Master Thesis

Simulation and control toolkit for small satellite projects

(Programovy balik pro simulaci a navrh rizeni malych satelitu)

Adam Śmiałek

February 2020

Abstract

Spacecraft project management calls for division of project lifetime into phases, with specific goals to be fulfilled at the end of each phase. During first few phases a Preliminary Design Review (PDR) has to be conducted, after which top-level hardware design is not to be changed. This thesis describes a process of creating and demonstrates a software framework supporting teams building small satellites - typically CubeSat student projects - during initial phases of conceptual design, mission planning, and selection and sizing of hardware components. The scope of the thesis covers review of available tools for satellite mission and control system design, then it proposes a self-made MATLAB/Simulink toolbox - Spacecraft Control Architecture Rapid Simulator (SCARS) Toolbox, as a open source tool with gentle learning curve and ease of reverse engineering approach. In further parts of the thesis examples of usage are provided, and conclusions and descriptions of problems are presented. In the end, this thesis should not only serve as a description of SCARS toolbox, but also as an insight on how to approach the task of building a small satellite simulation.

Contents

Abbreviations

ADCS Attitude Determination and Control System
AOCS Attitude and Orbit Control Systems
PDR Preliminary Design Review
ESA European Space Agency
NASA National Aeronautics and Space Administration
SCARS Spacecraft Control Architecture Rapid Simulator
SCARS Spacecraft Control Architecture Rapid Simulator
ESTEC European Space Research and Technology Centre
TEC-ECN Guidance, Navigation, and Control Systems Section
LEO Low Earth orbit
MEMS micro-electromechanical systems
A/VRW Angle/Velocity Random Walk
KSP Kerbal Space Program
kRPC Remote Procedure Call Server for KSP
VRML Virtual Reality Markup Language
WWW World Wide Web
STK Systems Tool Kit
REXUS/BEXUS Rocket and Balloon Experiments for University Students programme
GPS Global Positioning System
GNSS global navigation satellite systems
ECEF Earth-Centered, Earth-Fixed
ECI Earth-Centered Inertial
ECR Earth-Centered Rotational
DLR German Aerospace Center
iMQT ISIS Magnetorquer Board
LLA Geodetic Latitude, Longitude, Altitude
IMU Inertial Measurement Unit
NGA National Geospatial Intelligence Agency
NIR Near-infrared
DCM Direction Cosine Matrix
NED North east down
CGP Cold Gas Propulsion
PID Proportional-integral-derivative controller
LQR Linear Quadratic Regulator

1 Introduction

The idea to create a simulation and control toolkit for small satellite projects was a byproduct of work done on IRISC project by the author of this thesis. IRISC, or "InfraRed Imaging of astronomical targets with a Stabilized Camera", was a project realized as a part of Rocket and Balloon Experiments for University Students programme (REXUS/BEXUS) programme. The goal of the IRISC experiment was to obtain images in the Near-infrared (NIR) spectrum from astronomical targets. Possible targets included the Andromeda Galaxy, Pinwheel Galaxy, Iris Nebula, Eagle Nebula and Starfish Cluster. The images were obtained using a highly stabilized telescope with NIR camera mounted on a REXUS/BEXUS balloon. Author's responsibility covered the design of the control subsystem of the experiment. The stabilization was achieved by a gimbal-like system, to obtain high quality images while being on a moving platform[?]. While the design of the control system was not innovative, nevertheless it was a complicated task for a student with no previous practical experience in that field. This has proven to be especially challenging during early stages of experiment design.

The process of effective space-related project management - from the conception of the idea, through production, to disposal - features high costs and often various unpredictable risks. Due to this, a project life cycle is usually divided into distinct phases, allowing for introduction of conducting product reviews within rigid time-frames. An example of such a workflow, adopted by most major agencies such as ESA[2] and NASA[3], is a division of the project life cycle into phases, as it can be seen on Figure 1.1. While the design of a project is often an iterative process, the phases and reviews that conclude them exist as a checkpoints, after which the design of the project is to be unchanged, on a level of details progressing as phases do. For example, as one can see, Phase B is usually ended by the Preliminary Design Review (PDR). In the case of a spacecraft, for the PDR, a major architecture parameters have to be defined, such as volume and weight ramifications, top-level designs of solutions for major requirements have to be presented - for a practical example: for high-resolution Earth observation mission the type of the actuators which fulfills precision requirements has to be chosen.

