## Orbit Simulation Toolkit - OSTK -

### Documentation of Models, Methods and Implementation

Bachelor of Science Degree Thesis by Vitali Müller



September 13, 2010

Max-Planck-Institute for Gravitational Physics (Albert-Einstein-Institute) Gottfried Wilhelm Leibniz Universität Hannover

## Orbit Simulation Toolkit - OSTK -

### Documentation of Models, Methods and Implementation

1 <sup>st</sup> Examiner:	PrivDoz. Dr. Gerhard Heinzel Albert-Einstein-Institut, Hannover
2 <sup>nd</sup> Examiner:	Prof. Dr Ing. Jürgen Müller Institut für Erdmessung, Hannover
Supervisors:	Marina Dehne Dr. Benjamin Sheard Albert-Einstein-Institut, Hannover

# Selbstständigkeitserklärung

Ich versichere, die vorliegende Arbeit selbstständig und nur mit den angegebenen Quellen und Hilfsmitteln angefertig zu haben, sowie die Zitate deutlich kenntlich gemacht zu haben.

Vitali Müller Hannover, den 13. September 2010

# Contents

1	Intr	roduction 5
	1.1	OSTK Overview
	1.2	History
	1.3	About this Thesis
	1.4	Results of Simulations
		1.4.1 Ranging Sensitivity
		1.4.2 Ranging Noise 9
		1.4.3 Attitude
<b>2</b>	Sate	ellite Dynamics 13
	2.1	Kepler's Solution
		2.1.1 Keplerian Elements
		2.1.2 Perturbations of Keplerian Orbits
	2.2	Rigid Body Solution
		2.2.1 Translation
		2.2.2 Rotation
		2.2.3 Forces and Torques
		1
3	OST	ΓK Structure 21
	3.1	OSTK commands
		3.1.1 config.txt
		3.1.2 mission.txt
	3.2	Basic OSTK classes
		3.2.1 class tmemman in memman.h 24
		3.2.2 class tfileman in fileman.h
		3.2.3 class tdatalyzer in datalyzer.h
	3.3	Basic OSTK types
		3.3.1 tVec
		3.3.2 tTensor
		3.3.3 tStateVec 25
		3.3.4 tStateQuat
4	Coo	ordinate Systems 27
	4.1	Rotations
		4.1.1 Rotation Matrix
		4.1.2 Convention: Rotation Axis and Angle
		4.1.3 How to check if a matrix is a rotation matrix?
		4.1.4 How to calculate the rotation axis and angle from a rotation matrix? 28
		4.1.5 Euler Angles
		4.1.6 Quaternions
		4.1.7 Relation between Quaternions and Rotation Matrices
	4.2	Angular Velocity
		4.2.1 How to calculate $\vec{\omega}$ and $\dot{\vec{\omega}}$ from a Rotation Matrix?
		4.2.2 How to calculate $\vec{\omega}$ and $\dot{\vec{\omega}}$ from Quaternions?
	4.3	Transformation between two coordinate systems
		4.3.1 Transformation of a position vector: $\Sigma' \longrightarrow \Sigma$
		4.3.2 Transformation of a velocity vector: $\Sigma' \longrightarrow \Sigma$
		4.3.3 Transformation of a velocity vector: $\Sigma' \longrightarrow \Sigma$ (rotation only) 35
		4.3.4 Transformation of an acceleration vector: $\Sigma' \rightarrow \Sigma$

		4.3.5 Inverse Transformations: $\Sigma \longrightarrow \Sigma'$
		4.3.6 Transformation of a Matrix
	4.4	USTK TriadStateMat
		4.4.1 Definition: USIK IrladStateMat
		4.4.2 How to validate a TriadStateMat?
		4.4.3 How to setup a new TriadStateMat?
		4.4.4 Changing the order of Coordinate Axes
	4.5	OSTK class tCOF in COF.h 38
		4.5.1 Validation
	4.6	Coordinate Systems
		4.6.1 Earth-Centered Coordinate Systems
		4.6.2 Satellite-Centered Coordinate Systems
		4.6.3 Summary
	_	
5	Ear	th's Gravity Field 43
	5.1	Gravity and Gravitation
	5.2	Static Gravitational Field 44
		5.2.1 Geoid and Ellipsoid 44
		5.2.2 Tide Systems
		5.2.3 Including new Earth Geopotential Models (EGM) in OSTK 47
		5.2.4 OSTK class tSHmodels in SHmodels.h
		5.2.5 Practical Considerations
	5.3	General Tide Theory
		5.3.1 Tide Generating Potential TGP
		5.3.2 Spectral Analysis
		5.3.3 Tidal catalogs
		5.3.4 Hartmann & Wenzel Potential Catalog: HW95
	5.4	Solid Earth Tides
		5.4.1 Rizos/Stolz method
		5.4.2 IERS Solid Earth Tide Model
	5.5	Ocean and Atmospheric Tides 56
	0.0	551 Harmonic Analysis 57
		$5.5.2$ Doodson-Warthurg Phase Correction $v_{\rm c}$ 58
		5.5.2 Document of the content of $\chi_s$
		5.5.7 Model: FD52004
		5.5.5 Formats 50
		$5.5.6  \text{Admittance Theory} \qquad \qquad$
	56	Short Term Variational AOD1P De Aliasing
	5.0	Droduct 62
		$5.61  OCTV \ alorge \ alorge \ \pm AOD1P \ b \qquad 62$
		5.0.1 OSIK class: class tAUDIB III AUDIB.II
	57	0.0.2 Configuration
	5.7	
	0.8	Summary & validation $\ldots \ldots \ldots$
6	Mod	delling Satellites 67
U	6 1	Surfaces 67
	6.2	Drag coefficient 68
	0.2 6.2	Mag Distribution 68
	0.3 6.4	Implementation 60
	0.4	Implementation
		$0.4.1  \textbf{3-uniensional models}  \dots  \dots  \dots  \dots  \dots  \dots  \dots  \dots  \dots  $
7	For	res & Torques 73
•	7 1	Farth's Cravity Field 73
	1.1	711 Acceleration $73$
		71.9 Gravity Gradient Torque $79$
		7.1.2 Gravity Grautent Torque
		71.4 Torque due to Acceleration of Fourt Masses
	7 9	Direct 3rd Body Acceleration 74
	1.4	$721  \text{Acceleration} \qquad 74$
	79	Drog 77
	1.3	Diag

	7.3.1       Acceleration	75 75
8 Co	onclusion & Outlook	77
A Ti	me derivatives of a unit vector	79
А.	1 First time derivative of $ \vec{r}  = r \dots \dots$	79
А.	2 First time derivative of $\vec{e}$	79
А.	3 Time derivative of $\vec{s}$	79
А.	4 Second time derivative of $\ddot{\vec{e}}$	80
А.	5 OSTK function diffunitvec()	80
в Sp	pherical Multipole Expansion	81
В.	1 Definitions	81
	B.1.1 Legendre polynomials of first kind	81
	B.1.2 Associated Legendre polynomials of first kind	82
	B.1.3 Spherical Harmonics	82
	B.1.4 Normalization	84
В.	2 Gravitational Potential Expansion	85
	B.2.1 Determination of the Geopotential Field	87
В.	3 Derivatives of the Geopotential in Spherical Coordinates	87
В.	4 Transformation Spherical Derivatives to Cartesian Derivatives	88
В.	5 Calculating directly Cartesian Derivatives	90
В.	6 Recurrence Relations for $\bar{P}_{l,m}(\cos \theta)$ and Derivatives	90
В.	7 Spherical Harmonic expansion of a scalar field with two parameters	91

# Abbreviations and Nomenclature

$\xrightarrow{q}$	quaternions
^	a hat denotes always a matrix
AOCS	Attitude and Orbit Control System
ASD	amplitude spectral density
BFS	Body Fixed System
CHAMP	CHAllenging Minisatellite Payload, GFZ Potsdam
$\operatorname{CoM}$	Center of Mass
CSR	Center for Space Research, University of Texas
EGM	Earth geopotential model; Earth gravitational model
GFZ	GeoForschungsZentrum Potdsam, Potsdamm
GOCE	Gravity Field and Steady-State Ocean Circulation Explorer, ESA
GRACE	Gravity Recovery and Climate Experiment
GUI	Graphical User Interface
IAU	International Astronomical Union
ICRF	International Celestial Reference Frame
IERS	International Earth Rotation and Reference System Service
LEO	Low Earth Orbit ( $< 800$ km height)
LISA	Laser Interferometer Space Antenna
MoI	Moment of Inertia
MTQ	Magnetic-Torquers
PAS	Principal Axis System
S/C	Spacecraft
$\mathrm{SH}(\mathrm{E})$	Spherical Harmonic (Expansion)
SOFA	SOFA (Standards of Fundamental Astronomy) software libraries, IAU
SRP	Solar Radiation Pressure
TGP	tide generating potential

### Chapter 1

## Introduction

The Orbit Simulation Toolkit OSTK is a program written in C/C++ with the purpose to simulate satellite orbits. The development started in 2008 at the Albert-Einstein-Institute (AEI) in Hannover with the aim to provide orbital informations and parameters like doppler shifts between two satellites. These informations are helpful for the conception of satellite based laser interferometers, which can be used to measure the inter-satellite distance. While the distance variations can be used, on the one hand to map the Earth gravity field and on the other hand to detect gravitational waves in space.

For later application the joint ESA and NASA mission Laser Interferometer Space Antenna (LISA) is planned, but not to be launched earlier than  $2020^1$ . This mission consists of three satellites in a heliocentric orbit, arranged in an equilateral triangle with 5 million km side length. Laser links between the satellites will measure distance variations on picometer level, which can be caused by space-time deformations due to gravitational waves.

Wheras the satellite pair GRACE is an example for a mission with purpose to map Earth's gravity field. These satellites were launched in 2002 and are equipped microwave links<sup>2</sup>. The microwave links measure the distance fluctations, which are also caused by changes in Earth's gravity field. These length measurements can be used reconstruct a map of the gravity field.

Other closely related missions are

- CHAMP: this single satellite was launchned in 2002 and provides accurate orbit informations due to GPS and accelerometer, which can also used to create Earth gravity maps
- GOCE: this drag-free satellite uses gradiometer to measure directly Earth's gravity field

Variations in Earth's gravity field provide many useful informations. On the first sight they are caused by mass shifts, but a closer look will break down the sources into effects related to hydrology, topography, oceans, atmosphere, Earth's crust and core. Therefore the whole geoscientific community benefits from such maps and models.

Orbit simulations, which use these environmental models, are important on this field. For example in GRACE they are used to provide reference orbits, whereby the residuals between reference and GRACE orbits are the input for new Earth gravity models [Gruber and Flechtner, 2007].

Furthermore the results of simulations are used for the development of new satellite hardware or for the conception of new satellite missions or constellations. Although there are commercial orbit simulator available, they are often limited in their field of application and not free available or accessable. Furthermore scientific applications require very precise knowledge of the used models and methods, which is often not provided in proprietary software.

<sup>&</sup>lt;sup>1</sup>ESA homepage: http://sci.esa.int/science-e/www/object/index.cfm?fobjectid=47277

 $<sup>^{2}</sup>$ dual band on each satellite to correct for ionospheric perturbations (cf. GPS L1 and L2)

#### 1.1**OSTK** Overview

OSTK is being developed as a scientific orbit simulator, with the aim to provide knowledge in satellite dynamics. It excels in the diversity of models and methods, which simultaneously enable to verify the implementations and get to know different point of views on subjects. It is very flexible and adoptable, but on the other hand requires understanding of satellite dynamics, the models and the structure of the simulator. It is surely no out-of-the-box software.

The extend can be described with:

#### • Environmental Models

- Earth Gravity Models
  - \* various static models included (EGM96, ITG03, GGM03, EGM08, ...) and easy expandable
  - \* ocean & atmosphere tide (EOT08a and FES2004)
  - \* two solid earth tide models (IERS solution and Rizos/Stolz)
  - \* short-term variations via AOD1B de-aliasing product provided by GFZ
  - \* tide generating potential using Hartman & Wenzel 1995 potential catalog
- Earth Atmosphere Density Models
  - \* exponential model [Montenbruck and Gill, 2000]
  - \* Harries-Priester: exponential with diurnal variations [Montenbruck and Gill, 2000]
  - \* NRL-MSI-00 model considering solar & geomagnetic activity [Picone et al., 2002]
- Earth's Rotation
  - \* only z-Rotation
  - \* IAU-2000 solution considering precession & nutation and polar motion (external library:  $SOFA^3$ )
- Celestial Body Position
  - \* JPL DE405 ephemeris catalog<sup>4</sup>[Standish, 1998]

#### • Physical or Mathematical Models

- Newton's Second Gravitational Law in a pseudo-inertial frame
- Euler's Moment equations, moment of inertia, principal axes system computation
- Attitude using quaternions
- Forces & Torques:
  - \* Earth's gravitational acceleration (two methods)
  - \* Earth's gravity gradient torque considering satellite's total mass distribution or monopole approximation using moment of inertia tensor
  - \* direct 3rd celestial body acceleration due to Sun, Moon, Venus & Jupiter
  - \* direct solar radiation pressure considering sun-distance and with fixed or dynamic cross-section area (via 3d S/C model)
  - \* drag with fixed or dynamic cross-section area (via 3d S/C model)
- Coordinate system transformation methods considering pseudo-forces, capable of handling rotation matrices, angular velocity and quaternions
- Time systems like UTC, UT1, TDB, Julian Date & transformations considering leap seconds and earth's rotation variations
- Propagator
  - Kepler
  - general Runge Kutta with various coefficients based on [Heinzel, 1992]
  - 12th order Adams/Bashforth PECE multistep integrator based on [Goetzelmann, 2003]

<sup>&</sup>lt;sup>3</sup>SOFA (Standards of Fundamental Astronomy) software libraries, IAU

<sup>&</sup>lt;sup>4</sup>original JPL source code was ported to C++ by Benjamin Sheard, AEI

#### • Satellite Models

- only point mass
- point mass with fixed parameter like cross-section area or drag coefficient
- 3-d satellite model including surface properties and mass distribution, preliminary actuators and magnetic dipole moments

### 1.2 History

In September 2008 the OSTK development began with a simulator for Kepler orbits. It was a C/C++ program and designed for Linux. Later a graphical user interface based on OpenGL was implemented, to visualize the orbits and to allow direct user input during simulations. To take various perturbations into account, a numerical Runge Kutta integrator was adopted from [Heinzel, 1992], as well as support for geopotential models in spherical harmonics expansion format as implemented. In 2009 various perturbation models were included accounting for drag, solar radiation pressure and direct 3rd celestial body acceleration. To be able to validate the implementation, various models were taken into account (e.g. three atmosphere density models: NRL-MSI, exponential, Harries-Priester). The JPL DE405 catalog took over the ephemeris calculation. In August/September 2009 Prof. Li Guangyu<sup>5</sup> reviewed OSTK at the AEI and further helped at the implementation of further improvements (e.g. the solar eclipse functions for solar radiation pressure). In addition a new multistep integration method was introduced. By the end of 2009 a new version from scratch was written - with the aim to improve the structure of the source code. Furthermore OSTK reference orbits were calculated and provided to the GRACE L1B processing group<sup>6</sup> In 2010 OSTK was adopted to be compatible with Microsoft Windows and the doxygen source code documentation<sup>7</sup> was introduced. An external program was written, to reconstruct the geopotential field from satellite orbits and ranging data. Later O. Hartwig<sup>8</sup> helped at the OSTK development by validating the simulator (checking energy conservation, forward-backward-integration) and documentation of various parts of the source code. Furthermore he reviewed, improved and documented the Kepler functions, implemented AOD1B product support and extrapolated OSTK external data.

#### **1.3** About this Thesis

During my bachelor thesis time, started in April 2010 and ended in September 2010, I have done following:

- 1. simulation of pendulum orbits for a GRACE follow on mission; computation of the influence of every single geopotential coefficient on range measurements expressed in power spectral density (PSD) (fig. 1.1)
- 2. computation of colored noise from an interferometer PSD noise model using matlab; introducing this noise into simulated orbit sets of a pendulum and non-pendulum orbit; recovering the geopotential coefficients and calculation of the geoid errors (fig. 1.2)
- 3. implementation and validation of ocean & atmosphere tide models into OSTK (results in sec. 5.8)
- 4. implementation of IERS solid Earth tide models (results in sec. 5.8)
- 5. computation of a new GRACE L1B dataset including a) the residual acceleration between FES04 and EOT08 ocean tide model (as noise) b) interferometer noise model from 2.
- 6. programming and providing a Microsoft Windows version of OSTK that is configured to compute only arcs of satellite orbits - should be used for recovering the geopotential field with the energy balance approach

<sup>&</sup>lt;sup>5</sup>retired; former Purple Mountain Observatory, China

<sup>&</sup>lt;sup>6</sup>a collaboration of various geodesy institutes

 $<sup>^7\</sup>mathrm{doxygen}$  is a program, which creates proper documentations from comments in the source code  $^8\mathrm{student},$  former AEI

- 7. implementation of 3d-satellite models and satellite's attitude in OSTK:
  - introducing quaternions, Moment of Inertia, Euler's Moment equations
  - modifying the numerical integrator, solving the equations of rotation numerically
  - adapting the GUI
  - implementation of functions to calculate the cross-section area dependent on attitude for drag and solar radiation pressure
  - rough validation of source code by examining if physical effects like nutation and precession for a spinning top are visible
- 8. computation of the torque required to rotate a satellite in a pendulum-orbit (fig. 1.3 and 1.4)

Then I realized that further OSTK development will lead to the simulation of attitude and orbit control systems (AOCS), wherefore a proper preparation of the simulator would be benefiting. Furthermore OSTK reached to my knowledge the state-of-the-art in modeling Earth's gravity field. Hence I froze the OSTK proceeding development and started to solidify the achieved status by

- restructuring the implementations of all Earth gravity models (static, ocean & atmosphere tide, solid earth tide, AOD1B); introducing one class, that can handle all spherical harmonic expansions for the gravity as well as for the geomagnetic field
- writing a class, that can be used for all coordinate transformations, e.g. for attitude of the satellite as well as for Earth-Space coordinate system transformations and that is capable to use of rotation matrices, quaternions and angular velocity
- rewriting the 3d-satellite models, so they are capable to handle actuators (thrusters, magnetic torquers) and are able to approximate the satellite's mass density (with point masses on a grid)
- unifying the methods of handling perturbations; introducing new perturbations like gravity gradient torque

Although the next section shows some results of simulations, the purpose of this thesis is to provide a documentation or manual about the physical and mathematical models and methods that are actually used and information about their implementation in OSTK. However, this is not a complete documentation, yet. It will be hopefully a living document, so further chapters will be included or supplemented.

"Every physics thesis starts with Einstein, Newton or Adam & Eva."

Here Newton's Gravitational Law is used to introduce the Kepler orbits in chapter 2, followed by the classical solution for the rigid body movement. It requires basic knowledge in orbital mechanics and concludes with the basic concept of simulating the attitude and orbit.

In chapter 3 a rough overview about the structure of OSTK is given and some terms are introduced, that may appear in later chapters.

The chapter 4 summarizes general formulas for coordinate transformations and describes the coordinate systems.

The 5th chapter is the main chapter of this thesis. It provides the background for all included Earth gravity models as well as the formulas. Furthermore the Appendix B contains useful mathematical relations and derivations for this chapter.

Whereby chapter 6 covers the actual definition of satellites in OSTK and introduces the approach to model them.

Some formulas for the forces and torques acting on the Satellite are summarized in chapter 7.

In chapter 8 the conclusion and outlook is located.

#### 1.4 Results of Simulations

#### 1.4.1 Ranging Sensitivity

The GRACE inter-satellite distance variations are measured with a microwave system and are influenced by the gravity field and perturbations like drag or orbital parameters. A subject of research is the optimal reconstruction of the gravity signal from length measurements. The opposite direction, the influence of Earth's gravity signal on ranging measurements, can be used to estimate performance requirements for future ranging systems.

In fig. 1.1 the influence of two high-degree geopotential coefficients on a range measurement in terms of amplitude spectral density is shown. A pendulum orbit<sup>9</sup> with 200 km along track separation and with  $\pm 45^{\circ}$  maximum yaw angle variation was considered here. The orbital frequency peak (1 revolution) is located at  $10^{-4}$  Hz. The theory of these frequency patterns is described in [Thomas, 1999].

An estimated noise level of a laser interferometer is also illustrated. As shown such an interferometer would be able to measure even these high coefficients (in the main bandwidth). All parameters depend on the satellite's orbit as well as on the satellite constellation.



Figure 1.1: For this plot the range accelerations due to geopotential coefficients (l=110, m=2) and (l=120, m=120) in a pendulum GRACE orbit (L = 200 km, h = 380 km) were calculated. The range accelerations were transformed into an amplitude spectral density (ASD) and integrated in the frequency domain to obtain the range ASD (denoted as PSD in the plot). For comparison an estimated laser interferometer noise model was plotted.

#### 1.4.2 Ranging Noise

For fig. 1.2 the influence of inter-satellite ranging noise was analyzed. Ephemeris sets for a GRACE like and for a pendulum orbit were computed and noise with an amplitude of  $10 \text{ nm}/\sqrt{\text{Hz}}$  was superimposed in both cases. As next step the geopotential coefficients were recovered using the acceleration approach. The spatial distribution of the differences between input and recovered coefficients in terms of geoid height are shown. For the pendulum orbit the acceleration approach yielded results with lower noise in the coefficients. However, the noise level is very low and near to the numerical accuracy of the implemented acceleration

 $<sup>^{9}</sup>$  the orbital planes of the satellites are shifted at the equator (Right Ascension in Kepler elements, cf. sec. 2.1.1); these orbits have a nearly constant along-track distance and a time-variant cross-track component



approach. Further simulations, also with other approaches, should be done to solidify the superior performance of an pendulum orbit.

Figure 1.2: Influence of white noise  $10 \text{ nm}/\sqrt{\text{Hz}}$  in ranging observables in a GRACE like configuration with along track distance of 200 km and satellite height h=380 km in a pendulum orbit (cross track separation 50 km) and non-pendulum orbit case (cross track separation 0 km).

#### 1.4.3 Attitude

In this section attitude simulation results are presented. Under the assumption of a circular orbit, only acceleration due to the monopole term and negligible perturbation torques, a satellite can rotate torque free with the orbital frequency, so that the bottom side is always parallel to Earth's surface. For the fig. 1.3 a nearly circular orbit for two satellites was chosen. The satellites had an along track separation of about 200 km. Shown is the required torque to assure a pointing towards the other satellite (torque acts on the follower satellite). The torque is given in the principal axes (= body fixed) system and is nearly zero for a point mass Earth (the small variation in the y-component is due to the small ellipticity of the orbit). Activating higher moments of the geopotential causes large variations in the y-component mainly due to considered oblateness of the Earth.



Figure 1.3: Torque in principal axis system required to assure pointing towards the other satellite; high, nearly circular orbit in a GRACE like orbit configuration; no perturbations like gravity gradient torque

#### 1.4. RESULTS OF SIMULATIONS

In fig. 1.4 the previous simulation was repeated for a pendulum orbit. Torques along every principal axis have to be applied to warrant the pointing. These preliminary results look promising, because on the one hand the implementation seems to be physical correct and on the other hand the torques in the pendulum orbit were roughly, concurring compared with a plot provided by an independent Institute. However, further verification has to be done.



Figure 1.4: Torque in principal axis system required to assure pointing towards the other satellite; pendulum orbit case; on the left plot the roll axis is fixed w.r.t. to inertial space, while on the right plot all axes vary; no perturbations like gravity gradient torque

### Chapter 2

## Satellite Dynamics

Describing an artificial satellite (lat. "satelles" – attendant) is a subject of classical mechanics. A preliminary solution can be obtained by assuming the satellite R is orbiting only the planet P and both objects are point masses. The trajectory  $\vec{r}(t)$  of the Satellite is the solution of the well-known two-body problem.

#### 2.1 Kepler's Solution

It is reasonable to assume that the artificial satellite has negligible mass in comparison to the planet ( $m_{\rm R} \ll m_{\rm p}$ ). Hence the satellite will not influence the motion of the planet. The differential equations of this problem are given by gravitational acceleration due to a point mass (Newton's Gravitational Law):

$$\ddot{\vec{r}}(t) = -\frac{G \cdot m_{\rm p}}{|\vec{r}(t)|^3} \ \vec{r}(t), \tag{2.1}$$

where we have chosen the planet's center as origin of the coordinate system and G as the gravitational constant. The vector of angular momentum is a conserved quantity and is defined as

$$\vec{L}(t) := \vec{r}(t) \times m_{\rm R} \cdot \dot{\vec{r}}(t).$$
(2.2)

If the direction of this vector is fixed, the motion of the satellite takes place in a fixed orbital plane with normal vector  $\vec{L}$ .

The time derivative of  $\vec{L}$  is the torque  $\vec{T}$ 

$$\vec{T}(t) := \vec{L} = \vec{r}(t) \times \vec{F}(t) \tag{2.3}$$

with external force  $\vec{F}(t) = m_{\rm R} \cdot \vec{r}(t)$ . The force vector of the satellite points towards the planet and the position vector  $\vec{r}$  in opposite direction; hence, the cross-product and consequently the torque on the satellite is zero. The motion of a satellite in the unperturbed, Keplerian case, takes place in a fixed orbital plane.

Examining the three Keplerian Laws and switching from Cartesian Coordinates to Keplerian Elements (cf. next section 2.1.1) will lead to a solution for the differential equation (2.1) with four different orbit types: circular (e = 0), elliptic (0 < e < 1), parabolic (e = 1) and hyperbolic (e > 1) depending on the initial conditions, whereas the elliptic orbit is the most important one for LEO satellites. Several difficulties occur when all orbital types have to be considered in the calculations, like singularities and undefined Keplerian Elements for special orbital cases (e.g. circular-equatorial). The general solution (with special cases) is extensively discussed in [Vallado and McClain, 2007]. Unfortunately, there is no general closed-form solution for the transformation between Cartesian Coordinates and Keplerian Elements, because the equation for the eccentric anomaly E of the form

$$E(t) - e\sin(E(t)) = M(t) \tag{2.4}$$

has to be solved numerically, e.g. iteratively using the Newton Method.

Kepler's solution can also be applied to the general two-body problem, where  $m_{\rm R} \ll m_{\rm p}$  is not valid. In the absence of an external force field, the center of mass (CoM) of such a system moves uniformly. The definition of the CoM  $\vec{R}$  for a system consisting of n point masses is

$$\vec{R} = \frac{1}{M} \sum_{i=1}^{n} m_i \cdot \vec{r_i} \quad \text{with total mass} \quad M = \sum_{i=1}^{n} m_i.$$
(2.5)

One can obtain the solution of the general two-body problem by separating the trajectories  $\vec{r}(t)$  and  $\vec{p}(t)$  of both objects into a movement of the CoM  $\vec{R}(t)$  and into a movement caused due to the previously discussed simplified Keplerian solution:

$$\vec{r}(t) = \vec{R}(t) + \vec{r}_{\text{Kepler}}(t) \tag{2.6}$$

$$\vec{p}(t) = \vec{R}(t) + \vec{p}_{\text{Kepler}}(t) \tag{2.7}$$

The preconditions of the simplified Keplerian problem arise by switching to reduced masses and a coordinate system with origin at the CoM.

#### 2.1.1 Keplerian Elements

The six Keplerian Elements are a set of variables, which can be used to describe Kepler orbits in a radial symmetric gravitational field. There is a unique transformation [Vallado and McClain, 2007] between the position and velocity and the six Keplerian Elements, which are:

• Eccentricity e: defines the oblateness of the orbital trajectory:

$$e = \frac{r_{\rm a} - r_{\rm p}}{r_{\rm a} + r_{\rm p}} \tag{2.8}$$

where  $r_{\rm a}$  is the radius of apoapsis (farthest distance between focus of the ellipse and trajectory = semi-major axis a) and  $r_{\rm p}$  is the radius of periapsis<sup>1</sup> (closest distance = semi-minor axis b). Danger of confusion causes the not-related definition of the eccentricity of an ellipse:

$$e = \frac{\sqrt{a^2 - b^2}}{a}.\tag{2.9}$$

- Semi-Major Axis a: defines the size of the orbital ellipse.
- Inclination *i*: the angle between the orbital plane and a reference plane (usually Earth's equatorial plane). For an equatorial orbit  $i = 0^{\circ}$  and  $i = 90^{\circ}$  corresponds to a polar orbit.
- Right Ascension of the Ascending Node  $\Omega$ : angle between the ascending node (intersection line of equatorial and orbital plane) and the x-axis of the inertial frame.
- Argument of Perigee  $\omega$ : angle between ascending node and orbital perigee.
- Mean Anomaly M at a specific Epoch: position of the satellite in orbit expressed as a fraction of  $2\pi$  (without geometrical meaning). 0 or  $2\pi$  denotes a satellite at the perigee, whereby a satellite with  $M = \pi$  is located at the apogee. M changes uniformly with time. The true anomaly  $\nu$  and eccentric anomaly E have a geometrical interpretation and can be derived from M.

As already mentioned for special cases some elements may be undefined and new elements have to be introduced. The Keplerian Elements are especially useful for conception of new satellite orbits, because the first five elements have an geometrical meaning, as shown in figure 2.1, while the cartesian state vector of position and velocity provides no (direct) information about the shape of the total orbit.

 $<sup>^1</sup>$  "perapsis" is the general term; related to Sun, Moon, Earth the words "perihelion", "perigee", "periastron" (resp.) are common



Figure 2.1: Keplerian Elements, Image from Wikipedia

#### 2.1.2 Perturbations of Keplerian Orbits

Handling perturbations with Keplerian Elements is difficult (e.g. acceleration due to moon or deceleration due to atmospheric drag). There exists a perturbation theory, which for example is described in [Sidi, 2006]. The influence of tides (and general perturbations) on Keplerian Orbits is also very well explained in [Baur, 2002, ch. 4: Kaula-Theory]. One can obtain an analytical expression for the change rate of the Keplerian Elements for conservative and non-conservative perturbation forces:

$$\frac{de}{dt}, \frac{da}{dt}, \frac{d\omega}{dt}, \frac{d\Omega}{dt}, \frac{di}{dt}$$

For example the change of the orbital radius for a circular orbit (= semi-major axis a) due to drag can be described with [Sidi, 2006, eq. 2.7.17]

$$\frac{da}{dt} = -\rho \sqrt{a \cdot GM} \frac{C_{\rm d} \cdot A_{\rm c}}{m} \tag{2.10}$$

where  $\rho$  is the air density,  $C_{\rm d}$  the drag coefficient,  $A_{\rm c}$  the cross-section area perpendicular to the air flux and *m* is the mass of the satellite. It is usually necessary to integrate the equations numerically to obtain the orbital elements for a particular time. Nevertheless the influence of various perturbation parameters becomes directly visible in these equations.

### 2.2 Rigid Body Solution

A more advanced approach to calculate the motion of a satellite is to assume a satellite as a rigid body and to use a numerical integrator. On the one hand, this ansatz does not idealize the satellite as a point mass, and on the other hand, a numerical integrator is (nearly) independent of the considered accelerations acting on the satellite. It is also possible to use only a numerical integrator and to handle the satellite as a point mass, if a 3-d model of the satellite is not available.

A rigid body in classical mechanics is defined as a body with fixed mass distribution with respect to a body fixed coordinate system (BFS). For continuous systems with mass density  $\rho$  and  $[\rho] = \text{kg/m}^3$  this can expressed with

$$\frac{d}{dt}\rho(\vec{r}_{|\rm BFS}(t)) = 0 \tag{2.11}$$

or in the discrete case with an arbitrary number of point masses with position  $\vec{r_i}$ :

$$\frac{d}{dt}\vec{r}_{i,|\rm BFS}(t) = 0 \qquad \forall i.$$
(2.12)

It is convenient to choose the origin of the body fixed system (BFS) at the center of mass (CoM), which can be calculated using eq. (2.5). The subscript |BFS indicates the coordinate system or basis of a vector, as described in chapter 4.

