\tilde{\mathbf{f}}(t,\mathbf{x}) - \mathbf{f}(t,\mathbf{x}) \right) \right\| = \left\| \frac{\partial V}{\partial \mathbf{x}} \tilde{\mathbf{f}}(t,\mathbf{x}) - \frac{\partial V}{\partial \mathbf{x}} \mathbf{f}(t,\mathbf{x}) \right\| < \frac{1}{4} V_1(\mathbf{x}) \tag{3.20b}$$

$$\begin{aligned} \left| \frac{1}{8} \left(\boldsymbol{\omega}_{ob}^b \right)^{\mathrm{T}} (\tilde{\mathbf{J}} - \mathbf{J}) \boldsymbol{\omega}_{ob}^b + \frac{3}{8} \left(\omega_0^2 \left(\mathbf{c}_3^b \right)^{\mathrm{T}} (\tilde{\mathbf{J}} - \mathbf{J}) \mathbf{c}_3^b - (\tilde{J}_z - J_z) \right) \right. \\ \left. + \frac{1}{8} \omega_0^2 \left((\tilde{J}_x - J_x) - \left(\mathbf{c}_1^b \right)^{\mathrm{T}} (\tilde{\mathbf{J}} - \mathbf{J}) \mathbf{c}_1^b \right) \right| < \frac{1}{8} \left(\boldsymbol{\omega}_{ob}^b \right)^{\mathrm{T}} \mathbf{J} \boldsymbol{\omega}_{ob}^b + \frac{3}{8} \left(\omega_0^2 \left(\mathbf{c}_3^b \right)^{\mathrm{T}} \mathbf{J} \mathbf{c}_3^b - J_z \right) \\ + \frac{1}{8} \omega_0^2 \left(J_x - \left(\mathbf{c}_1^b \right)^{\mathrm{T}} \mathbf{J} \mathbf{c}_1^b \right) \end{aligned} \tag{3.20c}$$

which is true for small perturbations in the range

$$\tilde{\mathbf{J}} = \mathrm{diag}(\tilde{J}_x, \tilde{J}_y, \tilde{J}_z) < \mathrm{diag}(2J_x, 2J_y, 2J_z) = 2\mathbf{J}$$

or referring to equation (3.8) for the eigenvalue conditions

$$\lambda_{\max}(\tilde{\mathbf{J}}) < 2\lambda_{\max}(\mathbf{J})$$

[22] also shows a systematic way to construct the new Lyapunov function of Theorem 3.2 explicitly. The construction follows the steps of the proof and begins with

$$k_1(s) = s\,, \qquad k_{2,3}(s) = s + p_{2,3}(s)s = (1 + p_{2,3})s$$

where the last equality holds because p_2 and p_3 are constants. The next lemma defines the Matrosov strict Lyapunov construction functions.

Lemma 3.2 (Lemma 3.1 in [22]) *The functions* $\{U_i\}$ *defined by*

$$U_1(x) = V_1(x) \qquad \text{and} \qquad U_i(x) = k_i(V_1(x)) + V_i(x) \qquad \text{for } i \geq 2$$

satisfy

$$2k_i(V_1(x)) + \bar{p}(V_1(x)) \geq U_i(x) \geq V_1(x) \qquad \text{for } i = 2, \ldots, j \quad \text{and all } x \in \mathscr{X}$$

The Matrosov strict Lyapunov construction functions for the present problem are

$$U_1(\mathbf{x}) = V_1(\mathbf{x})\,, \qquad U_2(\mathbf{x}) = (1 + p_2)V_1(\mathbf{x}) + V_2(\mathbf{x})\,, \qquad U_3(\mathbf{x}) = (1 + p_3)V_1(\mathbf{x}) + V_3(\mathbf{x})$$

The very systematic approach of the proof is too complicated with the equations of the set (3.17), so *Remark 3.5* to one example in [22] (not quoted here) is followed. The time derivatives are approximated

$$\begin{aligned}
\dot{U}_1(\mathbf{x}) &= -\mathscr{N}_1(\mathbf{x}) \\
\dot{U}_2(\mathbf{x}) &\leq -(1 + p_2)\mathscr{N}_1(\mathbf{x}) - \mathscr{N}_2(\mathbf{x}) + \nu_2 \left(\boldsymbol{\omega}_{ob}^b\right)^{\mathrm{T}} \boldsymbol{\omega}_{ob}^b \\
\dot{U}_3(\mathbf{x}) &\leq -(1 + p_3)\mathscr{N}_1(\mathbf{x}) - \mathscr{N}_3(\mathbf{x}) + \nu_4 \left(\boldsymbol{\omega}_{ob}^b\right)^{\mathrm{T}} \boldsymbol{\omega}_{ob}^b + \nu_5 \left(\mathbf{c}_3^b\right)^{\mathrm{T}} \mathbf{c}_3^b
\end{aligned}$$

and the next suggestion done using the first condition of Assumption 3.2, see equation (3.19),

$$\begin{aligned}\dot{U}_2(\mathbf{x}) + \dot{U}_3(\mathbf{x}) &\leq -\mathcal{N}_1(\mathbf{x}) - \mathcal{N}_2(\mathbf{x}) - \mathcal{N}_3(\mathbf{x}) + (\nu_2+\nu_4)\left(\boldsymbol{\omega}_{ob}^b\right)^{\mathrm{T}}\boldsymbol{\omega}_{ob}^b + \nu_5\left(\mathbf{c}_3^b\right)^{\mathrm{T}}\mathbf{c}_3^b \\ &\leq -V_1(\mathbf{x}) + (\nu_2+\nu_4)\left(\boldsymbol{\omega}_{ob}^b\right)^{\mathrm{T}}\boldsymbol{\omega}_{ob}^b + \nu_5\left(\mathbf{c}_3^b\right)^{\mathrm{T}}\mathbf{c}_3^b\end{aligned}$$

Now the Lyapunov function needs to be guessed. Clearly

$$V(\mathbf{x}) = U_2(\mathbf{x}) + U_3(\mathbf{x}) + (\phi_2+\phi_3+\phi_2\phi_3)V_1(\mathbf{x}) + \phi_3 V_2(\mathbf{x})$$

is greater or equal to $V_1(\mathbf{x})$. The time derivative is shown to be negative definite

$$\begin{aligned}\dot{V}(\mathbf{x}) &= \dot{U}_2(\mathbf{x}) + \dot{U}_3(\mathbf{x}) + (\phi_2+\phi_3+\phi_2\phi_3)\dot{V}_1(\mathbf{x}) + \phi_3\dot{V}_2(\mathbf{x}) \\ &\leq -V_1(\mathbf{x}) + (\nu_2+\nu_4)\left(\boldsymbol{\omega}_{ob}^b\right)^{\mathrm{T}}\boldsymbol{\omega}_{ob}^b + \nu_5\left(\mathbf{c}_3^b\right)^{\mathrm{T}}\mathbf{c}_3^b - (\phi_2+\phi_3+\phi_2\phi_3)\mathcal{N}_1(\mathbf{x}) \\ &\quad + \phi_3\left(-\mathcal{N}_2(\mathbf{x}) + \nu_2\left(\boldsymbol{\omega}_{ob}^b\right)^{\mathrm{T}}\boldsymbol{\omega}_{ob}^b\right) \\ &\leq -V_1(\mathbf{x}) + (\phi_2+\phi_3)\mathcal{N}_1(\mathbf{x}) + \phi_3\mathcal{N}_2(\mathbf{x}) - (\phi_2+\phi_3+\phi_2\phi_3)\mathcal{N}_1(\mathbf{x}) - \phi_3\mathcal{N}_2(\mathbf{x}) + \phi_2\phi_3\mathcal{N}_1(\mathbf{x}) \\ &\leq -V_1(\mathbf{x})\end{aligned}$$

