Attitude Control and Determination System *for* **DTUsat1**

by
Torsten Lorentzen
c958425

CubeSat project
Department of Automation, Ørsted•DTU
Technical University of Denmark
Supervised by Prof. Mogens Blanke

Last edited 1st February 2002

Abstract

The simulations and control algorithm used during the work on the **DTUsat1** project is presented. The work is partly made in coorporation with the rest of the ACDS team and will result in a flight version of the ACDS system to be used on **DTUsat1**.

Contents

Chapter 1

Introduction

1.1 Introduction

Several subsystems in a satellite may require a stable satellite for better performance or to function at all. For instance radio communication will require less power if the antenna(s) can be pointed towards the Earth and the solar panels may increase power output if properly directed towards the Sun. Several of the discussed payloads also requires a stabilized platform where pitch, roll and yaw are controlled.

The above described requirements states that an attitude control system is needed. This report describes the relevant attitude control and determination strategies available for small satellites with special emphasis on the methods selected for **DTUsat1**. The modelling, simulations and control algorithms for the ACDS system used in the **DTUsat1** project is presented

The work is partly made in cooperation with the rest of the ACDS team, consisting of Jan Hales, Martin Pedersen and Klaus Krogsgaard.

I have made clear which part of the work is made in coorporation with others and where necessary there work is quoted and referenced.

1.1.1 Scope of the report

This report and the work described, covers only part of a ACDS system for a satellite. The work has primarily bin driven by the fact that we are building a satellite and that the work should end up being used for this. The focus is for that reason very focused on the preliminary work of modelling the satellite and its surroundings. A simple control algorithm is suggested and tested, that will detumble the satellite and bring it to a stable equilibria. The onboard sensor hardware currently under development are briefly introduced, but it is not the focus of this work, for a fuller description of the ADS

sensors see [2]. The actuators consisting of a set of three magnetotorquers are described in detail in section 7.2.1.

The suggested control algorithms and onboard models have not been implemented in a flight version, but only in MatLab. A description the work still needed before a flight version of the ACDS system is ready can be found at the end of the report in section 8.1

Chapter 2

Problem statement

The problem considered in this report consist of controlling a satellite in a low Earth orbit by means of only magnetic actuators, the satellite **DTUsat1**

The two payloads carried onboard **DTUsat1** requires a knowledge of the attitude of the satellite and some means of controlling the attitude.

The camera requires that the satellite does not spin to fast in order to take pictures with a short shutter time. The camera also requires attitude determination, to be able to take interesting pictures of for instance Denmark.

The tether experiment also requires a low spin of the satellite during deployment of the tether, so that the tether is not entangled in the antennas or spun around the satellite.

The antenna configuration of **DTUsat1** does not require a fully stabilised satellite in order to get in contact with the Earth, but in order to get the best possible radio signal a stabilised satellite is required

2.1 Requirements

The specific requirements from the different subsystems are examined in this section[1]

2.1.1 Camera

The camera group have given the requirement:

- The satellites rate with relation to a fixpoint on the ground, should be less than $0.005 rad/s$.

[1] It is important to note that the detailed requirements from the payloads were given to the ACDS group very late in the design phase, where the overall design strategy and selection of sensors and actuators was already in place

- Attitude estimate in order to take interesting pictures

2.1.2 Tether

The requirements from the Tether subsystem are divided into general requirement and requirements, during and after deployment of the tether:
General requirements:

- Attitude estimate
- Geomagnetic field estimate
- $\pm 10^{\circ}$ attitude control accuracy in any direction

During deployment:

- Low spin rate
- Attitude change of $15 - 30^{\circ}$ during deployment
- Compensation from additional disturbances caused by deployment

After Deployment:

- Only damping control after deployment of the tether

Furthermore if possible the tether group would like an estimate of the disturbance torque that the satellite is exposed to due to tether deployment. This requires that the compensation of the disturbance by the control system are evaluated to estimate the additional disturbance during deployment.

2.1.3 Radio

There is no requirements from the radio group, this is due to the fact that it was decided, due to the fact that the radiocommunication is one of the main criteria of success for the entire satellite, that the antennas should be able to transmit and receive radio signals even if the attitude control system is not working.

2.1.4 Power

A power budget that describes the allowable maximum power consumption of the different subsystems can be found at the **DTUsat1** homepage[10]. and the power constraints by the the available surface area and the effectiveness of the solar cells and the power subsystem. The power budget yields the following constraints to the magnetotorquers

Detumbling phase: A maximum power consumption of 150mW in total

Operational phase: A maximum power consumption of 50mW in total

2.2 Constraints

The constraints on the ACDS system are the limitations in available space, power and weight. The constraints in mass and size are determined by the CubeSat concept:

Mass: The **CubeSat** concept prescribes a maximum weight of $1kg$

Volume: A **CubeSat** is a cube with a volume of 10cm^3

Chapter 3

General Problem analysis

The requirements from the payloads are analysed and commented and the general control problem for a satellite attitude controller is briefly introduced in this section and the most common actuators and sensors are named.

3.1 Analysis of requirements

At the point of time where the ACDS system was selected, the requirements from the tether and camera was not fully analysed and where very relaxed. The selection of sensors and actuators where therefore based primarily on the constraints. In [3] a tradeoff between different actuators and sensors where made and the ACDS instrument package decided:

- Sensors
  - Sun sensor
  - Magnetometer
- Actuators
  - Magnetotorquers

The attitude requirement from the camera group is very strict, the 0.005rad/s. pointing towards a fixed point on the ground. If the satellite is fully stabilised and the satellites attitude is fixed in inertial coordinates, the satellite will rotate if seen from a fixed point on Earth due to its velocity, using triangular approximation seen on figure 3.1 the requirement can be expressed in pitch rate:

$$sin^{-1}\left(\frac{7.5\text{km}}{600\text{km}}\right) = 0.0125\text{rad/s} \tag{3.1.1}$$

This means that in order to achieve the required pointing the ACDS system has to maintain a pitch rate of:

$$0.0125\text{rad/s} \pm 0.005\text{rad/s} \tag{3.1.2}$$

and in the same time keep roll and yaw below the required pointing accuracy. The attitude pitch rate requirement seems to be very extreme when taking

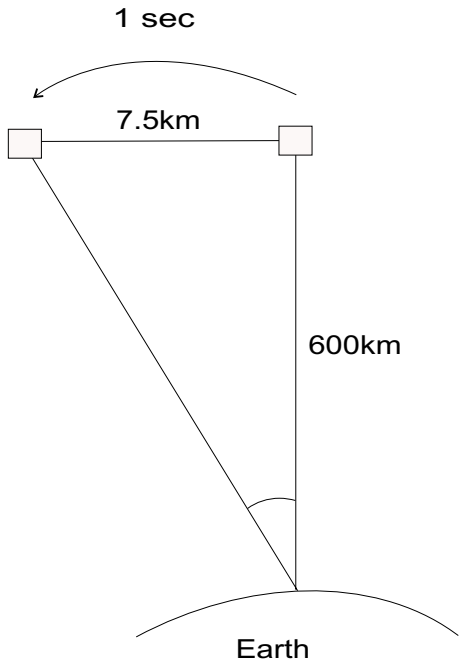


Figure 3.1: Calculations of attitude requirement from camera

into consideration that the only means of applying control torque are magnetotorquers. Very accurate control are normally performed using momentum wheels as they are much faster and much more accurate than other type of actuators. The requirement was first obtained late in the project and has therefore not been taking into consideration in the controller design.

The tether system has a $\pm 10^\circ$ requirement to the attitude and requires the possibility to set a new reference attitude and uncouple everything but the damping from the control algorithm. The Ørsted satellite has shown that a attitude accuracy of $\pm 10^\circ$ is feasible for purely magnetic control, the reference change requires use of active control

Accessability to the onboard models and attitude estimates are planned to be made accessible through a software interface and to be collected as housekeeping data.

3.2 General control problem

The attitude control problem consist consist if two problems; stabilizing the satellite in a environment with disturbances, such as air drag and gravity gradient, and performing attitude maneuvers to fulfill the payload requirements.

The first of the two problems are often solved by using passive control such as spinning the satellite or/and using gravity gradient control, where the weight distribution of the satellite has a stabilizing effect. The latter can for instance be seen on the danish Ørsted satellite that has an 8 meter long boom that has a passive stabilizing effect.

Attitude maneuvers are performed by using a control law and some sort of actuators such as momentum wheels, booster rockets or magnetotorquers.

3.2.1 Simplified attitude dynamics

The attitude dynamic equations for a satellite can under the assumption that it has a diagonal inertia matrix and small rotational speed be approximated as

$$\begin{aligned} T_d x + T_{cx} &= I_x \ddot{\Phi} \\ T_d y + T_{cy} &= I_y \ddot{\Theta} \\ T_d z + T_{cz} &= I_z \ddot{\Psi} \end{aligned} \tag{3.2.1}$$

A basic control law for (3.2.1) can be found in [9]:

$$\begin{aligned} T_c x &= K_x(\Phi_{com} - \Phi) + K_{xd}\dot{\Phi} = K_x \Phi_E + K_{xd}\dot{\Phi} \\ T_c y &= K_y(\Theta_{com} - \Theta) + K_{yd}\dot{\Theta} = K_y \Theta_E + K_{yd}\dot{\Omega} \\ T_c z &= K_z(\Psi_{com} - \Psi) + K_{zd}\dot{\Psi} = K_z \Psi_E + K_{zd}\dot{\Psi} \end{aligned} \tag{3.2.2}$$

This is three uncoupled PI controllers with the Euler angle errors and the angle rates as parameters. They can be designed using standard techniques where the desired dynamics of the three separate systems such as bandwidth and damping, are used as design parameters. This controller is sufficient for small changes in the angles, but for large changes there are several problems with (3.2.2). The equations can become singular or the controller can be saturated. For fast attitude maneuvers the cross couplings between the three axis cannot be ignored, because of the gyro effect introduced.

Chapter 4

Satellite modelling

This chapter contains the general dynamic equation of motion for a satellite and a linearised version of the same. The equations are in a later chapter used to find controllers for the satellite system.