As stated by classical mechanics (and shown later) the motion can be separated into a translation of the CoM and a rotation around an axis through the CoM. In first case, we assume the total mass M of the body is concentrated at the center of mass position  $\vec{R}(t)$  and all forces act on this point. Newton's second law states:

$$\vec{F}(t) = \vec{F}_1(t) + \vec{F}_2(t) + \ldots + F_N(t) = M \cdot \vec{R}(t)$$
 (2.13)

where  $\vec{R}$  is with respect to an inertial frame.

In the other case we calculate the torques  $T_i$  (also called moments) acting on each point mass and sum up to obtain the total torque

$$\vec{T} = \sum_{i=1}^{n} \vec{T_i}.$$
 (2.14)

This torque will cause a rotation around an axis through the CoM or will change the rotation (if the body rotates by initial conditions).

#### 2.2.1 Translation

To acquire the trajectory  $\vec{R}(t)$  of the satellite's CoM, we need the initial position  $\vec{R}(t_0)$ and velocity  $\dot{\vec{R}}(t_0)$  as well as the acceleration due to gravitational attraction  $\ddot{\vec{R}}_g(t)$ , which can be caused by the primary (the Earth) and by other celestial bodies like the Sun or Moon.

In addition there are several perturbation forces or accelerations  $\vec{R}_{\rm p}(t)$ , which are usually non-conservative and dependent on the velocity  $\vec{R}$ , attitude  $\vec{q}$  or other properties of the satellite. Therefore we write

$$ec{R}_{
m p}(t,ec{R},ec{R},ec{q})$$

The trajectory  $\vec{R}(t)$  can be calculated using:

$$\dot{\vec{R}}(t) = \dot{\vec{R}}(t_0) + \int_{t_0}^t \left[ \ddot{\vec{R}}_g(t'') + \ddot{\vec{R}}_p(t'', \vec{R}, \vec{q}, \dot{\vec{R}}) \right] dt''$$
(2.15)

$$\vec{R}(t) = \vec{R}(t_0) + \int_{t_0}^t \dot{\vec{R}}(t') dt'$$
(2.16)

or combining the equations to:

$$\vec{R}(t) = \vec{R}(t_0) + \int_{t_0}^t \dot{\vec{R}}(t_0) + \left(\int_{t_0}^t \ddot{\vec{R}}_g(t'') + \ddot{\vec{R}}_p(t'', \vec{R}, \vec{q}, \dot{\vec{R}}) dt'\right) dt'$$
(2.17)

$$= \vec{R}(t_0) + \dot{\vec{R}} \cdot (t_0)(t - t_0) + \int_{t_0}^t \left[ \int_{t_0}^{t'} \left[ \ddot{\vec{R}}_g(t'') + \ddot{\vec{R}}_p(t'', \vec{R}, \vec{q}, \vec{R}) \right] dt'' \right] dt'$$
(2.18)

However, there is no analytical solution for such a general integral equation. Given initial conditions a numerical integrator can be used to solve this equation and to obtain the position and the velocity of the CoM at a particular time.

#### 2.2.2 Rotation

Until now we have described the evolution of the position of the satellite's (or rigid body's) CoM. We now consider the attitude which is equal to the orientation of the body fixed coordinate system in an inertial frame. An intuitive and geometrical approach to describe an attitude is to use a rotation axis with unit length  $\vec{u}(t)$  and an angle  $\alpha(t)$ . We further define the angular velocity vector as the actual rotation axis times the actual angular velocity [Fogiel, 1987]:

$$\vec{\omega}(t) = \vec{u}(t) \cdot \dot{\alpha}(t)$$
 with unit  $[|\vec{\omega}|] = \text{rad/s}$  (2.19)

#### 2.2. RIGID BODY SOLUTION

#### Moment of Inertia

The moment of inertia (with respect to a rotation axis  $\vec{u}$ ) for a discrete rigid body is given by

$$I(\vec{u}) = \sum_{i=1}^{n} m_i \cdot d_i^2 \qquad \text{with unit} \qquad [I] = \text{kg} \cdot \text{m}^2$$
(2.20)

where  $d_i$  is the distance of every point mass from the rotation axis. Although we denote the rotation axis often as a normalized vector  $\vec{u}$  (or use the angular velocity vector  $\vec{\omega}$ ), the origin of the rotation axis is not explicitly defined and can be shifted along the rotation axis. The center of rotation can be even outside the body, but all points along the rotation axis will not move due to the rotation. It is possible to generalize the scalar moment of inertia (MoI) to a tensor and and to calculate the MoI for arbitrary rotation axes.

#### Moment of Inertia Tensor

The Moment of Inertia tensor for a rigid body is defined as [Williams, 1996, p.281]

$$\hat{I}_{|BFS} = \sum_{i=1}^{n} m_i \cdot \begin{bmatrix} y_i^2 + z_i^2 & -x_i \cdot y_i & -x_i \cdot y_i \\ -y_i \cdot x_i & x_i^2 + z_i^2 & -z_i \cdot y_i \\ -z_i \cdot x_i & -y_i \cdot z_i & x_i^2 + y_i^2 \end{bmatrix} \quad \text{with units} \quad [I_{i,j}] = \text{kg} \cdot \text{m}^2 \quad (2.21)$$

where  $\vec{r}_i = (x_i, y_i, z_i)^{\mathsf{T}}$  is the point mass' position in the BFS system. Note that the tensor depends on the coordinate system, especially on coordinate system's origin. The Moment of Inertia with respect to a rotation axis  $\vec{u}_{|\text{BFS}}$  going through the origin is given by:

$$I(\vec{u})_{|BFS} = \vec{u}_{|BFS}^{\mathsf{T}} \cdot \hat{I}_{|BFS} \cdot \vec{u}_{|BFS}.$$
(2.22)

We are allowed to separate the satellite's motion into a translation and a rotation, because the rotation axis points always through the center of mass. Hence the CoM will not move, due to the rotation.

#### Principal Axis

The MoI tensor  $\hat{I}$  is a real,  $3 \times 3$  symmetric matrix and hence always diagonalizable (e.g. with the Jacobi method). The eigenvectors are called the principal axes of the body and form a new coordinate system. The eigenvalues are the principal moments of inertia. The physical and mathematical description of rotations is easier in the principal axis system (PAS).

#### Angular Momentum

The angular momentum  $\vec{L}$  for a point mass was defined in eq. (2.2). For a rigid body the angular momentum can be easier computed with the product of MoI tensor and angular velocity [Williams, 1996, p.282 eq. 6-36]:

$$\vec{L}_{|\rm BFS} = \hat{I}_{|\rm BFS} \cdot \vec{\omega}(\rm BFS) \tag{2.23}$$

$$\vec{L}_{|\text{PAS}} = \hat{I}_{|\text{PAS}} \cdot \vec{\omega}(\text{PAS}) \tag{2.24}$$

(2.25)

where  $\hat{I}_{|PAS}$  is a diagonal matrix and the argument of  $\vec{\omega}$  denotes the rotating system.

#### **Euler's Moment Equations**

As mentioned before, the torque or moment is given by

$$\vec{T}_{|\text{inertial}} = \dot{\vec{L}}_{|\text{inertial}}$$
 (2.26)

where we can now use the later derived equation (4.66) to obtain the change rate in a rotating system:

$$\vec{T}_{|\text{inertial}} = \dot{\vec{L}}_{|\text{inertial}} = \vec{\omega}(\text{PAS}) \times \tilde{\mathbf{L}}_{|\text{inertial}} + \hat{\mathbf{Q}} \cdot \dot{\vec{\mathbf{L}}}_{|\text{PAS}}$$
 (2.27)

with  $\hat{Q} = \hat{Q}_{|\text{inertial}}^{|\text{PAS}}$  being the rotation matrix to rotate a vector from PAS to inertial system. Multiplying with the inverse rotation matrix  $\hat{Q}^{\mathsf{T}} = \hat{Q}_{|\text{PAS}}^{|\text{inertial}}$  and keeping in mind, that  $\vec{\omega}$  has the direction of the rotation axis and is an eigenvector of  $Q^{\mathsf{T}}$ , leads to

$$\vec{T}_{|\text{PAS}} = \vec{\omega}_{|\text{PAS}} \times \cdot \vec{L}_{|\text{PAS}} + \dot{\vec{L}}_{|\text{PAS}}.$$
(2.28)

The MoI tensor is diagonal, thus this equation can be written in components:

$$T_x = I_{xx} \dot{\omega}_x + \omega_y \,\omega_z (I_{zz} - I_{yy}), \qquad (2.29)$$

$$T_y = I_{yy} \dot{\omega}_y + \omega_x \,\omega_z (I_{xx} - I_{zz}), \qquad (2.30)$$

$$T_z = I_{zz} \,\dot{\omega}_z + \omega_x \,\omega_y (I_{yy} - I_{xx}),\tag{2.31}$$

or rearranged

$$\dot{\omega}_x = \frac{(T_x - \omega_y \ \omega_z (I_{zz} - I_{yy}))}{I_{xx}} \tag{2.32}$$

$$\dot{\omega}_y = \frac{(T_y - \omega_x \; \omega_z (I_{xx} - I_{yy}))}{I_{yy}} \tag{2.33}$$

$$\dot{\omega}_z = \frac{(T_z - \omega_x \ \omega_y (I_{yy} - I_{xx}))}{I_{zz}} \tag{2.34}$$

These are three coupled, nonlinear, differential equations of first order for  $\vec{\omega}(t)$ . They are referred to as Euler's Moment Equations and describe the relation between applied external torque  $\vec{T}$  and rotational behavior of a rigid body, like Newton's Second Law  $\vec{F} = m\vec{r}$  describes the relation between applied force and **linear** motion of a rigid body. The torque can be caused for example by the Earth's gravity gradient<sup>2</sup> or drag or other perturbations.

#### Quaternions

Problems arise when we want to calculate the rotation axis and angle (hence the attitude) from the angular velocity. Other methods to describe the attitude are given by rotation matrices or Euler angles, but they also cause problems due to singularities. We can avoid these problems by using quaternions  $\vec{q}$ , which are discussed in detail in sec. 4.1.6. Quaternions can describe rotations and can be defined by a rotation axis  $\vec{u}$  and angle  $\alpha$ 

$$\underline{q}(\vec{u},\alpha)$$

There are transformation formulas (see sec. 4.2.2):

$$q, \dot{q} \Rightarrow \vec{\omega}$$
 (2.35)

$$\dot{q}, \ddot{q} \Rightarrow \dot{\vec{\omega}}$$
 (2.36)

$$\vec{\omega} \Rightarrow \underline{q} \ \underline{\dot{q}} \tag{2.37}$$

$$\underline{\dot{q}}, \dot{\vec{\omega}} \Rightarrow \underline{\ddot{q}} \tag{2.38}$$

 $<sup>^{2}</sup>$ gravity gradient torque is caused due to the inhomogeneity of the gravitational attraction inside the satellite

If the initial orientation  $\vec{u}(t=0)$  and angle  $\alpha(t=0)$  is given, as well as the initial angular velocity  $\vec{\omega}(t=0)$  one can calculate (with the methods described in 4.1.6) the initial quaternions

$$\underline{q}(t=0), \qquad \underline{\dot{q}}(t=0)$$

Euler's Moment Equations (2.32) - (2.34) provide  $\dot{\omega}$ , while eq. (2.38) yields to  $\underline{\ddot{q}}(t=0)$ . Thus, if we know the attitude of the satellite represented by  $\underline{q}(t), \underline{\dot{q}}(t)$  at an arbitrary point in time and if we can calculate the torque  $\vec{T}$  acting on the satellite at this time, which usually depends on the satellite's attitude and position, then eq. (2.32) - (2.34) and (2.38) provide the second derivative of the quaternion  $\underline{\ddot{q}}(t)$ . Again, we can use a numerical integrator to integrate the quaternions and to obtain the attitude for all times.

#### 2.2.3 Forces and Torques

The methods previously described allow to solve the Rigid Body problem and hence to determine the position and attitude of a satellite in orbit for an arbitrary point in time, if

- the initial conditions are given: position of CoM  $\vec{R}(t)$ , velocity of CoM  $\vec{R}(t = 0)$ , attitude and change rate of attitude  $\underline{q}(t = 0)$ ,  $\underline{\dot{q}}(t = 0)$ .
- the gravitational and non-gravitational accelerations on the CoM  $\ddot{\vec{R}}_g(t)$ ,  $\ddot{\vec{R}}_p(t, \vec{R}, \vec{R}, \vec{q})$ , and the torques  $\vec{T}(t)$  are known (for all times).

These external forces and torques acting on the satellite are referred to:

- Earth's gravity field,
- Gravity field of celestial bodies: like Sun, Moon, Venus, Jupiter,
- Drag due to (residual) atmosphere,
- Solar Radiation Pressure (SRP),
- Earth's albedo SRP (light reflected back from Earth),
- Earth's magnetic field,

and have to be modeled and computed. In addition satellites are usually partly autonomous. They use sensors like:

- Cameras or Photo-Diodes (Sun Sensor, Star Sensor),
- Gyroscopes,
- Antennas, GPS Receiver,
- Magnetometer,

and are able to actuate (produce an acceleration or torque) to keep a desired orbit or attitude with

- Magnetic-Torquers MTQ,
- Reaction & Momentum Wheels,
- Thrusters.

This interaction between measurement and actuation is summarized in an Attitude and Orbit Control System AOCS .

Although so many different effects need to be taken into account to simulate or predict satellites, they all can be handled with the Rigid Body solution. OSTK is now being developed to simulate the satellite's center of mass position, the attitude and the Attitude and Orbit Control System AOCS (long-term aim). Figure 2.2 summarizes and explains the relation of this three fields in the context of the previously described CoM position  $\vec{R}$  and attitude  $\vec{q}$  for one satellite. Starting point are the initial conditions for the position, velocity and attitude. These state vectors are introduced into the integration loop. The quaternions can be used to compute the angular velocity. Then the environmental models use the state vectors to provide the accelerations and torques acting on the satellite. As next step the AOCS can simulate a measurement of this state vector, e.g. by simulating noise or uncertainties. This measured quantities (denoted with subscript M) are compared with reference values and the deviation (error) is passed to the control algorithms. Their task is provide a proper actuation signal for thruster or torquers, which directly produce an actuation force  $\vec{F}_A$  or torque  $\vec{T}_A$ . In the end the unmodified torques and forces (from the environmental models) as well as the actuation forces are added, and converted to angular and linear acceleration. Then the integrator calculates a state vector at a later point in time, and the loop repeats.



Figure 2.2: Basic approach to combine orbit, attitude and AOCS simulation

## Chapter 3

## **OSTK Structure**

This is a preliminary chapter concerning the practical work with the simulator.

The Orbit Simulation Toolkit is structured in a modular way. These modules are called classes (or instances of a class). The global class is calles OSTK, which summarizes all functions. The main components of the OSTK class are the subclass SIM and GUI. Later one is optional, because the program is able to run in a console mode without graphical output. However the in-time visualisation of vectors or values is very useful for debugging or verification. Furthermore the GUI enables user interaction while a simulation is running.

The SIM class contains all models and the scientific part. Various satellites, models and integration methods can be set up. This class links them to obtain the integration loop, which was shown in 2.2. A schematic overview is given on the next two pages. Additional description about the single sub-classes can be found in sec. 3.2 on page 24.

### 3.1 OSTK commands

The simulator can be controlled and configured with commands. These commands can be located in configuration files or can be typed in by the user in the graphical user interface. They all are processed in the source code file commands.cpp. The syntax of such an OSTK command is always

type command = param1 param2 ....

where only type and command are compulsory. For example sim start will start the simulation, while read egm = /externdata/egm96.dat will load geopotential coefficients.

#### 3.1.1 config.txt

This file is executed at the startup stage. It will set up different paths (for logfiles or programs) and load the external data, like

- leap seconds for time system conversions
- Earth's orientation parameter like polar axis position (polar motion) and length of day (LOD) for the IAU-2000 transformation between the celestial and terrestrial coordinate system (cf sec. 4.6 about coordinate systems)
- geopotential & tide coefficients for Earth's gravity field
- JPL DE405 ephemeris file for computation of celestial body position
- AP geomagnetic & F107 solar activity coefficients, which are required by the NRL-MSI atmosphere model

#### 3.1.2 mission.txt

After the startup stage this file sets up the actual simulation scenario. It defines the simulation time, duration and the initial conditions of the satellites, next to the perturbations and desired models. Further description about setting up satellites can be found in sec. 6.4.

### OSTK

uses

MEM

vars

meth

#### uses:

#### MEMMAN

methods: mymalloc(...) myfree(...)

myresize(...) list status

#### struct: SETTINGS

vars:

file paths, time, active text site (gui), active satellite in gui

#### PERFTIME eta-time

vars:

simulation speed

methods: calc\_newaverage()

#### **FILEMAN**

MEMMAN

uses:

methods:

openfile(...)

closefile(...) writeline(...) readline(...)

mem2file(...) file2mem(...) list status

#### DATALYZER

#### uses: MEMMAN

set\_columns

save2file, loadfile, direct mem access

set\_rows

**FILEMAN** 

#### methods: init()

global accessable storage container

interpolation differentiation

vars:

#### vector: AtTimeCommands commands executed at particular time(s)

#### methods:

OSTK(...): init memory manager, file manager load config & mission file, start GUI init(...):

#### **COMMAND functions:** process commands exec\_command(string);

sub-functions: mp, sim, sys, mission, read, texture, image

LOADFILE functions: load external data leapsec, EOP, EGM, HW95-Tides **AP-coeff, Solar Activity,** IERS-SET, AOT

**TEXTWINDOW functions for GUI:** defines the text Earth, other Planets, Satellite CoM (acc, status), Satellite 3d Model, Time, Evaluations, Status Line

	GUI
•	OpenGL GLUT library
	DevIL image library
MAN	
	windows: Main, Text, Cons, GT
	Console-Class
	Camera-Class
	array: Images, Textures
	array: Planets
	array: Vectors
	array: Points
	various toythuffor
	modoci fullcoroon 3d modol
	large console
nods:	
	GLUT callback functions:
	OnFisplay
	OnReshape,
	OnKeyboard,
	OnMouse, OnTimer, OnIdle
	DRAW functions:
	Vector
	Planet
	Point
	CoordSys
	3d-Model
	Textpage
	get_pressed_key()
	get consolecmd()

**SIM Class** - next page -

#### bool use\_gui speedmode sim2gui\_data(): updates the GUI-data on\_idle(): do next step in simulation on\_timer(): write GUI textwindow call sim2gui\_data() request display refresh start simulation w/o GUI run\_nogui(): sim\_start(): create simlog, open ephemeris logfiles, set running flag sim\_stop(): counterpart to sim\_start() interact(): process hotkey or console cmd create\_simlog(): writes all textpages into a file get\_systemtimedate(): get actual time&date start other application, start\_app(...): e.g. texteditor

GUI-CALLBACK functions: only wrapped

create\_dir(...): cons(...):

create directory (simlog) write text to console (GUI or not)

#### **SIM class**

**MEMMAN** 

FILEMAN

#### **CelestialBodies class**

calculates celestial body position, stores GM values

#### **EGMTides class**

computation of Earth's gravity field, tide models, AOD1B and geopotential derivatives

#### **Atmosphere class**

estimates atmosphere density; three models; can use solar & geomagnetic activity

#### **Earth Rotation class**

delivers rotation matrices for celestial terrestrial coordinate transformations; IAU-2000 or only z-rotation

**Time class** conversion between time systems: UT1, UTC, TDB, TT, Julian Date

vector <RungeKuttaIntegr> various RK methods can be used in one simulation

various Multistep integration methods can be used in one simulation

the Kepler propagator

vector <Satellite> multiple satellite can be simulated at once

Coordinate Sys. Transf. tCOF GCRF2ITRF

#### vars:

current\_time stores actual simulation time

### Satellite

name mass

vars:

CoM StateVec: position, velocity, acc. BFS StateQuat: attitude in quaternions kepler elements

propagator\_type and id

**Coordinate System Transformation:** 

tCOF GCRF2NTW, GCRF2RSW, GCRF2sGCRF

### 3d model

vars:

surfaces volume units point masses, mass distribution thrusters, magnetic dipole moments

Mol tensor, principal axes

**Coordinate System Transformation:** 

tCOF sGCRF2BFS, BFS2PAS

vector <ForceTorqueParam> this vector defines the force & torque models acting on this satellite and stores parameter necessary to calculate them

#### COF

Coordinate Frame transformations for vectors & matrices

#### vars:

origin position rotation matrix angular velocity vector angular acceleration vector timestamp

methods:

set(...) OLD2NEW(...) NEW2OLD(...)

#### methods:

**set\_environment(new\_time)** all environmental models will be set to the new\_time

**calc\_sat\_acc(satellite\_id);** this function calculates all torques and forces acting on the satellite using of the following methods:

> EGM\_ForceTorque(---) SRP\_ForceTorque(...) DRAG\_ForceTorque(...) CELBODIES\_ForceTorque(...) MAGNETIC\_ForceTorque(...)

#### 3.2 Basic OSTK classes

#### 3.2.1 class tmemman in memman.h

In OSTK an own memory manager is implemented, which provides malloc, resize and free functions. At startup of OSTK a Global Table is allocated for the Memory Manager. In this Global Table all memory allocations are logged. The memory manager has a function called list\_status, which will list all current memory allocations and the purpose of them. This should avoid memory leaks. Unfortunatly the memory address of the Global Table has to be passed to every sub-class, which needs a memory manager instance.

#### 3.2.2 class tfileman in fileman.h

This class is similiar to the memory manager with the difference, that it provides functions to read and write files.

#### 3.2.3 class tdatalyzer in datalyzer.h

Datalyzers are global accessable tables of long double values. The datalyzer class uses the memory manager to allocate space for the tables, and can use the file manager to write the tables into a file. The table size can be allocated dynamically. Furthermore functions for interpolation, differentiation or calculation of sums and averages are implemented. The data can be accessed directly via a pointer for accelerated computations.

#### 3.3 Basic OSTK types

The most important type is called ldouble and is on unix systems an abbreviation for long double. Windows systems have problems to handle long double, hence on windows machines it is usally a double. The definitions can be found in ldbl.h. Another basic type is a quaternion quat, which is defined in quat.h:

```
struct tquat{ ldouble q[4]; };
typedef quat;
```

OSTK often handles physical values, which have for example a unit. These physical values are stored and passed through OSTK in the classes located in **basictypes.h**. All these classes have the following properties in common:

- **timestamp**: In general physical values are time dependent. The timestamp indicates, at which simulation time the values was calculated. The numerical integrator, for example, prints out a warning if two accelerations with different timestamps had been passed. This should avoid irregular computation of accelerations (due to bugs or commented out lines).
- unit: useful for output (graphical or in logfiles) is set automatically
- cdsys: describes the coordinate system this value belongs to
- satid: usually a physical value belongs to a satellite, this is the id of the satellite

#### 3.3.1 tVec

In this type a 3-dimensional vector ldouble a[3] is stored. It can have one of the following types:

```
position, velocity, acceleration, torque, direction
```

The unit of this vector is chosen automatically.

#### 3.3.2 tTensor

A gravity gradient tensor or the moment of inertia tensor can be hold here.

#### 3.3.3 tStateVec

This type consists of a position, velocity and acceleration vector. It is used for satellite's CoM. This type is passed to the integrator.

#### 3.3.4 tStateQuat

This type consists of a quaternion and the first two time derivatives  $\underline{q}, \underline{\dot{q}}, \underline{\ddot{q}}$ . This type is used to express the attitude of the satellite. It is also passed to the integrator.

### Chapter 4

## **Coordinate Systems**

This chapter introduces the basics for coordinate system transformation and summarizes the used coordinate systems in OSTK. For example, it is convenient to calculate the Earth's gravitational acceleration in an Earth-fixed frame and to transform the acceleration into a space-fixed frame. Or the torques due to drag and solar radiation pressure are usually calculated in the body-fixed frame, but need to be rotated into the principal axis system to be considered in Euler's Moment equations.

#### 4.1 Rotations

Let us assume two coordinate systems with the same origin O. The coordinate system  $\Sigma$  is a fixed, inertial system with basis vectors:

$$\vec{e}_x = \begin{pmatrix} 1\\0\\0 \end{pmatrix}, \quad \vec{e}_y = \begin{pmatrix} 0\\1\\0 \end{pmatrix}, \quad \vec{e}_z = \begin{pmatrix} 0\\0\\1 \end{pmatrix}$$
(4.1)

while  $\Sigma'$  is rotated and has basis vectors  $\vec{e}_x \prime, \vec{e}_y \prime, \vec{e}_z \prime$ . A position vector  $\vec{r}$  can be expressed in both systems:

$$\vec{r} = r_x \vec{e}_x + r_y \vec{e}_y + r_z \vec{e}_z = r_x \vec{e}_x \prime + r_y \prime \vec{e}_y \prime + r_z \prime \vec{e}_z \prime = \vec{r} \prime$$
(4.2)

Note that the coefficients, denoted with the subscript  $|\Sigma$  or  $|\Sigma'$ , are not equal

$$\vec{r}_{|\Sigma} = \begin{pmatrix} r_x \\ r_y \\ r_z \end{pmatrix} \neq \begin{pmatrix} r'_x \\ r'_y \\ r'_z \end{pmatrix} = \vec{r}_{|\Sigma'}.$$
(4.3)

However the relation between both systems can be written as:

$$\vec{r}_{|\Sigma} = \begin{pmatrix} r_x \\ r_y \\ r_z \end{pmatrix} = \hat{R}_{|\Sigma'}^{|\Sigma'} \cdot \vec{r}_{|\Sigma'}.$$
(4.4)

#### 4.1.1 Rotation Matrix

The rotation matrix is a linear map  $\hat{R} : \mathbb{R}^3 \longrightarrow \mathbb{R}^3$ , which rotates a vector in the Euclidean Space. A rotation matrix has a basis, which is denoted with subscripts (indices). The system which is rotated is written in the superscript.

$$\hat{R}_{|\Sigma}^{|\Sigma'}: \Sigma' \longrightarrow \Sigma \tag{4.5}$$

The rotation matrix can be expressed in terms of basis vectors  $\vec{e_x'}, \vec{e_y'}, \vec{e_{z'}'}$ :

$$\hat{R}_{|\Sigma}^{|\Sigma'} = \begin{bmatrix} (\vec{e_x'})_x & (\vec{e_y'})_x & (\vec{e_z'})_x \\ (\vec{e_x'})_y & (\vec{e_y'})_y & (\vec{e_z'})_y \\ (\vec{e_x'})_z & (\vec{e_y'})_z & (\vec{e_z'})_z \end{bmatrix} = \begin{bmatrix} \uparrow & \uparrow & \uparrow \\ \vec{e_x'} & \vec{e_y'} & \vec{e_z'} \\ \downarrow & \downarrow & \downarrow \end{bmatrix}$$
(4.6)

An advantage of this definition is the simple way to calculate the time derivatives  $\hat{R}$  and  $\hat{R}$  by deriving every component of the matrix.

Another method to define the rotation matrix is to use the unit rotation axis  $\vec{u}_{|\Sigma} = (x, y, z)^T$ ,  $|\vec{u}| = 1$  and rotation angle  $\alpha$ . This method is used in axisangle2rotmatl(...) in lutils.cpp to compute the following matrix ([Jazar, 2010, p.92, eq. 3.5]:

$$\hat{R}_{|\Sigma}^{|\Sigma'} = \begin{bmatrix} \cos\alpha + x^2 \cdot (1 - \cos\alpha) & x \cdot y \cdot (1 - \cos\alpha) - z \cdot \sin\alpha & x \cdot z \cdot (1 - \cos\alpha) + y \cdot \sin\alpha \\ x \cdot y \cdot (1 - \cos\alpha) + z \cdot \sin\alpha & \cos\alpha + y^2 \cdot (1 - \cos\alpha) & y \cdot z \cdot (1 - \cos\alpha) - x \cdot \sin\alpha \\ x \cdot z \cdot (1 - \cos\alpha) - y \cdot \sin\alpha & y \cdot z \cdot (1 - \cos\alpha) + x \cdot \sin\alpha & \cos\alpha + z^2 \cdot (1 - \cos\alpha) \end{bmatrix}$$
(4.7)

The inverse of a rotation matrix and hence the inverse rotation or inverse basis transformation is given by the transposed matrix [Goldstein, 1991, eq. 4-34]

$$\left(\hat{R}_{|\Sigma'}^{|\Sigma'}\right)^{-1} = \left(\hat{R}_{|\Sigma'}^{|\Sigma'}\right)^{T} = \hat{R}_{|\Sigma'}^{|\Sigma} : \Sigma \longrightarrow \Sigma'.$$
(4.8)

#### 4.1.2 Convention: Rotation Axis and Angle

Since the transformation between rotation matrix and rotation axis and angle is not unique, the following convention is used in OSTK:

- The rotation axis is oriented in such a way, that the z-component is positive. If z = 0, then y > 0. If z = y = 0, then x > 0.
- The rotation angle has the range  $(-\pi, +\pi]$ , which can be achieved by using this mathematical C/C++ operation

$$\alpha = \operatorname{atan2}(\sin(\alpha), \cos(\alpha)) \tag{4.9}$$

The function boundRotationAxis(...) in lutils.cpp calculates the rotation axis and angle, obeying the convention, from arbitrary rotation axis and angle.

#### 4.1.3 How to check if a matrix is a rotation matrix?

A matrix  $\hat{R}$  is a rotation matrix, if both conditions are true:

$$\det(\hat{R}) = +1 \tag{4.10}$$

$$\hat{R} \cdot \hat{R}^T = \hat{1} \tag{4.11}$$

These conditions are checked by the function isRotationMatrix(...) in lutils.cpp

## 4.1.4 How to calculate the rotation axis and angle from a rotation matrix?

Using the definiton of a rotation matrix in eq. 4.7, one can easily prove that:

$$\cos \alpha = \frac{1}{2} \cdot \operatorname{trace}(\hat{R}) \tag{4.12}$$

and

$$\hat{Q} := \hat{R} - \hat{R}^T = 2\sin(\alpha) \cdot \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix}$$
(4.13)

Using the matrix  $\hat{Q}$ , we can define the vector

$$\vec{t} = \begin{pmatrix} Q_{2,1} - Q_{1,2} \\ Q_{0,2} - Q_{2,0} \\ Q_{1,0} - Q_{0,1} \end{pmatrix}$$
(4.14)
It is useful to define this vector using not only single matrix elements, but the difference or sum between the elements, to avoid numerical instabilities, if the rotation matrix gets deorthogonalized during mathematical operations. The normalized vector  $\vec{q} = \frac{\vec{t}}{|\vec{t}|}$  is parallel to the rotation axis. One has to check the orientation of the rotation axis and if necessary to reorient it, using the function boundRotationAxis(...) in lutils.cpp. If the z-component is non-zero, the following equation can be used:

$$\sin \alpha = \frac{Q_{1,0}}{2.0 \cdot t_z} \tag{4.15}$$

to get the sine of the rotation angle. The cosine is given in eq. 4.12, hence eq. 4.9 yields to the rotation angle. One has to consider the special cases, where  $\hat{Q} = 0$  in case of rotation angles of 0 or  $\pi$ . This method is used in rotmat2axisangle1(...) in lutils.cpp.