This proof is just a sketch, but the arguments are correct up to the constants (e.g. ϕ_2 and ϕ_3) which are not defined explicitly. The Lyapunov function won't be expressed with more details than with the construction functions as it is done above because inserting the equations (3.17) would just lead to confusion, the complexity is already high enough. But the inequality confirms the analysis of equations (3.20) (the same steps without dividing by 4) that the attitude stabilization with the controller (3.10) first presented in [40], improved in [5] and further investigated in [11] is robust in the sense presented in section 3.2, thus adding this thesis to the list.

3.4 Robustness Analysis of other stabilizing Controllers

Asymptotic stability of the origin of the perturbed system has been shown in the preceding section. But only the sketch of a proof is provided because a formal proof is very difficult. But one has not to restrict oneself just to analyze known controllers. Another way how the magnetic torque of equation (3.7) is controlled may solve the problem of stable robust spacecraft

attitude stabilization with an easier proof. This on one hand offers of course a lot of new possibilities but on the other hand complicates the task as one cannot see right away which path will lead to a result, meaning to a stabilizing controller or to a proof that the chosen approach does not work, or to a dead end.

If one now no longer sees the system as perturbed according to equation (3.15), but just analyzes the dynamics with the unknown matrix, the main problem in finding a new stabilizing controller is that the Lyapunov function (3.12) is no longer guaranteed to be positive definite, because $\tilde{J}_x > \tilde{J}_y > \tilde{J}_z$ won't hold in general. If it was so, the whole proof in [11] would still work because all inertia matrices for rigid bodies are positive definite and there is no further assumption than the mentioned inequality for the moments of inertia. And this chapter would be superfluous.

3.4.1 Controller with Attitude Feedback

Beside an unknown inertia the difficulties with the preceding section, that analyzes the controller (3.10) of [11], have their source in the fact that it makes the time derivative of the Lyapunov function only negative semidefinite because equation (3.12) only has a term for $\boldsymbol{\omega}_{ob}^b$ and no terms for the direction cosines c_{21}^b, c_{31}^b, c_{13}^b and c_{23}^b. One might just want to include them in the control torque to make the time derivative negative definite. Unfortunately this is not so easy.

As already stated several times, magnetorquers can only raise torques perpendicular to the earth's magnetic field. To rigorously enforce this the control variable, the magnetic dipole moment $\mathbf{m}^b$, is not chosen at free will but incorporates in any case a cross product with the surrounding magnetic field. So designing a control law according to the time derivative of the Lyapunov function (3.12) (first line) begins always with

$$\begin{aligned} \dot{V} &= \left(\boldsymbol{\omega}_{ob}^b\right)^{\mathrm{T}} \boldsymbol{\tau}_m^b = \left(\boldsymbol{\omega}_{ob}^b\right)^{\mathrm{T}} \left(-\mathbf{S}(\mathbf{B}^b)\mathbf{m}^b\right) = \left(\boldsymbol{\omega}_{ob}^b\right)^{\mathrm{T}} \left(-\mathbf{S}(\mathbf{B}^b)\mathbf{S}(\mathbf{r}^b)\mathbf{B}^b\right) \\ &= \left(\boldsymbol{\omega}_{ob}^b\right)^{\mathrm{T}} \left(-\mathbf{S}(\mathbf{B}^b)\mathbf{S}^{\mathrm{T}}(\mathbf{B}^b)\mathbf{r}^b\right) \end{aligned} \tag{3.21}$$

where $\mathbf{r}^b$ is a free to chose control law. It is now easy to include a term for $\boldsymbol{\omega}_{ob}^b$ that makes $\dot{V}$ negative semidefinite (as done in the control law of [11]), but one cannot easily add a term for the direction cosines when considering a general magnetic field. Then the matrix $\mathbf{S}(\mathbf{B}^b)\mathbf{S}^{\mathrm{T}}(\mathbf{B}^b)$ is just positive semidefinite and has no inverse.

That is also an explanation for the requirement (3.14) assumed by [2] that first allows the averaging method to be applicable. This chapter however aims to analyze robust attitude stabilization without posing any restricting condition on the earth's magnetic field.

Projection Method

There is at least a possibility to partially cope with the problem mentioned in the preceding paragraphs: the *projection method*. It is widely used and will also be part of the control algorithm for NUTS, see e.g. [38] and [29].

The logical inclusion of the attitude to the control law (3.10) follows from this extension to the Lyapunov function (3.11) with the attitude quaternion $\mathbf{q} = \begin{pmatrix} \boldsymbol{\epsilon}^{\mathrm{T}} & \eta \end{pmatrix}^{\mathrm{T}}$

$$V = \frac{1}{2}\left(\boldsymbol{\omega}_{ob}^{b}\right)^{\mathrm{T}} \tilde{\mathbf{J}} \boldsymbol{\omega}_{ob}^{b} + \frac{3}{2}\left(\omega_0^2 \left(\mathbf{c}_3^b\right)^{\mathrm{T}} \tilde{\mathbf{J}} \mathbf{c}_3^b - \tilde{J}_z\right) + \frac{1}{2}\omega_0^2 \left(\tilde{J}_x - \left(\mathbf{c}_1^b\right)^{\mathrm{T}} \tilde{\mathbf{J}} \mathbf{c}_1^b\right) + k\left(\boldsymbol{\epsilon}^{\mathrm{T}}\boldsymbol{\epsilon} + (1-\eta)^2\right) \tag{3.22}$$

where k is a scalar that must be large enough to ensure positive definiteness of V even for an unknown inertia matrix. Note that [29] expresses the Lyapunov function slightly differently and that the BODY axes x and y are swapped. Expanding the functions into an equation of scalars shows however that they're the same. The states to consider now are the angular velocity vector $\boldsymbol{\omega}^b$ and the quaternion $\mathbf{q}$ which can also substitute the direction cosines $c_{11}^b, \ldots, c_{33}^b$ according to the attitude representation methods of section 3.2.

The time derivative

$$\dot{V} = \left(\boldsymbol{\omega}_{ob}^{b}\right)^{\mathrm{T}} \left(\boldsymbol{\tau}_m^b + k\boldsymbol{\epsilon}\right) \tag{3.23}$$

could be made negative semidefinite with the torque

$$\boldsymbol{\tau}_m^b = -d\boldsymbol{\omega}_{ob}^{b} - k\boldsymbol{\epsilon}$$

that eliminates the attitude and leaves a negative quadratic function of $\boldsymbol{\omega}_{ob}^{b}$. d and k are two positive scalars. But this torque can in general not be generated by magnetorquers only. If it were so the proof for asymptotic stability of the origin of [11] would also work for an unknown inertia matrix as long as k is sufficiently large to make the Lyapunov function (3.22) always positive definite.