4.1 Coordinate systems

Several different coordinate systems are used to describe the motion and attitude of the satellite. The Earth centered ECEF and Inertial coordinates and the satellite centered orbit and body coordinate systems. The coordinate systems are defined in the same way as in [9]

ECEF: Earth Centred Earth Fixed longitude and latitude coordinates that rotates with the Earth. The natural coordinate system to express the geomagnetic field.

Inertial: Fixed Earth centred right orthogonal coordinate system used as inertial reference where the z axis is the Earth axis of rotation in the positive direction and the x axis is in the vernal equinox direction.

Orbit: Satellite centred coordinate system where the z axis points towards the centre of the Earth, the y axis is perpendicular to the orbital plane and the x axis completes a right orthogonal system. For circular orbits the x axis is aligned with the orbit velocity vector. The orbit coordinate system is often denoted the reference frame. Rotations around the axis of the orbit coordinate system will be denoted roll around the x axis, pitch around the y axis and yaw around the z axis.

body: the body coordinate system is a right orthogonal system fixed in the centre of the satellite body. The z axis is in same direction as the

camera lens and the x axis is in the structural rails direction. The satellite dynamic equations are expressed in this coordinate system.

4.2 Satellite Equations of Motion

For a rigid satellite with inertia tensor $\boldsymbol{I}_s$ and angular velocity ω equipped with magnetotorquers, that delivers the torque $\boldsymbol{N}_{mag}$, but without momentum wheels and exposed to the disturbance torque $\boldsymbol{N}_{dist}$ the dynamic equation is:

$$\dot{\omega}_{bi} = -\boldsymbol{I}_s^{-1}(\omega \times \boldsymbol{I}_s\omega) + \boldsymbol{I}_s^{-1}\boldsymbol{N}_{mag} + \boldsymbol{I}_s^{-1}\boldsymbol{N}_{dist} \tag{4.2.1}$$

where ω_{bi} is the rotation of the satellite body coordinates in relation to the inertial coordinate system. The magnetic torque N_{mag} is generated by the magnetic moment from the control system when it interacts with the geomagnetic field. The magnetotorquers generates a magnetic moment by sending current into a coil, each magnetotorquer produces the magnetic moment:

$$m_{c,b} = n_c \cdot i_c \cdot A_c \tag{4.2.2}$$

where n_c is the number of turns on the coil and A_c is the cross sectional area of the coil. The torque is the cross product between the generated magnetic moments and the geomagnetic field:

$$\boldsymbol{N}_{mag,b} = \boldsymbol{m}_{c,b} \times \boldsymbol{M}_b \tag{4.2.3}$$

The disturbance torque N_{dist} contains all the unmodelled torques the satellite is exposed to such as air drag and gravity gradient torque.

4.2.1 Satellite kinematics

The connection between the attitude and the angular velocities in the different coordinate systems are described by the kinematic equations. The satellites pointing can be found by integrating the angular velocities between the body and the inertial coordinates over time. This can be done quite elegant using quaternions, the rate between the inertial and body coordinates expressed in quaternions:

$$\dot{\boldsymbol{q}}_{oi} = \frac{1}{2}[\boldsymbol{Q}] \begin{bmatrix} \omega_{bx} \\ \omega_{by} \\ \omega_{bz} \end{bmatrix} \tag{4.2.4}$$

where

$$[\boldsymbol{Q}] = \begin{bmatrix} q_{oi4} & -q_{oi3} & q_{oi2} \\ q_{oi3} & q_{oi4} & -q_{oi1} \\ -q_{oi2} & q_{oi1} & q_{oi4} \\ -q_{oi1} & -q_{oi2} & -q_{oi3} \end{bmatrix} \tag{4.2.5}$$

The quaternion from the inertial to the body coordinates can be found by integrating the above expression. When using quaternions, singularities are avoided, this is not the case if the above equation is expressed using Euler angles.

In order to analyse the satellite system the dynamic equation (4.2.1) is linearised around a specific angular velocity.

4.2.2 Simple Linearised Satellite Model

The simplest linearised model of a satellite is to consider a non spinning satellite where the actuators are aligned with the principle axes of inertia. The satellite equations of motions is then simplified to:

$$\dot{\omega}_{bi} = \boldsymbol{I}_s^{-1}\boldsymbol{N}_{mag} + \boldsymbol{I}_s^{-1}\boldsymbol{N}_{dist} \tag{4.2.6}$$

This gives a transfer function from magnetic torque to each satellite angle on:

$$\frac{\boldsymbol{\Theta}(\boldsymbol{s})}{\boldsymbol{N}_{mag}(s)} = \frac{1}{\boldsymbol{I}_s s^2} \tag{4.2.7}$$

This means that in the simplified case, the satellite can be modeled as three uncoupled, undamped oscillators. It is then possible to design controllers using general SISO control theory. It is then necessary to verify the controllers by intensive testing as the stability is not guarantied because of the limitations in torque capability of magnetic control

4.2.3 Extended Linearised Satellite Model

The assumption that the satellite is not spinning has the effect that the three axes becomes uncoupled. When the satellite is spinning around one axis the gyro effect will influence on the other two axis. The linearisations used here are further described in [5][1] The cross coupling between the axes

[1]Note that in [5] the quaternions and the coordinate systems are defined differently from here

4.2.4 Unlinear satellite model

An unlinear satellite model seen on 4.1 was constructed in SimuLink, the model is used in the simulations to verify the controllers and is further described in 7.4

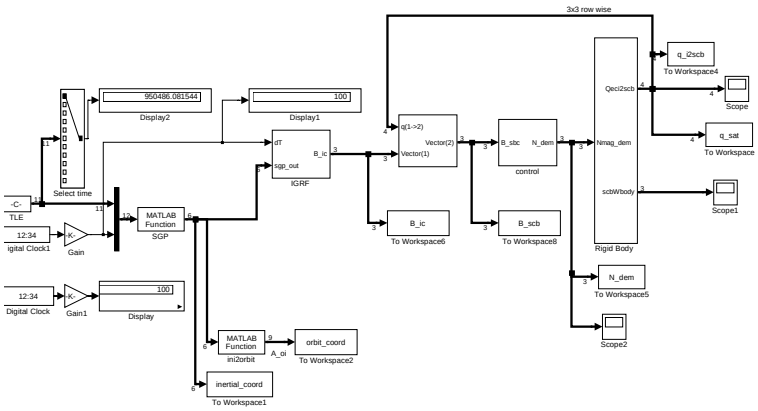


Figure 4.1: Simulink model of **DTUsat1**

4.3 Satellite specifications

The satellite consist of a cube with the external measurements 10x10x10cm with a weight of 1kg.

The structure consist of a aluminium honeycomb structure with aluminium rails placed on four parallel corner sides. The structure itself weights approximately 260g.

The inertia tensor is not known exactly as the mechanical structure is not final yet, the launch company has the requirements that the center of mass should be within two centimeters from the center of the satellite. A rough estimate of the inertia, can be calculated if the satellite structure is modelled as a uniformly mass distributed cube with four dense rails mounted in parallel at four of the corner sides. This was done in [3] and the inertia matrix was calculated as:

$$I_s = \begin{bmatrix} 6.858 \cdot 10^{-4}\text{kg m}^2 & 0 & 0 \\ 0 & 6.858 \cdot 10^{-4}\text{kg m}^2 & 0 \\ 0 & 0 & 8.164 \cdot 10^{-4}\text{kg m}^2 \end{bmatrix} \tag{4.3.1}$$

At the moment the most likely orbit for the satellite is a 600km prograde orbit with an inclination of 61°, due to the small size of the satellite and

the requirements to the mass distribution the disturbance from airdrag and gravity gradient are very small. The disturbances are combined and modelled as a periodic function for ease of calculations:

$$N_{dist} = N_0 \cdot \frac{2\pi t}{T_o} \tag{4.3.2}$$

where N_0 is 10^{-6}Nm.

For satellites where the mass distribution is not uniform, eg. because of solar panels or boom structures, the gravity gradient can be very significant and should be considered separately. Satellites containing momentum wheels should also be analysed with a non periodic disturbance to investigate momentum build-up in the wheels.

Chapter 5

Environment modelling

5.1 Orbit prediction

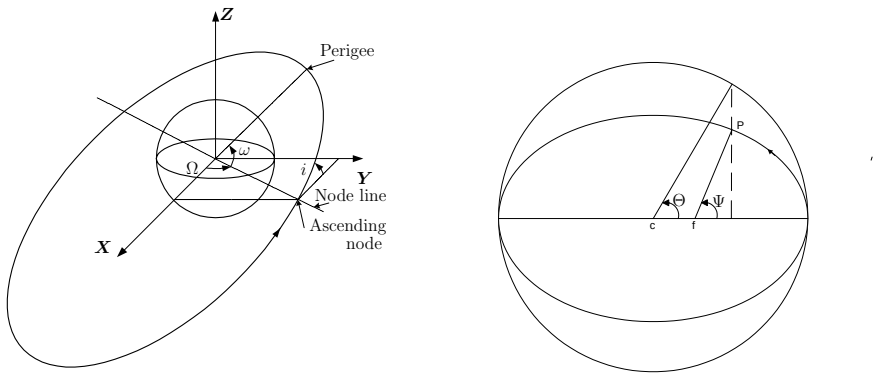
This section describes the implemented algorithms developed to predict the satellites position in orbit. The implemented functions referenced in this section was implemented in conjunction with Martin Pedersen and Klaus Krogsgaard from the **DTUsat1** *ACDS group.*

In order to determine the attitude of the spacecraft from inertial sensors, it is necessary to know the satellites position in space. To determine this, a model of the orbit dynamics is carried onboard the satellite.

It is possible to predict the position and velocity of low Earth orbiting objects with a good accuracy by solving a two body problem if the most important perturbations is accounted for. In order to predict a satellites position it is necessary to know a reference point and the elapsed time. NORAD[1] keeps track of all objects over a certain size currently in orbit and puts the information in the form of Two Line Element Sets (TLE's) at everyones disposal.

The elements in the NORAD TLE's are mean values of the classical orbit parameters generated by removing periodic variations, it is therefore necessary to use a prediction method that reconstructs the variations in a compatible way with the way they are removed. Five methods that can be used is presented in [4] with included Fortran source code. A MatLab version of the first and simplest of the these methods, the SGP[2] method, is implemented here.