#### 4.1.5 Euler Angles

It is possible to express every rotation with three Euler angles, often denoted as  $\Phi, \Theta, \Psi$ . These three angles describe rotations around the body fixed axes and have to be executed in a predefined order, because they do not commutate[Goldstein, 1991, p.119]. There are 12 possible conventions according to the sign and order for the Euler angles[Goldstein, 1991, p.120]<sup>1</sup>, and in addition the rotation matrices describing these single rotations are not always invertible. Hence they are not optimal to describe rotations, but still very common. In OSTK an arbitrary convention of Euler angles can be expressed with 6 characters, e.g. "+Z-X+Y", which defines the sign and order of rotations. In this example, at first a rotation around the z-axis is performed with angle  $\Phi$ , then a rotation around the X axis with angle  $-\Theta$  and in the end a rotation around the Y-axis with angle  $\Psi$ . These three steps transform a coordinate system  $\Sigma$  into a coordinate system  $\Sigma'$ . This overall transformation  $\hat{R}$  can be described as a product of three elementary rotation matrices:

$$\hat{R} = \hat{R}_Y(+\Phi) \cdot \hat{R}_X(-\Theta) \cdot \hat{R}_Z(+\Psi)$$
(4.16)

$$\hat{R}_X(\alpha) = \begin{bmatrix} 1 & 0 & 0\\ 0 & \cos(\alpha) & -\sin(\alpha)\\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix} \quad \hat{R}_Y(\alpha) = \begin{bmatrix} \cos(\alpha) & 0 & +\sin(\alpha)\\ 0 & 1 & 0\\ -\sin(\alpha) & 0 & \cos(\alpha) \end{bmatrix}$$
(4.17)

$$\hat{R}_Z(\alpha) = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0\\ \sin(\alpha) & \cos(\alpha) & 0\\ 0 & 0 & 1 \end{bmatrix}$$
(4.18)

The function EulerAngles2Rotmat(...) in lutils.cpp will calculate the resulting rotation matrix from a convention string and Euler angles. The inverse transformation is not yet implemented.

#### 4.1.6 Quaternions

Quaternions provide an elegant way to describe rotations in the three dimensional space. Equations presented here are mainly from [Gould, 2007]. Quaternions can be constructed by extending the  $\mathbb{R}$  with hyper-complex numbers i, j, k, similar to the familiar complex numbers:

$$q_{\rm c} = q_0 + iq_1 + jq_2 + kq_3 \tag{4.19}$$

The hyper-complex numbers obey special rules and do not commute [Gould, 2007, sec. 7.5]. Addition and subtraction is handled like in the  $\mathbb{R}^4$ :

$$r = \underline{q} + \underline{p} = (q_0 + p_0) + i \cdot (q_1 + p_1) + j \cdot (q_2 + p_2) + k \cdot (q_3 + p_3)$$
(4.20)

$$\underline{r} = \underline{q} - \underline{p} = (q_0 - p_0) + i \cdot (q_1 - p_1) + j \cdot (q_2 - p_2) + k \cdot (q_3 - p_3)$$
(4.21)

<sup>&</sup>lt;sup>1</sup>and Wikipedia

The product is more complicated and does not commute:

$$\underline{r} = \underline{q} \cdot \underline{p} \tag{4.22}$$

$$r_0 = q_0 \cdot p_0 - q_1 \cdot p_1 - q_2 \cdot q_2 - q_3 \cdot p_3 \tag{4.23}$$

$$r_{1} = q_{0} \cdot p_{1} + q_{1} \cdot p_{0} + q_{2} \cdot q_{3} - q_{3} \cdot p_{2}$$

$$r_{2} = q_{0} \cdot p_{0} - q_{1} \cdot p_{2} + q_{2} \cdot q_{3} + q_{2} \cdot p_{1}$$

$$(4.24)$$

$$(4.25)$$

$$r_{2} = q_{0} \cdot p_{2} - q_{1} \cdot p_{3} + q_{2} \cdot q_{0} + q_{3} \cdot p_{1}$$

$$r_{2} = q_{0} \cdot p_{2} + q_{1} \cdot p_{2} - q_{2} \cdot q_{1} + q_{2} \cdot p_{0}$$

$$(4.25)$$

$$(4.26)$$

$$r_3 = q_0 \cdot p_3 + q_1 \cdot p_2 - q_2 \cdot q_1 + q_3 \cdot p_0 \tag{4.20}$$

The complex conjugated quaternion is:

$$\underbrace{q}_{\rightarrow}^{*} = q_0 - iq_1 - jq_2 - kq_3 \tag{4.27}$$

(4.28)

and the length of a quaternion is

$$|\underline{q}| = \sqrt{\underline{q} \cdot \underline{q}^*} = \sqrt{q_0 \cdot q_0 + q_1 \cdot q_1 + q_2 \cdot q_2 + q_3 \cdot q_3}.$$
(4.29)

A rotation with unit rotation axis  $\vec{u} = (u_1, u_2, u_3)$  and rotation angle  $\varphi$  can be expressed with a unit quaternion

$$\underline{q} = \cos\left(\frac{\varphi}{2}\right) + (iu_1 + ju_2 + ku_3) \cdot \sin\left(\frac{\varphi}{2}\right).$$
(4.30)

A position vector  $\vec{r} = (x, y, z)$  can be rotated by defining the quaternion

$$\underline{r} = 0 + ix + jy + kz, \tag{4.31}$$

and using the following relation:

$$\underline{r'} = \underline{q} \cdot \underline{r} \cdot \underline{q}^* = (0, x', y', z') \tag{4.32}$$

where the prime denotes the rotated vector, like in

$$\vec{r'} = \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \hat{R}(u,\varphi) \cdot \vec{r}.$$
(4.33)

One can introduce an extended notation like for rotation matrices:

$$\underbrace{q}_{|\Sigma}^{|\Sigma'} \tag{4.34}$$

where we can see between which systems a quaternion rotates.

The basic operations for working with quaternions are shown in listing 4.1:

Listing 4.1: basic quaternion functions in quaternion.h

```
// add two quaternions
<u>quat</u> qtadd(<u>const</u> <u>quat</u> x, <u>const</u> <u>quat</u> y);
// subtract two quaternions
<u>quat</u> qtsub(<u>const</u> <u>quat</u> x, <u>const</u> <u>quat</u> y);
// multiply two quaternions
<u>quat</u> qtdot(<u>const</u> <u>quat</u> x, <u>const</u> <u>quat</u> y);
// set a quaternion with given components
<u>quat</u> qtset(<u>const</u> <u>ldouble</u> x,<u>const</u> <u>ldouble</u> y,<u>const</u> <u>ldouble</u> z,<u>const</u> <u>ldouble</u> w);
   conjugate quaternion
quat qtconj(const quat vec);
   norm/length of a quaternion
ldouble qtlength(const quat vec);
   convert rotation axis \mathcal{B} angle to quaternion
<u>quat</u> qtrotaxis2quat(<u>const</u> <u>ldouble</u> angle, <u>const</u> <u>ldouble</u> rotunitvec[3]);
// normalize a quaternion
quat qtnormalize(const quat vec);
    convert a "quaternion-vector" to usual vector in R^3
void qtget_vec(const quat qvec, ldouble vec[3]);
// rotate a vector using a quaternion
quat qtrotatevec(const ldouble vec[3], const quat unitquat);
// rotate a vector using a quaternion
void qtrotatevec(const ldouble vec[3],const quat unitquat, ldouble destvec[3])
     ;
```

# 4.1.7 Relation between Quaternions and Rotation Matrices

The relation between a rotation matrix and quaternion with length 1 is given by [Gould, 2007, eq. 17.34]

$$\hat{R} = \begin{bmatrix} \frac{1}{2} - q_2^2 - q_3^3 & q_1q_2 + q_0q_3 & q_1q_3 + q_0q_2 \\ q_1q_2 - q_0q_3 & \frac{1}{2} - q_1^2 - q_3^3 & q_2q_3 + q_0q_1 \\ q_1q_3 + q_0q_2 & q_2q_3 - q_0q_1 & \frac{1}{2} - q_1^2 - q_2^3 \end{bmatrix}$$
(4.35)

The inverse transformation can be found in [Gould, 2007, App. 17B]. Both transformations are implemented in OSTK as seen in Listing 4.2.

Listing 4.2: quaternion transformation functions in quaternion.h

```
// convert quaternion to rotation matrix
void qtquat2rotmat(quat q, ldouble destmat[3][3]);
// convert rotation matrix to quaternion
quat qtrotmat2quat(const ldouble r[3][3]);
```

# 4.2 Angular Velocity

The angular velocity  $\vec{\omega}$  of a point mass is given next to the definition from eq. (2.19) by [Williams, 1996, eq. 3-13]

$$\vec{\omega} = \frac{\vec{r} \times \dot{\vec{r}}}{|\vec{r}|^2} \quad \Leftrightarrow \quad \dot{\vec{r}} = \frac{d\vec{r}}{dt} = \vec{\omega} \times \vec{r}, \tag{4.36}$$

which connects the linear and the angular velocity. The direction of  $\vec{\omega}$  is the actual rotation axis and the magnitude is the angular velocity given in rad/s.

In the next steps the mathematical equations are derived to calculate the angular velocity of a coordiante system, which can be body fixed frame of the satellite. In a rotating frame with basis  $\{\vec{e}_x \prime, \vec{e}_y \prime, \vec{e}_z \prime\}$  the endpoints of the basis vectors can be assumed to be point-masses. Thus we get the three equations

$$\dot{\vec{e_x\prime}} = \frac{d\vec{e_x\prime}}{dt} = \vec{\omega}_x \times \vec{e_x\prime}$$
(4.37)

$$\dot{\vec{e_y\prime}} = \frac{d\vec{e_y\prime}}{dt} = \vec{\omega}_y \times \vec{e_y\prime}$$
(4.38)

$$\dot{\vec{e_z\prime}} = \frac{d\vec{e_z\prime}}{dt} = \vec{\omega}_z \times \vec{e_z\prime}.$$
(4.39)

Since the basis vectors keep their length during rotations, the velocity has to be perpendicular to the direction (cf. circular movement):

$$\vec{e_i'} \perp \vec{e_i'} \qquad i \in \{x, y, z\} \tag{4.40}$$

However, all three coordinate axes can not rotate independently and one can exploit some algebraic relations [Schnizer, 2003] which result in the fact, that all time-derivatives  $\dot{\vec{e_i}}$  are in one plane. Furthermore all  $\vec{\omega_i}$  are equal:

$$\vec{\omega} = \vec{\omega}_x = \vec{\omega}_y = \vec{\omega}_z. \tag{4.41}$$

The normal vector of this plane has the direction of  $\vec{\omega}$ , hence all time derivatives are perpendicular to the angular velocity:

$$\vec{\omega} \perp \vec{e_x'} \qquad \vec{\omega} \perp \vec{e_y'} \qquad \vec{\omega} \perp \vec{e_z'}$$
(4.42)

thus we can use the definition for the angular velocity:

$$\dot{\vec{e_i'}} = \frac{d\vec{e_i'}}{dt} = \vec{\omega} \times \vec{e_i'} \qquad i \in \{x, y, z\}$$
(4.43)

Calculating the dot-product with  $\vec{e_{i'}}$  for  $i \neq j$  yields to:

$$\vec{e_i'} \cdot \vec{e_j'} = (\vec{\omega} \times \vec{e_i'}) \cdot \vec{e_j'} = (\vec{e_i'} \times \vec{e_j'}) \cdot \vec{\omega} = \epsilon_{ijk} \ \vec{e_k'} \cdot \vec{\omega}$$
(4.44)

where we used the Levi-Cevita Symbol  $\epsilon_{ijk}$  and this relation for a right-handed coordinate system:

$$\vec{e}_i \times \vec{e}_j = \epsilon_{ijk} \ \vec{e}_k. \tag{4.45}$$

The product  $\vec{e_{k'}} \cdot \vec{\omega}$  is the k.th-coefficient  $\omega_{k'}$  in the basis  $\{\vec{e_{x'}}, \vec{e_{y'}}, \vec{e_{z'}}\}$ :

$$\vec{\omega} = (\vec{e_x'} \cdot \vec{\omega}) \cdot \vec{e_x'} + (\vec{e_y'} \cdot \vec{\omega}) \cdot \vec{e_y'} + (\vec{e_z'} \cdot \vec{\omega}) \cdot \vec{e_z'}$$
(4.46)

$$= \omega_x' \cdot \vec{e_x'} + \omega_y' \cdot \vec{e_y'} + \omega_z' \cdot \vec{e_z'}$$

$$(4.47)$$

$$=\vec{\omega}(\Sigma')\tag{4.48}$$

This is the angular velocity of the coordinate system  $\{\vec{e}_x\prime, \vec{e}_y\prime, \vec{e}_z\prime\}$  expressed in an arbitrary basis. The argument of  $\vec{\omega}$  denotes the system, which rotates. This lengthy derivation was done, because the angular velocity is not very intuitiv, as it will be shown in the next section.

# 4.2.1 How to calculate $\vec{\omega}$ and $\dot{\vec{\omega}}$ from a Rotation Matrix?

Using the definition of  $\hat{R}$  and  $\hat{R}$  and matrix multiplication one obeys with support of eq. 4.44 the skew-symmetric angular velocity tensor  $\hat{\omega}$ :

$$\dot{\hat{R}} \cdot \hat{R}^{T} = \begin{bmatrix} \uparrow & \uparrow & \uparrow \\ e_{x'}^{\prime} & e_{y'}^{\prime} & e_{z'}^{\prime} \end{bmatrix} \cdot \begin{bmatrix} \leftarrow & e_{x'}^{\prime} & \rightarrow \\ \leftarrow & e_{y'}^{\prime} & \rightarrow \\ \leftarrow & e_{z'}^{\prime} & \rightarrow \end{bmatrix} = \begin{bmatrix} 0 & -\omega_{z'} & +\omega_{y'} \\ +\omega_{z'} & 0 & -\omega_{x'} \\ -\omega_{y'} & +\omega_{x'} & 0 \end{bmatrix} = \hat{\omega} = -(\hat{\omega})^{T} \quad (4.49)$$

#### 4.2. ANGULAR VELOCITY

or with extended notation

$$\dot{\hat{R}}_{|\Sigma}^{|\Sigma\prime} \cdot \left(\hat{R}_{|\Sigma}^{|\Sigma\prime}\right)^T = \hat{\omega}(\Sigma\prime) \tag{4.50}$$

The diagonal elements vanish, because  $\vec{e_i} \prime \perp \vec{e_i} \prime$ . Differentiation with respect to time yields to the skew-symmetric angular acceleration tensor:

$$\frac{d}{dt}(\dot{\hat{R}}\cdot\hat{R}^{T}) = \ddot{\hat{R}}\cdot\hat{R}^{T} + \dot{\hat{R}}\cdot\dot{\hat{R}}^{T} = \begin{bmatrix} 0 & -\dot{\omega_{z}}\prime & +\dot{\omega_{y}}\prime \\ +\dot{\omega_{z}}\prime & 0 & -\dot{\omega_{x}}\prime \\ -\dot{\omega_{y}}\prime & +\dot{\omega_{x}}\prime & 0 \end{bmatrix} = \dot{\hat{\omega}}$$
(4.51)

The corresponding vectors are

$$\vec{\omega} = \frac{1}{2} \begin{pmatrix} \hat{\omega}_{2,1} - \hat{\omega}_{1,2} \\ \hat{\omega}_{0,2} - \hat{\omega}_{2,0} \\ \hat{\omega}_{1,0} - \hat{\omega}_{0,1} \end{pmatrix} \quad \text{and} \quad \dot{\tilde{\omega}} = \frac{1}{2} \begin{pmatrix} \dot{\hat{\omega}}_{2,1} - \dot{\hat{\omega}}_{1,2} \\ \dot{\hat{\omega}}_{0,2} - \dot{\hat{\omega}}_{2,0} \\ \dot{\hat{\omega}}_{1,0} - \dot{\hat{\omega}}_{0,1} \end{pmatrix}.$$
(4.52)

It is important to notice, that the angular velocity vector of a system  $\Sigma'$ 

$$\vec{\omega}(\Sigma t) = \begin{pmatrix} \omega'_x \\ \omega'_y \\ \omega'_z \end{pmatrix}$$
(4.53)

describes the angular velocity of the rotating frame with respect to a fixed frame in the rotating frame(!) basis. Furthermore the angular velocity vector is an eigenvector of the rotation matrix and is in the kernel (nullspace) of  $\hat{R}$  (eq. 4.42).

In addition there is a simple relation between the angular velocity tensor and vector:

$$\hat{\omega} \cdot \vec{r} = \vec{\omega} \times \vec{r} \tag{4.54}$$

for an arbitrary vector  $\vec{r}$ .

# 4.2.2 How to calculate $\vec{\omega}$ and $\vec{\omega}$ from Quaternions?

There are simple relations between the angular velocity and quaternions [Gould, 2007]

$$\frac{\dot{q}}{\dot{q}} = \begin{pmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{pmatrix} = \frac{1}{2} \hat{W} \begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ 0 \end{pmatrix} \tag{4.55}$$

$$\begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ 0 \end{pmatrix} = 2 \hat{W}^T \begin{pmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{pmatrix} \tag{4.56}$$

and for angular acceleration:

$$\frac{\ddot{q}}{\overrightarrow{q}} = \begin{pmatrix} \ddot{q}_0 \\ \dot{q}_1 \\ \ddot{q}_2 \\ \ddot{q}_3 \end{pmatrix} = \frac{1}{2} \hat{W} \begin{pmatrix} \dot{w}_1 \\ \dot{w}_2 \\ \dot{w}_3 \\ -2\sum \dot{q}_m^2 \end{pmatrix}$$
(4.57)

$$\begin{pmatrix} \dot{w}_1 \\ \dot{w}_2 \\ \dot{w}_3 \\ -2\sum \dot{q}_m^2 \end{pmatrix} = 2\hat{W}^T \begin{pmatrix} \ddot{q}_0 \\ \ddot{q}_1 \\ \ddot{q}_2 \\ \ddot{q}_3 \end{pmatrix}$$
(4.58)

where the matrix  $\hat{W}$  is an orthogonal matrix  $\hat{W}\hat{W}^T = \hat{W}^T\hat{W} = \hat{1}$ 

$$\hat{W}(\underline{q}) = \begin{bmatrix} -q_1 & q_0 & q_3 & -q_2 \\ -q_2 & -q_3 & q_0 & q_1 \\ -q_3 & q_2 & -q_1 & q_0 \\ q_0 & q_1 & q_2 & q_3 \end{bmatrix}$$
(4.59)

The implementation in OSTK is shown in listing 4.3

Listing 4.3: quaternion to angular velocity and acceleration (and v.v.) functions in quaternion.h

```
// convert angular velocity and acceleration to quaternion's derivatives
void qtomega2quat(const quat q, const ldouble omega[3], const ldouble omegadot
[3], quat &qd, quat &qdd);
// convert quaternion's derivat. to angular velocity and acc.
void qtquat2omega(quat q, quat qd, quat qdd, ldouble omega[3], ldouble omegadot
[3]);
```

# 4.3 Transformation between two coordinate systems

Let us assume an inertial frame  $\Sigma$  with basis vectors  $\vec{e}_x, \vec{e}_y, \vec{e}_z$  and a moving and rotating frame  $\Sigma'$  with basis vectors  $\vec{e}_x', \vec{e}_y', \vec{e}_z'$ . The origin of  $\Sigma'$  has the position  $\vec{P}$ , the velocity  $\dot{\vec{P}}$  and acceleration  $\ddot{\vec{P}}$ . Furthermore  $\Sigma'$  is rotated w.r.t.  $\Sigma$ . A point mass is located at position S (see Figure 4.1).



Figure 4.1: Scheme of the three coordinate systems (for simplicity only 2-dimensional)

There are various ways to transform the vector coefficients from one system to another. One can use only the rotation matrices or the angular velocity vector. Furthermore a shifted but not rotated frame called  $\tilde{\Sigma}$  and the vector  $\vec{Q} = S_{|\tilde{\Sigma}}$  are introduced. This vector abbreviates the formulas in the source code. The rotation matrix and angular velocity in the following sections are defined as previously

$$\hat{R} = \hat{R}_{|\Sigma'}^{|\Sigma'} : |\Sigma' \to |\Sigma$$
$$\vec{\omega} = \vec{\omega}(\Sigma')$$

The vector coefficient notation, which is useful for programming, as well as the vector notation, which is usually found in the textbooks, will be shown in the next sections.

# **4.3.1** Transformation of a position vector: $\Sigma' \longrightarrow \Sigma$

The transformation of the position vector is given by

$$\vec{S}_{|\Sigma} = \underbrace{\hat{R} \cdot \vec{S}_{|\Sigma'}}_{\vec{Q}} + \vec{P}_{|\Sigma}$$
(4.60)

where we know all variables on the right hand side. In vector notation

$$\vec{S} = \vec{S}' + \vec{P} \tag{4.61}$$

# **4.3.2** Transformation of a velocity vector: $\Sigma' \longrightarrow \Sigma$

Differentiation of 4.60 yields to [Williams, 1996]:

$$\vec{S}_{|\Sigma} = \hat{R} \cdot \vec{S}_{|\Sigma'} + \hat{R} \cdot \vec{S}_{|\Sigma'} + \vec{P}_{|\Sigma}$$

$$(4.62)$$

$$= \vec{\omega} \times \underbrace{\hat{R} \cdot \vec{S}_{|\Sigma'}}_{} + \hat{R} \cdot \dot{\vec{S}}_{|\Sigma'} + \vec{P}_{|\Sigma}$$

$$(4.63)$$

$$\dot{\vec{S}} = \vec{\omega} \times \vec{S}' + \dot{\vec{S}}' + \dot{\vec{P}}$$
(4.64)

The single terms are respectively related to

- the rotation of the system  $\Sigma'$
- the movement of the point mass with respect to  $\Sigma'$
- the movement of the system  $\Sigma'$

From the last equation we derive this relation:

$$\hat{R} \cdot \vec{S}_{|\Sigma'} = \vec{Q} - \vec{\omega} \times \vec{Q} \tag{4.65}$$

# 4.3.3 Transformation of a velocity vector: $\Sigma' \longrightarrow \Sigma$ (rotation only)

If we assume, that both coordinate systems have the same origin, the transformation formula simplifies to

$$\dot{\vec{S}}_{|\Sigma} = \vec{\omega} \times \vec{S}_{|\Sigma} + \hat{R} \cdot \dot{\vec{S}}_{|\Sigma'} \tag{4.66}$$

$$\vec{S} = \vec{\omega} \times \vec{S} + \vec{S}' \tag{4.67}$$

which is widely used in physical derivations.

# **4.3.4** Transformation of an acceleration vector: $\Sigma' \longrightarrow \Sigma$

Further differentiation will lead to pseudo forces (after lengthy transformations):

$$\ddot{\vec{S}}_{|\Sigma} = \ddot{\hat{R}} \cdot \vec{S}_{|\Sigma\prime} + \dot{\hat{R}} \cdot \dot{\vec{S}}_{|\Sigma\prime} + \dot{\hat{R}} \cdot \dot{\vec{S}}_{|\Sigma\prime} + \dot{\hat{R}} \cdot \dot{\vec{S}}_{|\Sigma\prime} + \dot{\hat{R}} \cdot \ddot{\vec{S}}_{|\Sigma\prime} + \ddot{\vec{P}}_{|\Sigma}$$
(4.68)

$$= \hat{R} \cdot \vec{S}_{|\Sigma'} + 2\hat{R} \cdot \vec{S}_{|\Sigma'} + \hat{R} \cdot \vec{S}_{|\Sigma'} + \vec{P}_{|\Sigma}$$
(4.69)

$$= \dot{\vec{\omega}} \times \vec{S}' + \vec{\omega} \times (\vec{\omega} \times \vec{S}') + 2 \cdot \vec{\omega} \times \dot{\vec{S}}' + \ddot{\vec{S}}' + \vec{\vec{P}}$$
(4.70)

This is the classical transformation formula [Williams, 1996, p.92]. The single summands are called:

- $\dot{\vec{\omega}} \times \hat{R}\vec{S}_{|\Sigma'}$ : tangential acceleration
- $2 \cdot \vec{\omega} \times \hat{R} \vec{S}_{|\Sigma|}$ : coriolis acceleration
- $\vec{\omega} \times (\vec{\omega} \times \hat{R}\vec{S}_{|\Sigma'})$ : centripetal acceleration
- $\hat{R} \cdot \ddot{\vec{S}}_{|\Sigma'}$ : acceleration in rotating frame
- $\vec{P}$ : acceleration of the origin

In  $\vec{Q}$  Notation this formula keeps the structure, but some sign changes will occur:

$$\ddot{\vec{S}}_{|\Sigma} = \dot{\vec{\omega}} \times \hat{R}\vec{S}_{|\Sigma'} + \vec{\omega} \times (\vec{\omega} \times \hat{R}\vec{S}_{|\Sigma'}) + 2 \cdot \vec{\omega} \times \hat{R}\dot{\vec{S}}_{|\Sigma'} + \hat{R} \cdot \ddot{\vec{S}}_{|\Sigma'} + \ddot{\vec{P}}_{|\Sigma}$$
(4.71)

$$= \dot{\vec{\omega}} \times \vec{Q} + \vec{\omega} \times (\vec{\omega} \times \vec{Q}) + 2 \cdot \vec{\omega} \times \hat{R}\vec{S}_{|\Sigma'} + \hat{R} \cdot \vec{S}_{|\Sigma'} + \vec{P}_{|\Sigma}$$

$$(4.72)$$

$$\stackrel{4.65}{=} \dot{\vec{\omega}} \times \vec{Q} + \vec{\omega} \times (\vec{\omega} \times \vec{Q}) + 2 \cdot \vec{\omega} \times (\vec{Q} - \vec{\omega} \times \vec{Q}) + \hat{R} \cdot \vec{S}_{|\Sigma'} + \vec{P}_{|\Sigma}$$
(4.73)

$$= \dot{\vec{\omega}} \times \vec{Q} - \vec{\omega} \times (\vec{\omega} \times \vec{Q}) + 2 \cdot \vec{\omega} \times \vec{Q} + \hat{R} \cdot \vec{S}_{|\Sigma'} + \vec{P}_{|\Sigma}$$

$$(4.74)$$

# 4.3.5 Inverse Transformations: $\Sigma \longrightarrow \Sigma'$

For the inverse direction it is easy to calculate the vectors  $\vec{Q}, \dot{\vec{Q}}$  and  $\ddot{\vec{Q}}$ 

$$\vec{Q} = \vec{S}_{|\Sigma} - \vec{P}_{|\Sigma} \tag{4.75}$$

$$\vec{Q} = \vec{S}_{|\Sigma} - \vec{P}_{|\Sigma} \tag{4.76}$$

$$\ddot{\vec{Q}} = \ddot{\vec{S}}_{|\Sigma} - \ddot{\vec{P}}_{|\Sigma} \tag{4.77}$$

and the inverse transformations are

$$\vec{S}_{|\Sigma'} = \hat{R}^T \cdot (\vec{S}_{|\Sigma} - \vec{P}_{|\Sigma}) = \hat{R}^T \cdot \vec{Q}$$
(4.78)

$$\vec{S}_{|\Sigma'} = \hat{R}^T \left( \vec{Q} - \vec{\omega} \times \vec{Q} \right) \tag{4.79}$$

$$\ddot{\vec{S}}_{|\Sigma'} = \hat{R}^T \left( \ddot{\vec{Q}} - \dot{\vec{\omega}}_{|\Sigma'} \times \vec{Q} - 2 \cdot \vec{\omega}_{|\Sigma'} \times \dot{\vec{Q}} + \vec{\omega}_{|\Sigma'} \times (\vec{\omega} \times \vec{Q}) \right)$$
(4.80)

or in matrix notation:

$$\vec{S}_{|\Sigma'} = \hat{R}^T \cdot \vec{Q} \tag{4.81}$$

$$\dot{\vec{S}}_{|\Sigma'} = \hat{R}^T \vec{Q} + \dot{R}^T \vec{Q} \tag{4.82}$$

$$\ddot{\vec{S}}_{|\Sigma\prime} = \hat{R}^T \ddot{\vec{Q}} + \ddot{\vec{R}}^T \vec{Q} + 2\dot{\vec{R}}^T \vec{Q}$$

$$\tag{4.83}$$

# 4.3.6 Transformation of a Matrix

A tensor of second kind (e.g. the MoI tensor or gravity gradient tensor) can be represented by a matrix M and can be transformed using this relations [Schulz, 2006]:

$$\hat{M}_{|\Sigma} = \hat{R} \cdot \hat{M}_{|\Sigma'} \cdot \hat{R}^T \tag{4.84}$$

$$\hat{M}_{|\Sigma'} = \hat{R}^T \cdot \hat{M}_{|\Sigma} \cdot \hat{R} \tag{4.85}$$

# 4.4 OSTK TriadStateMat

This definition will simplify the calculation of satellite-fixed coordinate systems in section 4.6.2. Another application can be found in sec. 6.4.1.

# 4.4.1 Definition: OSTK TriadStateMat

A TriadStateMat<sup>2</sup> in OSTK is a 3-dimensional tensor, in C/C++:

#### ldouble TriadStateMat[3][3][3]

where the first dimension is the transposed rotation matrix  $(= \hat{R}^T)$ , the second dimension is the first time derivative of  $\dot{\hat{R}}^T$  and the third dimension is  $\ddot{\hat{R}}^T$ . For example: Let G be a TriadStateMat, then

$$G[0] = \hat{R}^T, \quad G[1] = \hat{R}^T, \quad G[2][2] = \ddot{\vec{e}}_z^T, \quad G[0][1][0] = (\vec{e}_y)_x$$

#### 4.4.2 How to validate a TriadStateMat?

For debugging purpose it is useful to have a function which can validate a TriadStateMat. The following methods are implemented in check\_TriadStateMat(...) in lutils.cpp :

- use of isRotationMatrix(...) in lutils.cpp for  $G[0] = \hat{R}^T$
- check if all  $G[1][i] = \dot{\vec{e}}_i^T$  are in one plane using the triple product
- check if  $G[0][i] = \vec{e}_i^T$  is perpendicular to  $G[1][j] = \dot{\vec{e}}_j^T$
- check if all  $G[2][i] = \ddot{\vec{e}}_i^T$  are in one plane using the triple product

 $<sup>^2\</sup>mathrm{in}$  general a right handed coordinate system is also called "triad"

#### 4.4.3 How to setup a new TriadStateMat?

A very common problem in OSTK is to calculate a new TriadStateMat. Assume the given vectors  $\vec{p}$ ,  $\vec{p}$ ,  $\vec{q}$ ,  $\vec{q}$ ,  $\vec{q}$ ,  $\vec{q}$  but  $\vec{p}$  is not necessarily perpendicular to  $\vec{q}$ . We want to create a new coordinate system with x-axis parallel to  $\vec{p}$ . For simplicity we assume, that  $\vec{q}$  and  $\vec{p}$  are nearly perpendicular, and the z-axis of the new system shall be nearly in  $\vec{q}$  direction.

This example is adopted from the RSW-coordinate system in sec. 4.6.2:

The system's origin is the center of mass of the satellite. The y-axis is aligned with the satellite's velocity vector. And the x-axis points nearly towards the earth's center. It is only nearly, because the velocity and the vector to Earth's center are only perpendicular for circular orbits.

Note that  $\vec{p}$  and  $\vec{q}$  are usually not normalized. The x-axis is easily defined to

$$\vec{e_x} = \frac{\vec{p}}{p} \tag{4.86}$$

and we also define the unit q vector to

$$\vec{e_q} = \frac{\vec{q}}{q}.$$
(4.87)

The time derivatives of a unit vector can be calculated using the method described in Appendix A (diffunitvec(...) in lutils.cpp ), so we can reference on

$$\vec{e_x}, \ \dot{\vec{e_x}}, \ \vec{\vec{e_x}}, \ \vec{\vec{e_q}}, \ \vec{\vec{e_q}}, \ \vec{\vec{e_q}}, \ \vec{\vec{e_q}}$$

The z-axis is the normalized cross-product of both unit vectors:

$$\vec{e_z} = \frac{\vec{e_x} \times \vec{e_q}}{|\vec{e_x} \times \vec{e_q}|} \tag{4.88}$$

$$\dot{\vec{e}_z} = \frac{\vec{e_x} \times \vec{e_q} + \vec{e_x} \times \vec{e_q}}{|\vec{e_x} \times \vec{e_q} + \vec{e_x} \times \vec{e_q}|}$$
(4.89)

$$\ddot{\vec{e}_z} = \frac{\ddot{\vec{e}_x} \times \vec{e_q} + 2 \cdot \dot{\vec{e}_x} \times \ddot{\vec{e}_q} + \vec{e_x} \times \ddot{\vec{e}_q}}{|\vec{e}_x \times \vec{e}_q + 2 \cdot \dot{\vec{e}_x} \times \ddot{\vec{e}_q} + \vec{e}_x \times \ddot{\vec{e}_q}|}$$
(4.90)

and the y-axis is then

$$\vec{e_y} = -(\vec{e_x} \times \vec{e_z}) \tag{4.91}$$

$$\dot{\vec{e}_y} = -(\vec{e_x} \times \vec{e_z} + \vec{e_x} \times \vec{e_z}) \tag{4.92}$$

$$\ddot{\vec{e}}_y = -(\ddot{\vec{e}}_x \times \vec{e}_z + 2 \cdot \dot{\vec{e}}_x \times \dot{\vec{e}}_z + \vec{e}_x \times \ddot{\vec{e}}_z)$$
(4.93)

This method is implemented in newTriadStateMat(...) in lutils.cpp .