Combining the time derivative (3.23) with the torque parametrization as done in (3.21)

yields

$$\begin{aligned}\dot{V} &= \left(\boldsymbol{\omega}_{ob}^{b}\right)^{\mathrm{T}}\left(-\mathbf{S}(\mathbf{B}^{b})\mathbf{S}^{\mathrm{T}}(\mathbf{B}^{b})\mathbf{r}^{b} + k\boldsymbol{\epsilon}\right) \\ &= \left(\boldsymbol{\omega}_{ob}^{b}\right)^{\mathrm{T}}\left(-\frac{d}{\|\mathbf{B}^{b}\|^{2}}\mathbf{S}(\mathbf{B}^{b})\mathbf{S}^{\mathrm{T}}(\mathbf{B}^{b})\boldsymbol{\omega}_{ob}^{b} - \frac{k}{\|\mathbf{B}^{b}\|^{2}}\mathbf{S}(\mathbf{B}^{b})\mathbf{S}^{\mathrm{T}}(\mathbf{B}^{b})\boldsymbol{\epsilon} + k\boldsymbol{\epsilon}\right)\end{aligned}$$

if one uses the control law for NUTS [38]

$$\mathbf{m}^{b} = -\frac{1}{\|\mathbf{B}^{b}\|^{2}}\mathbf{S}(\mathbf{B}^{b})\left(d\boldsymbol{\omega}_{ob}^{b} + k\boldsymbol{\epsilon}\right) \tag{3.24}$$

The last two terms in $\dot{V}$ do not vanish so no statement about the definiteness is possible in general, but the inclusion of the $\frac{1}{\|\mathbf{B}^{b}\|^{2}}$ factor in the control law scales the torque such that a projection of the ideal torque onto the feasible region, i.e. the plain perpendicular to $\mathbf{B}^{b}$, is the same than the torque actually raised by the controller (3.24), see e.g. [24] for a derivation. One also can see intuitively that the division makes the magnitudes of the last two terms equal such that their contribution to $\dot{V}$ is small.
Note also that the control law (3.24) has the same structure than the control law (3.13) of [2], the first term is also the same than the control law (3.10) of [11] if the gain matrix is chosen to be $\mathbf{H} = \mathrm{diag}(d, d, d)$.

To summarize the last paragraphs, there is no easy way to include the attitude explicitly in the control law in such a way that attitude stabilization is asymptotically stable when magnetorquers are the only actuators. A rigorous proof, meaning a negative (semi-) definite Lyapunov function time derivative, is already impossible when there is no assumption about the properties of the geomagnetic field, even if the inertia matrix is perfectly known. Robustness issues with an unknown inertia are then of course even more difficult. It remains then only to follow the analysis and assumptions of [2].

Remarks on the Gravity Gradient Torque

Even if the magnitude of the gravity gradient torque is comparable with the torques generated by the coils on board, it is legitimate to suggest to remove the gravity gradient torque (3.9) from the analysis. The energy due to it and the potential energy in the Lyapunov function (3.11) are the reasons why $J_x > J_y > J_z$ must hold (that is the gravity gradient has stabilizing effect), which obviously is difficult to satisfy with an unknown inertia matrix. These two energies were included in [11] to have terms for four direction cosines in the Lyapunov

function, thus considering the satellite attitude in the analysis. The extended Lyapunov function (3.22) however has a positive definite term of the attitude with the last summand $k\left(\boldsymbol{\epsilon}^{\mathrm{T}}\boldsymbol{\epsilon}+(1-\eta)^2\right)$ and one can therefore remove the energy due to gravity gradient and potential without impeding the positive definiteness of the function. In fact it becomes easier.

The reduced function

$$V=\frac{1}{2}\left(\boldsymbol{\omega}_{ib}^{b}\right)^{\mathrm{T}}\tilde{\mathbf{J}}\boldsymbol{\omega}_{ib}^{b}+k\left(\boldsymbol{\epsilon}^{\mathrm{T}}\boldsymbol{\epsilon}+(1-\eta)^2\right)$$

is always positive definite for the state vector $\mathbf{x}=\begin{pmatrix}\left(\boldsymbol{\omega}_{ib}^{b}\right)^{\mathrm{T}} & \mathbf{q}^{\mathrm{T}}\end{pmatrix}^{\mathrm{T}}$ because every rigid body has a positive definite inertia matrix and $k>0$.
The time derivative using the kinetics (3.3), the known result from [29] and just the magnetic torque is

$$\dot{V}=\left(\boldsymbol{\omega}_{ib}^{b}\right)^{\mathrm{T}}\left(\boldsymbol{\tau}_{m}^{b}+k\boldsymbol{\epsilon}\right)$$

and thus almost equal to the time derivative (3.23) obtained in the analysis with gravity gradient. The only difference is that the angular velocity is given with respect to the inertial frame and not the ORBIT frame. The difference is however marginal when it comes to robustness of the controller and more a matter of defining what attitude stabilization is.

Anyway one arrives quasi at the same point than earlier or more precise just at the derivation of [2], where the gravity gradient torque is not considered either.

3.4.2 Sliding Mode Control

Sliding mode control is one of the most prominent nonlinear control design tools. The idea is to find a manifold in the state space on which all trajectories converge on the equilibrium and force all states not on the manifold to reach it with an appropriate controller. Thus two steps are needed to develop the controller: *sliding manifold design* and *sliding condition design*. Sliding mode control in general is explained for example in [19].

The PhD thesis [39] has a chapter where sliding mode control is used to get an attitude control law for a satellite with magnetorquers only. The papers [27] and [28] extend this approach. In the following their derivations will be analyzed in the light of this thesis, meaning the performance of the controller will be inspected for a general earth's magnetic field and an unknown satellite inertia matrix.

Sliding Manifold Design

The sliding manifold is defined as the hyperplane in the state space where the sliding variable $\mathbf{s}$ is zero. For satellite attitude dynamics the definition [39]

$$\mathbf{s} = \boldsymbol{\omega}_{ob}^{b} + k_q \boldsymbol{\epsilon} \tag{3.25}$$

has the desired effect that all trajectories where $\mathbf{s} = \mathbf{0}_{3\times1}$, converge on the equilibrium point $\boldsymbol{\omega}_{ob}^{b} = \mathbf{0}_{3\times1}$ and $\boldsymbol{\epsilon} = \mathbf{0}_{3\times1}$ exponentially fast. k_q is a positive design constant but can for more generality be replaced with a positive definite matrix. Note that just the imaginary part $\boldsymbol{\epsilon}$ of the quaternion is part of the sliding variable (3.25) because the unity requirement of the quaternion will force the real part η to reach ±1 (the desired attitude) on the sliding manifold.