[1] The North American Aerospace Defense Command
[2] Simplified General Perturbations



(a) Classical orbit parameters

(b) Relationship between mean and eccentric anomaly

Figure 5.1: Illustration of orbit parameters

5.1.1 The orbit parameters

The TLE contains mean values of the classical orbit parameters

a: the semimajor axis of the ellipse

e: the eccentricity of the ellipse

i: the inclination is the angle between the orbit plane (the ecliptic) and the equatorial plane

Ω: the right ascension of the ascending node is the angle in the ecliptic between the radius vector of the moving body and the radius vector to the orbits perigee

ω: the argument of perigee is the angle between the perigee radius vector and the node line (the line between the ascending and descending node)

M: the mean anomaly is defined as $2 \cdot \pi(\Delta t/P)$rad, where P is the period and Δt is the time since perigee passage. It describes the angle to the satellite at a given time if the orbit is circular. This is a parameter of no physical interest but only for ease of calculations. The relationship between the mean anomaly M, the true anomaly Θ and the eccentric anomaly Ψ is shown graphicly on figure 5.1(b)

5.1.2 SGP

The SGP solves Kepler's equation by numeric iteration and then it reconstructs the periodic variations that where removed when constructing the TLE. The SGP algorithm which is the simplest of the useable algorithms makes some simplifications during the reconstructions where the atmospheric drag are modelled as a periodic linear function.

5.2 Modelling of the Earth magnetic field

This section describes the model necessary to use the Earth magnetic field to determine a satellites attitude, this section is made in conjunction with the rest of the ACDS group, especially Jan Hales who implemented the model.

Using the Earth magnetic field for attitude control and determination requires a precise knowledge of the field. A magnetometer is part of the **DTUsat1**'s attitude determination system (ADS). The magnetometer measures a magnetic field strength vector, this is compared with an onboard reference map of the Earth's magnetic field to determine the satellites attitude. This requires knowledge of the satellites position in the orbit so the magnetometer as an ADS depends on both the orbit prediction and the onboard model of the geomagnetic field.

The model used on **DTUsat1** is a IGRF2000[3] model, which is the international standard model that is recommended for scientific use by the IAGA[4] from the year 2000 and forward. The model consist of a series of truncated spherical harmonic series describing the main field and its variations. It is based on empirical data from several different sources such as land based measurements, ships, planes and satellites [7]. The current version of the model with epoch in 2000 has been adapted to the new precise measurements from the Ørsted satellite.
The IGRF model is defined as:

$$\boldsymbol{V} = a\sum_{n=1}^{N}\sum_{m=0}^{n}(\frac{a}{r})^{n+1}(g_n^m cos(m\phi) + h_n^m sin(m\phi))\boldsymbol{P}_n^m cos(\theta) \qquad (5.2.1)$$

where $\boldsymbol{V}$ is the geomagnetic scalar potential, a is the mean radius of the Earth, r is the distance from the centre of the Earth, ϕ is the longitude, θ is the colatitude defined as 90° minus the latitude, g_n^m and h_n^m are spherical harmonic coefficients called Gauss coefficients of degree n and order m. N is the maximal spherical harmonical degree of the series expansion. $\boldsymbol{P}_n^m cos(\theta)$

[3]International Geomagnetic Reference Field
[4]International Association of Geomagnetism and Aeronomy

is the Schmidt quasi-normalised associated Legendre functions of degree n and order m calculated from (5.2.3).
The associated Legendre functions are normalised with the constants:

$$K_n = \begin{cases} \frac{1}{2^n} & ; m = 0 \\ \\ \frac{1}{2^n}\sqrt{\frac{2(n-m)!}{(n+m)!}} & ; m > 0 \end{cases} \tag{5.2.2}$$

which gives the normalized associated Legendre functions:

$$\boldsymbol{P}_n^m = \begin{cases} (\frac{1}{2^n n!})(\frac{d}{d\boldsymbol{x}})^n(\boldsymbol{x}^2-1)^n & ; m = 0 \\ \\ \sqrt{\frac{2(n-m)!}{(n+m)!}(1-\boldsymbol{x}^2)^m}(\frac{1}{2^n n!})(\frac{d}{d\boldsymbol{x}})^n(\boldsymbol{x}^2-1)^n & ; m > 0 \end{cases} \tag{5.2.3}$$

The model coefficients that are determined from empirical data are given in app. A.3.2, the model makes it possible to calculate the Magnetic Field Strength Vector for any given set of geocentric spherical coordinates r, ϕ and θ.

5.2.1 Implementation

The implementation is made in C by the Ørsted team and has generously been at our disposal. During the test phase the program was directly imported in MatLab. At present the most likely orbit for **DTUsat1** is a 600km 65° orbit, a field strength map generated for this orbit with the implemented IGRF2000 model is seen on figure 5.2. The map shows a good contingency with results calculated with the tool at [1]. .

5.3 Sun position prediction

The suns movement as seen from the Earth, i.e. the Inertial system, can be modelled given the classical orbital parameters as a two body problem where the major perturbations from eg. the Earth's oblateness are included. The simplified model described below was developed by Paul Schlyter in 1979 and is described in [8].

The onboard sunsensors are expected to have an accuracy of approximately two degrees, it is therefore required that the model of the suns position has the same or a higher degree of accuracy in order to get as good a determination as possible.

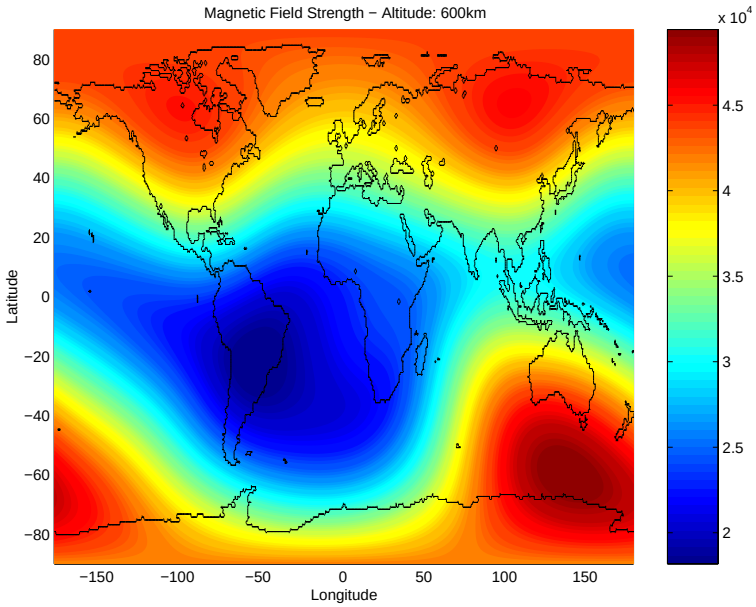


Figure 5.2: Earth magnetic field strength in 600km height over the surface

5.3.1 Computing the position

Given the classical orbit parameters seen in section 5.1.1, for the Earth's movement around the sun and reference time denoted *epoch* it is quite simple to calculate the suns position. The epoch used in these calculations are 0. Jan 2000 12,00 (that is 31. Dec. 1999).

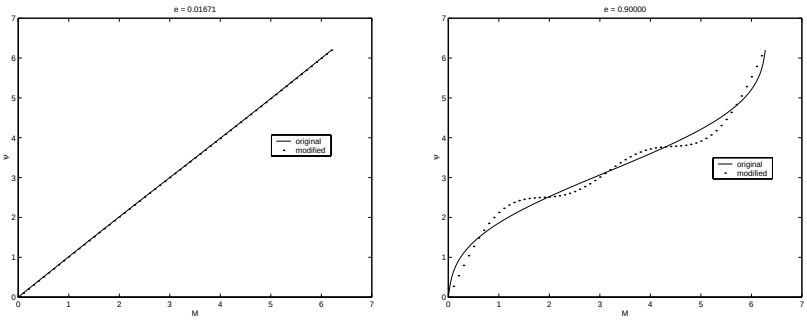
At first the eccentric anomaly Ψ is calculated from the mean anomaly M and the eccentricity e using Kepler's equation:

$$M = \Psi - e \cdot sin(\Psi) \tag{5.3.1}$$

Solving Kepler's equation to find Ψ has to be done iterativly as there is no closed form solution but for very near circular orbits [8] suggest a series expansion that results in the below approximation:

$$\Psi = M + e \cdot sin(M) \cdot (1 + e \cdot cos(M)) \tag{5.3.2}$$

Since the Earth's orbit around the sun has a very small eccentricity this approximation is sufficiently accurate, this can be seen on figure 5.3, where the accuracy of the approximation is shown by plotting the mean anomaly as a function of the eccentric anomalies for both formulaes. When the eccentric



(a) Earth eccentricity (b) Large eccentricity

Figure 5.3: Accuracy of the approximation (5.3.2) for small and large eccentricities

anomaly is calculated, the true anomaly is calculated from the following:

$$R_x = R \cdot cos(\Theta) = a \cdot (cos(\Psi) - e) \tag{5.3.3}$$

$$R_y = R \cdot sin(\Theta) = a \cdot (\sqrt{1 - e^2} \cdot sin(\Psi)) \tag{5.3.4}$$

$$\Theta = atan2(\frac{R_y}{R_x}) \tag{5.3.5}$$

and the distance to the sun is found

$$R = \sqrt{R_x^2 + R_y^2} \tag{5.3.6}$$

where $atan2$ is the four quadrant version of the inverse tangent function. The suns longitude is found from the argument at perihelion ω and the true anomaly Θ

$$lon_{sun} = \Theta + \omega \tag{5.3.7}$$

Expressed in rectangular coordinates in the ecliptic plane:

$$x_s = R \cdot cos(lon_{sun}) \tag{5.3.8}$$

$$y_s = R \cdot sin(lon_{sun}) \tag{5.3.9}$$

$$z_s = 0 \tag{5.3.10}$$

and finally rectangular coordinates in the inertial geocentric coordinate system:

$$xe = xs \tag{5.3.11}$$
$$ye = ys \cdot cos(e) \tag{5.3.12}$$
$$ze = ys \cdot sin(e) \tag{5.3.13}$$

The result has an accuracy well below an arc minute as long as the real time watch of the satellite is accurate.

5.4 Implementation of onboard models

This section describes the model implementations suggested for **DTUsat1**. *The models are at present only available in the versions used for test purpose implemented in MatLab.*

5.4.1 Requirements for onboard models

The main difference between the test and flight versions of the model is the available computer processing time. In orbit all the subsystems share the same OBC, it is therefore vital that the models are as simple as possible but still as accurate as necessary.