#### 4.4.4 Changing the order of Coordinate Axes

The permutation of the coordinate axes of a TriadStateMat can be done with this function:

Listing 4.4: lutils.cpp

```
change the axes order of a TriadStateMat
          example\ new-form\ string:
                     z x y z
       the 1st character indicates what the new X-axis will be
    (here the old z-axis)
       the 2nd character indicates what the new Y-axis will be
       the 3rd character indicates what the new Z-axis will be
the 4th character denotes which axis of the NEW TriadStateMat
can be changed (only the sign) to obtain a right
handed system! (if it was an anti-cyclical permutation)
        the 3rd character indicates what the new Z-axis
                                                                         will be
          @param[in] @param[in]
                               old Triad State Mat
                                                              old TriadStateMat
                               new form
                                                              string \ describes \ the \ permutation
                               newTriadStateMat
          @param[old]
                                                              new TriadStateMat
void change_TriadStateMatAxes(const ldouble oldTriadStateMat[3][3][3],
                    const char newform [4]
                    ldouble newTriadStateMat[3][3][3]);
```

# 4.5 OSTK class tCOF in COF.h

The COF (Coordinate Frame) class is used to calculate all coordinate transformations in OSTK. Every coordinate system is an instance of this class. Actually the following coordinate systems are defined in OSTK (explained in next section)

```
enum eCOFtype{ GCRF, ITRF, BFS, RSW, NTW, PAS, sGCRF, NoSys }
```

A coordinate system transformation is the related to an old basis and a new basis. Hence all instances are named after them.

#### tCOF GCRF2NTW;

To perform a proper coordinate transformation of a vector  $\vec{r}$  the basis of the vector must be known (old system). Then we need the knowledge of the origin of the new coordinate system

$$\vec{p}, \quad \dot{\vec{p}}, \quad \ddot{\vec{p}}$$

as well as information about the new basis vectors, which can be summarized in the rotation matrix, the angular velocity and acceleration

$$\hat{R}(\vec{e}_x, \vec{e}_y, \vec{e}_z), \quad \vec{\omega}, \quad \dot{\vec{\omega}}$$

or in a  ${\tt TriadStateMat}$  or in quaternions

$$q, \dot{q}, \ddot{q}$$

One of this variable set has to be passed to the instance, like,

```
GCRF2ITRF.set(OriginState, TriadStateMat, Time);
GCRF2ITRF.set(OriginState, QuaternionState, Time);
GCRF2ITRF.set(OriginState, Rotmat, Angular Vel, Angular Acc, Time);
```

Then it is possible to convert the types tVec, tStateVec, tTensor with functions OLD2NEW(...) or NEW2OLD(...), e.g.:

```
tVec accel;
accel.set( GCRF, acceleration, satid, time, ACCVEC);
GCRF2ITRF.OLD2NEW( accel, accel);
```

would convert the vector from GCRF to ITRF frame. A warning will appear, if the coordinate systems or timestamps of the vector and of the COF instance does not match.

#### 4.5.1 Validation

The implementation of the coordinate system formulas can be validated by integrating the equations of motions of a single point mass in an inertial (fixed) and in a rotating frame (with consideration of pseudo-forces) for a particular timespan. After appropriate coordinate transformation the position vectors have to match.

# 4.6 Coordinate Systems

The definition of an inertial frame or system<sup>3</sup> as introduced in Newtonian mechanics is a nontrivial task. Only an extra-galactic coordinate system with origin at the center of the galaxy approaches Newton's definition. Therefore it is necessary to find coordinate systems, which are sufficiently inertial for a particular application [Vallado and McClain, 2007, p.153]. Probably the best realization of an inertial frame in our solar system is the **International Celestial Reference Frame** ICRF (when relativistic effects can be neglected, otherwise

<sup>&</sup>lt;sup>3</sup>'coordinate frames' and 'coordinate system' are often used interchangeable (like in this document), nevertheless there is a difference. In [Vallado and McClain, 2007, p.153] is a footnote about this two expressions.

the Barycentric Celestial Reference System has to be considered). The origin is the Solar System Barycenter. Far stars measured by VLBI<sup>4</sup> are used to provide the orientation. The International Astronomical Union IAU provides definitions and transformation equations for various coordinate systems.

In OSTK the function updateCoordSys(...) in sim.cpp is responsible to setup all coordinate frames at a particular time.

#### 4.6.1 Earth-Centered Coordinate Systems

#### GCRF: Geocentric Celestial Reference Frame

This coordinate frame is the standard inertial frame of the Earth and sufficiently inertial for satellite's orbiting the Earth. The origin is at Earth's center. In earlier times the x-axis pointed towards the vernal equinox. However, the vernal equinox is not fixed and moves slowly over time due to precision and nutation. If a coordinate system is defined with principal direction towards the vernal equinox, an epoch (time) has to be chosen as reference. Newer realizations of the GCRF provided by the IAU do not use the vernal equinox anymore. Further information can be found in [Vallado and McClain, 2007], the IAU-2000 resolution and [McCarthy, 2004]. The equations of motion in OSTK are integrated in this frame. Relativistic effects are not considered yet in OSTK, but they have non-negligible effects for geodesy applications. Equations for relativistic corrections can be found in [Torsten Mayer-Gürr, 2006, eq. 3.82] and [McCarthy, 2004, 10.3, eq.1]. The GCRF is the space-fixed frame in OSTK.

#### **ITRF:** International Terrestrial Reference Frame

This frame origins at the geocenter and is the Earth-fixed frame in OSTK. In general this frame is connected to the Earth's topographic shape, measured by various stations on Earth (and influenced e.g. by tectonic motions). The x axis points towards the intersection between prime meridian and equator and the z-axis is nearly aligned with z-axis of the GCRF frame. The y-axis supplements this right-handed coordinate system. The transformation between GCRF and ITRF frame involves only rotation. Three methods are actually implemented:

- IAU-2000: considering polar motion, nutation, precession and LOD (external file)<sup>5</sup>. The IAU SOFA library is used to calculate the rotation matrices.
- simplified rotation: only a rotation around z-axis with angle  $\alpha$  in rad:

 $\alpha = \text{MJD} \cdot 86400.0 \cdot 7.29211585531 \cdot 10^{-5} + 5.133658456$ 

where MJD is the modified Julian Date. This method is useful due to simplicity of the equation, thus simulated data can be compared easier in collaborations.

• no earth rotation; useful for validation of enery conservation of a spacecraft.

The angular acceleration is set to zero in every case. The ITRF orientation at a particular time is updated using this line:

GCRF2ITRF.set(NullState, RotationMatrix, VecOmega, NullVec, Time);

#### 4.6.2 Satellite-Centered Coordinate Systems

#### sGCRF: shifted GCRF

The shifted GCRF origins at the CoM of the satellite. It provides the frame to integrate the equations of rotation. The quaternion state vector, which describes the attitude of the satellite, refers to this system. The OSTK command to update the COF instance is

sat[n].GCRF2sGCRF.set(sat[n].CoMStateVec, UnityMatrix, NullVec, NullVec, Time);

<sup>&</sup>lt;sup>4</sup>Very Long Baseline Interferometry

#### $\mathbf{RSW}$

This coordinate frame is located at the satellite's center of mass. The x-axis points towards the Earth's center (Nadir direction), the z-axis is the cross-product of x-axis and velocity vector. Hence the velocity vector and the y-axis are nearly aligned (Along-Track axis). Because the velocity vector of the satellite and the Nadir direction are only perpendicular for circular orbits, both vectors can not create an orthonormal system (in general). The realization is done using a TriadStateMat. Let  $\vec{r}$  denote the satellite's CoM position. We define  $\vec{q}$  and time derivatives as follows (and write them in a matrix)<sup>6</sup>:

$$\hat{Q} = [\vec{q}, \dot{\vec{q}}, \ddot{\vec{q}}] = \begin{bmatrix} \leftarrow & \dot{\vec{r}} & \rightarrow \\ \leftarrow & \ddot{\vec{r}} & \rightarrow \\ 0 & 0 & 0 \end{bmatrix}$$
(4.94)

The matrix  $\hat{P}$  is defined as

$$\hat{P} = [\vec{p}, \dot{\vec{p}}, \ddot{\vec{p}}] = \begin{bmatrix} \leftarrow & -\vec{r} & \rightarrow \\ \leftarrow & -\vec{r} & \rightarrow \\ \leftarrow & -\vec{r} & \rightarrow \end{bmatrix}$$
(4.95)

Then the RSW frame is updated using this three lines of code:

```
newTriadStateMat(Q, P, TriadStateMat)
changeTriadStateMatAxes(TriadStateMat,'yxzz',TriadStateMat);
sat[n].GCRF2RSW.set(sat[n].CoMStateVec, TriadStateMat, Time);
```

#### NTW

This frame is similar to the RSW frame. The difference is, that the y-axis is aligned with the velocity (In-Track axis), and the x-axis has nearly nadir direction.

newTriadStateMat(P, Q, TriadStateMat)
sat[n].GCRF2NTW.set(sat[n].CoMStateVec, TriadStateMat, Time);

 $\hat{Q}$  and  $\hat{P}$  are like in the section before.

#### **BFS:** Body Fixed System

The origin of the BFS is the CoM of the satellite. The orientation is given by the satellite's attitude, which is described with the quaternion state vector  $(\underline{q}, \underline{\dot{q}}, \underline{\ddot{q}})$ . The quaternions refer to the sGCRF frame.

```
sGCRF2BFS.set(NullVec, sat[i].BFSQuatStateVec, Time);
```

Usually, the x-vector points to the front of the satellite (velocity direction), y-axis is to the left and z-axis in negative nadir direction.

#### **PAS:** Principal Axes System

The principal axis system is usually fixed w.r.t. the BFS system, except the mass distribution of the satellite changes. Therefore only the timestamp of the COF instance is updated:

#### BFS2PAS.set(Time);

The orientation of the PAS system is set, when the moment of inertia tensor is calculated (cf. sec. 6.4.1 about 3d S/C models).

```
BFS2PAS.set(NullVec, PASdirection, NullVec, NullVec, Time);
```

 $<sup>{}^{6}\</sup>ddot{\vec{q}}$  is set zero, because  $\ddot{\vec{r}}$  is not computed in OSTK. Hence it is not possible to calculate the angular acceleration  $\dot{\vec{\omega}}$  for this coordinate system. The RSW and NTW coordinate system transformations can not transform accelerations properly (the tangential acceleration will be zero

# 4.6.3 Summary

With the introduced methods and formulas OSTK is able to transform vectors and their first and second time derivatives, as well as tensors of second kind like MoI oder gravity gradient tensor to arbitrary coordinate systems. As example a state vector  $\vec{P} = (\vec{p}, \dot{\vec{p}}, \ddot{\vec{p}})$  given in the satellite's principal axes system, can be transformed (physically correct) to the ITRF system with:

BFS2PAS.NEW2OLD( P, P ); sGCRF2BFS.NEW2OLD( P, P ); GCRF2sGCRF.NEW2OLD( P, P ); GCRF2ITRF.OLD2NEW( P, P );

For the future relativistic effects could be considered in the transformations. This would be probably helpful for the simulation of precise LISA orbits.

# Chapter 5

# Earth's Gravity Field

The Earth's gravity field has the most important influence on the trajectory and attitude. In this chapter the models to simulate the Earth's gravity field are introduced and the equations are summarized.

The gravity field is not constant and the contributions can be divided into the following categories:

- static or slow changing gravitational field,
- time-independent tide contribution,
- time-dependent solid Earth tides,
- ocean tides,
- atmospheric tides,
- loading effects due to atmosphere and oceans ,
- residuals or short-time variations.

Before describing the models I first want to emphasize the following two definitions.

# 5.1 Gravity and Gravitation

In the terrestrial case the gravity<sup>1</sup> acceleration  $\vec{g}$  is the sum of gravitational acceleration  $\vec{v}$  and centrifugal acceleration  $\vec{f}$  [Torge, 1989]. Former is due to the presence of mass, latter due to the Earth's rotation around the z-axis (in Earth-fixed frame). Both accelerations are conservative and can be expressed as a gradient of a potential:

$$V = G \int_{V} \frac{\rho(\vec{r}\prime)}{|\vec{r} - \vec{r}\prime|} d^{3}r', \qquad \vec{v} = \vec{\nabla}V$$
(5.1)

As usual in geodesy the acceleration  $\vec{v}$  and potential V are connected without minus sign. This potential can be rewritten as a spherical harmonic (SH) expansion as long as it obeys Laplace's equation (App. B.2, eq. (B.76)):

$$V(r,\varphi,\theta) = \frac{GM}{r} \cdot \sum_{l=0}^{\infty} \sum_{m=0}^{l} \left(\frac{a}{r}\right)^{l} \bar{P}_{l,m}(\cos\left(\theta\right)) \cdot \left[\cos\left(m\varphi\right) \cdot \bar{C}_{l,m} + \sin\left(m\varphi\right) \cdot \bar{S}_{l,m}\right]$$
(5.2)

where  $\bar{C}_{l,m}$  and  $\bar{S}_{l,m}$  are the normalized Stoke's coefficients, which are derived in App. B.2 eq. (B.77) and eq. (B.77). GM is the product of gravitational constant and Earth's mass, whereby a is Earth's mean radius.  $\bar{P}_{l,m}(\cos(\theta))$  are the normalized associated Legendre polynomials of degree l and order m. The normalizations are discussed in App. B.1.4.

<sup>&</sup>lt;sup>1</sup>gravity and gravitation are forces, due to Newton's Second Law  $F = m \cdot a$  they are connected to the acceleration

The centrifugal acceleration is a pseudo-force, which occurs in a rotating frame, and can be described with (cf. centripetal acceleration in sec. 4.3.4)

$$\vec{f} = \vec{\omega} \times (\vec{\omega} \times \vec{r}) = -\vec{\nabla}C = \omega^2 \begin{pmatrix} x \\ y \\ 0 \end{pmatrix}$$
(5.3)

$$F = -\frac{\omega^2}{2}r^2\sin(\theta)^2 = -\frac{\omega^2}{2}\cdot(x^2 + y^2)$$
(5.4)

whereas  $(r, \varphi, \theta)$  are spherical and (x, y, z) cartesian coordinates in an Earth-fixed system. The centrifugal acceleration vanishes at the poles and is maximal at the equator. Earth's angular velocity can be approximated using [Vallado and McClain, 2007, eq. 3-64, p. 27]

$$\omega = 7.292 \ 115 \ 146 \ 706 \ 979 \cdot 10^{-5} \cdot \left(1 - \frac{\text{LOD}}{86400}\right) \frac{\text{rad}}{\text{s}}$$
(5.5)

where LOD is the length of day in seconds, as provided by the IERS. It is the difference between the astronomical measured day length and 86400 SI seconds<sup>2</sup>. The change of the Earth's rotation axis can be neglected, so no other pseudo forces have to be taken into account. The centrifugal potential is not harmonic

$$\Delta C = \vec{\nabla} \cdot \vec{\nabla} C = 2 \cdot \omega^2. \tag{5.6}$$

Thus it can not be expressed in spherical harmonics. A spherical harmonic expansion can only describe the gravitational potential, not the gravity potential! Often these terms are used interchangeable.

# 5.2 Static Gravitational Field

There are several Earth gravitational models in spherical harmonics expansion format available, for example the EGM96 model by NASA GSFC and NIMA [Lemoine, 1998]. Data from several satellite missions and gravimetric ground measurements were used to calculate C and S potential coefficients up to degree and order 360. In the recent years with CHAMP, GRACE and GOCE the number of published gravitational models increased. GFZ and CSR publishes monthly GRACE solutions<sup>3</sup>, and the ITG in Bonn<sup>4</sup> provides even daily coefficient sets. A tabular overview and download possibilities of EGM models since 1966 can be found on the website of the *International Centre for Global Earth Models (ICGEM)*<sup>5</sup>. The defining parameters for a gravitational potential model are:

- GM: gravitational constant times Earth's mass: scaling factor in SH expansion,
- a: mean Earth radius, scaling factor in SH expansion,
- maxdeg: maximum degree & order of expansion,
- Tidesystem: Treatment of permanent tides,
- Normalization: used normalization for  $C_{l,m}$  and  $S_{l,m}$  coefficient (cf. B.1.4),

and the actual data is usually provided in a 6 column textfile with degree l, order m, coefficient  $C_{l,m}$  and  $S_{l,m}$  and standard deviations or error estimations  $\Delta C_{l,m}$  and  $\Delta S_{l,m}$ .

### 5.2.1 Geoid and Ellipsoid

The geoid is an equipotential surface of the Earth's **gravity** field, which bests fit the mean sea level (MSL). The ellipsoid (International Reference Ellipsoid) is a best-fitting oblate ellipsoid of Earth's shape (or MSL) [Lowrie, 2009]. There are two ellipsoids common:

<sup>&</sup>lt;sup>2</sup>Système international d'unités (International System of Units)

<sup>&</sup>lt;sup>3</sup>http://podaac.jpl.nasa.gov/grace/data\_access.html#Level2

<sup>&</sup>lt;sup>4</sup>http://www.igg.uni-bonn.de/apmg/index.php?id=itg-grace2010#content12

<sup>&</sup>lt;sup>5</sup>http://icgem.gfz-potsdam.de/ICGEM/modelstab.html

- WGS84: The US-Department of Defense World Geodetic System was defined in 1984 by the NIMA [NIMA, 2004]. Several modifications have been performed in the past years. The actual defining parameters of this ellipsoid are: semi-major axis a = 6378137.0 m, flattening 1/f = 298.257223563, Earth's  $GM = 3986004.418 \cdot 10^8 \text{ m}^3/\text{s}^2$ , Earth's angular velocity  $\omega = 7292115 \cdot 10^{-11} \text{ rad/sec}$ . This ellipsoid is used for the Global Positioning System (GPS).
- **GRS80:** A slightly different ellipsoid is defined in the Geodetic Reference System 1980: a = 6378137.0 m, flattening 1/f = 298.257222101, Earth's  $GM = 3986005 \cdot 10^8 \text{ m}^3/\text{s}^2$ , Earth's angular velocity  $\omega = 7292115 \cdot 10^{-11} \text{ rad/sec}$ .

With respect to an ellipsoid it is possible to calculate geoid undulations (in meter) of an Earth **gravitational** potential model (one assumes that the centrifugal potential on the geoid and on the ellipsoid are equal, hence they cancel out). The geoid shape in meters can be approximately<sup>6</sup> calculated with [Wahr et al., 1998]

$$N(\varphi,\theta) = V(r,\varphi,\theta) \frac{r \cdot a}{GM}$$
(5.7)

$$=a\sum_{l=0}^{\infty}\sum_{m=0}^{l}\bar{P}_{l,m}(\cos\theta)\cdot\left[\cos\left(m\varphi\right)\cdot\bar{C}_{l,m}+\sin(m\varphi)\cdot\bar{S}_{l,m}\right]$$
(5.8)

and the geoid undulation is the difference between geoid shape and ellipsoid shape. A (small) change  $\Delta$  in the geoid can be approximately expressed in geoid undulation or in Equivalent Water Height (EWH), which is sometimes called Equivalent Water Thickness [Wahr et al., 1998]:

$$N^{\text{EWH}}(\varphi,\theta) = \frac{a \ \rho_{\text{ave}}}{3 \ \rho_{\text{water}}} \sum_{l=0}^{\infty} \sum_{m=0}^{l} \frac{2l+1}{1+k_l} \bar{P}_{l,m}(\cos\theta) \cdot \left[\cos\left(m\varphi\right) \cdot \Delta \bar{C}_{l,m} + \sin(m\varphi) \cdot \Delta \bar{S}_{l,m}\right]$$
(5.9)

$$[N] = meter (5.10)$$

where  $\rho_{\text{ave}} \approx 5517 \text{ kg/m}^3$  is Earth's mean density,  $\rho_{\text{water}} = 1025 \text{ kg/m}^3$  is the water density and  $k_l$  are the frequency independent load Love numbers (named after A. E. H. Love). A spatial geoid undulation plot in units of EWH describes how thick a layer of water on the geoid has to be, to cause this geoid undulation (it takes also the loading deformation into account and is the same as the later introduced ocean tide height). The product  $N^{\text{water}}(\varphi, \theta)$ .  $\rho_{\text{water}}$  has unit of mass per surface, and describes the necessary surface pressure on the geoid, to produce this undulation. For further information on this topic, e.g. mathematical description and methods, refer to: [Martin Losch, 2003, e.g. accurate geoid undulation formula], [Smith, 1998] and [Wahr et al., 1998]. These two simplified examples (loading effect neglected) should illustrate the effect and magnitudes of geoid undulations. To produce a global geoid undulation<sup>7</sup>

- 1 mm, the global Earth surface has to be loaded with  $\approx 1 \, \text{kg/m}^2$
- 1 cm, the global Earth surface has to be loaded with  $\approx 10 \, \text{kg/m}^2$ .

However, the spatial pattern of geoid undulation and EWH can deviate significantly, due to the loading effect. Further it is important to notice, that the Stoke's coefficient C and S are a spectral representation of the geopotential, while geoid (undulations) represent spatial informations. Often both representations provide useful information.

#### 5.2.2 Tide Systems

The geopotential models, the geoid and the ellipsoids are (or can be defined) with respect to a particular permanent tide system, while the geoid undulations are independent from the tide system [Smith, 1998]. This statement is not true, if the ellipsoid is only defined by semi-major axis a and flattening f (and without tidesystem) [Lemoine, 1998, eq. (11.1-1)]. There are three tide systems common:

 $<sup>^{6}\</sup>mathrm{a}$  more sophisticated formula has the latitude dependent normal gravity (ellipsoidal gravity) as prefactor and does not assume a=r

 $<sup>^7\</sup>mathrm{a}$  layer of water is assumed, which directly increases the MSL

- **Tide-free (or non-tidal):** This geoid would exist for a tide-free Earth with all (direct and indirect) effects of the Sun and the Moon removed.
- Mean Tide: This geoid would exist in the presence of the Sun and Moon (or, equivalently, if no permanent tidal effects are removed)
- Zero Tide: This geoid would exist if the permanent direct effects of the Sun and Moon are removed, but the indirect effect component related to the elastic deformation is retained.

The attraction of Sun, Moon and other planets causes a non-zero permanent deformation of the Earth's crust (cf. sec. 5.3.1).



Figure 5.1: Treatment of observations for tidal effects in the geopotential [from McCarthy 2004, IERS Conventions 2003, p. 11]

The figure 5.1 shows the procedure in handling tides in geopotential models. The mean tide system has averaged tides and is close to instantaneous observations (e.g. measured with a gravimeter on the Earth's surface). For the purpose of OSTK, where the direct gravitational acceleration due to a 3rd celestial body can be considered, the zero or the tide-free systems are appropriate (depending on the used solid earth tide and ocean & atmospheric tide model).

Fortunately, the transformation between permanent tide systems is very easy, because only the  $\bar{C}_{2,0}$  coefficient is affected significantly<sup>8</sup>. The formulas for the Earth are [Martin Losch, 2003, eq. (16)+(17)] and [McCarthy, 2004, eq. (11)+(12)]:

$$\bar{C}_{20}^{\text{mean}} - \bar{C}_{20}^{\text{zero}} = 4.4228 \cdot 10^{-8} \cdot (-0.31460) \tag{5.11}$$

$$\bar{C}_{2,0}^{\text{mean}} - \bar{C}_{2,0}^{\text{free}} = (1 + k_{2,0}) \cdot 4.4228 \cdot 10^{-8} \cdot (-0.31460)$$
(5.12)

<sup>&</sup>lt;sup>8</sup> one of the coefficients responsible for the oblateness of the Earth; the  $\bar{C}_{l,0}$  coefficients are responsible for the oblatenes

where  $k_{2,0}$  is the zero frequency Love number. These equations are implemented in the function shown in Listing 5.1.

Listing 5.1: tegmtides::get\_permC20Bias in iersSET.cpp

```
get permanent tide C_2, 0 bias for different tide transformations
   Converts C2,0 coefficients from one tide system into another tide system: just add the return value to the C20-coefficient and you get the new
      tidesystem
  literature: [How to Compute Geoid Undulations...] by Martin Losch, eq. 16 +
     17
  literature: IERS TN 32, p. 67, eq. 11
   @param[in] from
@param[in] to
                            eTideSystem of coefficients
                            eTideSystem you want
   @param[in] k20
                            k20 - Earth \ constant
                   bias for the coefficients C20 (to add)
*
   @return
ldouble tegmtides :: get_permC20Bias(
         eTideSystem from,
         eTideSystem to,
         ldouble k20);
```

#### 5.2.3 Including new Earth Geopotential Models (EGM) in OSTK

Including new EGM models is very easy. One has to copy the data text file with 1, m, C, S and optional standard deviations of C and S columns to the directory externdata/. Then two lines have to be added with the keyword "settings:" for the defining parameters, and "coeff:" for the starting the coefficients data, thus the file looks like:

```
# comment lines
# Defining Parameters:
# Model Name, GM value, '`a''-radius value, tide-system (zero, free,
# mean, none), normalization (4pi/semi)
#
settings: ggm03c 398600.44150E+09 6378136.30 zero 4pi
# 1 degree , m order , C, S, std-dev-C, std-dev-S
coeff:
2 0 -4.841693259705E-04 0.00000000000E+00 4.68460E-11 0.00000E+00
2 1 -2.189810040712E-10 1.467451636117E-09 7.75160E-12 7.81670E-12
2 2 2.439349093502E-06 -1.400284857733E-06 7.80670E-12 7.80760E-12
3 0 9.572141520460E-07 0.0000000000E+00 9.83230E-12 0.00000E+00
3 1 2.030465664514E-06 2.482058936058E-07 4.49800E-12 4.50570E-12
3 2 9.047902456370E-07 -6.189883025458E-07 6.75280E-12 6.75280E-12
....
```

Finally the configuration file has to be adopted to load the new model with this line:

```
# read EGM coefficients (parameter: maximum degree & file)
read egm = 85 externdata/egm96.dat
```

# 5.2.4 OSTK class tSHmodels in SHmodels.h

To simplify handling with spherical harmonic expansions of gravitational and geomagnetic potential fields, the class tSHmodels was implemented. An instance of this class can store the defining parameters (GM, radius a, tidesystem,...) as well as the (gravitational) Stokes' coefficients or (geomagnetic) Gauss' coefficients, as shown in figure 5.2. The coefficients have to use the geodesy  $4\pi$ -normalization (App. B.1.4) to be compatible with the associated Legendre Polynomials. Eventually a re-normalization has to be performed during loading procedure (the Gauss' coefficients are usually Schmidt semi-normalized). Furthermore this class provides methods to calculate spherical and cartesian first and second order derivatives of the potential. To obtain these, two methods can be used:

• calculating spherical derivatives and converting them to cartesian, as described in App. B.3 and in App. B.4

• direct calculation in cartesian coordinates with a method proposed in App. B.5

For these methods the associated Legendre polynomials and possibly Petrovaskaja Coefficients have to be calculated and stored. However, this class basically wraps low-level functions. Every Earth gravitational model, that is loaded in OSTK, is stored in an instance of this class.



Figure 5.2: Overview of the C/C++ class to handle spherical harmonic expansions

## 5.2.5 Practical Considerations

I recommend to use only one GM value and one Mean Earth radius a as scaling factor and only one tidesystem for the EGM models, otherwise difficulties may arise, while comparing or summing C and S coefficients from different sources. In OSTK reference parameters are defined in **constants.h**:

EARTH\_GM = EARTH\_EGM96\_GM = 
$$398600.44180 \cdot 10^9 \text{ m}^3/\text{s}^2$$
 (5.13)

$$EARTH_A = EARTH_EGM96_A = 6378137.00 m$$
 (5.14)

When C and S coefficients are loaded (using methods in filetemplates.cpp or in AOD1B.cpp), they are always re-scaled using these equations:

$$C_{l,m}(\text{new}) = \frac{GM}{\text{EARTH}_{GM}} \left(\frac{a}{\text{EARTH}_{A}}\right)^{l} C_{l,m}(\text{old})$$
(5.15)

$$S_{l,m}(\text{new}) = \frac{GM}{\text{EARTH}_{GM}} \left(\frac{a}{\text{EARTH}_{A}}\right)^{l} S_{l,m}(\text{old})$$
(5.16)

Hence, all Stokes' coefficients have the same scaling factor GM and a. The disadvantage is, that in general the monopole coefficient  $C_{0,0}(new)^{9}$  is only nearly 1. OSTK calculates the Earth's monopole acceleration using the potential

$$\mathbf{V}(r) = \frac{\mathbf{EARTH}_{\mathbf{G}}\mathbf{M}}{r}$$

For very accurate simulations it is recommended to adapt the reference GM and a values to the actual used geopotential model. The permanent tide system of Earth's static geopotential models is automatically converted to the zero tide system using the methods previously described.

# 5.3 General Tide Theory

In this section the basics concepts of tidal theory are introduced, which are necessary to understand the equations of the tidal models.

#### 5.3.1 Tide Generating Potential TGP

Let us assume two coordinate systems: the ICRF frame, which originates at the solar system barycenter and is a nearly inertial frame (neglecting relativistic effects), and the pseudo inertial GCRF frame with origin at the geocenter. For simplicity, we consider the axes of both systems are fixed and have the same orientation. A second celestial body with position  $\vec{P}$ , like the Moon or Sun, will cause an acceleration of the GCRF origin towards  $\vec{P}$ :

$$\ddot{\vec{O}}_{|ICRF}(GCRF) = \frac{GM_p}{|\vec{P}_{|GCRF}|^3} \vec{P}_{|GCRF}$$
(5.17)

For the acceleration at position  $\vec{r}$  follows from the general acceleration transformation formula (in sec. (4.3.4))

$$\ddot{\vec{r}}_{|ICRF} = \ddot{\vec{r}}_{|GCRF} + \ddot{\vec{O}}_{|ICRF} \tag{5.18}$$

$$\Leftrightarrow \tag{5.19}$$

$$\ddot{\vec{r}}_{|GCRF} = \ddot{\vec{r}}_{|ICRF} - \ddot{\vec{O}}_{|ICRF} \tag{5.20}$$

$$= \frac{GM_p}{|\vec{P}_{|ICRF} - \vec{r}_{|ICRF}|^3} (\vec{P}_{|ICRF} - \vec{r}_{|ICRF}) - \frac{GM_p}{|\vec{P}_{|GCRF}|^3} \vec{P}_{|GCRF}$$
(5.21)

$$= GM_p \left( \frac{(\vec{P}_{|GCRF} - \vec{r}_{|GCRF})}{|\vec{P}_{|GCRF} - \vec{r}_{|GCRF}|^3} - \frac{\vec{P}_{|GCRF}}{|\vec{P}_{|GCRF}|^3} \right)$$
(5.22)

This acceleration due to another planet in the GCRF frame is also called (direct) tidal acceleration and the direction on Earth's surface is shown in fig. 5.3.1.