One very nice side of this particular sliding manifold, is that it is robust in the sense of this thesis. The proof in [39] using Lyapunov theory is independent of the satellite inertia and the geomagnetic field. In fact for any spacecraft a controller that solely maintains the condition $\mathbf{s} = \mathbf{0}_{3\times1}$, will bring the attitude to the equilibrium because just the kinematics (3.2) appear in the proof. The positive definite Lyapunov function

$$V = \boldsymbol{\epsilon}^{\mathrm{T}}\boldsymbol{\epsilon} + (1-\eta)^2 = 2(1-\eta)$$

has a negative definite time derivative

$$\dot{V} = \left(\boldsymbol{\omega}_{ob}^{b}\right)^{\mathrm{T}} \boldsymbol{\epsilon} = -k_q \boldsymbol{\epsilon}^{\mathrm{T}}\boldsymbol{\epsilon}$$

on the manifold, where $\boldsymbol{\omega}_{ob}^{b} = -k_q\boldsymbol{\epsilon}$. The angular velocity is not explicitly part of the Lyapunov function but it is of course zero if the attitude does not change.

Sliding Condition Design

Up to this point the three sources, [39], [27] and [28] in chronological order, are equal. But now each derives an own sliding mode control law that ensures that any solution to the dynamical system, starting on the manifold or not, will eventually reach it and thus the equilibrium point, which by this way is proven to be globally uniformly asymptotically stable. The newer sources claim to have better convergence properties than the older ones and provide simulations to underline this fact.

The control laws are in chronological order

$$\mathbf{m}^b_{[39]} = \frac{1}{\|\mathbf{B}^b\|^2}\mathbf{S}(\mathbf{B}^b)\left(\frac{1}{\|\mathbf{s}\|^2}\left(\left(\boldsymbol{\tau}_{eq} - k_s\mathbf{s}\right)^{\mathrm{T}}\mathbf{s}\right)\mathbf{s}\right) \tag{3.26}$$

$$\mathbf{m}^b_{[27]} = \frac{1}{\|\mathbf{B}^b\|^2}\mathbf{S}(\mathbf{B}^b)\left(\frac{1}{\|\mathbf{s}\|^2}\left(\left(\boldsymbol{\tau}_{eq} - k_s\mathbf{I}_{3\times3}\left|\|\boldsymbol{\omega}^b_{ob}\| - k_q\|\boldsymbol{\epsilon}\|\right|\mathrm{sgn}(\mathbf{s})\right)^{\mathrm{T}}\mathbf{s}\right)\mathbf{s}\right) \tag{3.27}$$

$$\mathbf{m}^b_{[28]} = \frac{1}{\|\mathbf{B}^b\|^2}\mathbf{S}(\mathbf{B}^b)\left(\boldsymbol{\tau}_{eq} - k_g\left(\boldsymbol{\omega}^b_{ob} + k_q\boldsymbol{\epsilon}\right) - k_s\mathrm{sgn}(\mathbf{s})\right) \tag{3.28}$$

$$\begin{aligned}\boldsymbol{\tau}_{eq} = {}& \mathbf{S}(\boldsymbol{\omega}^b_{ob} + \omega_0\mathbf{c}^b_1)\left(\hat{\mathbf{J}}(\boldsymbol{\omega}^b_{ob} + \omega_0\mathbf{c}^b_1)\right) - 3\omega_0^2\mathbf{S}(\mathbf{c}^b_3)\left(\hat{\mathbf{J}}\mathbf{c}^b_3\right) + \omega_0^2\hat{\mathbf{J}}\mathbf{S}(\mathbf{c}^b_1)\boldsymbol{\omega}^b_{ob} \\ & - \frac{1}{2}k_q\hat{\mathbf{J}}\left(\boldsymbol{\omega}^b_{ob}\eta + \mathbf{S}(\boldsymbol{\omega}^b_{ob})\boldsymbol{\epsilon}\right)\end{aligned} \tag{3.29}$$

where $\boldsymbol{\tau}_{eq}$ aims to cancel the kinetics and kinematics with unknown inertia expressed in terms of the sliding variable

$$\begin{aligned}\tilde{\mathbf{J}}\dot{\mathbf{s}} = {}& -\mathbf{S}(\boldsymbol{\omega}^b_{ob} + \omega_0\mathbf{c}^b_1)\left(\tilde{\mathbf{J}}(\boldsymbol{\omega}^b_{ob} + \omega_0\mathbf{c}^b_1)\right) + 3\omega_0^2\mathbf{S}(\mathbf{c}^b_3)\left(\tilde{\mathbf{J}}\mathbf{c}^b_3\right) - \omega_0^2\tilde{\mathbf{J}}\mathbf{S}(\mathbf{c}^b_1)\boldsymbol{\omega}^b_{ob} \\ & + \frac{1}{2}k_q\tilde{\mathbf{J}}\left(\boldsymbol{\omega}^b_{ob}\eta + \mathbf{S}(\boldsymbol{\omega}^b_{ob})\boldsymbol{\epsilon}\right) + \boldsymbol{\tau}^b_m\end{aligned} \tag{3.30}$$

using the estimation $\hat{\mathbf{J}}$ to the unknown inertia matrix $\tilde{\mathbf{J}}$.

Equation (3.29) for $\boldsymbol{\tau}_{eq}$ is slightly different in all the three sources because of the respective definitions, here the (notationally harmonized) version of [39] is reproduced because it uses the same conventions than this thesis. One also recognizes the projection term with the cross product and scaling with the magnetometer measurement. A second projection is inside the large parentheses for the controllers (3.26) and (3.27) where only the desired torque components *parallel* to the sliding variable $\mathbf{s}$ are retained. The newest controller (3.28) does not use this because the newer approach *integral* sliding mode control is used.

There is an important problem with the sliding conditions above, namely that the (unknown) inertia matrix actually is needed in the $\boldsymbol{\tau}_{eq}$ term to really cancel out the torque. An easy workaround is provided by [39]. If $\hat{\mathbf{J}}$ used in $\boldsymbol{\tau}_{eq}$ is the best (constant) approximation of $\tilde{\mathbf{J}}$ and the difference $\Delta\mathbf{J} = |\tilde{\mathbf{J}} - \hat{\mathbf{J}}|$ is bounded (e.g. $\Delta\mathbf{J} \leq \lambda_{\max}(\tilde{\mathbf{J}})\mathbf{I}_{3\times3} - \lambda_{\min}(\tilde{\mathbf{J}})\mathbf{I}_{3\times3}$), then the control law will stabilize the attitude if

$$k_s > \sigma_{\max}(\Delta\boldsymbol{\tau}_{eq})$$

where $\sigma_{\max}(\Delta\boldsymbol{\tau}_{eq})$ is largest singular value of $\boldsymbol{\tau}_{eq}$ with $\Delta\mathbf{J}$ instead of $\hat{\mathbf{J}}$ in equation (3.29). Note that this only works well if the estimate of the inertia matrix is good, $\hat{\mathbf{J}}$ and $\tilde{\mathbf{J}}$ are of the same

order of magnitude.