5.4.2 Real time clock

All the models accuracy depends heavily on the accuracy of the real time clock, it has been proposed to implement the watch in software using the processors build in clock signal as reference. The CPU clock is based on an oscillating crystal that has a high accuracy, the accuracy is influenced by the large temperature deviations that the satellite is exposed to, and the deviation due to temperature can be as large as up to 1 second per 3 hours[5]. It should be possible to compensate the temperature drift, but it is not certain if this will be implemented or not so the above accuracy is assumed.

Because of the risk that the computer will reboot due to latch-up or faulty software, the clock is backed up once a second to flash ram. In the worst case the latch up will occur just before the clock is backed up and the lost time will be one second.[6] The probability of latch- up of the OBC depends

[5]Based on conversation with Nils Andersen, associate professor IAU

[6]The boot time itself should be accounted for when restoring the time

primarily on the orbit of the satellite and the selected hardware. The orbit is a low Earth orbit where the worst radiation is avoided. The hardware has not been flight tested before and the CPU is based on a technology that can be vulnerable to latch-up. There has not yet been a latch-up test of the OBC but it is *assumed* here that more than one latch-up in a 12 hour period is unlikely.

The clock will be synchronised during radio contact with the ground station if all goes well the satellite will be in contact with the ground station whenever possible. A radio contact period consist of up to three consecutive orbits, the maximum number of orbits between possible contact periods is 6 orbits which is approximately 10 hours, it is however quite likely that the radio contact with the ground station is interrupted sometimes due to eg. low battery power, a reasonable estimate is that the satellite is without contact in two successive contact periods. this means that the clock will not be updated and the clock offset will be able to accumulate for up to 30 hours. This yields a worst case situation where the real time clock will deviate with:

$$\Delta t_{exp} \approx \left(\frac{1}{3}\ \text{s/h} + \frac{1}{8}\ \text{s/h}\right) \cdot 30\ \text{h} \approx 14\text{s} \tag{5.4.1}$$

5.4.3 Orbit prediction

The precision of the orbit propagator is very vital, as all our attitude sensors are relative to external reference coordinate systems. In order to use there information it is therefore necessary to know the satellites position in some Inertial system.

The precision of the orbit propagator depends primarily on three parameters

- The precision of the real time clock
- The complexity of the model
- The intervals between TLE updates

Clock precision

The satellite moves through space with approx. 7 km/sec. this means that with the worst case time deviation of 15 seconds the position deviation caused by the real time clock will be:

$$\Delta P_{rtc} = 7.5\text{km/s} \cdot 14\text{s} = 105\text{km} \tag{5.4.2}$$

model complexity

Simulations from the Ørsted satellite project, shows that the SGP4 model has an accuracy well below 20 kilometres after 15 days [5]. The SGP model implemented here is not as precise and a larger deviation is expected. A conservative estimate of the accuracy of the SGP model would be an accuracy of 40 kilometres after 14 days.[7]

$$\Delta P_{SGP} = 40\text{km} \tag{5.4.3}$$

It is possible to get well written implementations of all the SGP algorithms written in C or in Fortran, the propagator is quite simple and only contains one iteration process namely the numerical solution to Kepler's equation, for low values of eccentricity the iteration converge well within ten iterations, making the entire function very efficient. Given that the precision of the SGP model is better than the possible offset due to the real time clock, it seems that it would be a waste of time implementing a more complex model.

TLE updates

The SGP model drifts with time because it does not take all perturbations into consideration, it is therefore necessary to update the model parameters and start the propagation from a new epoch. It is suggested that the model is updated once a week with TLE data from NORAD, this would give a maximum error of 20 kilometres in error free condition, using the same estimate on the model accuracy as above.

5.4.4 Overall precision

The overall precision of the orbit model in worst case is just before TLE update and with maximum deviation of the real time clock. This would give a position error of:

$$\Delta P_{max} = \Delta P_{SGP} + \Delta P_{rtc} = 35km + 105km = 140km \tag{5.4.4}$$

If the control algorithm was to use the information to home in on the ground station, this would mean that the satellite with a 600km altitude would have a pointing error of (using triangular approximation):

$$sin^{-1}(\frac{140}{600}) \approx 13.5^{\circ} \tag{5.4.5}$$

[7] This has not been examined and should be considered before the final implementation of the orbit model

This is not acceptable and mainly due to but it is quite easy to compensate the temperature drift of the RTC either in software or by operating the clock resonator at constant temperature.

5.4.5 Sun position model

The sun position model is very similar to the orbit propagator, and the accuracy depends on the same parameters, but the perturbing forces are smaller, because of the different proportions between the multiple bodies, so the propagator is more simple eg. the satellite motion is very influenced by the sun so it is not very accurate to consider the satellite motion as a two body problem, whereas the Earth movement around the sun is much less influenced (but still considerably) by other large masses such as other planets. The pointing accuracy to the sun is not affected in the same way as the pointing towards the Earth because of the larger distance.

5.4.6 magnet model

The IGRF2000 model is implemented in C and imported in MatLab. The model has 144 coefficients and takes the first 10 harmonics of the magnetic field into consideration. Simulations using this model where very slow and took a lot of processing power. In return a very precise model of the magnetic field is achieved with an accuracy of 50nT in RMS value[8].

Even though we are planning to fly a fast 32 bit processor the processing power needed for the above implementation is way to high. A tabularisation of the model stored in ROM or FLASH is therefore suggested where the magnetic field strength vector is tabularised for every 5 degrees longitude and latitude. Instead of calculating the field, the onboard computers work is reduced to a table look-up which is a lot faster. If the individual vector components are stored as 32 bit floating points this will require a permanent storage of the size:

$$32bit \cdot \frac{360^\circ \cdot 360^\circ}{5^\circ} \cdot 3 = 2.5Mbit \approx 200kB \tag{5.4.6}$$

A small section of the tabularised magnetic field vector, can be seen in app. A.4.

[8] According to [1] this accuracy can be achieved using the year 2000 coefficients

Chapter 6

Attitude Determination

6.1 Attitude Determination sensors

Attitude determination sensors are typically sensors that measures the relative angle between the spacecraft and some known objects, eg. the sun or the stars and rate sensors that measures the angular velocities between the spacecraft and an inertial system. Examples of rate sensors could be gyros either gimballed gyros with the drawback of moving parts or more recently laser gyros.

6.1.1 Attitude determination sensors on DTUsat1

This section describes the ADS sensors under development by Martin Pedersen and Jan Hales (sun sensors) and Klaus Krogsgaard (magnetometer) from the **DTUsat1** *ACDS group. Further information regarding the sensors and there interface is described in [2]*

Sunsensors

The sun sensors will be two directional slit sensors mounted on all the sides of **DTUsat1**. The sensors will presumably have an opening angle of 60° and an accuracy of approximately 2°.

Magnetometer

The magnetometer consist of three single axis magnetic field sensors placed orthogonal on each other. The sensor has a presumed accuracy of less than one degree.

6.2 Attitude determination

The sensors produce a vector that gives the direction from the spacecraft body coordinates to an external reference object. In order to determine the attitude it is necessary to know the current orbit position and respectively the direction from the Earth to the sun and the direction of the magnetic field vector in the current orbit position.

6.3 Evaluation of multiple attitude sensor output

This section describes the problem of estimating the attitude from several different vector measurements.

The magnetometer and the sun sensors, produces unit vectors in the measured direction, combining two or more measurements in an optimal way are normally done by solving Wahbas problem[6]. Wahbas problem is to find the orthogonal matrix $\boldsymbol{A}$ with determinant 1 that minimises the loss function:

$$L(A) \equiv \frac{1}{2}\sum_{i=1}^{n} a_i |\boldsymbol{b}_i - A\boldsymbol{r}_i|^2 \tag{6.3.1}$$

where $\{\boldsymbol{b}_i\}$ is a set of the measured vectors in space craft body coordinates, $\{\boldsymbol{r}_i\}$ are the corresponding vectors in inertial coordinates and $\{a_i\}$ are positive weights and n is the number of different measurements. Several different algorithms exists to solve the problem, in [6] several different algorithms are presented. The q-method is further investigated here.

The q-method expresses the $\boldsymbol{A}$ matrix in terms of quaternions as:

$$\boldsymbol{A} = (q_4^2 - |\boldsymbol{q}|^2)I + 2\boldsymbol{q}\boldsymbol{q}^T - 2q_4[\boldsymbol{q}\times] \tag{6.3.2}$$

where

$$[\boldsymbol{q}\times] = \begin{bmatrix} 0 & -q_3 & q_2 \\ q_3 & 0 & -q_1 \\ -q_2 & q_1 & 0 \end{bmatrix} \tag{6.3.3}$$

and $\boldsymbol{q}$ is the vector part of the quaternion.
The representation of $\boldsymbol{A}$ is a homogenous quadratic function of the quaternion which means that according to [6]:

$$tr(\boldsymbol{A}\boldsymbol{B}^T) = q^T\boldsymbol{K}q \tag{6.3.4}$$

where

$$\boldsymbol{B} = \sum a_i \boldsymbol{b}_i \boldsymbol{r}_i{}_{i=1}^{n} \tag{6.3.5}$$

and

$$K \equiv \begin{bmatrix} \boldsymbol{B} + \boldsymbol{B}^T - I\ tr(\boldsymbol{B}) & \boldsymbol{z} \\ \boldsymbol{z}^T tr(\boldsymbol{B}) & \end{bmatrix} \tag{6.3.6}$$

where

$$\boldsymbol{z} \equiv \begin{bmatrix} B_{23} - B_3 2 \\ B_{31} - B_1 3 \\ B_{12} - B_2 1 \end{bmatrix} = \sum a_i \boldsymbol{b}_i \times \boldsymbol{r}_i{}_{i=1}^{n} \tag{6.3.7}$$

The optimal quaternion can be found as the normalised eigenvector of $\boldsymbol{K}$ with the largest eigenvalue. Algorithms that can solve this problem are robust but slow, faster algorithms exists, but the q-method and the SVD algorithm also presented in [6] are the most robust.