The potential of this acceleration is called the **Tide-generating Potential TGP** [Wenzel, 1997, eq.3]<sup>10</sup>:

$$V(\vec{r}) = GM_p \left( \frac{1}{|\vec{P} - \vec{r}|} - \frac{1}{|\vec{P}|} - \frac{|\vec{r}|\cos(\Psi)}{|\vec{P}|^2} \right) \quad \text{with} \quad \Psi = \angle(\vec{P}, \vec{r})$$
(5.23)

 $<sup>^{9}</sup>$  corresponds to the acceleration due to a point mass

 $<sup>^{10}{\</sup>rm the}$  second term will vanish, if we differentiate V w.r.t.  $\vec{r}$ 



Figure 5.3: Tidal acceleration [from Tide-Generating Potential for the Earth, Hans-Georg Wenzel, 1997]

It is harmonic and can be expressed in terms of Legendre-Polynomials [Wenzel, 1997, (eq.4)], [Baur, 2002, (2.3)] or [Vallado and McClain, 2007, p. 573], very similar to the method described in App. B.2,

$$V(r,\theta,\varphi) = \frac{GM_p}{p} \sum_{l=2}^{\infty} \left(\frac{r}{p}\right)^l P_l(\cos(\Psi))$$
(5.24)

where we used spherical coordinates  $p = |\vec{P}|$  and  $\vec{r} = (r, \Theta, \varphi)$ . Furthermore, the potential can be written like a spherical harmonic expansion

$$V(r,\varphi,\theta) = \frac{GM_p}{p} \sum_{l=2}^{\infty} \sum_{m=0}^{l} \left(\frac{r}{p}\right)^l \frac{1}{2l+1} \bar{P}_{l,m}(\sin(\delta_p)) \bar{P}_{l,m}(\sin(\theta)) \cos(m(\varphi-\varphi_p)) \quad (5.25)$$

where  $\delta_p$  is the declination and  $\varphi_p$  the longitude of the celestial body P. The l = 2, m = 0 term has a non-negligible celestial body independent part (under the assumption p = const.). This part is the permanent tide contribution [Baur, 2002, eq. (2.14)]

$$V_{\text{perm},p}(r,\theta,\varphi) = \frac{GM_p \cdot r^2}{4 \cdot p^3} \left(1 - 3\sin^2(\theta)\right)$$
(5.26)

and causes the change of the  $C_{2,0}$  Stokes' coefficient in section 5.2.2.

#### 5.3.2 Spectral Analysis

The potential in eq. (5.25) is time-dependent, because the planetary position  $\delta_p$  and  $\varphi_p$  is time-dependent. A spectral analysis of the TGP shows, that the main contributions in the TGP are caused by discrete frequencies  $\omega_s$  [Wenzel, 1997] or [Torsten Mayer-Gürr, 2006, p. 28]. Hence the TGP can be split up into single tide constituents, the tidal waves s with frequency  $\omega_s$  (cf. Fourier series):

$$V(r,\varphi,\theta) = Re\left[\frac{GM_e}{a^2}\sum_{l=2}^{\infty}\sum_{m=0}^{l}\sum_{s}\left(\frac{a}{r}\right)^{l+1} H_{lms}(\omega_s) Y_{l,m}(\theta,\varphi) e^{i(\omega_s t)}\right]$$
(5.27)

where  $GM_e$  and a are the parameters of the Earth,  $Y_{l,m}$  the Spherical Harmonics, and  $H_{lms}(\omega_s)$  is the amplitude of the tidal wave s. This is the Cartwright-Edden-Tayler expansion [Baur, 2002]. The amplitude describes the vertical height change of the equipotential surface.

The frequencies  $\omega_s$  can be related to the Delaunay variables

• *l*: mean anomaly of the moon

- l': mean anomaly of the sun
- F: mean argument of latitude of moon
- D: mean longitude of the ascending node of the moon

and mean longitudes of the planets  $\Omega_{Moon}$ ,  $\Omega_{Sun}$ , ... which are used in earth's nutation theory, e.g. IAU-2000 in [McCarthy, 2004, chapter 5], or lunar orbit theory. These variables are (usually) given as polynomial expressions of time [Simon et al., 1994] and can also be used to calculate approximated planetary ephemeris. They have the dimension of radian (or degree) and are often referred to as *astronomical fundamental arguments*. The OSTK method in Listing 5.2 can be used to calculate the values for a particular time.

Listing 5.2: emgtides.h

```
calculate fundamental arguments
     delaunay variables l, l', F, D
     and moons mean longitude
     all results in radian and bounded to (-pi, pi]
     - fundamental arguments are used in tide modelling
     theory:
      [Simon et al. 1994: Precession formulae and mean elements of planets ]
[Daubrawa: Bahnstoerungen durch Ozeangezeiten]
    @param[in] JD
@param[out] fundarg[0]
@param[out] fundarg[1]
@param[out] fundarg[2]
@param[out] fundarg[3]
     @param[in]
                                           Julian Date Terrestrial Time
                     JD
                                           mean anomaly of the moon l
mean anomaly of the sun l
                                           mean argument of latitude of the moon
mean longitude of ascending node of the moon D
     @param[out] fundarg[4]
                                           mean longitude of moon Omega_moon
void fundamentalarguments(const ldouble JD_TT, ldouble fundarg[5]);
```

However, for the description of tides another variable set is more convenient, which can be derived directly from the fundamental arguments. These variables are called Doodson elements, named after Arthur Thomas Doodson. This parameter-set consists of six variables:

- $\tau = (\text{GMST} + \pi s)$  is the mean lunar time + 12 hr with a frequency of  $\approx 14.49 \text{ deg/hr}$  or a period of  $\approx 1.035 \text{ d}$ ,
- $s = (F + \Omega_{moon})$  is the mean lunar longitude with a frequency of  $\approx 0.55 \text{ deg/hr}$  or a period of  $\approx 27.322 \text{ d}$ ,
- h = (s D) is Sun's mean longitude with a period of  $\approx 1$  yr,
- p = (s l) is the mean longitude of Moon's mean perigee with a period of  $\approx 8.847$  yr,
- $N' = (-\Omega_{moon})$  is Sun's mean longitude with a period of  $\approx 18.613 \,\mathrm{yr}$ ,
- $p_s = (s D l')$  is Sun's mean perigee longitude with a period of  $\approx 20936$  yr,

where GMST is the Greenwich Mean Sidereal Time. The frequencies and periods refer to the J2000.0 epoch (01.01.2000, noon, Terrestrial Time) and are the first derivatives of the Doodson Elements evaluated at J2000.0 epoch. It will be convenient to write these time-dependent Elements in a vector

$$\vec{D}(t) = (\tau, s, h, p, N', p_s)^{\mathsf{T}}$$
(5.28)

Another vector

$$\vec{K}(s) = (k_1, k_2, k_3, k_4, k_5, k_6)^{\mathsf{T}}$$
(5.29)

consisting of six integer numbers and with dependency on the tidal wave s, allows the calculation of phase and frequency for every tidal wave:

$$\omega_s t = \vec{K}(s) \cdot \vec{D}(t), \tag{5.30}$$

$$\omega_s = \vec{K}(s) \cdot \dot{\vec{D}}(t). \tag{5.31}$$

Listing 5.3: emgtides.h

```
//! calculate Doodson Elements
/*!
* use the fundamental arguments to calculate doodson
* elements
*
* theory:
* theory:
* http://en.wikipedia.org/wiki/Arthur_Thomas_Doodson#Doodson_Numbers
* [Daubrawa: Bahnstoerungen durch Ozeangezeiten] p.13
*
* @param[in] time.JD_TT julian date terrestrial time
* @param[in] time.GMST Greenwich Mean Sidereal Time
* @param[out] DoodsonElements (tau, s, h, p, N', p-s)
*/
void calc_doodsonelements(const ttimepoint_ext time, ldouble DoodsonElements[6]);
```

Because the  $k_i$  can be negative, the tidal constituents are expressed in a Doodson Code, a format like '273.555', where every digit is simply related to the  $k_i$  number:

 $d_1d_2d_3.d_4d_5d_6 = (k_1)(k_2+5)(k_3+5).(k_4+5)(k_5+5)(k_6+5)$ 

The Doodson Code of a tidal wave consists of six positive integers. However, the first Doodson Element  $\tau$  has the highest frequency of about 1 day, hence the first Doodson Code digit  $d_1$ , which is multiplied with  $\tau$ , can be used to classify the tidal waves into [Wenzel, 1997]:

- $d_1 = m = 0$ : long periodic waves with period of 14 day up to 18.6 years which have a maximum at the poles
- $d_1 = m = 1$ : diurnal waves with period of about 24 hr which have a maximum at  $\pm 45^{\circ}$  latitude
- $d_1 = m = 2$ : semidiurnal waves with period of about 12 hr which have a maximum at the equator
- $d_1 = m = 3$ : tertidiurnal waves with period of about 8 hr

where the fact was used that m (in " $m(\varphi - \varphi_p)$ ") of eq. (5.25) has approximately the same effect as  $d_1$ .

#### 5.3.3 Tidal catalogs

In the last century several TGP catalogs consisting of Doodson numbers and amplitudes were derived (according to [Kudryavtsev, 2005])

- Doodson 1921: 381 tidal waves,
- Cartwright & Tayler 1971,
- Cartwright & Edden 1973,
- Büllesfeld 1985,
- Xi 1987 & 1989,
- Tamura 1987 & 1995,
- Hartmann & Wenzel 1995: 12935 tidal waves [Hartmann and Wenzel, 1995],
- Roosebeck 1996,
- Kudryavtsev 2005: ≈ 27000 tidal waves, for a period of 2000 years [Kudryavtsev, 2005].

At least the catalogs of Kudryavtsev and Hartmann & Wenzel consider also the Earth's flattening effect, which corrects the tidal potential to Earth's ellipticity (for further information see [Dahlen, 1993]).

#### 5.3.4 Hartmann & Wenzel Potential Catalog: HW95

This TGP model consists of a catalog of 12935 tidal waves, which are located in the file /externdata/hw95s.dat. Contributions due to Moon, Sun, Mercury, Venus, Mars, Jupiter and Saturn are considered. The model data is read-in using the function loadf\_hw95(...) in filetemplates.cpp . The catalog is stored in the datalyzer tegmtides.hw95. The acceleration due to the TGP at a particular time and position can be calculated using the method tegmtides.calc\_hw95tide\_deriv(...) in hw95.cpp . This model assumes a rigid Earth, which has no elastic tide deformations. Although the effects of the TGP on satellites can be modeled using the direct 3rd body acceleration, this model can be (probably) adpoted to simulate solid Earth tides of an elastic or anelastic earth by introducing Love numbers.

# 5.4 Solid Earth Tides

The Earth reacts to the tide generating potential TGP with deformations. This mass redistribution induces a change of the geopotential. Mathematically this can be described by introducing Love numbers  $k_{l.m.s}$  in eq. (5.27)

$$V(r,\varphi,\theta) = Re\left[\frac{GM_e}{a^2}\sum_{l=2}^{\infty}\sum_{m=0}^{l}\sum_{s}\left(\frac{a}{r}\right)^{l+1} k_{l,m,s} H_{l,m,s}(\omega_s) Y_{l,m}(\theta,\varphi) e^{i(\omega_s t)}\right]$$
(5.32)

An elastic Earth model reacts directly (without time-shift) to the TGP, hence the  $k_{l.m.s}$  are real numbers and no phase-shift will occur in  $e^{i(\omega_s t)}$ . Another difficulty arises through the fact, that the Love numbers are frequency dependent.

The method proposed in section 5.4.2 uses at first frequency independent Love numbers (independent of the tidal wave s)

$$k_{l,m} = k_{l,m,s}$$

and corrects in the next step for the frequency dependency using this Love numbers:

$$\delta k_{l,m,s} = k_{l,m,s} - k_{l,m}$$

#### 5.4.1 Rizos/Stolz method

One simple formula to calculate the solid earth tide acceleration due to Moon or Sun is given in [Rizos and Stolz, 1985] by

$$\ddot{\vec{r}} = \frac{k_{2,0}}{2} \frac{GM_b}{|\vec{R}_b|^3} \frac{a^5}{|\vec{r}|^4} \left( (3 - 15\cos(\Theta)^2) \frac{\vec{r}}{|\vec{r}|} + 6\cos(\Theta) \frac{\vec{R}_b}{|\vec{R}_b|} \right)$$
(5.33)

$$\Theta = \angle(\vec{R}, \vec{R}_b) \tag{5.34}$$

with Earth's Love number  $k_{2,0}$ , *a* the Earth's mean radius and *b* denoting either Moon or Sun. All vectors refer to the GCRF frame and the portions of Sun and Moon are added up. In this model the Sun's and Moon's gravitational potential is equal to the potential of a point mass. It has to be investigated if a permanent Earth tide contribution is included in this equation. For GPS satellites at a height of about 20000 km the magnitude of this acceleration is in the order of  $10^{-9} \frac{\text{m}}{\text{s}^2}$  or about 0.5ma - 1.0m after integration time of two days. However, this method provides no information about the gravity gradient, although it is possible to differentiate this equation to obtain it. The implementation is as provided in Listing 5.4.

#### 5.4.2 IERS Solid Earth Tide Model

This model is proposed in [McCarthy, 2004, ch. 6.1] and describes the calculation of corrections  $\Delta \bar{C}_{lm}$  and  $\Delta \bar{S}_{l,m}$  for the normalized geopotential Stokes' coefficients  $\bar{C}_{lm}$  and  $\bar{S}_{lm}$  up to degree l = 4.

#### Step 1: Frequency Independent Tide Contribution

The frequency independent tide contribution of the TGP can be expressed in complex form (for l = 2 and l = 3)

$$\Delta \bar{C}_{lm} + i\Delta \bar{S}_{lm} = \frac{k_{lm}}{2l+1} \sum_{j=2}^{3} \frac{GM_j}{GM_E} \left(\frac{a}{r_j}\right)^{l+1} \bar{P}_{lm}(\sin(\theta_j)) \ e^{im\varphi_j} \tag{5.35}$$

where j = 2 denotes the Moon and j = 3 denotes the Sun.  $(r_j, \theta_j, \varphi_j)$  is the geocentric position of the celestial body in spherical coordinates.  $GM_E$  and a are Earth's parameters and the  $k_{lm}$  are nominal (frequency independent) Love numbers (given in Table 6.1 in [McCarthy, 2004]). They have an imaginary part, if an anelastic Earth Model is considered. The complex equation can be rewritten as

$$CT := \sum_{j=2}^{3} \frac{GM_j}{GM_E} \left(\frac{a}{r_j}\right)^{l+1} \bar{P}_{lm}(\sin(\theta_j)) \cos\left(m\varphi_j\right)$$
(5.36)

$$ST := \sum_{j=2}^{3} \frac{GM_j}{GM_E} \left(\frac{a}{r_j}\right)^{l+1} \bar{P}_{lm}(\sin(\theta_j)) \sin\left(m\varphi_j\right)$$
(5.37)

$$\Delta \bar{C}_{lm} = \frac{CT \cdot \operatorname{Re}(\mathbf{k}_{lm}) - \operatorname{ST} \cdot \operatorname{Im}(\mathbf{k}_{lm})}{2l+1}$$
(5.38)

$$\Delta \bar{S}_{lm} = \frac{ST \cdot \text{Re}(k_{\text{lm}}) - \text{CT} \cdot \text{Im}(k_{\text{lm}})}{2l+1}.$$
(5.39)

The l=2 tides produces a contribution to the l=4 term, which is taken into account by<sup>11</sup>

$$CT := \sum_{j=2}^{3} \frac{GM_j}{GM_E} \left(\frac{a}{r_j}\right)^3 \bar{P}_{2m}(\sin(\theta_j)) \,\cos\left(m\varphi_j\right)$$
(5.40)

$$ST := \sum_{j=2}^{3} \frac{GM_j}{GM_E} \left(\frac{a}{r_j}\right)^3 \bar{P}_{2m}(\sin(\theta_j)) \sin(m\varphi_j)$$
(5.41)

$$\Delta \bar{C}_{4m} = \frac{CT \cdot Re(k_{2m}^+) - ST \cdot Im(k_{2m}^+)}{5}$$
(5.42)

$$\Delta \bar{S}_{4m} = \frac{ST \cdot Re(k_{2m}^+) - CT \cdot Im(k_{2m}^+)}{5}.$$
 (5.43)

where m = 0, 1, 2 and  $k_{2m}^+$  is given.

<sup>&</sup>lt;sup>11</sup>notice: the degree of the associated Legendre polynomials is 2, although the correction is for l = 4

#### Step 2: Frequency Dependency Correction

In this step corrections due to different tidal waves have to be applied. A catalog of tidal waves for the correction of (l = 2, m = 0), (l = 2, m = 1) and (l = 2, m = 2) coefficients is provided in Table 6.3 b), a), c) respectively. These data can be found in /externdata/iersSET.dat. The file contains 71 tidal waves with the following columns:

- Name of the tide according to Darwin (1888)
- *l.th* degree of correction
- *m.th* order of correction
- Doodson-Code of tide
- $\delta k_r$ , the real part of the correction for the  $k_{lm}$  Love number
- $\delta k_i$ , the imaginary part of the correction for the  $k_{lm}$  Love number
- in-phase (real) amplitude (ip), unit  $10^{-12}$
- out-phase (imaginary) amplitude (op), unit  $10^{-12}$

The data file is loaded into the memory during execution of the configuration file (config.txt)

```
# Read in tidal waves for IERS Solid Earth tide model
read iersset = externdata/ierSET.dat
```

The function loadf\_iersSET(...) in filetemplates.cpp is responsible for loading the data into a datalyzer tegmtides.iersSET, where the Doodson Code is directly converted to the  $\vec{K}$  vector.

The formula for the  $C_{2,0}$  coefficient is <sup>12</sup>

$$\Delta \bar{C}_{2,0} = \sum_{s=(2,0)} [ip \cdot \cos(\omega_s t) + op \cdot \sin(\omega_s t)]$$
(5.44)

where the sum only considers tides with l = 2, m = 0. The  $S_{l,0}$  coefficients are always zero and can not be corrected. For the (l = 2, m = 1) degree and order the formula can be written in real form as:

$$\Delta \bar{C}_{2,1} = \sum_{s=(2,1)} [ip \cdot \sin(\omega_s t) + op \cdot \cos(\omega_s t)], \qquad (5.45)$$

$$\Delta \bar{S}_{2,1} = \sum_{s=(2,1)} [ip \cdot \cos(\omega_s t) - op \cdot \sin(\omega_s t)], \qquad (5.46)$$

and for (l = 2, m = 2):

$$\Delta \bar{C}_{2,2} = \sum_{s=(2,2)} [ip \cdot \cos(\omega_s t) - op \cdot \sin(\omega_s t)], \qquad (5.47)$$

$$\Delta \bar{S}_{2,2} = \sum_{s=(2,2)} [-ip \cdot \sin(\omega_s t) - op \cdot \cos(\omega_s t)].$$
(5.48)

#### Step 3: Solid Earth Pole Tide

This effect is caused by the centrifugal effect of polar motion, and is taken into account with

$$m_1 = x_p - \bar{x}_p \tag{5.49}$$

$$m_2 = y_p - \bar{y}_p \tag{5.50}$$

$$\Delta \bar{C}_{2,1} = -1.333 \cdot 10^{-9} \cdot (m_1 - 0.0115 \ m_2) \tag{5.51}$$

$$\Delta \bar{S}_{2,1} = -1.333 \cdot 10^{-9} \cdot (m_2 + 0.0115 \ m_1) \tag{5.52}$$

where  $(x_p, y_p)$  are actual position of the Earth's pole and  $(\bar{x}_p, \bar{y}_p)$  are mean values.

 $<sup>^{12}</sup>$ The sign change in this formula in comparison to the original formula is due to the fact, that in table 6.3b) the negative amplitudes are given

#### Step 4: Permanent Tide

The previous steps are designed to correct a geopotential model for the effect of solid earth tide deformation, which is in the tide free system (not zero tide). If the geopotential model is in the zero tide system, the  $C_{2,0}$  coefficient should be biased using the method proposed in section. The following line of code provides the bias to change an  $C_{2,0}$  coefficient a the system TDS, defined in sec. 5.2.2, to the tide free system:

$$\Delta \bar{C}_{2.0} = \text{get\_permC20Bias(TDS, Tidefree, } k_{20})$$
(5.53)

#### Summary

The previously described contributions  $\Delta \bar{C}_{lm}$  and  $\Delta \bar{S}_{lm}$  in every step have to be summed up to obtain the final result and this can be stored in an instance of class tSHmodels in SHmodels.h , like implemented in Listing 5.5.

Listing 5.5: egmtides.h

```
calculate Solid Earth Tide defined by [IERS]
           this method calculates the corrections for
          C and S coefficients as defined by
          IERS 2003 Conventions.
           The Coefficients are stored in the SHmodel
                     egmtides::iersSET
     @param[in]
                     time
                     sunSphericalPosIJK
                                                        Sun \ 's \ spherical \ ITRF \ Pos
     @param[in]
                                                        Moon's spherical ITRF Pos
     @param[in]
                     moonSphericalPosIJK
                                                       Earth's Pole Parameters
     @param[in]
                    PoleParam
     @param [ in ]
                     elastic_earth
                                                       true, or false (anelasti
Tide System of EGM model
                                                                              (anelastic)
                    TDS
     @param [ in ]
void tegmtides :: set_IERS_SET (const ttimepoint_ext time,

        const
        ldouble
        sunSphericalPosIJK [3],

        const
        ldouble
        moonSphericalPosIJK [3],

        const
        ldouble
        PoleParam [4],

                                               eTideSystem TDS,
                                       const
                                       const bool elastic_earth)
```

# 5.5 Ocean and Atmospheric Tides

Most of ocean's water is considered in the static part of Earth's gravitational field, but as for example altimetry missions have shown, there are time-dependent variations of ocean's height in the order of 50 cm on free oceans and even higher in shallow water regions. This height variations are effects of the TGP and they cause a geopotential change due to massredistribution. Two effects have to be taken into account:

- the direct geopotential change due to presence or lack of tidal water mass,
- the deformation of Earth's crust due to the presence or lack of tidal water mass

The same effects occur for the atmosphere's mass.

In the past the prediction of ocean tides was very important especially for harbor cities and naval use. In 1775 Laplace derived a set of nonlinear PDEs to describe water flux in oceans for a barotropic case in two dimensions [Daubrawa, 2007], which are called Laplace's tidal equations (LTEs). Based on this George Howard Darwin derived in 1887 a tidal model. He introduced also names for different tidal constituents, which are still in use. [Schwiderski, 1980] published a global ocean tide model based on tidal gauge measurements, which fulfilled the LTEs. Nowadays, data from altimetry, direct tidal gauges and gravitational measurements are used to calculate global ocean tide models. The seperation of ocean and atmosphere tide signals can be difficult when only satellite based gravity observations are used.

#### 5.5.1 Harmonic Analysis

Under the assumption of linearity of the tidal gauge response on the TGP, one can separate the total tidal gauge into single harmonic constituents, the tidal waves s [Baur, 2002][eq. (3.17)]:

$$A(\varphi,\theta,t) = \sum_{s} A_s(\varphi,\theta,t) = \sum_{s} a_s(\varphi,\theta) \cdot \cos(\omega_s t + \chi_s + \delta_s(\varphi,\theta)) \quad , \quad [A] = m \quad (5.54)$$

This equation describes the tidal height at a particular time and position on Earth's sphere.  $a_s(\varphi, \theta)$  is the tidal amplitude, and the cosine term describes the time dependency. The frequency  $\omega_s$  of the tide is equal to the frequency of the solid earth tide, but the ocean tidal wave is shifted in phase  $\delta_s(\varphi, \theta)$ .  $\chi_s$  can be seen as a constant offset and will be defined in the next sec. 5.5.2. However, this equation is very basic, every scalar field on a sphere can be described with this equation. It can be expanded in spherical harmonics (as shown in App. B.7):

$$A(\varphi, \theta, t) = \sum_{l=0}^{\infty} \sum_{m=0}^{l} \sum_{s} \frac{\bar{P}_{lm}(\sin(\theta))}{2l+1} \cdot \\ \cdot \left[\cos(m\varphi)(\sin(\omega_{s}t + \epsilon_{lms}^{+} + \chi_{s}) \ \hat{C}_{lms}^{+} + \sin(\omega_{s}t + \epsilon_{lms}^{-} + \chi_{s}) \hat{C}_{lms}^{-}\right] \\ + \sin(m\varphi)(\cos(\omega_{s}t + \epsilon_{lms}^{+} + \chi_{s}) \ \hat{C}_{lms}^{+} - \cos(\omega_{s}t + \epsilon_{lms}^{-} + \chi_{s}) \hat{C}_{lms}^{-}\right]$$
(5.55)

where the  $\hat{C}^+$  and  $\epsilon_{lms}^+$  are called the prograde wave amplitude and  $\hat{C}^-$  and  $\epsilon_{lms}^-$  the retrograde wave. This notation follows Schwiderski 1983 [Baur, 2002]. The tidal amplitudes or gauges are usually calculated on grids, with for example 1° × 1° resolution. The maximum degree of the spherical harmonic expansion can be estimated with eq. (B.83) to

$$l_{max} = \frac{180^{\circ}}{\alpha} \tag{5.56}$$

The tidal height can be expressed as a geopotential change:

$$\Delta V(r,\varphi,\theta,t) = \frac{GM}{r} \frac{4\pi a^2}{M} \rho_w \sum_{l=0}^{\infty} \sum_{m=0}^{l} \sum_{s} \left(\frac{a}{r}\right)^l (1+k'_{lm}) \frac{\bar{P}_{lm}(\sin(\theta))}{2l+1}$$

$$[\cos(m\varphi)(\sin(\omega_s t + \epsilon^+_{lms} + \chi_s) \ \hat{C}^+_{lms} + \sin(\omega_s t + \epsilon^-_{lms} + \chi_s) \ \hat{C}^-_{lms})$$

$$+ \sin(m\varphi)(\cos(\omega_s t + \epsilon^+_{lms} + \chi_s) \ \hat{C}^+_{lms} - \cos(\omega_s t + \epsilon^-_{lms} + \chi_s) \ \hat{C}^-_{lms})] \quad (5.57)$$

where  $\rho_w$  is the density of water  $(1025 \text{ kg/m}^3)$ , the factor  $(4\pi a^2) \cdot \rho_w \cdot (\sum \sum ...)$  is the product of the Earth's surface area, water density and tide height and thus similar to a "mass of the tidal wave". The factor  $(1 + k'_l)$  accounts for the effect of direct gravitational potential as well as for the loading effect  $(k'_l)$  frequency independent load Love number). The corrections  $\Delta \bar{C}_{lm}$  and  $\Delta \bar{S}_{lm}$  of the normalized  $\bar{C}_{lm}$  and  $\bar{S}_{lm}$  coefficients are then [McCarthy, 2004, eq. (15b)+(15d)]:

$$\Delta \bar{C}_{lm} = F_{lm} \sum_{s} \sin(\omega_s t + \epsilon^+_{lms} + \chi_s) \ \hat{C}^+_{lms} + \sin(\omega_s t + \epsilon^-_{lms} + \chi_s) \ \hat{C}^-_{lms}, \tag{5.58}$$

$$\Delta \bar{S}_{lm} = F_{lm} \sum_{s} \cos(\omega_s t + \epsilon^+_{lms} + \chi_s) \ \hat{C}^+_{lms} - \cos(\omega_s t + \epsilon^-_{lms} + \chi_s) \ \hat{C}^-_{lms}, \tag{5.59}$$

$$F_{lm} = \frac{4\pi a^2 \rho_w}{M} \frac{1 + k'_l}{2l+1} = \frac{4\pi G \rho_w}{g_e M} \frac{1 + k'_l}{2l+1}$$
(5.60)

where G is the gravitational constant and  $g_e$  the mean equatorial gravity. If the coefficients  $\hat{C}_{lms}^{\pm}$  are unnormalized, the prefactor is:

$$F_{lm} = \frac{4\pi G\rho_w}{g_e M} \frac{1+k'_l}{2l+1} \sqrt{\frac{(l+m)!}{(2l+1)(l-m)!(2-\delta_{lm})}}.$$
(5.61)

# 5.5.2 Doodson-Wartburg Phase Correction $\chi_s$

=

As explained, the total ocean tide contribution can be devided up into single tidal waves. The main constituents are listed in table 5.2 and are responsible for about 90% of the ocean tide signal<sup>13</sup>. The phase correction for these main tidal constituents is defined in the following table 5.1. The first true condition (in the table) has to be used

Doodson Code Condition	$\chi_s$	Darwin Name
165.555	$+\frac{\pi}{2}$	$K_1$
055.565	$\pi^{-}$	$O_{m1}$
1xx.xxx	$-\frac{\pi}{2}$	
0xx.xxx	0	
2xx.xxx	0	
4xx.xxx	0	

Table 5.1: Doodson-Wartburg phase correction; x denotes an arbitrary digit

The following OSTK function returns the value for this correction:

Listing 5.6: egmtides.cpp

```
//! returns value of doodson-wartburg-phasecorrection
/*!
* ref: e.g.
* [Daubrawa: Bahnstoerungen durch Ozeangezeiten]
*
* @param[in] k six k parameters (doodson-codes-offset)
* @return offset in [rad]
*/
Idouble tegmtides::doodson_phasecorrection(int k[6]);
```

The more general definition of  $\chi_s$  is related to the sign of the amplitude in the TGP and on the tidal band (longperiod, diurnal, semidiurnal)<sup>14</sup> [McCarthy, 2004][Table 6.4].

period	Darwin Name	Doodson Code	Doodson Elements	Frequency	caused by
long	$O_{m1}$	055.565	N'	0.002206	М
long	$O_{m2}$	055.575	2N'	0.004413	$\mathbf{M}$
long	$S_a$	056.554	$h - p_s$	0.041067	$\mathbf{S}$
long	$S_{Sa}$	057.555	2h	0.082137	$\mathbf{S}$
long	$M_m$	065.455	s-p	0.544375	$\mathbf{M}$
long	$M_{f}$	075.555	2s	1.098033	$\mathbf{M}$
long	$M_{tm}$	085.455	3s-p	1.642048	$\mathbf{M}$
long	$M_{sq}$	093.555	4s-2h	2.113929	$\mathbf{M}$
diurnal	$Q_1$	135.655	$(\tau - s)(s - p)$	13.398661	Μ
diurnal	$O_1$	145.555	$\tau - s$	13.943036	Μ
diurnal	$P_1$	163.555	$\tau + s - 2h$	14.958931	$\mathbf{S}$
diurnal	$K_1$	165.555	$\tau + s$	15.041069	M+S
semidiurnal	$2N_2$	235.755	$2\tau - 2s + 2p$	27.895354	$\mathbf{M}$
semidiurnal	$N_2$	245.655	$2\tau - s + p$	28.439730	$\mathbf{M}$
semidiurnal	$M_2$	255.555	$2\tau$	28.984104	$\mathbf{M}$
semidiurnal	$S_2$	273.555	$2(\tau + s - h)$	30.000000	$\mathbf{M}$
semidiurnal	$K_2$	275.555	$2(\tau + s)$	30.082137	M+S
	$M_4$	455.555	4 au	57.968208	Μ

Table 5.2: Main tidal constituents; Frequency in degree / hour for J2000.0 epoch; M=Moon, S=Sun

#### 5.5.3 Model: FES2004

This model was computed using finite elements to solve the LTEs. The first version (FES94) was a pure hydrodynamical model, newer versions like FES2004 use also data assimilation

<sup>&</sup>lt;sup>13</sup>Torsten Mayer-Gürr: private communication

<sup>&</sup>lt;sup>14</sup>and it is predestine to cause confusion and anger

[Lyard et. al., 2006]. The following tidal waves are considered: M2, S2, K2, N2, 2N2, O1, P1, K1, Q1, Mf, Mtm, Mm, Msqm and M4. The solution is provided on a  $0.125^{\circ} \times 0.125^{\circ}$  grid and is state-of-the-art, as the figure 5.4 shows<sup>15</sup>.