One further advantage of sliding mode control in the eyes of this thesis, is that the analysis with unknown inertia matrix becomes easier as the Lyapunov function

$$V = \frac{1}{2}\mathbf{s}^{\mathrm{T}}\tilde{\mathbf{J}}\mathbf{s} \tag{3.31}$$

always is positive definite. Depending on the controller $\dot{V} = \mathbf{s}^{\mathrm{T}}\tilde{\mathbf{J}}\dot{\mathbf{s}}$ takes the form

$$\dot{V}_{[39]} \leq -\frac{k_s}{\|\mathbf{B}^b\|^2\|\mathbf{s}\|^2}\left(\mathbf{S}(\mathbf{B}^b)\mathbf{s}\right)^{\mathrm{T}}\left(\mathbf{S}(\mathbf{B}^b)\mathbf{s}\right) \tag{3.32}$$

$$\dot{V}_{[27]} \leq -\frac{k_s\mathrm{sgn}(\mathbf{s})^{\mathrm{T}}\mathbf{s}}{\|\mathbf{B}^b\|^2\|\mathbf{s}\|^2}\left|\|\boldsymbol{\omega}_{ob}^b\| - k_q\|\boldsymbol{\epsilon}\|\right|\left(\mathbf{S}(\mathbf{B}^b)\mathbf{s}\right)^{\mathrm{T}}\left(\mathbf{S}(\mathbf{B}^b)\mathbf{s}\right) \tag{3.33}$$

where the inequality comes from the fact that the dynamics (3.30) are not compensated exactly by the torque of equation (3.29) but dominated with help of the other term of the controller if k_s is sufficiently large.

The question is now if the time derivative of the Lyapunov function is negative definite to prove asymptotic stability and thus that all trajectories reach the sliding manifold. Clearly this is not the case in all scenarios for the first two controllers (3.26) and (3.27). The time derivatives (3.32) and (3.33) are zero not only for $\mathbf{s} = \mathbf{0}_{3\times1}$, but also when the sliding variable is parallel to the earth's magnetic field. In this case the control torque is also zero, hence the controller makes the equilibrium (at least) uniformly stable but not asymptotically.
To prove asymptotic stability using the controller (3.26) [39] again needs to assume a periodic geomagnetic field along the orbit. In addition k_q in equation (3.25), the definition of the sliding variable, must be quite large or $\mathbf{s}$, and thus the control torque raised by $\mathbf{m}^b$, will become small rapidly, this would harm the convergence properties. But on the other hand a too high value is not good either and could even lead to instability [39].
This is where [27] intervenes. In the general sliding mode control theory, controllers with discontinuous ($k_s\mathrm{sgn}(\mathbf{s})$) terms have better convergence properties than those with continuous ($k_s\mathbf{s}$) terms, because the control action is not too small for $\mathbf{s} \approx \mathbf{0}_{3\times1}$. But a lot of controllers have difficulties to actually do this, because very fast sign changes in the control action are needed. But magnetorquers can switch very rapidly, instantly if one compares the fast switching time with the slow attitude dynamics, so a discontinuous controller can be implemented. However $\mathrm{sgn}(\mathbf{s})$ instead of $\mathbf{s}$ makes the time derivative (3.32) of the Lya-

punov function indefinite. This is why [27] adds $\left|\|\boldsymbol{\omega}_{ob}^{b}\| - k_q\|\boldsymbol{\epsilon}\|\right|$. Now there are no cases where $\dot{V} > 0$ and the convergence properties are better with the controller (3.27) than with the controller (3.26). However a general surrounding magnetic field is not assumed either, it still needs to satisfy the known condition (3.14), that for example also [2] assumes.

For the newest paper [28], that uses the integral sliding mode control approach, the time derivative $\dot{V} = \mathbf{s}^{\mathrm{T}}\tilde{\mathbf{J}}\dot{\mathbf{s}}$ becomes

$$\dot{V}_{[28]} = -\mathbf{s}^{\mathrm{T}}\left(k_s\frac{\mathbf{S}^{\mathrm{T}}(\mathbf{B}^b)\mathbf{S}(\mathbf{B}^b)}{\|\mathbf{B}^b\|^2}\mathrm{sgn}(\mathbf{s}) - \boldsymbol{\tau}_d\right) \tag{3.34}$$

where $\boldsymbol{\tau}_d$ summarizes the disturbance torques. This looks similar to the Lyapunov function derivative (3.12), if one disregards the disturbance torque, with the important difference that $\mathbf{s}$ regroups both the angular velocity and the attitude, and not just the angular velocity as it is in equation (3.12). This means that the controller (3.28) achieves asymptotic stability of the equilibrium point if [28]

$$k_s > \sup_t\left(\max_i\left|\tau_{d_i}(t)\right|\right), \qquad t > t_0\,, \quad i = 1,2,3$$

i.e. the disturbance torque is bounded and the control gain k_s is large enough to make $\dot{V}$ negative definite.

With this result a solution to the problem inspected in this chapter is found. If the estimated inertia matrix $\hat{\mathbf{J}}$ in equation (3.29) is close to the real and unknown matrix $\tilde{\mathbf{J}}$, the model error can be seen as a part of the disturbance torques. If the condition above still holds despite the possibly higher disturbance, the controller will make the origin globally uniformly asymptotically stable in the robust sense assumed here, that is independent of the exact satellite inertia and with a general surrounding geomagnetic field.

The source provides a simulation that confirms the successful operation of the controller. And even if the inertia is known perfectly well, this simulation is interesting for this thesis as the requirement of [11], i.e. $J_x > J_y > J_z$, is *not* satisfied. This shows that integral sliding mode control can stabilize spacecraft where the gravity gradient has destabilizing effect, therefore it is legitimate to add the equivalent torque of a disturbed inertia matrix in the attitude dynamics to the disturbance torque.

To summarize the last paragraphs the three sliding mode controllers that were inspected, are all somewhat robust to perturbations in the satellite inertia. Obviously these perturba-

tions should not be too high. This counts especially for the oldest controller (3.26). When it comes to the properties of the earth's magnetic field, both [39] and [27] require it to be periodic. But the newest sliding mode controller presented in [28] satisfies very well the robustness demands of this thesis. One could even relax other assumptions as nearly every difference between the idealization in the modelling and the real world results in disturbance torques. And the controller is actually robust against bounded disturbance torques, where they originate is of secondary importance. So the controller of equation (3.28) is maybe the best control law to solve the problem of this chapter with reasonable complexity.

Online Parameter Estimation

The three sliding mode controllers (3.26), (3.27) and (3.28) in first place aim to completely cancel out the torque (3.29) due to kinetic energy, potential energy and gravity gradient. This is not perfectly possible when the inertia matrix is unknown. Until now a constant estimate was considered and the difference to the actual torque added to the disturbances in the hope that they're dominated by the other controller terms. Intuitively a better performance is possible if the estimate $\hat{\mathbf{J}}$ is not constant but updated based on the states of the system.

These paragraphs just show an idea how this can be implemented, but it is unclear whether this approach works. The high complexity of the online parameter estimation law makes an analysis of the convergence properties difficult (if not impossible). In addition the input to the system, the magnetic dipole moment generated by the coils, may not be sufficiently rich to satisfy the persistence of excitation requirement.