Chapter 7

Attitude Control

7.1 Attitude control requirements

The requirements to the control system is a pointing in the range of $\pm 10^{c}irc$ and a low spin rate, further more it should be possible to change the attitude reference during the different phases of the tether, after the tether is deployed the control system should only minimize the spin of the satellite and not try to control the attitude. The camera group came with a extreme demand for the rate accuracy, after the selection of actuators and sensors was made. The requirement is not considered feasible with the selected attitude hardware and is not taken into consideration in evaluation of the control system.

7.2 Actuators

This section describes the magnetotorquers that are used as actuators. The dimensioning section is made in conjunction with Jan Hales and Klaus Krogsgaard. The driver circuit presented in the end of the section is reused from the magnetometer and designed by Klaus Krogsgaard.

The satellite is equipped with magnetotorquers as the only attitude actuators. The selection of solely magnetic control was made in [3] as a trade of between weight, power consumption and available power. The magnetic approach means that the attitude control system has to take into consideration that it is not possible to control all three axis at the same time, as it is only possible to control roll and pitch at the poles and pitch and yaw at the equator due to the direction of the Earth magnetic field.

The magnetotorquers was dimensioned in [3] from the requirement to maximum available torque and maximum power consumption.

7.2.1 Dimensioning of magnetotorquers

The magnetotorquers are dimensioned from the constraints given by the physical size of the satellite, and the limitations in weight and available power. The satellites mechanical constraints employs a realisable cross area of 7cm by 7cm and a winding area of 5mm by 2.5mm. With a design goal of $P_{coil,max} = 50\text{mW}$ per coil from a 3.3V supply voltage U_s the current and resistance becomes:

$$I_{coil} = P_{coil}/U_s = 15.15\text{mA}$$
$$R_{coil} = U_s^2/P_{coil} = 218\Omega$$

The torque due to a magnetic dipole moment $\boldsymbol{m}$ in a magnetic field $\boldsymbol{B}$ is given by

$$\boldsymbol{N} = \boldsymbol{m} \times \boldsymbol{B} \tag{7.2.1}$$

where $\boldsymbol{m}$ is given by

$$\boldsymbol{m} = \boldsymbol{n}IS \tag{7.2.2}$$

where S is the surface spanned by the current I and $\boldsymbol{n}$ is the vector perpendicular to the surface with positive direction relative to I in a right orthogonal system. The cross area and the current – implied by equation 7.2.1 and 7.2.2 – has to be maximised in order to maximise the torque. This is achieved by making several turns of thin wire. By doing this and using round enameled wire one has to take the fill factor in to account when calculating the effective cross area of the coil.

The fill factor for e.g. a 0.22mm wire is approx. 66.9%, because of the non-ideal winding of the coil and the use of thinner wire, which results in the enamel to be be a larger fraction of the wire cross aread, a more realistic fill factor of 40% is used giving an effective coil cross area of:

$$A_{eff.coil} = 0.4 \cdot 2.510^{-3}\text{m} \cdot 5.010^{-3}\text{m} = 510^{-6}\text{m}^2 \tag{7.2.3}$$

The total resistance of the wire R_{tot} is given by the specific resistance of copper r_s, the number of turns n, the width of the coil x_t, the height of the coil y_t and the cross area of the wire A_{wire}:

$$R_{tot} = r_s \frac{n(x_t + y_t)2}{A_{wire}} \tag{7.2.4}$$

The coil current I is then given by the supply voltage V_{cc} and the total resistance R_{tot}:

$$I = \frac{V_{cc}}{R_{tot}}$$

The total current through the coil's cross area I_{tot} is given by the number of turns n and the coil current I:

$$I_{tot} = nI$$

The coil current I can now be found:

$$\begin{aligned} P &= V_{cc}I = R_{tot}I^2 \\ &= r_s \frac{n(x_t + y_t)2}{A_{wire}} \left(\frac{I_{tot}}{n}\right)^2 \\ &= r_s \frac{2(x_t + y_t)}{A_{wire}n} {I_{tot}}^2 \\ \Leftrightarrow {I_{tot}}^2 &= \frac{PA_{wire}n}{r_s 2(x_t + y_t)} \\ \Leftrightarrow I_{tot} &= \sqrt{\frac{PA_{wire}n}{r_s 2(x_t + y_t)}} \end{aligned}$$

For $P = 50\text{mW}$, $A_{wire}n = A_{eff.coil} = 5.00 \cdot 10^{-6}\text{m}^2$, $r_s = 1.724 \cdot 10^{-8}\Omega\text{m}$ and $x_t = y_t = 0.07\text{m}$:

$$I_{tot} = \sqrt{\frac{50\text{mW} \cdot 5.00 \cdot 10^{-6}\text{m}^2}{1.724 \cdot 10^{-8}\Omega\text{m} \cdot 2\,(0.07\text{m} + 0.07\text{m})}} \approx 7.2\text{A}$$

Given a magnetic field of $50\mu\text{T}$ in the same plane as the magnetotorquer. This produces the maximum torque:

$$\tau_{max} = AI_{tot}B = (0.07\text{m})^2 \cdot 7.2\text{A} \cdot 50\mu\text{T} \approx 1.75\mu\text{Nm}$$

The cross area of the wire as a function of effective area, supply voltage and coil power can be found by combining (7.2.4) with $A_{eff,coil} = nA_{wire}$ and $U^2 = PR$:

$$A_{wire} = \frac{\sqrt{2Pr_sA_{eff,coil}(x_t + y_t)}}{V_{cc}}$$

For the above given values the cross area is:

$$A_{wire} = \frac{\sqrt{2 \cdot 0.05\text{W} \cdot 1.724 \cdot 10^{-8}\Omega\text{m} \cdot 5.00 \cdot 10^{-6}\text{m}^2 \cdot (0.07\text{m} + 0.07\text{m})}}{3.3\text{V}}$$
$$= 10.5 \cdot 10^{-9}\text{m}^2$$

Leading to the following wire diameter and number of turns:

$$D_{wire} = 2 \cdot \sqrt{\frac{A_{wire}}{\pi}} = 2 \cdot \sqrt{\frac{7.2 \cdot 10^{-9}\text{m}^2}{\pi}} = 116\mu\text{m}$$
$$A_{wire}n = A_{eff.coil} \Leftrightarrow n = \frac{A_{eff.coil}}{A_{wire}} = \frac{5.00 \cdot 10^{-6}}{7.2 \cdot 10^{-9}\text{m}^2} = 475 \text{ turns}$$

weight

With a density of copper of 8960kg/m^3 the weight of one coil will be:

$$\pi \cdot (0.5 \cdot 11610^{-6}\text{m})^2 \cdot 475 \cdot 6.510^{-2}\text{m} \cdot 8960 = 11.7g \tag{7.2.5}$$

which gives a total weight of 35.1g. The present weight budget for the satellite has dedicated 35g as expected weight for the coils, so the weight is just at the limit.

Construction

At present the idea is that the coils are wound directly on the sides of the satellite, laid into grooves in the structure. Finite element calculations and vibration and shock test are needed to evaluate the need for encapsulating the coils in epoxy. Special two layer enameled wire where the outer layer melts during the winding process could also be an option to mechanically stabilise the coils.

Driver circuit

The driver circuit applies a current trough either direction of the coil by changing the voltage of $M1$ and $M2$ (PGND or V_{cc}). A pulse width modulated(PWM) signal from the control system drives the change in the voltage. The voltage at $M1$ is inverted with respect to the voltage at $M2$ due to the XOR which also works as an enable switch. The total current is determined by the duty cycle of the PWM signal. The selected configuration gives a total of three PWM signals to drive all magnetotorquers.

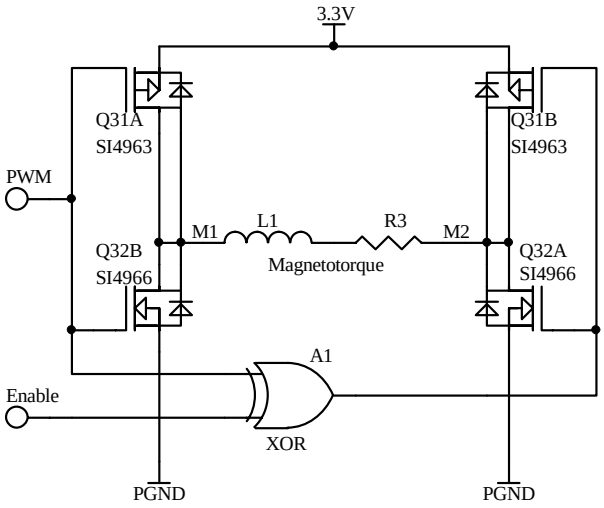


Figure 7.1: Schematic of the magnetotorquer and the driver

7.3 Attitude control

The satellite is equipped with magnetotorquers as the only attitude actuators. The torque generated by the interaction between the generated field and the Earth magnetic field, will always be perpendicular to the Earth magnetic field. This means that the attitude control system has to take into consideration, that it is not possible to control all three axis at the same time. It is only possible to control two directions at a time, namely roll and pitch at the poles and pitch and yaw at the equator. This is due to the direction of the Earth magnetic field and requires that the control problem is considered as a nonlinear and time-varying problem [11].

The requirements to the control system is very relaxed, first it is necessary to detumble the satellite and then a pointing in the range of $\pm 10^{c}irc$ is considered reasonable.

7.3.1 Controlling the Simple Linearised Model

To adequately control the pointing of the satellite system given in (4.2.7) it is necessary to have two measurements for each axis, an angle and an angular velocity.

$$\frac{\Theta(s)}{\boldsymbol{N}(s)} = \frac{1}{I_s} \frac{1}{s^2 + 2 \cdot \omega_{bw} \cdot \xi \cdot s + \omega_{bw}^2} \tag{7.3.1}$$

Where ω_{bw} is the satellite systems bandwidth, I_s is the satellite inertia and ξ is the damping of the system. A simple PD controller can be design with

the controller parameters as:

$$K_p = I_s \omega_{bw}^2 \tag{7.3.2}$$

$$K_d = 2 \cdot \eta I_S \cdot \sqrt{\frac{K_p}{I_s}} \tag{7.3.3}$$

If the bandwidth is set to 0.1 rad/sek and the damping to 1 the controller will have the following gain matrices:

$$\boldsymbol{K_p} = \begin{bmatrix} 0.710^{-5} & 0.0000 & 0.0000 \\ 0.0000 & 0.710^{-5} & 0.0000 \\ 0.0000 & 0.0000 & 0.810^{-5} \end{bmatrix} \tag{7.3.4}$$

$$\boldsymbol{K_d} = \begin{bmatrix} 0.09810^{-6} & 0.0000 & 0.0000 \\ 0.0000 & 0.09810^{-6} & 0.0000 \\ 0.0000 & 0.0000 & 0.12810^{-6} \end{bmatrix} \tag{7.3.5}$$

This controller is not feasible in this context as it is still not possible to control all three directions simultaneously, but the above design can give an idea of the magnitude of the controller gain.