			M2 Difference	s					S2 Difference	S	
Model	mean amp	amp RMS	mean phase	phase RMS	vec RMS	Model	mean amp	amp RMS	mean phase	phase RMS	vec RMS
	cm	cm	degrees	degrees	cm		cm	cm	cm	degrees	cm
SCW80	-0.0355	4.7150	-0.1216	14.7067	9.324	SCW80	0.1351	2.0438	0.1072	17.4946	7.276
CSR3	-0.0758	2.3430	-1.3799	11.7010	5.729	CSR3	0.1036	1.503	-1.0573	19.8562	2.6159
CSR4	-0.0760	2.2330	-1.0949	13.8589	6.733	CSR4	0.0793	1.4639	-0.7248	18.0633	2.5868
ORI96	-0.1194	2.0560	-2.8351	17.9949	6.228	ORI96	0.0346	1.4693	-4.6152	22.5627	4.0159
NAO99	-0.1922	2.8610	-2.9516	11.7955	5.222	NAO99	0.1198	1.4559	-4.4916	22.9656	6.5667
GOT00	-0.2609	2.0460	-1.1239	10.6726	5.867	GOT00	-0.065	1.5433	-2.0676	17.3499	2.3983
FES952	-0.0807	2.4900	-0.3070	16.4062	7.258	FES952	0.3484	2.018	-2.8913	17.7022	2.7601
FES02	0.0344	1.6660	-1.3924	7.5686	3.907	FES02	0.1372	1.3841	-3.102	11.973	2.2087
FES2004	0.0052	1.6310	-1.4294	6.9666	3.8	FES2004	0.1074	1.3571	-2.5927	11.348	2.1344
TPXO6	-0.1735	1.6810	-0.9515	11.3570	5.416	TPXO6	-0.1172	1.3689	-2.2182	14.3029	2.2101
			K1 Differences	S				(	D1 Differences	5	
Model	mean amp	amp RMS	mean phase	phase RMS	vec RMS	Model	mean amp	amp RMS	mean phase	phase RMS	vec RMS
	cm	cm	cm	degrees	cm		cm	cm	cm	degrees	cm
SCW80	-0.6013	1.6807	1.0264	19.8698	3.5384	SCW80	-0.3986	1.6067	2.2815	19.8580	1.8188
CSR3	-0.2946	1.6737	-3.9425	16.0491	2.6336	CSR3	-0.0786	1.6567	-5.0962	25.3865	2.9475
CSR4	-0.2291	1.6011	-3.2593	16.6586	2.3194	CSR4	-0.0830	1.6324	-2.7052	22.0498	2.2845
ORI96	-0.4566	1.9863	0.4734	22.7853	2.8583	ORI96	-0.0262	1.7623	0.1914	23.4517	2.0004
NAO99	-0.4451	2.6677	-2.1754	14.5764	3.0438	NAO99	-0.2910	2.2081	0.2135	20.6251	2.7928
GOT00	-0.1566	1.4396	-3.0230	15.0035	3.4144	GOT00	-0.0834	1.5073	-1.0304	15.5629	1.6893
FES952	-0.2871	1.6599	-4.6677	15.5268	3.5390	FES952	-0.0787	1.7693	-2.5369	19.1204	2.2045
FES02	-0.2247	1.3740	-1.3247	11.9505	1.9096	FES02	-0.0329	1.5519	-0.7191	16.5683	1.6327
FES2004	-0.1619	1.2497	-0.8837	16.9003	1.8348	FES2004	-0.0034	1.5251	-0.5854	17.8026	1.5696
TPXO6	-0.1681	1.3270	-2.2193	8.6096	1.8055	TPXO6	-0.1277	1.5333	-1.2184	11.8327	1.6124

Figure 5.4: Summary table comparing the ten global tidal models to tide gauge data. Mean and Root Mean Squared (RMS) differences are presented for amplitude and phase. source:Applied Modelling and Computation Group, London

## 5.5.4 Model: EOT08a

For this model a thirteen year time series of different altimeter missions was processed to calculate the tidal waves: 2N2, K1, K2, M2, M4, N2, O1, P1, Q1, S2 [Savcenko and Bosch, 2008]. The long-periodic tides (Om1, Om2, Sa, Ssa, Mm, Mf, Mtm, Msq) are adopted from FES2004 model. This model should be more accurate in shallow water regions, especially the new EOT10a version (which is not implemented yet). The FES2004 and EOT08a model are implemented in OSTK.

### 5.5.5 Formats

To my knowledge the following formats for ocean tide models exist :

#### Format: Grids

The data is given on a worldwide grid with a particular resolution, usually in NetCDF format. FES2004 grid data is available through AVISO<sup>16</sup>, the EOT08a model can be downloaded via DGFI ftp<sup>17</sup>. Unfortunately the harmonic ansatz (5.54) is not applicable for long timescales<sup>18</sup>, because the amplitude as well as the phase have long-periodic (18 yr) lunar perturbations. Which is taken into account with the following formula for the EOT08a model <sup>19</sup>:

$$A(\varphi,\theta,t) = \sum_{s} A_{s}(\varphi,\theta,t) = \sum_{s} f(t) \cdot a_{s}(\varphi,\theta) \cdot \cos(\omega_{s}t + u_{s}(t) + \delta_{s}(\varphi,\theta)) \quad , \quad [A] = m$$
(5.62)

For every latitude & longitude grid point a complex amplitude H is provided. The magnitude of H is the amplitude  $a_s(\varphi, \theta)$ , the phase of H is  $\delta_s(\varphi, \theta)$ . The values f(t) and u(t) are the

<sup>&</sup>lt;sup>15</sup>src: http://amcg.ese.ic.ac.uk/index.php?title=Local:Global\_Tidal\_Models

<sup>&</sup>lt;sup>16</sup>http://www.aviso.oceanobs.com/index.php?id=1279

<sup>&</sup>lt;sup>17</sup>ftp://ftp.dgfi.badw.de/pub/EOT08a

<sup>&</sup>lt;sup>18</sup>it can be used when further longperiodic tidal waves are introduced

<sup>&</sup>lt;sup>19</sup>Roman Savcenko DGFI; private communication

nodal factor and nodal angle. The computation of them is lengthy and burdensome and can be found in [Schureman, 1958], [Pugh, 1987] or in the software [Flater, 2010]. OSTK does not support grid formats, because this format is not appropriate to calculate potential derivatives (accelerations).

#### Format: EPOS

This file in fortran-style format is for example used at GFZ. An official documentation of this format is not available (to my knowledge), but the main structure can be deduced<sup>20</sup>. The following parameters are provided after the keyword in this files:

- **OTIDMOD**: maximum degree & order, water density
- **OTIDLDEF**: load  $k'_l$  numbers for every degree
- **ATIDCOEF**: Atmosphere tidal wave; Darwin name, Doodson code, degree l, order m, normalization(-1 =  $4\pi$ -normalized), prograde wave amplitude  $C_{lms}^+$  in [cm], prograde phase  $\epsilon_{lms}^+$  in [deg], retrograde wave amplitude  $C_{lms}^-$  in [cm], retrograde phase  $\epsilon_{lms}^-$  in [deg]
- **OTIDCOEF**: Ocean tidal wave constituent; parameters as in ATIDCOEF

The correction of the geopotential Stokes' coefficients can be computed with eq. (5.58) - (5.60). The normalization with  $F_{lm}$  as well as the calculation of the Doodson-Wartburg phase correction is performed once directly after loading the EPOS file.

#### **OSTK** class: tAOTEPOSmodels(...) in AOTmodels.h

This class is written to store the EPOS data, as well as to perform the calculation of geopotential Stokes' coefficients. An instance of class tSHmodels in SHmodels.h is used to store the C and S coefficients. Hence acceleration and gravity gradient are easily obtained. The vector<tidalwaves> contains informations like Doodson-Code and Doodson-Wartburg phase correction about the used tides in the EPOS file.

 $<sup>^{20}\</sup>mathrm{and}$  was confirmed by private communication with Roman Savcenko, DGFI



Figure 5.5: C/C++ class to handle atmospheric & ocean tide models in EPOS format

#### **Implementation of EPOS-Models**

In the config.txt the following line tells OSTK to load an EPOS file.

# Read Atmosphere-Ocean-Tide (AOT) model in EPOS format; # name, maxdegree, filepath read aotepos = EOTO8aEPOS 60 externdata/EPOS-EOT08.dat

The function loadf\_aot\_epos(...) in filetemplates.cpp reads the data into memory (datalyzers) and initializes the tAOTEPOSmodels instance properly. The model name (first parameter) is used in the mission.txt to setup satellite's with this model! In original EPOS files a few OSTK header lines have to be added (they are self-explanatory).

#### Format: TMG

This format is called TMG (Torsten Mayer Gürr) in OSTK, because he computed and provided these files to me (as well as the EPOS files, for EOT08a and FES2004). It consists of a cos and a sin file per tidal constituent, with C and S coefficients in every file, hence four numbers are provided (per degree, order and tidal wave):  $cos_{lm}c$ ,  $cos_{lm}c$ ,  $sin_{lm}c$ ,  $sin_{lm}c$ . The correction in the geopotential Stokes' coefficients is calculated using this equation:

$$\Delta \bar{C}_{lm} = \sum_{s} \cos(\omega_s t) \, \cos_{lm} c + \sin(\omega_s t) \, \sin_{lm} c, \qquad (5.63)$$

$$\Delta \bar{S}_{lm} = \sum_{s} \cos(\omega_s t) \, \cos_{lm} s + \sin(\omega_s t) \, \sin_{lm} s. \tag{5.64}$$

This representation is equivalent to eq. (B.115) and (B.116). The advantage of this representation is, that no phase correction or  $F_{lm}$  factor has to be applied, hence it is less error-prone to implement. Also the computation is faster due to fewer mathematical operations.

#### OSTK class: class tAOTTMGmodels in AOTmodels.h

This class has the same intention as the previous one, but it is structured different. It uses only the class tSHmodels in SHmodels.h to store the data in a vector<tidalwaves>.



Figure 5.6: C/C++ class to handle atmospheric & ocean tide models in TMG format

#### **Implementation of TMG-Models**

Because this format consists of different files, here the line in the config.txt the contains a directory.

```
# Read Atmosphere-Ocean-Tide (AOT) model using TMG format;
# name, maxdegree, path
read aotepos = EOTO8aTMG 60 externdata/EOT08a.gfc/
```

OSTK needs a file called **tides** in this directory, where a the tidal waves and corresponding sin and cos files are specified. However, again the function <code>loadf\_aot\_tmg(...)</code> in <code>filetemplates.cpp</code> reads the data into memory (datalyzers) and initializes the <code>tAOTTMGmodels</code> instance properly. The model name (first parameter) is here also used in the <code>mission.txt</code> to setup satellite's with this model!

#### 5.5.6 Admittance Theory

In EOT08a and FES2004 only major tidal constituents are provided, which explain about 90% of the tidal signal. It is possible to compute the the minor tidal constituents with Admittance Theory. However, this method is complicated, basic introductions can be found in [Baur, 2002]. It is not being considered to implement this method at present.

#### 5.6Short-Term Variations: AOD1B De-Aliasing Product

To consider residuals, uncertainties and short time mass-variations, for example caused due to hydrology, recently support for the AOD1B data product was implemented in OSTK. The documentation of this data product can be found in [Frank Flechtner, 2007]. Several geo-data sources (like global surface pressure data provided by ECMWF<sup>21</sup>) are evaluated to compute gravitational correction coefficients  $\Delta C_{lm}$  and  $\Delta S_{lm}$  every 6 hours. Daily data files can be downloaded via  $ISDC^{22}$ .

#### 5.6.1OSTK class: class tAOD1B in AOD1B.h

This class has the task to load the AOD1B data from files into memory (during simulation) and to interpolate the  $\Delta C_{lm}$  and  $\Delta S_{lm}$  coefficients. The linearly interpolated data is stored in an instance of the class tSHmodels in SHmodels.h , where it can be differentiated to obtain the acceleration and gravity gradient.

	AOD1B tAOD1B in AOD1B.h
This class ca	an load, stores and interpolates the AOD1B coefficients.
vars:	maxdeg, next-interpolation-timepoint, AOD1B data-type (glo, ocn, atm)
datalyzer:	
	actual AOD1B data set dat1
	next AOD1B data set dat2
	SHmodel for C and S coefficients (result)) tSHmodels CS
methods:	
	load new AOD1B data when neccessary load_data();
	interpolate when neccessary and calculate C and S set_time();
The init	tialisation is mainly done by the command set aod1b = in mission.txt file

Figure 5.7: C/C++ class to handle AOD1B data

#### 5.6.2Configuration

The data files have to be located in externdata/aod1b/ without any modification. The configuration line can be found in the mission.txt:

```
# AOD1B data parameters: time, type, rate, use_customtime, maxdeg
# name, maxdegree, path
set aod1b = UTC 01 01 2006 00 01 00 glo 3600 1 40
```

where

- UTC 01 01 2006 00 01 00: is a UTC date and time string, which has to be present. This date and time can be used, if the data files are not available for the used simulation time.
- type: defines the data type glo (global), ocn (ocean) or atm (atmosphere) as described in [Frank Flechtner, 2007]
- 3600: interpolation rate in seconds. Here every 3600 seconds a linear interpolation will be performed and new  $\Delta C_{lm}$  and  $\Delta S_{lm}$  will be computed.

<sup>&</sup>lt;sup>21</sup>European Centre for Medium-Range Weather Forecasts

<sup>&</sup>lt;sup>22</sup>http://isdc.gfz-potsdam.de

- 1: this one indicates, that the custom time should be used. If this flag is set to zero, the data file of the actual simulation time will be used (or it will be tried, to use it).
- maxdeg: maximum degree of coefficients to consider

# 5.7 OSTK class: class tegmtides in egmtides.h

The class tegmtides in egmtides.h summarizes all models, that are implemented in OSTK to simulate Earth's gravitational field.

EGMT	IDES		
<i>tegmtides</i> in	n egmtides.h		
This class contains various subclasses to simulate well es time-dependent effects like Ocean Tides or h	the effects of the static Earth Gravitational Field as Hydrology (AOD1B).		
static Earth Gravitational Models (EGM) are stored in an abritrary long vector of SHmodels - called SHM.	The HW95 Tidal Potential Catalogue resulting CS coefficients are stored in a SHmodel. The acceleration is computed not in the SHmodel, because HW95 uses slichtly different formulas.		
vector <tshmodels> SHM</tshmodels>	tSHmodel HW95CS		
	tormoder masos		
EPOS Atmospheric & Ocean Tide Models are stored in			
an abritrary long vector of tAOTEPOSmodels:	The HW95 Tidal Potential Catalogue data (the tidal waves and amplitudes) are stored in a datalyzer		
vector <tauimodels> AUIEPUS</tauimodels>	datalyzer HW95		
TMG Atmospheric & Ocean Tide Models are stored in an abritrary long vector of tAOTTMGmodels:	The IERS Solid Earth Tide resulting CS coefficients are stored in a SHmodel.		
vector <taottmgmodels> AOTTMG</taottmgmodels>	tSHmodel iersSETCS		
This SHmodel is store the result, when summing various models to calculate the acceleration at once.	An instance of the class tAOD1B handles loading and interpolation of the AOD1B De-Aliasing Product data.		
This SHmodel is store the result, when summing various models to calculate the acceleration at once.	An instance of the class tAOD1B handles loading and interpolation of the AOD1B De-Aliasing Product data.		
This SHmodel is store the result, when summing various models to calculate the acceleration at once. tSHmodel TotalCS methods:	An instance of the class tAOD1B handles loading and Interpolation of the AOD1B De-Aliasing Product data. tAOD1B AOD1B		
This SHmodel is store the result, when summing various models to calculate the acceleration at once. tSHmodel TotalCS methods:	An instance of the class tAOD1B handles loading and interpolation of the AOD1B De-Aliasing Product data. tAOD1B AOD1B		
This SHmodel is store the result, when summing various models to calculate the acceleration at once.  ISHmodel TotalCS  methods: initialize sub-classes void init();	An instance of the class tAOD1B handles loading and interpolation of the AOD1B De-Aliasing Product data. tAOD1B AOD1B calculate tidal acceleration with HW-95 Model void calc_hw95tide_deriv();		
This SHmodel is store the result, when summing various models to calculate the acceleration at once. <b>tSHmodel TotalCS</b> methods: initialize sub-classes void init(); get model-id from model-name int get_modelno();	An instance of the class tAOD1B handles loading and interpolation of the AOD1B De-Aliasing Product data. tAOD1B AOD1B calculate tidal acceleration with HW-95 Model void calc_hw95tide_deriv(); calculate Solid Earth Tide defined by [IERS] void set_IERS_SET();		
This SHmodel is store the result, when summing various models to calculate the acceleration at once.           tSHmodel TotalCS           methods:           initialize sub-classes           yoid init();           get model.in from model-name           int get_modelno();           get model name from id           yoid get_modelname();	An instance of the class tAOD1B handles loading and interpolation of the AOD1B De-Aliasing Product data. <b>LAOD1B AOD1B</b> calculate tidal acceleration with HW-95 Model void calc_hw95tide_deriv(); calculate Solid Earth Tide defined by [IERS] void set_IERS_SET(); converts C2,0 coefficients from one tide system into another tide system		
This SHmodel is store the result, when summing various models to calculate the acceleration at once. <b>tSHmodel TotalCS</b> methods: initialize sub-classes void init(); get model-id from model-name int get_modelno(); get model name from id void get_modelname(); calculate acceleration and gravity gradient due to	An instance of the class tAOD1B handles loading and interpolation of the AOD1B De-Aliasing Product data. <b>LAOD1B AOD1B</b> calculate tidal acceleration with HW-95 Model void calc_hw95tide_deriv(); calculate Solid Earth Tide defined by [IERS] void set_IERS_SET(); converts C2.0 coefficients from one tide system into another tide system double get_permC20Bias();		
This SHmodel is store the result, when summing various models to calculate the acceleration at once. <b>tSHmodel TotalCS</b> methods: initialize sub-classes void init(); get model-id from model-name int get_modelno(); get model name from id void get_modelname(); calculate acceleration and gravity gradient due to pointmass term	An instance of the class tAOD1B handles loading and interpolation of the AOD1B De-Aliasing Product data. tAOD1B AOD1B calculate tidal acceleration with HW-95 Model void calc_hw95tide_deriv(); calculate Solid Earth Tide defined by [IERS] void set_IERS_SET(); converts C2.0 coefficients from one tide system into another tide system double get_permC20Bias(); write Stoke's coefficients from all used models to files		

Figure 5.8: OSTK class responsible for dealing with Earth gravity models

# 5.8 Summary & Validation

With the shown models it is possible to simulate the Earth's gravity field very accurately. A comparison between real (e.g. GRACE) satellite ephemeris and OSTK ephemeris is outstanding. The Ocean Tide models were validated with a matlab implementation provided by Torsten Mayer-Gürr (ITG Bonn). The following geoid plots show the contribution due to the different models at 20.03.2006 - 14:09:00 UTC. No correction of the center of mass (first three coefficients) was applied.


Figure 5.9: Static Gravitational Field

The most important part on satellite's acceleration is of course due to the static part of the gravitational field, which is shown fig. 5.9. At about  $60^{\circ}$  West and  $40^{\circ}$  South the gravitational signal due to the Andes is visible. Pictured is the difference between geoid shape and ellipsoid shape.



Figure 5.10: Solid Earth Tide

The next plot shows the geoid deformation due to solid Earth tides, which has correct shape (cf. image 5.3.1). An Ellipsoid was not subtracted. The Moon has higher influence (in comparison to the Sun) and is located at  $151.4^{\circ}$  W,  $25.1^{\circ}$  S. At the opposite side of the Earth is the other bulge.





The ocean tides have most of the signal over oceans (as expected). An Ellipsoid was not subtracted.



Figure 5.12: AOD1B geoid plot; the used data file is from 01.01.2006 - 00:01:00 UTC This plot is from the AOD1B De-Aliasing product and accounts for effects like hydrology.

## Chapter 6

## Modelling Satellites

This chapter describes approaches to model satellites. A satellite is in general a body, consisting of material. It has a center of mass (CoM), an expansion and six degrees of freedom (3 translation, 3 attitude or rotation). At first one can consider an idealized satellite consisting only of a point mass with a particular position and velocity. However, a point mass has no expansion and therefore no attitude. Furthermore forces which depend on the satellite's shape and on the surface area or surface properties, like the drag or solar radiation pressure, can not be considered. This can be avoided by assigning static values to the point mass e.g. a surface area or other parameters like a drag-coefficient and surface's reflectivity. Thus these effects can be approximatively considered. It is only approximatively because the drag and solar radiation pressure are not depended on the total surface area of the satellite, but on the cross-section area perpendicular to the air or photon flux, which is time and position depended.

## 6.1 Surfaces

To improve the accuracy of the simulation the introduction of a 3-dimensional satellite model is common. The satellite's shape can be complex, therefore one can try to approximate the (outer) shape by primitive elements like triangles (or quadrangles). In addition one can assign surface parameters for every primitive surface like the reflectivity (diffuse, geometrical) or absorption coefficients. The surface properties for the GRACE satellites can be found in [Srinivas Bettadpur, 2007, p.22].

The 3d-satellite model is defined in a body fixed coordinate system (BFS). Three points  $\vec{a}_i$ ,  $\vec{b}_i$ ,  $\vec{c}_i$  define the triangular primitive surface  $S_i$ . The normal vector of this area is

$$\vec{n}_{i} = \frac{(\vec{b}_{i} - \vec{a}_{i}) \times (\vec{c}_{i} - \vec{a}_{i})}{|(\vec{b}_{i} - \vec{a}_{i}) \times (\vec{c}_{i} - \vec{a}_{i})|}$$
(6.1)

while the magnitude of area is

$$A_{i} = \frac{|(\vec{b}_{i} - \vec{a}_{i}) \times (\vec{c}_{i} - \vec{a}_{i})|}{2}.$$
(6.2)

For later calculation of the torque, the centroid is important, given by

$$\vec{C}_i = \frac{1}{3}(\vec{a}_i + \vec{b}_i + \vec{c}_i).$$
(6.3)

It is useful to define the normal vector to the outside direction of the satellite. The total surface area is simply the sum over all primitive areas

$$A = \sum_{i}^{N} A_i. \tag{6.4}$$

The cross-section area of a primitive element perpendicular to a vector  $\vec{v}$  (with direction towards the spacecraft) is

$$A_{c,i}(v) = |A_i \cdot \vec{n}_i \cdot \vec{v}| = |A_i \cdot \cos(\alpha)| \qquad \alpha = \measuredangle(\vec{v}, \vec{n}_i).$$

$$(6.5)$$

For the computation of the whole cross-section area one has to take into account, that surfaces may be at the back of the satellite (w.r.t. vector  $\vec{v}$ ). For satellites with convex shape, the dot product  $\vec{n}_i \cdot \vec{v}$  is positive for surfaces at the back (because the angle is > 90°). Hence we have to sum only over elements with negative dot product:

$$A_{\rm c}(v) = \sum_{i:(\vec{n}_i \cdot \vec{v} < 0)} |A_i \cdot \vec{n}_i \cdot \vec{v}|$$
(6.6)

## 6.2 Drag coefficient

The drag coefficient  $C_D$  of a satellite is important for the drag calculation and depends on the spacecraft's shape, attitude and surface [Wertz, J. R., 2001]. A mathematical description is given by

$$C_{\rm D} = \gamma \ C_{\rm DS} + \delta \ C_{\rm DD} + (1 - \gamma - \delta) \ C_{\rm DA} \tag{6.7}$$

where  $C_{\rm DS}$  describes the specular reflection,  $C_{\rm DD}$  the diffuse reflection and  $C_{\rm A}$  the absorption. Equations for these coefficients are [Wertz, J. R., 2001, p.70]

$$C_{\rm DS} = \frac{4}{A} \int_{S} \cos^3(\alpha) \, \mathrm{d}S \tag{6.8}$$

$$C_{\rm DD} = 2 + \frac{1}{A} \int_S \cos^2(\alpha) \, \mathrm{d}S \tag{6.9}$$

$$C_{\rm DA} = 2 \tag{6.10}$$

where A is the cross-sectional area, S the front-surface and  $\alpha$  the angle of incidence (0° <  $\alpha < 90^{\circ}$ ). For a flat plate the coefficients are given  $C_{\rm DS} = 4 \cdot \cos^2(\alpha)$  and  $C_{\rm DD} = 2 + \cos(\alpha)$ , hence the drag coefficient of a satellite approximated with primitive triangular surfaces is:

$$C_{\rm D} = \sum_{i:(\vec{n}_i \cdot \vec{v} < 0)} \left[ 4\gamma \cos^2(\alpha) + \delta(2 + \cos(\alpha)) + (1 - \gamma - \delta) \cdot 2 \right], \qquad \alpha = |\measuredangle(\vec{v}, \vec{n}_i)| \quad (6.11)$$

Unfortunately  $\gamma$  and  $\delta$  have to be estimated or determined by experiments. They obey  $\gamma + \delta < 1$ . The "standard value" for  $C_{\rm D}$  is 2 [Fichter, 2004].

## 6.3 Mass Distribution



Figure 6.1: An OSTK primitive volume unit is defined by two triangles (123) and (456) and consists of 8 surface

The knowledge of the mass distribution of a spacecraft is necessary to calculate the MoI tensor, which is essential for proper solving of Euler's Moment equations and for computation of the principal axes. In addition the gravitational acceleration in the satellite is not homogeneous and yields to the gravity gradient torque, which also depends on the mass



Figure 6.2: Internal view of GRACE satellite instruments; Image from CSR webpage



Figure 6.3: left: OSTK spacecraft model with an interior flat plate; **right**: mass distribution of plate and surfaces, approximated with point masses

distribution. A straightforward method is to approximate the continuous mass distribution with a finite number of point masses on a grid with fixed resolution. For a solid satellite one can introduce primitive volume units, for example as shown in fig. 6.1, which have for simplicity a homogeneous mass density. This volume unit is defined by two triangles (123) and (456). Two volume units can be used, to approximate a solid cube, as shown in fig. 6.4 (right). As further example a GRACE satellite was approximated. The interior is shown in fig. 6.2. On the main equipment platform most of the devices are mounted, which was assumed as flat plate (with 300 kg mass). In fig. 6.3 this flat plate and the outer surfaces were approximated with 4588 point masses. In fig. 6.4 (left) is the effect of an additional 300 kg mass in the equipment plate visible.

Although the use of so many points may be to slow for long-term orbit simulations, these method can provide useful informations how mass shifts inside the satellite influence the MoI tensor or the PAS.

## 6.4 Implementation

OSTK can handle a variable number of satellites. A satellite in OSTK is an instance of the class tSatellites, located in satellites.h, which summarizes various properties and subclasses. The satellite for a particular simulation is defined in the mission.txt file. In the following list the members of the tSatellite class are described, as well as the command in the mission.txt to configure them.



Figure 6.4: **left**: misalignment between PAS (short axes) and BFS due to a 300 kg mass (red) in a 550 kg spacecraft; **right**: approximation of a cubic mass distribution

• char name[]: satellite's name (useful for output)

mp name = GRACE-A

• ldouble mass: satellite's mass (in kg)

mp mass = 450.0

• unsigned char CoM\_color[3]: satellite's color (only for GUI output)

mp color = 
$$127 \ 0 \ 0$$

• ldouble char CoM\_radius: satellite's radius (only for GUI output)

mp radius = 2.0

• tStateVec CoMStateVec: CoM state vector (position, velocity and acceleration). The initial values for the position (in meter) and velocity (in meter/seconds) can be defined with

mp posvel = GCRF -4147374.85 572065.28 5343977.61 5959.31 -931.46 4728.11

or

```
mp posvel = ITRF -4147374.85 572065.28 5343977.61 5959.31 -931.46 4728.11
```

where the first argument denotes the coordinate system of the data. Another method is to use Kepler Elements

mp posvel = Kepler a ecc nu incl omega argp

where **a** is the semi-major axis in meter, **ecc** is the eccentricity of the orbit (unitless), **nu** is the true anomaly, **incl** is the inclination of the orbit, **omega** is the Right Ascension of the Ascending Node and **argp** is the Argument of Perigee. The last four arguments have units of degree.

• tkepl\_elem kepl: Kepler Elements of the point mass. They will be computed regularly, if the following line is present:

• vector<tFTmodels> FTmodels: various perturbations forces and torques can be activated. Every activated perturbation is an entry in this vector and can store additional informations about the satellite (like the drag-coefficient).

• PropagatorType Propagator; unsigned int PropagatorId; the satellite's movement can be computed by one of the following methods:

```
RUNGEKUTTAintegrator, MULTISTEPintegrator, KEPLERmethod
```

The Runge Kutta or the Multistep integrator can be activated by one of the following commands:

mp prop = rk METHODNAME

mp prop = multi METHODNAME

The PropagatorId will be assigned properly due to the METHODNAME.

The Kepler propagator is enabled with:

mp prop = kepler

### 6.4.1 3-dimensional Models

The 3 dimensional satellite models were separated into the class tsat3d in sat3d.cpp. The actual data is located in text files where all surfaces, volume units, thrusters or magnetic dipole moments of the spacecraft are defined. The following command loads the 3-d model:

mp3d satfile = sat1.mp3d

The MoI tensor can be defined directly using the matrix elements I<sub>xx</sub>, I<sub>yy</sub>, I<sub>zz</sub>, I<sub>xy</sub>; I<sub>xz</sub>, I<sub>yz</sub>:

mp3d momofinertia = 150 130 15 ; 15 10 30

or can be calculated from the mass distribution of the satellite model:

```
mp3d momofinertia = massdistr
```

In both cases the MoI tensor is diagonalized using the Jacobi method located in jacobi-diag.h. The eigenvectors are the principal axes (PAS) and the eigenvalues the principal moments. Furthermore the initial attitude of the satellite has to be defined. Actually three methods are provided:

mp3d initorient = RSW

the BFS will be aligned with the RSW system introduced in sec. 4.6.2, this means: satellite's front (BFS x axis) points in along track (nearly velocity) direction

mp3d initorient = NTW

as before, but the NTW system is chosen.

#### mp3d initorient = sat SATNAME

the satellite will point towards another satellite. A triadStateMat is used to calculate the orientation.

## Chapter 7

## Forces & Torques

In this chapter the forces (acceleration) and torques acting on the satellite are summarized.

## 7.1 Earth's Gravity Field

#### 7.1.1 Acceleration

As stated in App. B or in chapter 5 the Earth's gravity potential can be written as

$$V(r,\lambda,\theta) = \frac{GM}{r} \cdot \sum_{l=0}^{l_{max}} \left(\frac{a}{r}\right)^l \sum_{m=0}^l \bar{P}_{l,m}(\cos\theta) \cdot \left[\cos\left(m\lambda\right) \cdot \bar{C}_{l,m} + \sin(m\lambda) \cdot \bar{S}_{l,m}\right]$$
(7.1)

and the acceleration is

$$\ddot{\vec{r}} = \vec{\nabla}_{xyz} \cdot V = \begin{pmatrix} V_x \\ V_y \\ V_z \end{pmatrix}.$$

The two methods to obtain this acceleration are described in App. B.3 and App. B.5

## 7.1.2 Gravity Gradient Torque

The gravity gradient  $\hat{G}$  (gradient of the acceleration) is the hessian matrix of the potential:

$$\hat{G} = \begin{bmatrix} \frac{\partial^2 V}{\partial x \partial x} & \frac{\partial^2 V}{\partial x \partial y} & \frac{\partial^2 V}{\partial x \partial z} \\ \frac{\partial^2 V}{\partial y \partial x} & \frac{\partial^2 V}{\partial y \partial y} & \frac{\partial^2 V}{\partial y \partial z} \\ \frac{\partial^2 V}{\partial z \partial x} & \frac{\partial^2 V}{\partial z \partial y} & \frac{\partial^2 V}{\partial z \partial z} \end{bmatrix} = \begin{bmatrix} V_{xx} & V_{xy} & V_{xz} \\ V_{yx} & V_{yy} & V_{yz} \\ V_{zx} & V_{zy} & V_{zz} \end{bmatrix} = \begin{bmatrix} \uparrow & \uparrow & \uparrow \\ \frac{\partial \vec{a}}{\partial x} & \frac{\partial \vec{a}}{\partial y} & \frac{\partial \vec{a}}{\partial z} \\ \downarrow & \downarrow & \downarrow \end{bmatrix} = \hat{\nabla}_{xyz} \cdot V$$
(7.2)

Due to Clairaut's theorem (sometimes also called Schwarz's theorem or Young's theorem) the second derivatives are symmetric, hence the gravity gradient has 3 independent diagonal elements and 3 independent off-diagonal elements.