The estimate of the inertia shall no longer be constant, thus an update law $\dot{\hat{\mathbf{J}}}$ must be found. Since the unknown parameters do not appear inside the nonlinearities of the system, a classical approach leads to the update law. The Lyapunov function for the sliding mode control (3.31) is augmented with the squared differences of the estimated and real values

$$V = \frac{1}{2}\mathbf{s}^{\mathrm{T}}\tilde{\mathbf{J}}\mathbf{s} + \frac{1}{2\gamma_x}(\hat{J}_x - \tilde{J}_x)^2 + \frac{1}{2\gamma_y}(\hat{J}_y - \tilde{J}_y)^2 + \frac{1}{2\gamma_z}(\hat{J}_z - \tilde{J}_z)^2$$

When the control law is equation (3.28), the time derivative becomes (see equation (3.34))

$$\dot{V} = -\mathbf{s}^{\mathrm{T}}\left(k_s \frac{\mathbf{S}^{\mathrm{T}}(\mathbf{B}^b)\mathbf{S}(\mathbf{B}^b)}{\|\mathbf{B}^b\|^2}\operatorname{sgn}(\mathbf{s}) - \boldsymbol{\tau}_d\right) + \frac{\dot{\hat{J}}_x}{\gamma_x}(\hat{J}_x - \tilde{J}_x) + \frac{\dot{\hat{J}}_y}{\gamma_y}(\hat{J}_y - \tilde{J}_y) + \frac{\dot{\hat{J}}_z}{\gamma_z}(\hat{J}_z - \tilde{J}_z)$$

Let now the only disturbance torque be the remains because of the wrong inertia matrix

in equation (3.29), then

$$\dot{V} = -\mathbf{s}^{\mathrm{T}}\left(\frac{\mathbf{S}^{\mathrm{T}}(\mathbf{B}^b)\mathbf{S}(\mathbf{B}^b)}{\|\mathbf{B}^b\|^2}\left(k_s\mathrm{sgn}(\mathbf{s}) - \mathbf{S}(\boldsymbol{\omega}_{ob}^b + \omega_0\mathbf{c}_1^b)\left((\hat{\mathbf{J}}-\tilde{\mathbf{J}})(\boldsymbol{\omega}_{ob}^b + \omega_0\mathbf{c}_1^b)\right)\right.\right.$$
$$\left.\left. +3\omega_0^2\mathbf{S}(\mathbf{c}_3^b)\left((\hat{\mathbf{J}}-\tilde{\mathbf{J}})\mathbf{c}_3^b\right) - \omega_0^2(\hat{\mathbf{J}}-\tilde{\mathbf{J}})\mathbf{S}(\mathbf{c}_1^b)\boldsymbol{\omega}_{ob}^b + \frac{1}{2}k_q(\hat{\mathbf{J}}-\tilde{\mathbf{J}})\left(\boldsymbol{\omega}_{ob}^b\eta + \mathbf{S}(\boldsymbol{\omega}_{ob}^b)\boldsymbol{\epsilon}\right)\right)\right)$$
$$+\frac{\dot{\hat{J}}_x}{\gamma_x}(\hat{J}_x - \tilde{J}_x) + \frac{\dot{\hat{J}}_y}{\gamma_y}(\hat{J}_y - \tilde{J}_y) + \frac{\dot{\hat{J}}_z}{\gamma_z}(\hat{J}_z - \tilde{J}_z)$$

This time derivative must be resolved in a function just containing scalars, that is the vectors are broken down into their components, the cross products and the matrix multiplication with $\mathbf{S}^{\mathrm{T}}(\mathbf{B}^b)\mathbf{S}(\mathbf{B}^b)$ calculated and finally the dot product $\mathbf{s}^{\mathrm{T}}$ with the resulting vector carried out. There are then very long and complicated factors multiplied with $(\hat{J}_x - \tilde{J}_x)$, $(\hat{J}_y - \tilde{J}_y)$ and $(\hat{J}_z - \tilde{J}_z)$. These terms can be compensated with the update laws that need to be defined

$$\dot{\hat{J}}_x = \gamma_x(\ldots), \qquad \dot{\hat{J}}_y = \gamma_y(\ldots), \qquad \dot{\hat{J}}_z = \gamma_z(\ldots)$$

such that the sums are zero. The dots stand for the very long factors that consist just of known variables (angular velocity components, direction cosines and components of $\mathbf{B}^b$). The gains γ_x, γ_y and γ_z are further positive constants that need to be tuned.

With this online parameter estimation, the sliding mode controller of [28] does not lose its good properties. On the contrary one can expect $\boldsymbol{\tau}_{eq}$ with the varying estimate for the inertia matrix, to cancel out the undesired torque better. This holds however only if the difference $(\hat{\mathbf{J}}-\tilde{\mathbf{J}})$ converges to zero, this is the condition that requires persistence of excitation of the system. Unfortunately there is legitimate doubt to this, because the satellite attitude dynamics are very slow and the magnetic dipole moment generated by the coils not expected to change fast either. To make things worse the update laws are very long and errors easily made during the implementation. The developer also needs again to tune gains. Finally the states of the system and the flux density of the geomagnetic field must be known and the inertia matrices $\tilde{\mathbf{J}}$ and $\hat{\mathbf{J}}$ diagonal. The latter can change if the principal axes of BODY no longer are those used in the design phase because of some change in the inertia. Then the update law becomes even more complicated and the persistence of excitation requirement stronger because more parameters must be estimated online. So in theory this is an improvement but the practical use is dubious. Anyway sliding mode control remains one of the best methods to do robust attitude stabilization with magnetorquers as only actuators.

3.4.3 Further Strategies

Beside the strategies developed and explained with larger details in the preceding subsections, some more advanced nonlinear control design techniques have been used for spacecraft attitude stabilization using magnetorquers. They won't be detailed out here, only some comments whether they're suited to solve the present problem, will be given.

Dynamic Neural Network

One of the most advanced techniques for nonlinear control design are *neural networks*, that are based on *dynamic neural units*. They can in general cope with any kind of nonlinearity and, in short terms, extend conventional adaptive laws for online estimation of unknown parameters.

Such a network "behind" a more classical attitude controller has very good robustness properties. Not just for perturbations in the system dynamics, such as a change in the spacecraft inertia, but also for example in case of actuator faults. [6] and [41] are two examples where a spacecraft with magnetorquers only is stabilized using dynamic neural units.

[41] even explicitly describes its controller as being intended for CubeSats. However once again the surrounding geomagnetic field is assumed to be periodic along the orbit of the CubeSat. Another limitation, that also appears in [6], is the higher complexity of the controller. There are a lot of parameters to tune and a lot of calculations to be done by the ADCS in orbit. Even if the power of modern microcontrollers increased very much in the past decades and still is increasing, there are doubts whether dynamic neural networks are suited for CubeSat missions.

So in conclusion while controllers based on dynamic neural networks seem one of the best methods to solve the present problem, a formal proof is not given and it is questionable whether the increased complexity compared to the PD controllers mentioned earlier (and which have proven themselves in the field numerous times), is worth the effort.

Fuzzy Logic

A completely different approach to the control design problem is to use linguistic terms to classify measurements and control variables of a system as for example "very high", "high", "normal", "low" and "very low". *If-then-rules*, i.e. the control laws, state which value must be applied to the control variable dependent on the classifications of the measurements.

The process to convert the measurement values to the classes and the process to convert the classes of the control variables to values are called *fuzzyfication* and *defuzzification* respectively. The whole system is called *fuzzy logic controller*.