7.3.2 B-dot control

Detumbling the satellite is a necessary requirements in order to make the payloads work properly and is therefore a direct criteria of success for the satellite.

A simple control strategy called B-dot control, based on rate measurements of the local magnetic field by the magnetometer is presented in [5]. B-dot control minimises the satellites angular rate change in relation to the local geomagnetic field. The controller consist of a rate feedback and a constant term to make the satellite act like a permanent magnet where the oscillations are damped by the feedback term. The feedback signal expressed as the required magnetic moment is:

$$\boldsymbol{m}_c = -k\dot{\boldsymbol{B}} - m_{const} \tag{7.3.6}$$

Where k is a constant gain and m_{const} is a constant vector.

This simple controller uses the magnetometer for rate measurements and does not require any knowledge of the field or the orbit. The control parameters have been found via simulations starting with the magnitudes found in the previous section.

Simulations resulted in the following controller parameters:

$$Herskalstaafundneregulatorparametre XXX \tag{7.3.7}$$

It is important to notice that the use of this controller, can affect the use of the magnetometer. If one of the satellite axis is constantly aligned with the local magnetic field, the magnetometer will not be able to measure the rate around the axis, this means that it is impossible to determine the attitude of the satellite from the magnetometer alone. The control algorithm itself is not affected by this, but the payloads require attitude determination and it is therefore necessary for the sunsensors and the sun position model to work in order to fulfill the payload requirements.

7.3.3 Three axis magnetic control

The Ørsted satellite shows the feasibility for purely magnetic three axis control and is described in detail [5]. To design and verify this type of controller in depth knowledge of nonlinear control theory is needed, and it is beyond the scope of this project to investigate this further, but it would be an interesting master project. This section contains the performed simulations on and a description of a SimuLink model of **DTUsat1**. The B-dot controller analysed in section 7.3.2 is implemented in the model and tested.

7.4 Simulation model

The simulink model consist of the environment models, a controller and a model of the satellite dynamics and kinematics.

7.4.1 Coordinate transformation

All coordinate transformation in the SimuLink model are made using direction cosines expressed with quaternion notation as in [9, app. A.4.3].

7.4.2 SGP model

The implemented SGP algorithm is based on the FORTRAN version of the algorithm seen in [4]. The major difference between the MatLab and the FORTRAN implementation is that the MatLab version is given a time vector and calculates the orbit parameters for the entire vector. The FORTRAN version is given a single time and calculates the orbit parameters for that time only. The MatLab implementation of the SGP algorithm seen in app. A.1.1, takes a modified TLE as input argument. The raw TLE is read from a text file by the MatLab function loadTLE.m that can be seen in app. A.1.2.

This function transforms the units in the TLE, eg. it transform from degrees to radians and returns a modified TLE that can be read by sgp.m.

The resulting positions and velocities from the SGP is given in Earth radii and Earth radii per minute, this is transformed to kilometers and kilometers per second in the runSGP MatLab script that can be seen in app. A.1.4.

The matlab implementation was verified using the TLE example in [4], the result matches the results from the FORTRAN implementation with four digits accuracy.

7.4.3 Magnetic field model

The IGRF model is imported as a C function directly in SimuLink. The model uses ecef coordinates internally which means that the inertial coordinates are transformed to ecef coordinates, the field is calculated and the result is then transformed back to the inertial frame. The transformation from inertial coordinates to ecef coordinates are quite simple, the ecef coordinates rotates one revolution each sidereal day around the Earth spin axis. The inertial frames z-axis is aligned with the spin axis, so the transformation is a pure rotation of the inertial frame around the z-axis. The SimuLinkmodel of the IGRF model with the coordinate transformations can be seen on 7.2

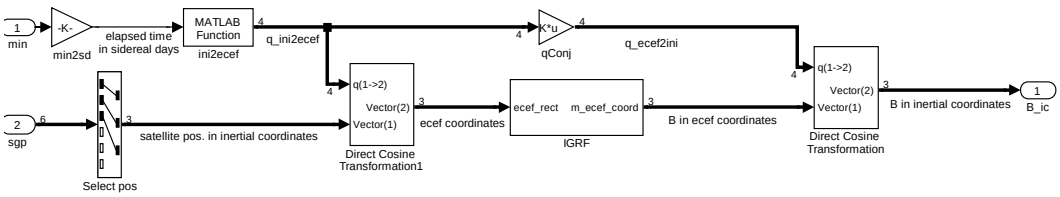


Figure 7.2: Simulink implementation of the IGRF model with coordinate tranformations to and from ecef coordinates

7.4.4 Sun position model

The sun position model was not used during the simulations because of the selected B-dot controller, that made it superfluous since it only uses the magnetometer. Before this was realised the sun model was already implemented in MatLab and can be seen in appendix A.2.1.

7.4.5 Satellite dynamics

The satellite dynamics described by (4.2.1) can be seen on figure 7.3

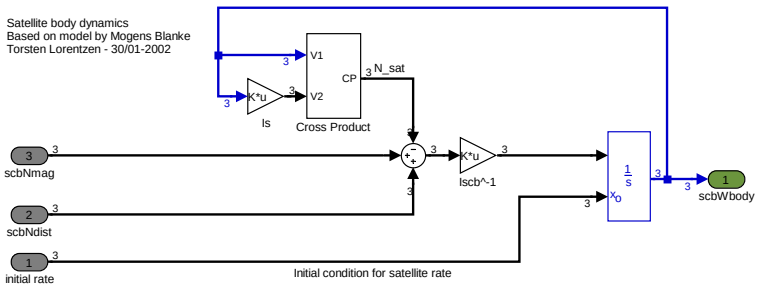


Figure 7.3: The satellite dynamic equation as SimuLink diagram

In order to verify the theoretic controllers, a non-linear satellite model is constructed in SimuLink. The model contains the implemented SGP model, the IGRF magnetic field model and a model of the satellite dynamics.

Chapter 8

Future considerations

8.1 DTUsat1: What is missing

What is needed to finish the **DTUsat1** project

8.2 past DTUsat1

Chapter 9

Conclusion

The work described in this report consisted in developing environmental models for a satellite orbit and investigate and design an attitude determination and control system.

9.0.1 Environment models

The environmental models consisting of a SGP model, that describe the satellite orbit, an IGRF model that describes the geomagnetic field and a sun position model where all implemented in MatLab.

The SGP model was evaluated by computing the orbit given the test example in [4]. The result matched the FORTRAN version with four digits accuracy.

The IGRF model was generously set to our disposition by the Ørsted team implemented in C. Simulations showed a good contingency with general knowledge of the geomagnetic field, the south atlantic anomaly and the minimum and maximum fields fits nicely to values calculated by the tool at [1].

The sun position model is implemented in MatLab, but is not thoroughly tested due to the fact that it is not used in the simulations. Comparison with simulations performed using spenvis or satellite toolkit, can be used to verify the model.