The gravity gradient torque is caused due to inhomogeneity of the gravitational acceleration over the expansion of the satellite.

Acutally three methods are implemented to calculate the gravity gradient Torque on the Satellite.

#### Monopole Approximation

This method is derived in [Fichter, 2004]. It approximates the gravity gradient torque using Earth's monopole (point mass) acceleration. The formula is

$$\vec{T}_{|sGCRF} = 3\frac{GM}{|\vec{r}|^3} \cdot \vec{e}_{sat} \times (\hat{I}|sGCRF \cdot \vec{e}_{sat}) \qquad \text{with} \qquad \vec{e}_{sat} = \frac{\vec{r}}{r}$$
(7.3)

where  $\vec{r}$  is the position of the satellite in the GCRF frame and  $\hat{I}$  is the Moment of Inertia Tensor as defined in 2.21, but rotated into the sGCRF system. The advantage of this formula is that only the MoI tensor is required. However it neglects higher multipole moments of the geopotential.

## 7.1.3 Torque due to Acceleration of Point Masses

As mentioned in ch. 6 the mass distribution of the space craft can be approximated with point masses. One very slow method is not to use the gravity gradient, but to calculate the gravitational acceleration and torque on every point mass (of the spacecraft) and to sum the torques:

$$\vec{T}_{|sGCRF} = \sum_{i=0}^{N} \vec{r}_{i,|sGCRF} \times m_i \cdot \vec{a}_{i,|GCRF}$$
(7.4)

Here  $\vec{r}_{i,|sGCRF}$  denotes the position of the i-th masspoint in the sGCRF frame and  $\vec{a}_{i,|GCRF}$  is the acceleration at the position of the i-th masspoint. Because the calculation of the acceleration is CPU intensiv, this method is only useful to validate other computational methods.

#### 7.1.4 Torque due to Gravity Gradient and Point Masses

This method is similiar to the previous one, but faster. We can approximate the acceleration at the position of the i-th point mass with a 3-dimensional taylor expansion (at the center of mass) and neglect the quadratic terms.

$$\vec{r}_{i,|GCRF} = \vec{r}_{CoM,|GCRF} + \vec{r}_{i,|sGCRF} \tag{7.5}$$

$$\vec{a}_{i,|GCRF}(\vec{r}_i) = \vec{a}_{i,|GCRF}(\vec{r}_{CoM}) + G(\vec{r}_{CoM}) \cdot \vec{r}_{i,|sGCRF}$$
(7.6)

and insert this result in 7.4 yields to

**N** 7

$$\vec{T}_{|sGCRF} = \sum_{i=0}^{N} \vec{r}_{i,|sGCRF} \times m_i \cdot \left( \vec{a}_{i,|GCRF}(\vec{r}_{CoM}) + \hat{G}(\vec{r}_{CoM}) \cdot \vec{r}_{i,|sGCRF} \right)$$
(7.7)

$$=\underbrace{\left(\sum_{i=0}^{N} m_{i} \ \vec{r}_{i,|sGCRF}\right)}_{=0} \times \vec{a}_{i,|GCRF}(\vec{r}_{CoM}) + \sum_{i=0}^{N} \vec{r}_{i,|sGCRF} \times m_{i} \cdot \hat{G}(\vec{r}_{CoM}) \cdot \vec{r}_{i,|sGCRF}$$
(7.8)

$$=\sum_{i=0}^{N} \vec{r}_{i,|sGCRF} \times m_i \cdot \hat{G}(\vec{r}_{CoM}) \cdot \vec{r}_{i,|sGCRF}$$

$$\tag{7.9}$$

The first sum vanished, because it is proportional to the center of mass, but in the sGCRF frame. And the sGCRF origins at the CoM, hence it is zero.

This method needs only the gravity gradient and the approximated mass distribution.

#### Validation

All three methods were compared. The torque of both later methods matched very well, while the monopole approximation deviated about 1%. Further research can be done, to figure out how this deviation influences the attitude and what dependency exists to the satellite's dimension.

## 7.2 Direct 3rd Body Acceleration

#### 7.2.1 Acceleration

The equation for the direct 3rd celestial body acceleration was derived in 5.22 or can be found in [Vallado and McClain, 2007, eq. 8-34, p. 571]

$$\ddot{\vec{r}}_{\text{Planet}} = GM_P \left( \frac{\vec{P} - \vec{r}}{|\vec{P} - \vec{r}|^3} - \frac{\vec{P}}{|\vec{P}|^3} \right)$$
(7.11)

The torque is negligible due to the long distance to the planets.

## 7.3 Drag

## 7.3.1 Acceleration

The acceleration due to drag can be described with [Fichter, 2004] and [Vallado and McClain, 2007, eq. 8-28]

$$\ddot{\vec{r}}_{\text{Drag}} = -\frac{1}{2} \frac{C_{\text{D}} A_{\text{C}}}{m} \rho_{\text{air}} \left( \vec{v}_r \cdot \vec{v}_r \right)$$
(7.12)

where  $\vec{v}_r$  is the velocity relative to the atmosphere,  $A_{\rm C}$  the cross-section area and  $\rho_{air}$  the air density. The drag coefficient  $C_{\rm D}$  was introduced in sec. 6.2. This acceleration can be calculated for every (primitive) surface.

## 7.3.2 Torque

The drag force acts homogenously on the surface, hence the torque is the cross-product of centroid position and force vector:

$$\vec{T}_{|\rm BFS,Drag} = \vec{C}_{|\rm BFS} \times (m \cdot \ddot{\vec{r}}_{|\rm BFS,Drag})$$
(7.13)

## Chapter 8

# **Conclusion & Outlook**

In this thesis an overview about the extent of OSTK was given. The Earth gravity field models were emphasized. Next to the basic mathematical methods, which are mainly located in the appendix and explain why so many interesting geophysical effects can be expressed with simple coefficients, it was tried to point out sticking points, like the different normalizations, diversity of formats or rare transformation formulas for spherical second order derivatives. The chapter 5 provided a summary of the terminology and of the models, which are necessary to simulate precise satellite orbits. All together a reader should be able to implement these models on his own in a little while.

Furthermore this thesis demonstrates, that OSTK is now capable of modeling the Earth's gravity field with all major contributions. Together with the simulation of attitude, which is now understood as the results in chapter 1 show, and the introduction of 3-dimensional spacecraft models, which were preliminary explained in chapter 6, a new level of accuracy and new research fields can be reached within the next time.

However there are still various things to do: the implementation of the IGRF geomagnetic model is sill outstanding, the solar radiation pressure torque needs to be considered and a validation of the newly introduced torque perturbations has to be done.

In addition the simulation of precise LISA orbits can be a challenge in the future.

# Appendix A Time derivatives of a unit vector

Given is a vector in direction  $\vec{r} = (x(t), y(t), z(t))^T$  and the time derivatives  $\dot{\vec{r}}$  and  $\ddot{\vec{r}}$ . We want to calculate  $\vec{e}, \vec{e}, \text{and } \ddot{\vec{e}}$ , where

$$\vec{e} = \frac{\vec{r}}{r} \tag{A.1}$$

with  $|\vec{r}| = r = \sqrt{x^2(t) + y^2(t) + z^2(t)}$ .

=

## A.1 First time derivative of $|\vec{r}| = r$

$$\frac{d}{dt}r = \frac{1}{\sqrt{x^2(t) + y^2(t) + z^2(t)}} (x(t) \cdot \dot{x}(t) + y(t) \cdot \dot{y}(t) + z(t) \cdot \dot{z}(t)))$$
(A.2)

$$= -\frac{\vec{r}}{r} \cdot \vec{r}$$
(A.3)

$$\vec{e} \cdot \vec{r}$$
 (A.4)

## A.2 First time derivative of $\dot{\vec{e}}$

$$\frac{d}{dt}\vec{e} = \frac{\dot{\vec{r}}}{r} + \vec{r} \cdot \frac{d}{dt}\left(\frac{1}{r}\right) = \frac{\dot{\vec{r}}}{r} - \vec{r} \cdot \left(\frac{\dot{r}}{r^2}\right) = \frac{\dot{\vec{r}}}{r} - \left(\frac{\vec{r}}{r}\right) \cdot \left(\frac{\dot{r}}{r}\right)$$
(A.6)

$$= = \frac{\dot{\vec{r}}}{r} - \vec{e} \cdot \left(\frac{\dot{r}}{r}\right) \tag{A.7}$$

(A.8)

with the definition of a vector  $\vec{s} := \frac{\dot{\vec{r}}}{r}$  we get

$$\dot{\vec{e}} = \frac{d}{dt}\vec{e} = \vec{s} - \vec{e} \cdot \left(\frac{\dot{r}}{r}\right) = \vec{s} - \vec{e} \cdot (\vec{e} \cdot \vec{s}) \tag{A.9}$$

## A.3 Time derivative of $\vec{s}$

$$\dot{\vec{s}} = \frac{\ddot{\vec{r}}}{r} - \frac{\dot{\vec{r}} \cdot \dot{r}}{r^2}$$
(A.10)

$$= \frac{\ddot{\vec{r}}}{r} - \vec{s} \cdot \frac{\dot{r}}{r} \tag{A.11}$$

$$= \frac{\ddot{\vec{r}}}{r} - \vec{s} \cdot (\vec{e} \cdot \vec{s}) \tag{A.12}$$

(A.13)

## A.4 Second time derivative of $\ddot{\vec{e}}$

According to (A.9):

$$\ddot{\vec{e}} = \dot{\vec{s}} - \dot{\vec{e}} \cdot (\vec{e} \cdot \vec{s}) - \vec{e} \cdot \frac{d}{dt} (\vec{e} \cdot \vec{s})$$

$$= \dot{\vec{s}} - \dot{\vec{e}} \cdot (\vec{e} \cdot \vec{s}) - \vec{e} \cdot (\dot{\vec{e}} \cdot \vec{s} + \vec{e} \cdot \dot{\vec{s}})$$
(A.14)
(A.15)

## A.5 OSTK function diffunitvec(...)

This method to calculate the unit vector and it's time derivatives is implemented in diffunitvec(...) in lutils.cpp.

# Appendix B Spherical Multipole Expansion

The motivation for this chapter is given by the general problem to describe vector fields like Earth's gravitational field or an electrostatic field outside the source. Let us assume a volume V, where the field  $\vec{F}(\vec{r})$  is generated. It can be a mass distribution or charge distribution. Outside the volume V is no source, hence the divergence is zero

$$\nabla \cdot \vec{F}(\vec{r}) = 0. \tag{B.1}$$

Furthermore  $\vec{F}(\vec{r})$  shall be conservative, hence it can be expressed by a scalar potential  $\phi(\vec{r})$  with

$$\vec{F}(\vec{r}) = \pm \nabla \phi(\vec{r}). \tag{B.2}$$

The minus sign is usually convention in physics, but it is often neglected in geophysics or geodesy. The Laplace equation is valid for this problem

$$\Delta\phi(\vec{r}) = \nabla^2\phi(\vec{r}) = 0. \tag{B.3}$$

## **B.1** Definitions

### B.1.1 Legendre polynomials of first kind

The Legendre polynomials  $P_l(x)$  are solutions of Legendre's differential equation [Riley, 2006, 18.1, p.577]

$$(1 - x^2) \cdot \frac{d^2y}{dx^2} - 2x\frac{dy}{dx} + l(l+1) = 0$$
(B.4)

where we used the notation  $y = P_l(x)$ . Another representation is known as Rodrigues' formula:

$$P_l(x) = \frac{1}{2^l \cdot l!} \frac{d^l}{dx^l} (x^2 - 1)^l.$$
 (B.5)

The calculation is possible via recurrence relations [Riley, 2006, 18.23-18.26]. Usually the polynomials are normalized to satisfy

$$P_l(1) = 1.$$

One interesting application can be found when examining the inverse distance between two points in 3-dimensions. They can be expressed in spherical coordinates with position vectors  $\vec{r}(r, \theta, \varphi)$  and  $\vec{q}(r', \theta', \varphi')$ . The inverse distance is

$$\frac{1}{|\vec{r} - \vec{q}|} = \frac{1}{\sqrt{r^2 + r'^2 - 2rr'\cos\alpha}}$$
(B.6)

$$= \frac{1}{r} \cdot \frac{1}{\sqrt{1 + (r'/r)^2 - 2(r'/r)\cos\alpha}} \quad \text{with} \quad \alpha = \angle(\vec{r}, \vec{q}). \tag{B.7}$$

We can use a Taylor expansion on the right fraction to obtain this helpful relation [Riley, 2006, 18.22, p.585] and [Montenbruck and Gill, 2000, 3.6, p.56]

$$\frac{1}{|\vec{r} - \vec{q}|} = \frac{1}{r} \sum_{l=0}^{\infty} \left(\frac{r'}{r}\right)^l P_l(\cos(\theta)).$$
(B.8)

This sum converges if r' < r (cf. geometric series). We used the convention in spherical coordinates with co-latitude, however some authors use latitude then  $P_l(\sin(\theta))$  has to be used.

### B.1.2 Associated Legendre polynomials of first kind

The associated Legendre polynomials  $P_l^m(x)$  of degree l and order m are the solutions of the following equation [Riley, 2006, 18.28]:

$$(1-x^2) \cdot \frac{d^2y}{dx^2} - 2x\frac{dy}{dx} + \left(l(l+1) - \frac{m^2}{1-x^2}\right)y = 0$$
(B.9)

with the notation  $y = P_l^m(x)$  and -l < m < l. The degree and order are integer numbers. It is obvious, that if m = 0 the equation simplifies to the Legendre's differential equation and we get

$$P_l^0(x) = P_l(x) \qquad \forall l. \tag{B.10}$$

The relation to the non-associated Legendre polynomials is given by

$$P_l^m(x) = (1 - x^2)^{(m/2)} \cdot \frac{d^m}{dx^m} P_l(x) \qquad m > 0$$
(B.11)

and for negative m we can use [Riley, 2006, 18.33]

$$P_l^{-m}(x) = (-1)^m \cdot \frac{(l-m)!}{(l+m)!} \cdot P_l^m(x) \qquad m > 0$$
(B.12)

The functions are often normalized using a normalization factor  $N_l^m$ :

$$\bar{P}_l^m(\cos\theta) = P_l^m(\cos\theta) \cdot N_l^m \tag{B.13}$$

In addition this notation is commonly used in physics (be careful with the position of l and m):

$$P_{l,m}(x) = (-1)^m P_l^m(x)$$
(B.14)

whereas  $(-1)^m$  is referred as Condon–Shortley phase. However this factor is neglected in geodesy and in this derivation. The following notation will be used instead:

$$P_{l,m}(x) = \sqrt{2 - \delta_{m,0}} \cdot P_l^m(x) \tag{B.15}$$

It will will be shown subsequently, why this factor was introduced. The function  $P_{l,m}$  will be used in real spherical harmonics, while  $P_l^m$  is for complex spherical harmonics.

### **B.1.3** Spherical Harmonics

The complex spherical harmonics (SH)  $Y_{l,m}(\theta, \varphi)$  are eigenfunctions of the Laplace operator, when we neglect the radial portion and are defined as [Riley, 2006, 18.45]

$$Y_l^m(\theta,\varphi) = \bar{P}_l^m(\cos(\theta)) \cdot e^{im\varphi}$$
(B.16)

$$= N_l^m \cdot P_l^m(\cos(\theta)) \cdot e^{im\varphi}.$$
 (B.17)

According to eq. (B.12) and eq. (B.17), we get

$$Y_l^{-m}(\theta,\varphi) = N_l^m \cdot P_l^m(\cos(\theta)) \cdot e^{-im\varphi}$$
(B.18)

$$= (-1)^m \cdot [Y_l^{+m}(\theta, \varphi)]^*$$
 (B.19)

#### **B.1.** DEFINITIONS

The spherical harmonic functions are an orthonormal and complete set of functions on the unit sphere. Hence arbitrary complex functions  $f(\theta, \varphi)$  can be expressed in the basis of the spherical harmonics

$$f(\theta,\varphi) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} c_l^m Y_l^m(\theta,\varphi)$$
(B.20)

with complex coefficients  $c_l^m$  (cf. Fourier transformation).

One important theorem is the spherical harmonic addition theorem, which connects the Legendre polynomials and the SH functions [Riley, 2006, 18.49].

$$P_l(\cos(\alpha)) = \underbrace{\left(\frac{1}{N_l^m}\right)^2 \frac{(l-m)!}{(l+m)!}}_{\kappa_l^m} \sum_{m=-l}^l Y_l^m(\theta,\varphi) \cdot [Y_l^m(\theta',\varphi')]^*$$
(B.21)

$$=\kappa_l^m \sum_{m=-l}^l Y_l^m(\theta,\varphi) \cdot [Y_l^m(\theta',\varphi')]^*$$
(B.22)

where  $\alpha$  is the angle between two position vectors  $\vec{r}(r, \theta, \varphi)$  and  $\vec{q}(r', \theta', \varphi')$  given in spherical coordinates and \* denotes the complex-conjugation.

The first orthonormalized complex SH are

$$Y_0^0(\theta,\varphi) = \sqrt{\frac{1}{4\pi}}, \qquad Y_1^0(\theta,\varphi) = \sqrt{\frac{3}{4\pi}}\cos(\theta), \qquad Y_1^{\pm 1}(\theta,\varphi) = \mp \sqrt{\frac{3}{8\pi}}\sin(\theta)\exp\pm i\varphi$$
(B.23)

#### **Real Spherical Harmonics**

If we have only real functions, we can use the real SH functions defined with

$$Y_{l,m}(\theta,\varphi) = \begin{cases} \bar{P}_l^m(\cos\theta)\cos(m\varphi) & m \ge 0\\ \bar{P}_l^{|m|}(\cos\theta)\sin(|m|\varphi) & m < 0 \end{cases}$$
(B.24)

The advantage of definition B.15 and B.24 becomes obvious, when we rearrange the sum in B.22 (note that the lhs of B.22 is real, hence the rhs):

$$\sum_{m=-l}^{l} Y_l^m(\theta,\varphi) \cdot [Y_l^m(\theta',\varphi')]^*$$
(B.25)

$$=\sum_{m=1}^{l} \left( Y_{l}^{m} \cdot [Y'_{l}^{m}]^{*} + Y_{l}^{-m} \cdot [Y'_{l}^{-m}]^{*} \right) + \left( Y_{l}^{0} \cdot [Y'_{l}^{0}]^{*} \right)$$
(B.26)

$$=\sum_{m=1}^{l} (N_l^m)^2 [P_l^m(\cos\theta) P_l^m(\cos\theta') e^{im(\varphi-\varphi')} + P_l^m(\cos\theta) P_l^m(\cos\theta') e^{im(\varphi'-\varphi)}]$$
(B.27)

$$+ (N_l^0)^2 [P_l^0(\cos\theta) P_l^0(\cos\theta')]$$
(B.28)

$$=\sum_{m=1}^{l} [\bar{P}_l^m(\cos\theta)\bar{P}_l^m(\cos\theta') \cdot 2 \cdot \cos\left(m(\varphi-\varphi')\right)] + \cdot [\bar{P}_l^0(\cos\theta)\bar{P}_l^0(\cos\theta')]$$
(B.29)

$$=\sum_{m=0}^{l} (2-\delta_{m,0}) \cdot \bar{P}_l^m \bar{P'}_l^m \cdot \cos\left(m(\varphi-\varphi')\right)$$
(B.30)

The prefactor is caused by the zero order terms. The cosine addition theorem states:

$$\cos\left(m(\varphi - \varphi')\right) = \cos\left(m\varphi\right)\cos\left(m\varphi'\right) + \sin\left(m\varphi\right)\sin\left(m\varphi'\right) \tag{B.32}$$

and finally we can transform  $\bar{P}_l^m$  to  $\bar{P}_{l,m}$  using B.15 and exploit the theorem:

$$\sum_{m=-l}^{l} Y_l^m(\theta,\varphi) \cdot [Y_l^m(\theta',\varphi')]^*$$
(B.33)

$$=\sum_{m=0}^{l} (2-\delta_{m,0}) \cdot \bar{P}_l^m \bar{P'}_l^m \cdot \cos\left(m(\varphi-\varphi')\right)$$
(B.34)

$$=\sum_{m=0}^{l}\bar{P}_{l,m}\bar{P}'_{l,m}\cdot\cos\left(m(\varphi-\varphi')\right)$$
(B.35)

$$=\sum_{m=0}^{l}\bar{P}_{l,m}\bar{P'}_{l,m}\cdot\left[\cos\left(m\varphi\right)\cos\left(m\varphi'\right)+\sin\left(m\varphi\right)\sin\left(m\varphi'\right)\right]$$
(B.36)

$$=\sum_{m=0}^{l} [Y_{l,m}Y_{l,m}' + Y_{l,-m}Y_{l,-m}']$$
(B.37)

We are now able to rewrite the spherical harmonic addition theorem B.22 to the real case [Montenbruck and Gill, 2000, eq. 3.8].

$$P_l(\cos(\alpha)) = \underbrace{\left(\frac{1}{N_l^m}\right)^2 \frac{(l-m)!}{(l+m)!}}_{\kappa_l^m} \sum_{m=-l}^l Y_l^m(\theta,\varphi) \cdot [Y_l^m(\theta',\varphi')]^*$$
(B.38)

$$=\kappa_l^m \sum_{m=-l}^l Y_l^m(\theta,\varphi) \cdot [Y_l^m(\theta',\varphi')]^*$$
(B.39)

$$=\kappa_l^m \sum_{m=0}^l [Y_{l,m} Y'_{l,m} + Y_{l,-m} Y'_{l,-m}]$$
(B.40)

$$=\kappa_l^m(2-\delta_{m,0})\sum_{m=0}^l \bar{P}_{l,m}(\cos\theta)\bar{P}_{l,m}(\cos\theta')\cdot\left[\cos\left(m\varphi\right)\cos\left(m\varphi'\right)+\sin\left(m\varphi\right)\sin\left(m\varphi'\right)\right]$$
(B.41)

$$= \kappa_{lm} \sum_{m=0}^{l} \bar{P}_{l,m}(\cos\theta) \bar{P}_{l,m}(\cos\theta') \cdot \left[\cos\left(m\varphi\right)\cos\left(m\varphi'\right) + \sin\left(m\varphi\right)\sin\left(m\varphi'\right)\right]$$
(B.42)

## B.1.4 Normalization

The diversity of normalizations can cause confusion. The factor  $(2 - \delta_{m,0})$  appears only in real SH representation. The transformations are

$$N_{l,m} = \sqrt{(2 - \delta_{m,0})} \cdot N_l^m, \tag{B.43}$$

$$\kappa_{l,m} = (2 - \delta_{m,0}) \cdot k_l^m, \tag{B.44}$$

$$\bar{P}_{l,m} = N_{l,m} \cdot P_l^m. \tag{B.45}$$

According to [Wieczorek, 2010] the most common normalization factors for the real case are:

#### **Racah or Orthonormalization**

Commonly used in seismology and physics, especially in quantum mechanics, these SH functions can be understood as probabilities.

$$(N_{l,m})^{ON} = \sqrt{(2 - \delta_{m,0}) \frac{2l+1}{4\pi} \frac{(l-m)!}{(l+m)!}}$$
(B.46)

$$(\kappa_{l,m})^{ON} = \frac{4\pi}{2l+1} \tag{B.47}$$

$$\int_{-1}^{+1} \bar{P}_{l,m}(x) \cdot \bar{P}_{l',m'}(x) \, dx = \frac{(2 - \delta_{m,0})}{2\pi} \delta_{l,l'} \tag{B.48}$$

$$\int_{Sphere} Y_{l,m}(\Omega) \cdot Y_{l',m'}(\Omega) \ d\Omega = \delta_{m,m'} \delta_{l,l'} \tag{B.49}$$

### Schmidt Semi-Normalization

Usually this occurs in the field of magnetism (e.g. geomagnetic field):

$$(N_{l,m})^{SSM} = \sqrt{(2 - \delta_{m,0})\frac{(l-m)!}{(l+m)!}}$$
(B.50)

$$(\kappa_{l,m})^{SSM} = 1 \tag{B.51}$$

$$\int_{-1}^{+1} \bar{P}_{l,m}(x) \cdot \bar{P}_{l',m'}(x) \, dx = \frac{2(2-\delta_{m,0})}{2l+1} \delta_{l,l'} \tag{B.52}$$

$$\int_{Sphere} Y_{l,m}(\Omega) \cdot Y_{l',m'}(\Omega) \ d\Omega = \frac{4\pi}{2l+1} \delta_{m,m'} \delta_{l,l'}$$
(B.53)

#### Geodesy $4\pi$ -Normalization

This normalization is used for Earth's gravitational field models or ocean tide models. Often coefficients with this normalization are referred to as "fully normalized".

$$(N_{l,m})^{GED} = \sqrt{(2 - \delta_{m,0})(2l+1)\frac{(l-m)!}{(l+m)!}}$$
(B.54)

$$(\kappa_{l,m})^{GED} = \frac{1}{2l+1} \tag{B.55}$$

$$\int_{-1}^{+1} \bar{P}_{l,m}(x) \cdot \bar{P}_{l',m'}(x) \, dx = 2(2 - \delta_{m,0})\delta_{l,l'} \tag{B.56}$$

$$\int_{Sphere} Y_{l,m}(\Omega) \cdot Y_{l',m'}(\Omega) \ d\Omega = 4\pi \cdot \delta_{m,m'} \delta_{l,l'} \tag{B.57}$$

Unnormalized

$$(N_{l,m})^{UN} = 1 (B.58)$$

$$(\kappa_{l,m})^{ON} = \frac{(l-m)!}{(l+m)!}$$
 (B.59)

$$\int_{-1}^{+1} \bar{P}_{l,m}(x) \cdot \bar{P}_{l',m'}(x) \, dx = \frac{2}{2l+1} \frac{(l+m)!}{(l-m)!} \delta_{l,l'} \tag{B.60}$$

$$\int_{Sphere} Y_{l,m}(\Omega) \cdot Y_{l',m'}(\Omega) \ d\Omega = \frac{4\pi}{(2-\delta_{m,0})} \cdot \frac{(l+m)!}{(l-m)!} \delta_{m,m'} \delta_{l,l'}$$
(B.61)

## **B.2** Gravitational Potential Expansion

Now we can derive the SH representation of Earth's gravitational field. The general gravitational potential of an arbitrary mass distribution  $\rho(\vec{r}\prime)$  in Volume V is given by

$$\phi(\vec{r}) = G \int_{V} \frac{\rho(\vec{r}')}{|\vec{r} - \vec{r}'|} d^{3}r'.$$
(B.62)

with  $\vec{r}$  as the position vector of the evaluation point and  $\vec{r'}$  the integration vector inside the volume.

We can now use eq. B.8 and eq. B.22 to instruct the complex SH functions in this equation:

$$\phi(\vec{r}) = G \int_{V} \frac{\rho(\vec{r}')}{r} \cdot \sum_{l=0}^{\infty} \left(\frac{r'}{r}\right)^{l} P_{l}(\cos(\theta)) d^{3}r'$$
(B.63)

$$=G\int_{V}\frac{\rho(\vec{r}\prime)}{r}\cdot\sum_{l=0}^{\infty}\left(\frac{r'}{r}\right)^{l}\kappa_{l}^{m}\sum_{m=-l}^{l}Y_{l}^{m}(\theta,\varphi)\cdot\left[Y_{l}^{m}(\theta',\varphi')\right]^{*}d^{3}r'$$
(B.64)

$$= \frac{G}{r} \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \left(\frac{r'}{r}\right)^{l} Y_{l}^{m}(\theta,\varphi) \int_{V} \rho(\vec{r'}) \cdot \kappa_{l}^{m} \cdot [Y_{l}^{m}(\theta',\varphi')]^{*} d^{3}r'$$
(B.65)

$$= \frac{G}{r} \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \frac{1}{r^l} Y_l^m(\theta,\varphi) \sqrt{\kappa_l^m} \int_V (r\prime)^l \cdot \rho(\vec{r}\prime) \sqrt{\kappa_l^m} \cdot [Y_l^m(\theta',\varphi')]^* \ d^3r'$$
(B.66)

$$=G\sum_{l=0}^{\infty}\sum_{m=-l}^{l}\frac{1}{r^{l+1}}Y_{l}^{m}(\theta,\varphi)\cdot\sqrt{\kappa_{l}^{m}}\cdot Q_{l}^{m}$$
(B.67)

whereas we introduced the general spherical moments, which can be imaginary:

$$Q_l^m := \int_V (rt)^l \cdot \rho(\vec{r}t) \sqrt{\kappa_l^m} \cdot [Y_l^m(\theta',\varphi')]^* \ d^3r'$$
(B.68)

For l = 0, m = 0 the moment is proportional to the total mass M inside the volume (monopole)

$$Q_0^0 = \int_V 1 \cdot \rho(\vec{r'}) \sqrt{\kappa_0^0} \cdot [Y_0^0(\theta', \varphi')]^* \ d^3r'$$
(B.69)

$$= \int_{V} \rho(\vec{r}') \sqrt{\frac{1}{N_{0}^{0}} \cdot N_{0}^{0} \cdot [P_{0}^{0}(\cos(\theta')]^{*} d^{3}r'}$$
(B.70)

$$= \int_{V} \rho(\vec{r'}) \sqrt{N_0^0} \cdot P_0(\cos(\theta')) \ d^3r' \tag{B.71}$$

$$= \sqrt{N_0^0} \int_V \rho(\vec{r}') \cdot 1 \ d^3 r'$$
(B.72)

$$=\sqrt{N_0^0} \int_V \rho(\vec{r'}) \ d^3r' = \sqrt{N_0^0} M. \tag{B.73}$$

If we calculate the potential using eq. B.67 only for l = 0, m = 0, we get the potential of a point mass with mass M.

We now switch to real SH functions and introduce the Earth radius a and Earth's mass M and obtain the Spherical Harmonic expression for Earth's gravitational field:

$$\phi(\vec{r}) = G \int_{V} \frac{\rho(\vec{r'})}{r} \cdot \sum_{l=0}^{\infty} \left(\frac{r'}{r}\right)^{l} \kappa_{l}^{m} \sum_{m=-l}^{l} \left[Y_{l,m}(\theta,\varphi) \cdot Y_{l,m}(\theta',\varphi') + Y_{l,-m}(\theta,\varphi) \cdot Y_{l,-m}(\theta',\varphi')\right] d^{3}r'$$
(B.74)

$$= \frac{GM}{r} \cdot \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \left(\frac{a}{r}\right)^{l} \bar{P}_{l,m}(\cos\theta) \frac{\kappa_{l}^{m}}{M} \int_{V} \left(\frac{r'}{a}\right)^{l} \rho(\vec{r'}) [\cos(\theta) \cdot Y_{l,m}' + \sin(\varphi) \cdot Y_{l,-m}'] d^{3}r'$$
(B.75)

$$= \frac{GM}{r} \cdot \sum_{l=0}^{\infty} \sum_{m=0}^{l} \left(\frac{a}{r}\right)^{l} \bar{P}_{l,m}(\cos\theta) \cdot \left[\cos\left(m\varphi\right) \cdot \bar{C}_{l,m} + \sin(m\varphi) \cdot \bar{S}_{l,m}\right]$$
(B.76)

,

with normalized Stokes coefficients

$$\bar{C}_{l_m} = \frac{\kappa_l^m}{M} \int_V \left(\frac{r'}{a}\right)^{\iota} \rho(\vec{r'}) \cdot \bar{P}_{l,m}(\cos\theta') \cdot \cos\left(m\varphi'\right) \, d^3r' \tag{B.77}$$

$$\bar{S}_{l,m} = \frac{\kappa_l^m}{M} \int_V \left(\frac{r'}{a}\right)^l \rho(\vec{r'}) \cdot \bar{P}_{l,m}(\cos\theta') \cdot \sin\left(m\varphi'\right) \, d^3r' \tag{B.78}$$

It is obvious, that

$$\bar{S}_{l,m=0} = 0 \quad \forall l \tag{B.79}$$

and as previous the l = 0, m = 0 term is proportional to the total mass (and here unity due to normalization)

$$\bar{C}_{0,0} = 1.$$
 (B.80)

In addition it can be shown, that  $\bar{C}_{1,0}, \bar{C}_{1,1}, \bar{S}_{1,1}$  are proportional to the CoM coordinates x, y, z respectively. The infinite sum can be truncated at an appropriate degree  $l_{max}$ . If we assume the CoM as the origin of our coordinate system, we obtain:

$$\phi(r,\theta,\varphi) = \frac{G \cdot M}{r} \cdot \left[ 1 + \sum_{l=0}^{l_{max}} \sum_{m=2}^{l} \left(\frac{a}{r}\right)^{l} \bar{P}_{l,m}(\cos\theta') \cdot (\cos\left(m\varphi\right) \cdot \bar{C}_{l,m} + \sin(m\varphi) \cdot \bar{S}_{l,m}) \right]$$
(B.81)

The spatial resolution of a spherical harmonic expansion (on the earth's surface) with maximum degree  $l_{max}$  is

$$A = \frac{20000 \,\mathrm{km}}{l_{max}} \tag{B.82}$$

or in degree:

$$\alpha = \frac{180^{\circ}}{l_{max}} \tag{B.83}$$

## B.2.1 Determination of the Geopotential Field

Actually the equations B.77 and B.78 are difficult to calculate, because Earth's mass density is not known to high precision. The geopotential models are usually calculated using observations of the gravity field and/or satellite orbits combined fitting procedures (for example Least-Squares).