Fuzzy logic control has not been investigated much for satellite applications, however [13] proposes an easy fuzzy controller to stabilize a spacecraft actuated by magnetorquers only. The simulations indicate good behavior also in presence of a constant disturbance torque, which could be interpreted as the result of the perturbed inertia matrix.

Unfortunately fuzzy logic controllers, due to their different nature, are difficult to analyze for stability properties, at least according to the formulation and with the methods of Lyapunov. But in general it requires quite significant changes in the model parameters to destabilize the system if the fuzzyfication was well done (which can be a long and difficult process), because small perturbations won't change the classification of the measurements.

To summarize, it is doubtful if a fuzzy logic controller has any advantage over the well-known and proven PD controllers. For this thesis at least they're not an alternative as formal stability proofs that are comparable to proofs for the other controllers of this chapter, do not exist. The only way would be to approximate the fuzzy logic controller with an analytical expression and insert it into an appropriate Lyapunov function. This is an elaborate task, just an approximation of the actual controller and would presumably in the end lead to very similar results than those of the control laws analyzed with more detail.

3.5 Conclusion and further Work

Several conclusions can be taken from this chapter. The most obvious one is that there are considerable difficulties to formally prove that attitude control of a spacecraft with magnetorquers works in general. One has to restrict on special cases an assume certain properties e.g. of for the geomagnetic field. Nonetheless magnetic attitude control has proven itself in the field and theoretical proofs exist for reasonable models of the spacecraft and surrounding field, this thesis adds two more to this list.

The special case this chapter looks at, is described in detail in section 3.2. The attitude is to be aligned to an outer reference frame of the satellite and this without assuming any special property for the geomagnetic field and allowing the inertia matrix of the rigid body to be unknown within upper and lower bounds, these must however be somewhat tight.

Section 3.3 shows then in its second part that a nonlinear D controller solves the problem. The approach leading to the solution is to use the stability proof of [11] (where the inertia is known and has a special form giving stabilizing effect) and to build a new and strict Lyapunov function. Unfortunately it is too elaborate to find the explicit form of this new Lyapunov function, but the requirements of a theorem are satisfied and this theorem guarantees that the function exists and gives bounds. The bounds are sufficient to prove that the controller stabilizes the spacecraft in the scenario of this chapter.
Nonetheless would it be nice to find a more formal proof for this claim, but this is left as further work. The main conclusion, namely that the control law (3.10) is robust, is more important than the actual tight bounds on the tolerance for the unknown inertia matrix.

Section 3.4 shows several attempts to solve the given problem.
The nonlinear PD controller cannot be proven to be robust in the sense of this thesis, however the paper [2] shows that this controller (equation (3.13)) has very good and robust stabilizing effect if the geomagnetic field takes a property that is fulfilled in a lot of orbits, especially those with high inclination. Therefore the attitude control law (3.24) for the NTNU Test Satellite, is a good choice between robustness and ease of implementation and operation.
The sliding mode control approach, especially the integral sliding mode controller of equation (3.28), is one of the best ways to solve the robust attitude stabilization problem. It probably works even better for the case of an unknown inertia matrix if this unknown parameter is estimated with an adaptive control law. However the expectations to this extension are not very high, because the persistence of excitation requirement for adaptive laws to converge may not be fulfilled, especially if nine (the inertia matrix may lose its diagonal property under operation) parameters need to be estimated. But the sliding mode control in itself is robust to almost any kind of perturbation.
Finally an outlook to two other strategies is given which most likely can stabilize the attitude robustly. However these more advanced techniques need considerably more processor power and are more difficult to implement, the dynamic neural network approach because of the complex nature of neural networks, the fuzzy logic approach because of the large number of it-then-rules that need to be defined.

This chapter remained purely theoretical. It would have been an enrichment to extend the simulations of the sources with variable inertia. This would also have given the opportunity to verify the claims. But the time was to short and this left as further work.

Bibliography

[1] Brown, R. G. and Hwang, P. Y. C. (2012). *Introduction To Random Signals And Applied Kalman Filtering: With MATLAB Exercises*. John Wiley, 4th edition.

[2] Celani, F. (2015). Robust three-axis attitude stabilization for inertial pointing spacecraft using magnetorquers. *Acta Astronautica*, 107:87–96.

[3] Chulliat, A., Macmillan, S., Alken, P., Beggan, C., Nair, M., Hamilton, B., Woods, A., Ridley, V., Maus, S., and Thomson, A. (2015). The US/UK World Magnetic Model for 2015-2020. Technical report, National Geophysical Data Center, NOAA.

[4] Compston, D. (2014). MATLAB File Exchange - International Geomagnetic Reference Field (IGRF) Model. `http://www.mathworks.com/matlabcentral/fileexchange/34388-international-geomagnetic-reference-field--igrf--model`.

[5] Damaren, C. J. (2002). Comments on "Fully magnetic attitude control for spacecraft subject to gravity gradient". *Automatica*, 38:2189.

[6] Das, S., Sinha, M., and Misra, A. (2012). Dynamic Neural Units for Adaptive Magnetic Attitude Control of Spacecrafts. *Journal of Guidance, Control, and Dynamics*, 35(4):1280–1291.

[7] Davis, J. (2004). Mathematical Modeling of Earth's Magnetic Field. Technical report, Virginia Tech, Blacksburg, VA 24061.

[8] Fossen, T. I. (2011). *Handbook of Marine Craft Hydrodynamics and Motion Control*. John Wiley & Sons Ltd., 1st edition.

[9] Ghuffar, S. (2009). Design and Implementation of Attitude Determination Algorithm for the Cubesat UWE-3. Master's thesis, Julius-Maximilians-Universität Würzburg, Faculty of Computer Science, Institute of Robotics and Telematics.

[10] Glatzmaier, G. A. (1995). The Geodynamo. http://es.ucsc.edu/~glatz/geodynamo.html.

[11] Gravdahl, J. T. (2004). Magnetic attitude control for satellites. *43rd IEEE Conference on Decision and Control, Atlantis, Paradise Island, Bahamas*, pages 261–266.

[12] Haave, H. R. (2014). Implementation of Attitude Estimation and a Look at the UWE CubeSat. Project thesis, Norwegian University of Science and Technology, Department of Engineering Cybernetics.

[13] Heydari, A., Pourtakdoust, S. H., and Heydari, H. (2009). Magnetic Attitude Control Using Fuzzy Logic. *2009 IEEE Multi-conference on Systems and Control, Saint Petersburg, Russia*, pages 456–460.

[14] Hoots, F. R. and Roehrich, R. L. (1980). Spacetrack Report no. 3 - Models for Propagation of NORAD Element Sets. Technical report, United States Department of Defense.

[15] International Association of Geomagnetism and Aeronomy (IAGA) (2014). The 12th Generation International Geomagnetic Reference Field. http://www.ngdc.noaa.gov/IAGA/vmod/igrf.html.

[16] International Earth Rotation and Reference Systems Service (IERS) (2015a). IERS Bulletin A - Rapid Service/Prediction of Earth Orientation. http://www.iers.org/IERS/EN/Publications/Bulletins/bulletins.html.

[17] International Earth Rotation and Reference Systems Service (IERS) (2015b). IERS Bulletin B - Monthly Earth Orientation Parameters. http://www.iers.org/IERS/EN/Publications/Bulletins/bulletins.html.