9.0.2 Attitude determination

Appendix A

Environment models implementation

A.1 SGP matlab implementation

A.1.1 sgp.m

```
% - - - - - - - - - - - - - - - - - - - - - - - - - -
% This code is a matlab implementation of the SGP Fortran code in the
% Startrack report #3.
%
% The matlab code was derived mainly by porting and editing the
% FORTRAN routines in NORAD's Spacetrack report #3.
% According to a statement in that report, the document is free
% of copyrights and open to unlimited public distribution.
%
% This matlab code is distributed under the same conditions
%
%
% Created 15/10-2001 by
% Klaus Krogsgaard & Torsten Lorentzen
% - - - - - - - - - - - - - - - - - - - - - - - - - -
%
% SGP model for predicting orbit position.
%
function [R,dR] = sgp(modTLE,dT)

% - - - - - - - - - - - - - - - - - - - - - - - - - -
% Constants given by the modified NORAD TLE (all at epoch)
% - - - - - - - - - - - - - - - - - - - - - - - - - -

t0 = modTLE(1);

dn0 = modTLE(2); % first time derivative of the mean motion

ddn0 = modTLE(3); % second time derivative of mean motion

Bstar = modTLE(4); % Drag parameter

i0 = modTLE(5); % inclination
```

```
Ohm0 = modTLE(6); % right ascension of the ascending node

e0 = modTLE(7); % eccentricity

w0 = modTLE(8); % argument of perigee

M0 = modTLE(9); % mean anormaly

n0 = modTLE(10); % mean motion

revNo = modTLE(11); % Number of revolutions before epoch
% - - - - - - - - - - - - - - - - - - - - - - - - - -
% Other constants
% - - - - - - - - - - - - - - - - - - - - - - - - - -
GM = 398603 * 10^9;
er = 6378.135;
ke = 0.0743669161; %converted to er/min ^3/2
% aE = 6378160; % equatorial radius of the Earth
aE = 1;% give result in Earth radii
min_per_day = 1440;
sec_per_day = 86400;
km_per_er = er;


J2 = 5.413080*10^-4 * 2; % second gravitational zonal harmonic of the Earth

J3 = -0.253881*10^-5; % third gravitational zonal harmonic of the Earth

J4 = -0.62098875*10^-6 * 8/3; % fourth gravitational zonal harmonic of the Earth


% - - - - - - - - - - - - - - - - - - - - - - - - - -
% Local constants
% - - - - - - - - - - - - - - - - - - - - - - - - - -
a1 = (ke/n0)^(2/3);
delta1 = 3/4 * J2 * (aE/a1)^2 * (3*((cos(i0))^2)-1)/((1-e0^2)^(3/2));
a0 = a1 * (1 - 1/3*delta1 - delta1^2 - 134/81*delta1^3);
p0 = a0 * (1 - e0^2);
q0 = a0 * (1-e0);
L0 = M0 + w0 + Ohm0;
dOhm = - (3/2) * J2 * (aE/p0)^2 * n0 * cos(i0);
dw = (3/4) * J2 * (aE/p0)^2 * n0 * (5*(cos(i0))^2-1);

% - - - - - - - - - - - - - - - - - - - - - - - - - -
% Secular effects of drag and gravitation
% - - - - - - - - - - - - - - - - - - - - - - - - - -
a = a0 * ( n0 ./ (n0 + 2*(dn0/2)*(dT) + 3*(ddn0/6)*dT.^2)).^(2/3); % vector
e = zeros(1,length(dT)); % vector
for i=1:length(a)
if ( a(i)>q0 )
e(i) = 1 - q0/a(i);
else
e(i) = 10^-6;
end;
end;

p = a .* (1-e.^2); % vector
OhmS0 = Ohm0 + dOhm * dT; % vector
wS0 = w0 + dw * (dT); % vector
Ls = mod(L0 + (n0 + dw + dOhm)*(dT) + dn0/2 * (dT).^2 + ddn0/6 * (dT).^3,2*pi); % vector
```

```
% - - - - - - - - - - - - - - - - - - - - - - - - - -
% Long period periodic effects
% - - - - - - - - - - - - - - - - - - - - - - - - - -
axNSL = e .* cos(wS0); % vector
ayNSL = e .* sin(wS0) - 1/2 * J3/J2 * aE./p * sin(i0); % vector
L = mod(Ls - 1/4 * J3/J2 * aE./p .* axNSL * sin(i0) * (3+5*cos(i0))/(1+cos(i0)),2*pi); % vector

% - - - - - - - - - - - - - - - - - - - - - - - - - -
% Iteration
% - - - - - - - - - - - - - - - - - - - - - - - - - -
tol = 10^-8;
U = mod(L - OhmS0,2*pi); % vector
Ew1 = U; % vector
Ew2 = Ew1; % vector
dEw = (U - ayNSL.*cos(Ew1) + axNSL.*sin(Ew1) - Ew1) ./ (-ayNSL.*sin(Ew1) - axNSL.*cos(Ew1) + 1); % vector
for i=1:length(dEw)
if abs(dEw(i)) > 1
Ew2(i) = Ew1(i) + sign(dEw(i)); %vector
else
Ew2(i) = Ew1(i) + dEw(i);
end;
end;
for i=1:length(Ew1)
while abs(dEw(i))>tol
Ew1(i) = Ew2(i);
dEw(i) = (U(i) - ayNSL(i).*cos(Ew1(i)) + axNSL(i).*sin(Ew1(i)) - Ew1(i)) ./ ...
(-ayNSL(i).*sin(Ew1(i)) - axNSL(i).*cos(Ew1(i)) + 1);

if abs(dEw(i)) > 1
Ew2(i) = Ew1(i) + sign(dEw(i)); %vector
else
Ew2(i) = Ew1(i) + dEw(i);
end;
end;
end;

ecosE = axNSL.*cos(Ew2) + ayNSL.*sin(Ew2); % vector
esinE = axNSL.*sin(Ew2) - ayNSL.*cos(Ew2); % vector
SQeL = axNSL.^2 + ayNSL.^2; % vector
pL = a.*(1 - SQeL); % vector
r = a.*(1 - ecosE); % vector
dr = ke * sqrt(a)./r .* esinE; % vector
rdv = ke * sqrt(pL)./r; % vector
sinu = a./r .* (sin(Ew2) - ayNSL - axNSL.*esinE./(1+sqrt(1-SQeL))); % vector
cosu = a./r .* (cos(Ew2) - axNSL + ayNSL.*esinE./(1+sqrt(1-SQeL))); % vectro
u = mod(atan2( sinu,cosu ),2*pi); % vector

% - - - - - - - - - - - - - - - - - - - - - - - - - -
% Short term perturbations
% - - - - - - - - - - - - - - - - - - - - - - - - - -
rk = r + 1/4 * J2 * aE^2./pL * (sin(i0))^2 .* cos(2*u); % vector
uk = u - 1/8 * J2 * (aE./pL).^2 * (7 * (cos(i0))^2 - 1) .* sin(2*u); % vector
Ohmk = OhmS0 + 3/4 * J2 * (aE./pL).^2 * cos(i0) .* sin(2*u); % vector
ik = i0 + 3/4 * J2 * (aE./pL).^2 * sin(i0) * cos(i0) .* cos(2*u); % vector


% - - - - - - - - - - - - - - - - - - - - - - - - - -
% Unit orientation vectors
% - - - - - - - - - - - - - - - - - - - - - - - - - -
R = zeros(length(uk),3); % vector
dR = zeros(length(uk),3); % vector
for i=1:length(uk)
```

```
M_vec = [ -sin(Ohmk(i))*cos(ik(i)) cos(Ohmk(i))*cos(ik(i)) sin(ik(i)) ]; % vector
N_vec = [ cos(Ohmk(i)) sin(Ohmk(i)) 0 ]; % vector
U_vec = M_vec * sin(uk(i)) + N_vec * cos(uk(i)); % vector
V_vec = M_vec * cos(uk(i)) - N_vec * sin(uk(i)); % vector

% - - - - - - - - - - - - - - - - - - - - - - - - - -
% Position and velocity
% - - - - - - - - - - - - - - - - - - - - - - - - - -
R(i,:) = rk(i) * U_vec; % vector
dR(i,:) = dr(i) * U_vec + rdv(i) * V_vec; % vector
end;
% - - - - - - - - - - - - - - - - - - - - - - - - - -
% Transforming position to Cartesian coordinates in kilometers and velocity to kilometers/sec
% - - - - - - - - - - - - - - - - - - - - - - - - - -
R = R * km_per_er /aE;
dR = dR * km_per_er / aE * min_per_day / sec_per_day;
```

A.1.2 loadTLE.m

```
%------------------------------------------------------------%
% Loads NORAD TLE file and converts the data to the
% units rad and minutes.
%
% Performs approx. 98% syntax check of the file. The onely
% thing thatr is not checked is whether there is a correct
% number of spaces in the file. However this should be
% unnecessary since the function should output an error on
% on all possible data failures.
%
% The converison part of the function can be excluded. Then
% it is a 'clean' loader.
%
% -1 is returned in ret if an error has occured.
%
% Description of the different returned parameter is given
% below in the source code when they are extracted.
%
% 09/10-2001 Martin Pedersen
%------------------------------------------------------------%
function [TLE, satName, class, IDpiece] = loadTLE(filename)


%-------------------------------------%
% Initialise return variables
%-------------------------------------%
ret = 0; satName = 0; satID1 = 0; class = 0; IDyear = 0; IDlaunchNO = 0; IDpiece = 0;
EpYear = 0; EpJD = 0; EpTime = 0; SGPdragp1 = 0; SGPdragp2 = 0; SGP4dragp = 0;
Eph = 0; ElementNo = 0; satID2 = 0; orbInc = 0; rghtAsc = 0; orbEcc = 0; argPer = 0;
meanAno = 0; meanMo = 0; revNo = 0; checkZ2 = 0;



%-------------------------------------%
% Load file
%-------------------------------------%
fid = fopen(filename, 'r'); %Open file
line0 = fgetl(fid);
line1 = fgetl(fid);
```

```
line2 = fgetl(fid);
fclose(fid);

if (line0==-1)
disp('Not a valid TLF file (line 1).');
ret=-1;
return;
elseif (line1==-1)
disp('Not a valid TLF file (line 2).');
ret=-1;
return;
elseif (line2==-1)
disp('Not a valid TLF file (line 3).');
ret=-1;
return;
elseif (length(line1)>69)
disp('TLE line 1 too long.');
ret=-1;
return;
elseif (length(line2)>69)
disp('TLE line 2 too long.');
ret=-1;
return;
elseif (length(line1)<69)
disp('TLE line 1 too short.');
ret=-1;
return;
elseif (length(line2)<69)
disp('TLE line 2 too short.');
ret=-1;
return;
elseif (line1(1)~='1')
disp('TLE line 1 not labeled correct.');
ret=-1;
return;
elseif (line2(1)~='2')
disp('TLE line 2 not labeled correct.');
ret=-1;
return;
end



%--------------------------------------%
% Get the individual elements
%  - See also the TLE sepecification
%--------------------------------------%
% Name of satellite
satName = line0;
% LINE 1:
% Satellite ID number (NNNNN)
satID1 = readPart(line1(3:7),'%d');
% Security classification (U=Unclassified, 1 char)
class = readPart(line1(8),'%c');
% International Designator: last two digits of launch year (YY)
IDyear = readPart(line1(10:11),'%d');
% --,,--: Launch number of the year (NNN)
IDlaunchNO = readPart(line1(12:14),'%d');
% --,,--: Piece of the launch (NNN)
IDpiece = readPartV(line1(15:17),'%s',3);
% Epoch year (YY)
EpYear = readPart(line1(19:20),'%d');
```

```
% Epoch Julian day (DDD)
EpJD = readPart(line1(21:23),'%d');
% Epoch time; fraction of one day (.FFFFFFFF)
EpTime = readPart(line1(24:32),'%f');
% SGP drag parameter (<|1|) (.NNNNNNNN)
sign = readPart(line1(34),'%c');
SGPdragp1 = readPart(line1(35:43),'%f');
if (sign=='-')
SGPdragp1 = -1*SGPdragp1;
end
% SGP drag2 parameter ((.)NNNNN-N (last 10's exp))
sign = readPart(line1(45),'%c');
[SGPdragp2, len] = readPart(line1(46:50),'%d');
exponent = -1*readPart(line1(52),'%d');
SGPdragp2 = SGPdragp2*10^(exponent-len);
if (sign=='-')
SGPdragp2 = -1*SGPdragp2;
end
% SGP4 drag parameter (NNNNN-N (last 10's exp))
sign = readPart(line1(54),'%c');
[SGP4dragp, len] = readPart(line1(55:59),'%d');
exponent = -1*readPart(line1(61),'%d');
SGP4dragp = SGP4dragp*10^(exponent-len);
  if (sign=='-')
SGP4dragp = -1*SGP4dragp;
end
% Ephemeris type (N)
Eph = readPart(line1(63),'%d');
% Element sat number (NNNN)
ElementNo = readPart(line1(65:68),'%d');
% Check sum for line one (N)
checkZ1 = readPart(line1(69),'%d');
% LINE 2:
% Satellite ID number (NNNNN)
satID2 = readPart(line2(3:7),'%d');
% Orbital inclination (NNN.NNNN)
orbInc = readPart(line2(9:16),'%f');
% Right Ascension (NNN.NNNN)
rghtAsc = readPart(line2(18:25),'%f');
% Orbital Eccentricity (NNNNNNN)
orbEcc = readPart(line2(27:33),'%d')/10^7;
% Argument of perigee (NNN.NNNN)
argPer = readPart(line2(35:42),'%f');
% Mean Anomaly (NNN.NNNN)
meanAno = readPart(line2(44:51),'%f');
% Mean Motion (NN.NNNNNNNN)
meanMo = readPart(line2(53:63),'%f');
% Revolution number (NNNNN)
revNo = readPart(line2(64:68),'%d');
% Check sum for line two (N)
checkZ2 = readPart(line2(69),'%d');


%-------------------------------------%
% Check checksum
%-------------------------------------%
if (checkChkSum( line1(1:(length(line1)-1)) )~=checkZ1)
disp('Incorrect check sum for TLE line 1.');
ret = -1;
return;
elseif (checkChkSum( line2(1:(length(line2)-1)) )~=checkZ2)
disp('Incorrect check sum for TLE line 2.');
```

```
ret = -1;
return;
end;


%-------------------------------------%
% Return loaded values in vector
%-------------------------------------%
TLE = [ret satID1 IDyear  IDlaunchNO...
EpYear  EpJD  EpTime  SGPdragp1 SGPdragp2 SGP4dragp Eph...
ElementNo satID2 orbInc rghtAsc orbEcc argPer meanAno...
meanMo revNo];
```