## B.3 Derivatives of the Geopotential in Spherical Coordinates

To calculate the acceleration due to the geopotential  $V = \phi$  we need the derivatives of B.76. One straightforward way is to calculate the derivatives in spherical coordinates and transform then to cartesian.

$$\frac{\partial V}{\partial r} = -\frac{GM}{r^2} \cdot \sum_{l=0}^{l_{max}} (l+1) \left(\frac{a}{r}\right)^l \sum_{m=0}^l \bar{P}_{l,m}(\cos\theta) \cdot \left[\cos\left(m\lambda\right) \cdot \bar{C}_{l,m} + \sin(m\lambda) \cdot \bar{S}_{l,m}\right] \quad (B.84)$$

$$\frac{\partial V}{\partial \lambda} = +\frac{GM}{r} \cdot \sum_{l=0}^{l_{max}} \left(\frac{a}{r}\right)^l \sum_{m=0}^{l} \bar{P}_{l,m}(\cos\theta) \cdot m \cdot \left[\cos(m\lambda) \cdot \bar{S}_{l,m} - \sin\left(m\lambda\right) \cdot \bar{C}_{l,m}\right]$$
(B.85)

$$\frac{\partial V}{\partial \theta} = +\frac{GM}{r} \cdot \sum_{l=0}^{l_{max}} \left(\frac{a}{r}\right)^l \sum_{m=0}^l \frac{d\bar{P}_{l,m}(\cos\theta)}{d\theta} \cdot \left[\cos\left(m\lambda\right) \cdot \bar{C}_{l,m} + \sin\left(m\lambda\right) \cdot \bar{S}_{l,m}\right]$$
(B.86)

Second order derivatives are necessary for the gravity gradient:

$$\frac{\partial^2 V}{\partial r \partial r} = +\frac{GM}{r^3} \cdot \sum_{l=0}^{l_{max}} (l+1)(l+2) \left(\frac{a}{r}\right)^l \sum_{m=0}^l \bar{P}_{l,m}(\cos\theta) \cdot \left[\cos\left(m\lambda\right) \cdot \bar{C}_{l,m} + \sin(m\lambda) \cdot \bar{S}_{l,m}\right]$$
(B.87)

$$\frac{\partial^2 V}{\partial \lambda \partial \lambda} = -\frac{GM}{r} \cdot \sum_{l=0}^{l_{max}} \left(\frac{a}{r}\right)^l \sum_{m=0}^l \bar{P}_{l,m}(\cos\theta) \cdot m^2 \cdot \left[\sin(m\lambda) \cdot \bar{S}_{l,m} + \cos\left(m\lambda\right) \cdot \bar{C}_{l,m}\right] \quad (B.88)$$

$$\frac{\partial V}{\partial \theta \partial \theta} = +\frac{GM}{r} \cdot \sum_{l=0}^{l_{max}} \left(\frac{a}{r}\right)^l \sum_{m=0}^l \frac{d^2 \bar{P}_{l,m}(\cos\theta)}{d\theta^2} \cdot \left[\cos\left(m\lambda\right) \cdot \bar{C}_{l,m} + \sin(m\lambda) \cdot \bar{S}_{l,m}\right]$$
(B.89)

$$\frac{\partial^2 V}{\partial r \partial \lambda} = -\frac{GM}{r^2} \cdot \sum_{l=0}^{l_{max}} (l+1) \left(\frac{a}{r}\right)^l \sum_{m=0}^{l} \bar{P}_{l,m}(\cos\theta) \cdot m \cdot \left[\cos(m\lambda) \cdot \bar{S}_{l,m} - \sin(m\lambda) \cdot \bar{C}_{l,m}\right]$$
(B.90)

$$\frac{\partial^2 V}{\partial r \partial \theta} = -\frac{GM}{r^2} \cdot \sum_{l=0}^{l_{max}} (l+1) \left(\frac{a}{r}\right)^l \sum_{m=0}^l \frac{d\bar{P}_{l,m}(\cos\theta)}{d\theta} \cdot \left[\cos\left(m\lambda\right) \cdot \bar{C}_{l,m} + \sin(m\lambda) \cdot \bar{S}_{l,m}\right]$$
(B.91)

$$\frac{\partial^2 V}{\partial \lambda \partial \theta} = +\frac{GM}{r} \cdot \sum_{l=0}^{l_{max}} \left(\frac{a}{r}\right)^l \sum_{m=0}^l \frac{d\bar{P}_{l,m}(\cos\theta)}{d\theta} \cdot m \cdot \left[\cos(m\lambda) \cdot \bar{S}_{l,m} - \sin(m\lambda) \cdot \bar{C}_{l,m}\right]$$
(B.92)

This equations are implemented in the following function: calc\_sh\_deriv(...) in SHutils.cpp

# B.4 Transformation Spherical Derivatives to Cartesian Derivatives

The transformation of spherical derivatives to cartesian is constituted by the nabla operator in spherical coordinates [Riley, 2006, Table 10.3, p.363]

$$\vec{\nabla}_{xyz}V = \frac{\partial V}{\partial r} \cdot \vec{e}_r + \frac{1}{r}\frac{\partial V}{\partial \theta} \cdot \vec{e}_\theta + \frac{1}{r\sin(\theta)}\frac{\partial V}{\partial \lambda} \cdot \vec{e}_\lambda \tag{B.93}$$

It is possible to use a matrix-vector notation, where the prefactors are written in a diagonal matrix  $\hat{B}$  and the spherical unit vectors into a matrix  $\hat{R}$ :

$$\vec{\nabla}_{(r\theta\lambda)}V = \begin{pmatrix} \frac{\partial V}{\partial r} & \frac{\partial V}{\partial \theta} & \frac{\partial V}{\partial \lambda} \end{pmatrix}^{T}$$
(B.94)

$$\hat{R} = \begin{bmatrix} 1 & | & | \\ \vec{e_r} & \vec{e_\theta} & \vec{e_\lambda} \\ \downarrow & \downarrow & \downarrow \end{bmatrix} = \begin{bmatrix} \sin(\theta) \cos(\lambda) & \cos(\theta) \cos(\lambda) & -\sin(\lambda) \\ \sin(\theta) \sin(\lambda) & \cos(\theta) \sin(\lambda) & \cos(\lambda) \\ \cos(\theta) & -\sin(\theta) & 0 \end{bmatrix}$$
(B.95)

$$\hat{B} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{r} & 0 \\ 0 & 0 & \frac{1}{r\sin(\theta)} \end{bmatrix}$$
(B.96)

$$\begin{pmatrix} V_x \\ V_y \\ V_z \end{pmatrix} = \vec{\nabla}_{xyz} \cdot V = \underbrace{\hat{R}\hat{B}}_{\hat{Q}} \cdot \vec{\nabla}_{(r\theta\lambda)} \cdot V = \hat{Q} \cdot \vec{\nabla}_{(r\theta\lambda)} \cdot V = \hat{Q} \cdot \begin{pmatrix} V_r \\ V_\theta \\ V_\lambda \end{pmatrix}$$
(B.97)

We can express this transformation with a single matrix-vector multiplication. The matrix  $\hat{Q}$  and the derivatives w.r.t. spherical coordinates (for later use) are:

$$\hat{Q} = \begin{bmatrix} \sin(\theta)\cos(\lambda) & \cos(\theta)\cos(\lambda)/r & -\sin(\lambda)/(r\sin(\theta)) \\ \sin(\theta)\sin(\lambda) & \cos(\theta)\sin(\lambda)/r & \cos(\lambda)/(r\sin(\theta)) \\ \cos(\theta) & -\sin(\theta)/r & 0 \end{bmatrix}$$
(B.98)

$$\frac{\partial \hat{Q}}{\partial r} = \begin{bmatrix} 0 & -\cos(\theta)\cos(\lambda)/r^2 & +\sin(\lambda)/(r^2\sin(\theta)) \\ 0 & -\cos(\theta)\sin(\lambda)/r^2 & -\cos(\lambda)/(r^2\sin(\theta)) \\ 0 & +\sin(\theta)/r^2 & 0 \end{bmatrix}$$
(B.99)

$$\frac{\partial \hat{Q}}{\partial \theta} = \begin{bmatrix} \cos(\theta)\cos(\lambda) & -\sin(\theta)\cos(\lambda)/r & +\sin(\lambda)\cos(\theta)/(r\sin(\theta)^2) \\ \cos(\theta)\sin(\lambda) & -\sin(\theta)\sin(\lambda)/r & -\cos(\lambda)\cos(\theta)/(r\sin(\theta)^2) \\ -\sin(\theta) & -\cos(\theta)/r & 0 \end{bmatrix}$$
(B.100)

$$\frac{\partial \hat{Q}}{\partial \lambda} = \begin{bmatrix} -\sin(\theta)\sin(\lambda) & -\cos(\theta)\sin(\lambda)/r & -\cos(\lambda)/(r\sin(\theta)) \\ \sin(\theta)\cos(\lambda) & \cos(\theta)\cos(\lambda)/r & -\sin(\lambda)/(r\sin(\theta)) \\ 0 & 0 & 0 \end{bmatrix}$$
(B.101)

The transformation of the gravity gradient is more challenging. Preliminary we try to calculate the derivatives of  $V_x$  using the matrix Q.

$$\begin{pmatrix} V_{xx} \\ V_{yx} \\ V_{zx} \end{pmatrix} = \vec{\nabla}_{xyz} \cdot V_x = \hat{Q} \cdot \vec{\nabla}_{(r\theta\lambda)} V_x$$
(B.102)

$$= \hat{Q} \cdot \begin{pmatrix} \frac{\partial}{\partial r} \\ \frac{\partial}{\partial \theta} \\ \frac{\partial}{\partial \lambda} \end{pmatrix} (\hat{Q} \cdot \vec{\nabla}_{(r\theta\lambda)} V)_x$$
(B.103)

It is convenient to do further calculations with index notation. The indices now denote matrix elements and derivatives are written as  $\frac{\partial}{\partial}$ . Latin and Greek indices go from 1 to 3, but concerning differentiation the Greek indices refer to spherical derivatives while Latin indices stand for cartesian derivatives.

$$\frac{\partial^2 V}{\partial i \ \partial k} = \sum_{\alpha=1}^3 Q_{i\alpha} \frac{\partial}{\partial \alpha} (\hat{Q} \cdot \vec{\nabla}_{(r\theta\lambda)} V)_k \tag{B.104}$$

$$=\sum_{\alpha=1}^{3}\sum_{\beta=1}^{3}Q_{i\alpha}\frac{\partial}{\partial\alpha}(Q_{k\beta}\cdot(\vec{\nabla}_{(r\theta\lambda)}V)_{\beta})$$
(B.105)

$$=\sum_{\alpha=1}^{3}\sum_{\beta=1}^{3}Q_{i\alpha}\frac{\partial}{\partial\alpha}(Q_{k\beta}\cdot\frac{\partial}{\partial\beta}V)$$
(B.106)

now we use the product rule to obtain the end result:

sian

$$\frac{\partial^2 V}{\partial i \ \partial k} = \sum_{\alpha=1}^3 \sum_{\beta=1}^3 Q_{i\alpha} \left( \frac{\partial Q_{k\beta}}{\partial \alpha} \cdot \frac{\partial V}{\partial \beta} + Q_{k\beta} \cdot \frac{\partial^2 V}{\partial \alpha \partial \beta} \right) \tag{B.107}$$

The implementation in the source code of the last equation is very simple:

Listing B.1: gravity gradient: conversion from spherical to carte-

```
for (i=0; i < 3; i++)
for (k = 0; k<=i; k++)
for (a=0; a<3; a++)
for (b=0; b<3;b++){
    Vcart[i][k] += Q[i][a]*(dQ[a][k][b]*R[b]+Q[k][b]*Vsph[a][b]);
}</pre>
```

Unfortunately the conversion from spherical to cartesian derivatives has a singularity at the poles. However the satellite has to be very near to the poles to cause problems, what practically never occurred in the simulations. One can validate the equation (B.107) by calculating directly the cartesian derivatives, as stated in the next section.

## **B.5** Calculating directly Cartesian Derivatives

It is also possible to calculate directly the cartesian coordinates, as shown by [Cunningham, 1970] for unnormalized Stoke's Coefficients. [Petrovskaya and Vershkov, 2010] has extended this method also for  $4\pi$ -normalized coefficients. The latter method is implemented in OSTK. For calculation of the first derivatives one has to calculate 6 coefficients  $C_x, C_y, C_z, S_x, S_y, S_z$  from the Stoke's Coefficient C and S. Then there is a summation formula similar to the spherical harmonic expansion to yield the acceleration. Next to the advantage of calculating directly the cartesian derivatives, also at the poles, this method needs only the associated Legendre polynomials (without derivatives). However, it is not convenient in the case of time dependent Stoke's Coefficients due to performance issues.

## **B.6** Recurrence Relations for $\bar{P}_{l,m}(\cos \theta)$ and Derivatives

There are various recurrence formulas for the associated Legendre polynomials. In [Montenbruck and Gill, 2000] and [Riley, 2006] are formulas for unnormalized ones. In OSTK are Geodesy  $4\pi$  normalized formulas implemented based on [Wenzel, 1985]. The second derivatives are not in [Wenzel, 1985] and were self-derived. The following function is responsible for the evaluation for a particular co-latitude  $\theta$ :

Listing B.2: eval\_Plm() in SHutils.cpp

```
evaluate Associated Legendre polynomials Plm(cos(theta)) and derivatives
    this function calculates & evalutes the Associated Legendre polynomials of
    first kind:
 *
 *
    Plm(cos(theta))
    and the first two derivatives (w.r.t. theta) dPlm and ddPlm
 *
    for all degrees l \ll MAXDEG and orders m \ll MAXDEG
    uses: geodesy 4-pi normalization
 *
    based on Wenzel 1995 - "Hochaufloesende Kugelfunktionsmodelle fuer
 *
    das Gravitationspotential der Erde"
         the result is written in a long-double array with 5 columns and N rows,
         whereas N = (MAXDEG) * (MAXDEG+1)/2 + MAXDEG
         the array starts at [0][0] and ends at [N][4].
         P[MAXDEG][MAXDEG](cos theta) is the last element.
         the columns are:
         0:Plm, 1:dPlm, 2:ddPlm, 3:1 degree, 4:m order
                THETA
                                   co-latitude of the evaluation point
 *
    @param[in]
               MAXDEG
    @param[in]
                                   maximum degree for calculation
    @param[out] mPlm
                                   memory for values (5-col ldouble array)
*/
void eval_Plm(
          ldouble THETA,
          unsigned int MAXDEG,
          ldouble *mPlm);
```

There is a paper by [Holmes and Featherstone, 2002], in which actual methods for calculation even of high degree ( $l_{max} = 2700$ ) associated Legendre polynomials and derivatives is presented. One difficulty is to ensure numerical stability of the polynomials.

# B.7 Spherical Harmonic expansion of a scalar field with two parameters

Assuming the scalar field given in eq. (5.54) on page 57 which has an amplitude A and phase  $\delta$  at every point on a sphere:

$$A(\varphi,\theta,t) = \sum_{s} A_{s}(\varphi,\theta,t) = \sum_{s} a_{s}(\varphi,\theta) \cdot \cos(\omega_{s}t + \chi_{s} + \delta_{s}(\varphi,\theta))$$
(B.108)

or for simplicity only

$$A_s(\varphi, \theta, t) = a_s(\varphi, \theta) \cdot \cos(\omega_s t + \chi_s + \delta_s(\varphi, \theta))$$
(B.109)

we can use cosine addition theorem to rewrite this equation as

$$A(\varphi, \theta, t) = a_s(\varphi, \theta) \left( \cos(\omega_s t + \chi_s) \cos(\delta_s(\varphi, \theta)) - \sin(\omega_s t + \chi_s) \sin(\delta_s(\varphi, \theta)) \right)$$
  
= 
$$\underbrace{a_s(\varphi, \theta) \cos(\delta_s(\varphi, \theta))}_{(B.110)} \cos(\omega_s t + \chi_s) - \underbrace{a_s(\varphi, \theta) \sin(\delta_s(\varphi, \theta))}_{(B.110)} \sin(\omega_s t + \chi_s)$$

Further assuming that the underbraced parts are harmonic, we can expand this terms with real spherical harmonics. Using geodesy  $4\pi$  normalization introduces the factor  $\kappa_{lm} = \frac{1}{2l+1}$  as described in B.1.4:

$$a_{s}(\varphi,\theta)\cos(\delta_{s}(\varphi,\theta)) = \sum_{l=0}^{\infty} \sum_{m=0}^{l} \left(K_{lms}\cos(m\varphi) + L_{lms}\sin(m\varphi)\right) \frac{\bar{P}_{lm}(\sin(\theta))}{2l+1}$$
$$a_{s}(\varphi,\theta)\sin(\delta_{s}(\varphi,\theta)) = \sum_{l=0}^{\infty} \sum_{m=0}^{l} \left(M_{lms}\cos(m\varphi) + N_{lms}\sin(m\varphi)\right) \frac{\bar{P}_{lm}(\sin(\theta))}{2l+1}$$
(B.111)

where K, L, M, N are coefficients. Four coefficients are necessary per degree, order and tidal wave to describe the total scalar field:

$$A_{s}(\varphi,\theta,t) = \sum_{l=0}^{\infty} \sum_{m=0}^{l} \frac{\bar{P}_{lm}(\sin(\theta))}{2l+1} [\left(K_{lms}\cos(m\varphi) + L_{lms}\sin(m\varphi)\right)\cos(\omega_{s}t + \chi_{s}) + \left(M_{lms}\cos(m\varphi) + N_{lms}\sin(m\varphi)\right)\sin(\omega_{s}t + \chi_{s})]$$
(B.112)

Exploitation of addition theorems for  $\cos(\omega_s t + \chi_s)$  and  $\sin(\omega_s t + \chi_s)$  and rearranging yields with  $(cb = \cos(\chi_s), sb = \sin(\chi_s))$ 

$$A_{s}(\varphi,\theta,t) = \sum_{l=0}^{\infty} \sum_{m=0}^{l} \frac{\bar{P}_{lm}(\sin(\theta))}{2l+1}$$

$$[\cos(m\varphi)(\cos(\omega_{s}t) \underbrace{(cb \ K_{lms} + M_{lms} \ sb)}_{C^{+}_{lms} + C^{-}_{lms}} + \sin(m\varphi)(\cos(\omega_{s}t) \underbrace{(cb \ L_{lms} + N_{lms} \ sb)}_{S^{+}_{lms} - S^{-}_{lms}} + \sin(\omega_{s}t) \underbrace{(cb \ N_{lms} - L_{lms} \ sb))}_{C^{+}_{lms} - C^{-}_{lms}} (B.113)$$

$$A_s(\varphi,\theta,t) = \sum_{l=0}^{\infty} \sum_{m=0}^{l} \frac{\bar{P}_{lm}(\sin(\theta))}{2l+1}$$
(B.114)

$$\left[\cos(m\varphi)(\cos(\omega_s t) \ (C^+_{lms} + C^-_{lms}) + \sin(\omega_s t)(S^+_{lms} + S^-_{lms}))\right]$$
(B.115)

$$+\sin(m\varphi)(\cos(\omega_s t) \ (S^+_{lms} - S^-_{lms}) + \sin(\omega_s t)(C^+_{lms} - C^-_{lms}))]$$
(B.116)

Another convention uses two amplitudes and two phases, instead of four amplitudes (coefficients), to describe this field. The transformation can be performed with [McCarthy, 2004, eq. 14]:

$$C_{lms}^{\pm} - iS_{lms}^{\pm} = -i\hat{C}_{lms}^{\pm} e^{i(\epsilon_{lms}^{\pm} + \chi_s)}$$
 (B.117)

where  $\hat{C}_{lms}^{\pm}$  are the new coefficients and  $\epsilon_{lms}^{\pm}$  the two phases. After the use of some (messy) addition theorems, the field can be written as:

$$A_{s}(\varphi,\theta,t) = \sum_{l=0}^{\infty} \sum_{m=0}^{l} \frac{\bar{P}_{lm}(\sin(\theta))}{2l+1} \\ [\cos(m\varphi)(\sin(\omega_{s}t + \epsilon^{+}_{lms} + \chi_{s}) \ \hat{C}^{+}_{lms} + \sin(\omega_{s}t + \epsilon^{-}_{lms} + \chi_{s})\hat{C}^{-}_{lms}) \\ + \sin(m\varphi)(\cos(\omega_{s}t + \epsilon^{+}_{lms} + \chi_{s}) \ \hat{C}^{+}_{lms} - \cos(\omega_{s}t + \epsilon^{-}_{lms} + \chi_{s})\hat{C}^{-}_{lms})]$$
(B.118)

## Bibliography

- [Baur, 2002] Baur, O. (2002). "Ozeangezeitenlösungen aus Bahnstörungen erdnaher Satelliten", Dipl. Thesis.
- [Cunningham, 1970] Cunningham, L. E. (1970). "On the computation of the spherical harmonic terms needed during the numerical integration of the orbital motion of an artificial satellite". Celestial Mechanics and Dynamical Astronomy 2, 207–216.
- [Dahlen, 1993] Dahlen, F. A. (1993). "Effect of the Earth's ellipticity on the lunar tidal potential". Geophysical Journal International 113, No. 1 (1993),250.
- [Daubrawa, 2007] Daubrawa, J. (2007). "Bahnstörungen durch Ozeangezeiten" Dipl. Thesis. Geodätisches Institut der Universität Stuttgart.
- [Fichter, 2004] Fichter, W. (2004). "Verlesungsskriptum: Advanced Spacecraft Navigation and Control". Universität Stuttgart.
- [Flater, 2010] Flater, D. (2010). "Congen and Libcongen Library: libcongen, a C++ library for generating the speeds, equilibrium arguments".
- [Fogiel, 1987] Fogiel, D. M. (1987). "The Essentials of Mechanics I". Research and Education Association, New York.
- [Frank Flechtner, 2007] Frank Flechtner (2007). "AOD1B Product Description Document for Product Releases 01 to 04: GRACE 327-750 (GR-GFZ-AOD-01)".
- [Goetzelmann, 2003] Goetzelmann, M. (2003). "Simulation von Satellitenbahnen unter Berücksichtigung von direkten Gezeiteneffekten und Umsetzung in ein C-Programm". In Studienarbeit; Geodätisches Institut; Universität Stuttgart.
- [Goldstein, 1991] Goldstein, H. (1991). "Klassische Mechanik". 11., korrigierte aufl. edition, Aula-Verl., Wiesbaden.
- [Gould, 2007] Gould, H. (2007). "An introduction to computer simulation methods: Applications to physical systems". 3. ed. edition, Pearson Addison Wesley, San Francisco, Calif., Munich.
- [Gruber and Flechtner, 2007] Gruber, T. and Flechtner, F. (2007). "Vereinfachte Darstellung der GRACE Datenanalyse". In DFG Schwerpunktsprogramm 1257: Massentransporte.
- [Hartmann and Wenzel, 1995] Hartmann and Wenzel (1995). "The HW95 tidal potential catalogue". Geophys. Res. Lett. 22, 3553–3556.
- [Heinzel, 1992] Heinzel, G. (1992). "Beliebig genau Moderne Runge-Kutta-Verfahren zur Lösung von Differentialgleichungen". c't Computer Magazin (GER) 8/92.
- [Holmes and Featherstone, 2002] Holmes, S. A. and Featherstone, W. E. (2002). "A unified approach to the Clenshaw summation and the recursive computation of very high degree and order normalised associated Legendre functions". Journal of Geodesy 76, 279–299.
- [Jazar, 2010] Jazar, R. N. (2010). "Theory of Applied Robotics: Kinematics, Dynamics, and Control (2nd Edition)".

- [Kudryavtsev, 2005] Kudryavtsev, S. (2005). "Advanced Harmonic Development of the Earth Tide Generating Potential". In A Window on the Future of Geodesy, (Sansò, F., ed.), vol. 128, of International Association of Geodesy Symposia pp. 465–470. Springer Berlin Heidelberg.
- [Lemoine, 1998] Lemoine, F. G. (1998). "The Development of the joint NASA GSFC and the National Imagery and Mapping Agency (NIMA) Geopotential Model EGM96", vol. 1998-206861, of NASA/TP. National Aeronautics and Space Administration Goddard Space Flight Center and Available from NASA Center for AeroSpace Information and National Technical Information Service [distributor], Greenbelt Md., Hanover MD, Springfield VA.
- [Lowrie, 2009] Lowrie, W. (2009). "Fundamentals of geophysics: Includes bibliographical references and index". 2. ed., completely rev. and updated, repr. edition, Cambridge Univ. Press, Cambridge.
- [Lyard et. al., 2006] Lyard et. al. (2006). "Modelling the global ocean tides: modern insights from FES2004". Ocean Dynamics 56, 394–415.
- [Martin Losch, 2003] Martin Losch (2003). "How to compute Geoid Undulations (Geoid Height Relative to a Given Reference Ellipsoid) from Spherical Harmonics Coefficients for Satellite Altimetry Applications".
- [McCarthy, 2004] McCarthy, D. D. (2004). IERS conventions: (2003), vol. 32, of IERS technical note. Verl. des Bundesamtes für Kartographie und Geodäsie, Frankfurt am Main.
- [Montenbruck and Gill, 2000] Montenbruck, O. and Gill, E. (2000). "Satellite orbits: Models, methods, and applications". Springer, Berlin.
- [NIMA, 2004] NIMA (2004). "Technical Report TR8350.2: DoD World Geodetic System 1984: Department of Defense World Geodetic System 1984, Its Definition and Relationships With Local Geodetic Systems".
- [Petrovskaya and Vershkov, 2010] Petrovskaya, M. and Vershkov, A. (2010). "Construction of spherical harmonic series for the potential derivatives of arbitrary orders in the geocentric Earth-fixed reference frame". Journal of Geodesy 84, 165–178.
- [Picone et al., 2002] Picone et al. (2002). "NRLMSISE-00 empirical model of the atmosphere: Statistical comparisons and scientific issues". J. Geophys. Res. 107, 1468.
- [Pugh, 1987] Pugh, D. T. (1987). "Tides, surges and mean sea-level: [a handbook for Engineers and Scientists]". Wiley, Chichester.
- [Riley, 2006] Riley, K. F. (2006). "Mathematical methods for physics and engineering". 3. ed., reprinted. edition, Cambridge Univ. Press, Cambridge.
- [Rizos and Stolz, 1985] Rizos, C. and Stolz, A. (1985). "Force Modelling for GPS Satellite Orbits". In Proc. of First Int. sym. on Precise Position. with GPS Rockville, Maryland, pp.87-99.
- [Savcenko and Bosch, 2008] Savcenko, R. and Bosch, W. (2008). "EOT08a empirical ocean tide model from mutli-mission satellite altimetry". In "DGFI Report No. 81, Deutsches Geodätisches Forschungsinstitut (DGFI), München, 2008".
- [Schnizer, 2003] Schnizer, B. (2003). "Vorlesungsskriptum Analytische Mechanik 2003, Kap. 8, ITP Universität Graz".
- [Schulz, 2006] Schulz, H. (2006). "Physik mit Bleistift. Das analytische Handwerkszeug des Naturwissenschaftler". Verlag Harri Deutsch.
- [Schureman, 1958] Schureman, P. (1958). "Manual of harmonic analysis and prediction of tides", vol. 98, of Special Publication / US Dep. of Commerce. US Coast and Geodetic Survey. Repr. with corrections. edition, U.S. Gov. Print. Off., Washington.

- [Schwiderski, 1980] Schwiderski, E. W. (1980). "On charting global ocean tides". Rev. Geophys. 18, 243–268.
- [Sidi, 2006] Sidi, M. J. (2006). "Spacecraft dynamics and control: A practical engineering approach", vol. 7, of Cambridge aerospace series. Reprint. edition, Cambridge Univ. Pr., Cambridge.
- [Simon et al., 1994] Simon et al. (1994). "Numerical expressions for precession formulae and mean elements for the Moon and the planets". \aap 282, 663–683.
- [Smith, 1998] Smith, D. A. (1998). "There is no such thing as The' EGM96 geoid: Subtle points on the use of a global geopotential model.". IGeS Bulletin No. 8 International Geoid Service, Milan, Italy, pp. 17–28.
- [Srinivas Bettadpur, 2007] Srinivas Bettadpur (2007). "GRACE Product Specification Document: GRACE 327-720: CRS Texas".
- [Standish, 1998] Standish, E. (1998). "JPL Planetary and Lunar Ephemerides, DE405/LE405". PL IOM 312.F-98-048.
- [Thomas, 1999] Thomas, J. B. (1999). "An Analysis of Gravity-Field Estimation Based on Inter-satellite Dual One-Way Biased Ranging". "JPL Publication 98-15, p. 3-13, May 1999.".
- [Torge, 1989] Torge, W. (1989). Gravimetry. de Gruyter, Berlin.
- [Torsten Mayer-Gürr, 2006] Torsten Mayer-Gürr (2006). "Gravitationsfeldbestimmung aus der Analyse kurzer Bahnbögen am Beispiel der Satellitenmissionen CHAMP und GRACE"
   - Diss.
- [Vallado and McClain, 2007] Vallado, D. A. and McClain, W. D. (2007). "Fundamentals of astrodynamics and applications", vol. 21, of Space technology library. 3. ed. edition, Microcosm Press [u.a.], Hawthorne, Calif.
- [Wahr et al., 1998] Wahr et al. (1998). "Time variability of the Earth's gravity field: Hydrological and oceanic effects and their possible detection using GRACE". J. Geophys. Res. 103, 30205–30229.
- [Wenzel, 1985] Wenzel, H.-G. (1985). "Hochauflösende Kugelfunktionsmodelle für das Gravitationspotential der Erde", vol. 137, of Wissenschaftliche Arbeiten der Fachrichtung Vermessungswesen der Universität Hannover. Univ. Fachbereich Bauingenieur- u. Vermessungswesen, Hannover.
- [Wenzel, 1997] Wenzel, H.-G. (1997). "Tide-generating potential for the Earth". In Tidal Phenomena, (Wilhelm et. al., ed.), vol. 66, of Lecture Notes in Earth Sciences pp. 9–26. Springer Berlin / Heidelberg.
- [Wertz, J. R., 2001] Wertz, J. R. (2001). "Mission Geometry; Orbit Constellation Design and Managment". Microcosm Press.
- [Wieczorek, 2010] Wieczorek, M. (2010). "SHTOOLS v. 2.5: Documentation on http://www.ipgp.fr/ wieczor/SHTOOLS/www/conventions.html".
- [Williams, 1996] Williams, J. H. (1996). "Fundamentals of Applied Dynamics". Wiley, New York, NY.