[18] Kelso, T. S. (2015). CelesTrak. http://www.celestrak.com/.

[19] Khalil, H. K. (2002). *Nonlinear Systems*. Prentice Hall, 3rd edition.

[20] Loría, A., Panteley, E., Popović, D., and Teel, A. R. (2002). An extension of Matrosov's theorem with application to stabilization of nonholonomic control systems. *Proceedings of the 41st IEEE Conference on Decision and Control, Las Vegas, Nevada USA*, pages 1528–1533.

[21] Lovera, M. and Astolfi, A. (2006). Global Magnetic Attitude Control of Spacecraft in the Presence of Gravity Gradient. *IEEE Transactions on Aerospace and Electronic Systems*, 42(3):796–805.

[22] Malisoff, M. and Mazenc, F. (2009). *Constructions of Strict Lyapunov Functions*. Communications and Control Engineering. Springer, 1st edition.

[23] National Aeronautics and Space Administration (2007). Polynomial Expressions for Delta T. `http://eclipse.gsfc.nasa.gov/SEcat5/deltatpoly.html`.

[24] Pignède, A. (2014). Detumbling of the NTNU Test Satellite. Project thesis, Norwegian University of Science and Technology, Department of Engineering Cybernetics.

[25] Psiaki, M. L. (2001). Magnetic Torquer Attitude Control via Asymptotic Periodic Linear Quadratic Regulation. *Journal of Guidance, Control, and Dynamics*, 24(2):386–394.

[26] Rein, Ø. (2014). Developing an ADCS Prototype for NTNU Test Satellite. Master's thesis, Norwegian University of Science and Technology, Department of Engineering Cybernetics.

[27] Sofyalı, A. and Jafarov, E. M. (2012). Purely Magnetic Spacecraft Attitude Control by Using Classical and Modified Sliding Mode Algorithms. *12th IEEE Workshop on Variable Structure Systems, VSS'12, Mumbai, Maharashtra*, pages 117–123.

[28] Sofyalı, A. and Jafarov, E. M. (2014). Integral Sliding Mode Control of Small Satellite Attitude Motion by Purely Magnetic Actuation. *19th IFAC World Congress, Cape Town, South Africa*, pages 7947–7953.

[29] Tudor, Z. (2011). Design and Implementation of Attitude Control for 3-axes Magnetic Coil Stabilization of a Spacecraft. Master's thesis, Norwegian University of Science and Technology, Department of Engineering Cybernetics.

[30] United States' National Oceanic and Atmospheric Administration's National Geophysical Data Center (NOAA/NGDC) and British Geological Survey (BGS) (2009). The enhanced magnetic model. `http://www.ngdc.noaa.gov/geomag/EMM/`.

[31] United States' National Oceanic and Atmospheric Administration's National Geophysical Data Center (NOAA/NGDC) and British Geological Survey (BGS) (2013). The high Definition Geomagnetic Model. `http://www.ngdc.noaa.gov/geomag/hdgm.shtml`.

[32] United States' National Oceanic and Atmospheric Administration's National Geophysical Data Center (NOAA/NGDC) and British Geological Survey (BGS) (2014c). Geomagnetism Frequently Asked Questions. https://www.ngdc.noaa.gov/geomag/faqgeom.shtml.

[33] United States' National Oceanic and Atmospheric Administration's National Geophysical Data Center (NOAA/NGDC) and British Geological Survey (BGS) (2014b). Magnetic Field Calculators. https://www.ngdc.noaa.gov/geomag-web/#igrfwmm.

[34] United States' National Oceanic and Atmospheric Administration's National Geophysical Data Center (NOAA/NGDC) and British Geological Survey (BGS) (2014a). The World Magnetic Model. http://www.ngdc.noaa.gov/geomag/WMM/DoDWMM.shtml.

[35] Vallado, D. A. (2013). *Fundamentals of Astrodynamics and Applications*. Space Technology Library. Microcosm Press, 4th edition.

[36] Vallado, D. A., Crawford, P., Hujsak, R., and Kelso, T. S. (2006). Revisiting Spacetrack Report #3: Rev 2. Technical report, American Institute of Aeronautics and Astronautics.

[37] Vallado, D. A. and Kelso, T. S. (2013). Earth Orientation Parameter and Space Weather Data for Flight Operations. *23rd AAS/AIAA Space Flight Mechanics Meeting, Kauai, HI*, pages 2679–2698.

[38] Westgaard, M. F. (2014). Hardware Testing of Attitude Control for the NTNU Test Satellite. Project thesis, Norwegian University of Science and Technology, Department of Engineering Cybernetics.

[39] Wiśniewski, R. (1996). *Satellite Attitude Control Using Only Electromagnetic Actuation*. PhD thesis, Aalborg University, Department of Control Engineering.

[40] Wiśniewski, R. and Blanke, M. (1999). Fully magnetic attitude control for spacecraft subject to gravity gradient. *Automatica*, 35:1201–12014.

[41] Zhang, M., Yin, L., and Qiao, L. (2014). Adaptive Fault Tolerant Attitude Control for Cube Satellite in Low Earth Orbit based on Dynamic Neural Network. *International Journal of Innovative Computing, Information and Control*, 10(5):1843–1852.

Curriculum Vitae

Name:	**Antoine François Xavier PIGNEDE**
Gender:	Male
Date of birth:	15 October 1991
Place of birth:	Darmstadt, Germany
Address:	Edgar B. Schieldropsvei 15 Apart. 13, N–7033 Trondheim
Nationality:	French
Email :	afpigned@stud.ntnu.no
Telephone:	+47 94494918

Language Skills

Written and spoken skills in *French* and *German*, native languages, *English* and *Norwegian*, fluent.

Education

- 1997 - 2001: Primary School, Heinrich-Heine-Schule in Darmstadt
- 2001 - 2010: Secondary School, Edith-Stein-Schule in Darmstadt
  Degree: Abitur (1.1)
- 2010 - 2013: University, Technische Universität in Darmstadt
  Degree: Bachelor of Science Electrical Engineering and Information Technology, Specialization Automation Systems (1.45)
  Student assistant during the last four semesters
- 2013 - 2016: Double Degree

  - 2013 - 2015: MSc Engineering Cybernetics, Field of Study Control Engineering at Norges teknisk-naturvitenskapelige universitet in Trondheim
  - 2015 - 2016: MSc Electrical Engineering and Information Technology, Specialization Automation Systems at Technische Universität in Darmstadt

Computer Skills

Participation in a computer writing course (2007).
Good knowledge of

- Office programs, i.e. Word, Excel and PowerPoint
- Typesetting markup language LaTeX
- Programming languages Java, Python, C/C++, JavaScript. MATLAB programming language and SIMULINK graphical package. Modelica modelling language
- Ubuntu Linux operating system

Experience

- Summer job (full time) at Sensorlink AS in Trondheim from 30 June to 22 August 2014 and from 1 June 2015 to 24 July 2015
- Part time job at Sensorlink AS in Trondheim from 3 September 2014 to 28 May 2015

Hobbies and Other Activities

- 15 years of classical piano courses
- 3 marathons, 2 triathlons (Olympic distance) and a lot of cycling
- Norwegian amateur radio licence