A.1.3 loadTLE.m

```
%-------------------------------------%
% Convert to rad and min
%-------------------------------------%

%  - Constants given by the NORAD TLE (all at epoch)

function [modTLE] = convertTLE(TLE)

meanMo = TLE(19);
orbEcc = TLE(16);
orbInc = TLE(14);
meanAno = TLE(18);
argPer = TLE(17);
rghtAsc = TLE(15);
SGPdragp1 = TLE(8);
SGPdragp2 = TLE(9);
SGP4dragp = TLE(10);
EpYear = TLE(5);
EpJD = TLE(6);
EpTime = TLE(7);
revNo = TLE(20);
n0 = 2*pi*meanMo/1440; % Mean motion (rad/min)
e0 = orbEcc; % Eccentricity (0.0<=e0>=1.0)
i0 = pi*orbInc/180; % Inclination (rad)
M0 = pi*meanAno/180; % Mean anomaly (rad)
w0 = pi*argPer/180;  % Argument of perigee (rad)
Ohm0 = pi*rghtAsc/180; % Right ascension of the ascending node (rad)
dn0 = 2*2*pi*SGPdragp1/(1440^2); % Rirst time derivative of mean motion(rad/min^2)
ddn0 = 6*2*pi*SGPdragp2/(1440^3); % Second time derivative of mean motion (rad/min^3)
Bstar = SGP4dragp; % SGP4 type drag coefficient

% Mean year (400 year period) in days:
meanYearDays = (400*365 + 4 * (100/4 - 1) + 1) / 400;

% Time of epoch (since y2k)
t0 = (EpYear*meanYearDays + EpJD + EpTime)*1440;


modTLE = [t0 dn0 ddn0 Bstar i0 Ohm0 e0 w0 M0 n0 revNo];
```

A.1.4 init.m

```
clear;
%--------------------------
% prints latex table if toLatex = 1
%--------------------------
toLatex = 0;

TLE = loadTLE('norad.tle');
modTLE = convertTLE(TLE);
global dT

%dT = [0 360 720 1080 1440];
dT = 0:10:1000;
[R dR] = sgp(modTLE,dT);
R
dR

%disp(sprintf('Time \t\t\tposition\t\t\t\t\tvelocity'))
%disp(sprintf('\t     x\t\t     y\t\t      z\t\t    dx\t\t    dy\t\t    dz'))
%for i= 1:length(dT),
% disp(sprintf('%d\t%f\t%f\t%f\t%f\t%f\t%f',dT(i),R(i,:),dR(i,:)))
%end


if (toLatex==1),
disp(sprintf('Time\t&\t\t\t&\t   position\t&\t\t\t&\t\t\t&\tvelocity\t&\\\\'))
disp(sprintf('&\t\tx &\t\ty &\t\tz &\t\tdx & dy\t\t& dz\\\\'))
for i= 1:length(dT),
disp(sprintf('%d & %f & %f & %f & %f & %f & %f\\\\'...
,dT(i),R(i,:),dR(i,:)))
end
end
```

A.1.5 readPart.m

```
%---------------------------------------------------------%
% Reads a variable of ANSI C-type type from string.
%
% Prints 'ERROR not a TLE file (failed on: string |
% with type type): ERRMSG'
% if there is an error. ERRMSG is sscanf's error message.
%
% An error has occured if ERRMSG!=''
%
% type e.g.: '%d', '%f', '%s'
%
% This function is used as an assisting function in:
% loadTLF.m
%
% 09/10-2001 Martin Pedersen
%---------------------------------------------------------%
function [ret, charsRead] = readPart(string, type)

[ret, rlen, ERRMSG, charsRead] = sscanf(string, type);

if (~isempty(ERRMSG))
disp(sprintf('ERROR not a TLE file (failed on: %s | with type: %s): %s', string, type, ERRMSG));
ret = -1;
return;
```

```
end;

charsRead = charsRead - 1;

if (isempty(ret))
ret = 0;
end

return;
```

A.1.6 checkChkSum.m

```
%-----------------------------------------------------------%
% Check whether or not string has check sum Z
%
% The TLE check sum algorithm is used.
%
% This function is used as an assisting function in:
% loadTLF.m
%
% 09/10-2001 Martin Pedersen
%-----------------------------------------------------------%
function cs = checkChkSum(string)

cs = 0;

for i = 1:length(string)
if (~isempty(str2num(string(i))))
cs = cs + str2num(string(i));
elseif (string(i)=='-')
cs = cs + 1;
end
end

cs = mod(cs,10);

return;
```

A.2 Sun prediction matlab implementation

A.2.1 sun.m

```
function [xe, ye, ze] = sun(y,m,D,UT)

d = 367*y - 7 * ( y + (m+9)/12 ) / 4 + 275*m/9 + D - 730530;

d = d + UT/24.0;


% Orbital elements for the sun
w = pi/180*(282.9404 + 4.70935E-5 * d)
e = (0.016709 - 1.151E-9 * d)
M = pi/180*(356.0470 + 0.9856002585 * d)
ecl = (23.4393 - 3.563E-7 * d);
E = M + e * sin(M) * ( 1.0 + e * cos(M) );

xv = cos(E) - e;
```

```
yv = sqrt(1.0 - e*e) * sin(E);

v = atan2( yv, xv );
r = sqrt( xv*xv + yv*yv );
r = 1.495e8*r; % convert to km
lonsun = v + w;

xs = r * cos(lonsun);
ys = r * sin(lonsun);

xe = xs;
ye = ys * cos(ecl);
ze = ys * sin(ecl);

RA  = atan2( ye, xe );
Dec = atan2( ze, sqrt(xe*xe+ye*ye) );
```

A.3 IGRF model matlab implementation

A.3.1 magmod.m

```
%
% Print contour plot for B-field strength in height radius
%
% Input:  radius in km, lat (-90 to 90 deg.), longitude (0 to 360 deg.)
% Output: in cartesian coord. (ecef)
%
% 30/10-2001 Jan Hales
% 02/11-2001 Martin Pedersen
%
function bv=magmod(r,lat,long)
sim('magfield', 0, simset,[0, r lat long]);
bv=yout;
```

A.3.2 IGRF2000 coefficients

A.4 Tabularized magnetic field generation)

```
% getBvec create a table of Bfield vectors indexed by x,y,z coordinates
% Torsten Lorentzen - december 2001


% Get B-field vector
b = zeros(181,360,3);
%Mean radius + altitude (it's a circular orbit)
radius = 6371.0 + 600;
h = waitbar(0,'Calculating B-field vector')
for lat=-90:1:90
for long=-179:1:180
b(lat+91, long+180,:) = getB(radius, lat, long);
waitbar((long+91)*(lat+180)/64800,h)
end;
end;
close(h)
save bvec.mat b
```

Bibliography

[1] NASA Goddard Space Flight Center. Dgrf/igrf geomagnetic field model 1945-2005, 1999. http://nssdc.gsfc.nasa.gov/space/model/models/igrf.html.

[2] Jan Hales, Martin Pedersen, and Klaus Krogsgaard. Satellite construction. Technical report, Department of Automation, Ørsted.DTU, 2002. Midterm Project.

[3] Jan Hales, Martin Pedersen, Klaus Krogsgaard, and Torsten Lorentzen. Attitude control and determination system for dtusat1. Technical report, Department of Automation, Ørsted.DTU, 2001. Requirement specification.

[4] Felix R. Hoots and Ronald L. Roehrich. Spacetrack report no. 3. Technical report, Aerospace Defense Center, 1980. http://celestrack.com/NORAD/documentation/spacetrk.pdf.

[5] Rafa l Wiśniewski. *Satellite Attitude Control Using Only Electromagnetic Actuation*. PhD thesis, 1996.

[6] F. Landis Markley and Daniele Mortari. How to estimate attitude from vector observations. 1999.

[7] International Association of Geomagnetism and Aeronomy. International geomagnetic reference field - epoch 2000, 2000. http://www.ngdc.noaa.gov/IAGA/wg8/igrf2000.html.

[8] Paul Schlyter. Computing planetary positions, 1979. http://hotel04.ausys.se/pausch/comp/ppcomp.html#5.

[9] Marcel J. Sidi. *Spacecraft Dynamics and Control - A Practical Engineering Approach*. Cambridge University Press, 1997.

[10] DTUsat team. The homepage of the dtusat project. www.dtusat.dtu.dk.

[11] Rafal Wiśniewski and Mogens Blanke. Fully magnetic attitude control for spacecraft subject to gravity gradient. *Automatica 35*, 1999.