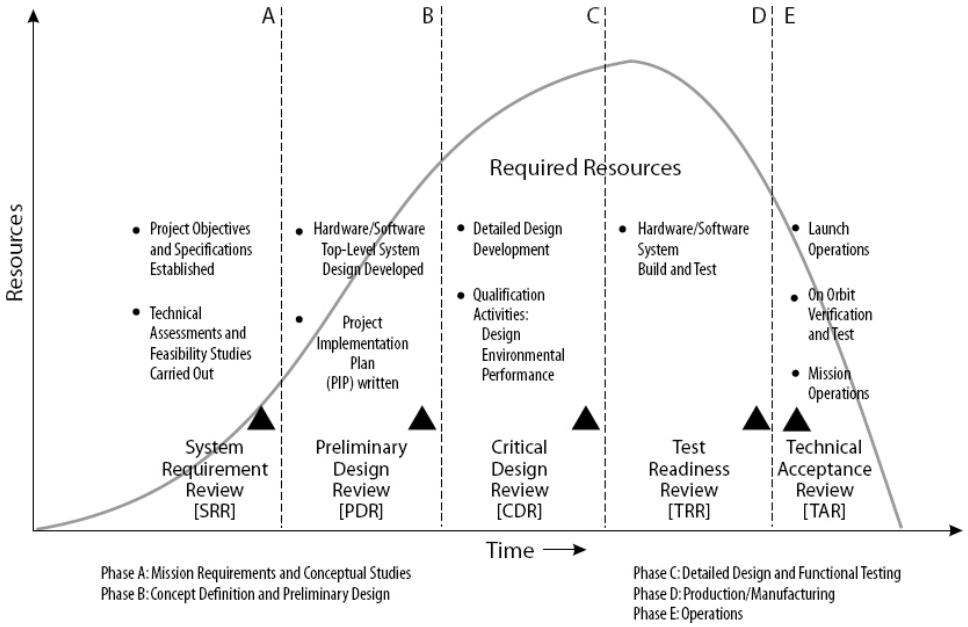


Figure 1.1: Typical space project phases and its life cycle [1]

Taking the experience of how challenging and time-consuming was the process of learning how to produce a reliable simulation of IRISC control system and the knowledge of significance of preliminary design, author decided to produce and publish a simulation, which would be useful for purposes of initial spacecraft design and for beginner engineers to learn how to build a reliable simulation.

1.1 Scope

The thesis covers the process of development of the toolbox for rapid prototyping of satellite's control systems. Chapter 1 describes the aim of this work and discusses the topic of prototyping tools. Chapter 2 goes into detail about the architecture of Spacecraft Control Architecture Rapid Simulator (SCARS), its features and methods of implementation. Also here there are described the ways of connecting SCARS with various visualization tools. Chapter 3 explains the documentation and usage of SCARS, while in Chapter 4 there are examples showing how the toolbox can be used in real life applications. Finally, Chapter 5 discusses the conclusions from the development process and the possibilities for improvements of SCARS.

1.2 Aim

The aim of this thesis work is to build and provide a ready to use open source product - a toolbox for small and low budget satellite projects. The toolbox features allow for a initial design of spacecraft's Attitude Determination and Control System (ADCS), which means that they allow for, i.a. simulation of spacecraft orbit, testing the feasibility of various actuation methods and testing the effectiveness of different control algorithms in given use cases. That software would then allow smaller and inexperienced teams of spacecraft designers to better prepare for design milestones like PDR, when there is not enough time to create a full simulation of their spacecraft ADCS subsystems. Besides the toolbox being a tool for practical use, the thesis also serves as as a review of available solutions, so it can be used by future control engineers as a learning material. The idea is, that some parts of the proposed model could be removed from the model as the students, for the learning purposes, would be tasked with designing a substitution.

For the purposes of later evaluation of how the solutions proposed in this thesis fulfil the goals stated in the preceding paragraph, a following list of objectives was compiled:

- **Conduct a review of existing tools for preliminary spacecraft design**, focusing on mission planning and Attitude and Orbit Control Systems (AOCS) subsystem;
- **Create a spacecraft dynamics and AOCS model**, to be used with minimal set-up;
- **Assemble a library of models**, to be used by other beginner control engineers;
- **Provide a documentation of the toolbox**, explaining not only the purpose and operating principles of individual parts, but the process of using the toolbox to conduct a preliminary design of spacecraft AOCS subsystem;
- **Share the toolbox to be available online**, with principles of open-source software in mind.

Furthermore, the objectives that are set for the design of the toolbox itself are described in Chapter 2.1.

1.3 Already existing tools

The idea for a toolbox allowing for spacecraft mission design and AOCS simulation is not a new one. There are various solutions available, ranging

from very robust commercial software packages to open-source implementations of individual features for use as a part of MATLAB framework.

The aim of this section is to prove that, while the solutions for spacecraft prototyping are available, there is still a need for open-source, easy-to-sue and modify toolbox. Further below is a compiled list of selected software solutions that fit the most objectives stated in subsection 1.2, with included explanation what they are lacking that discussed toolbox should have.

1.3.1 MATLAB CubeSat Simulation Library

CubeSat Simulation Library is a part of Aerospace Blocks created by MathWorks Aerospace Products Team. Using it one can model motion and dynamics of CubeSats and nanosatellites. It provides the most basic features, like the simulation of pre-set attitude scenarios, basic actuators and sensors models and integration with MATLAB's Virtual World visualization tools.

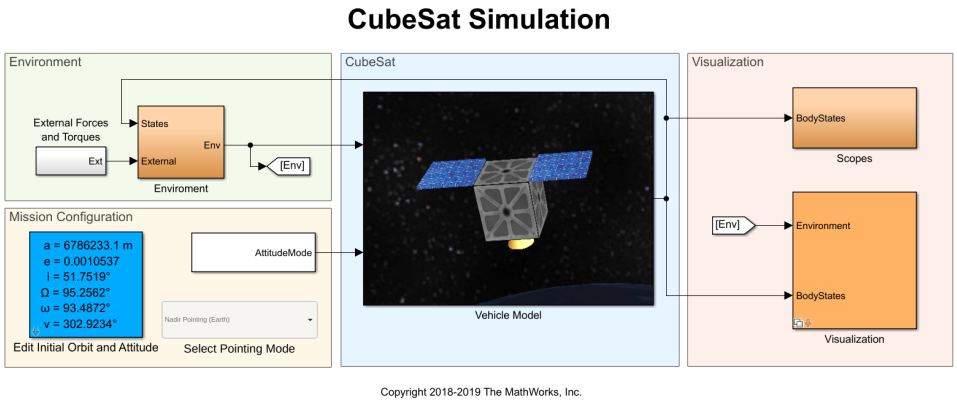


Figure 1.2: Top-level view of the example project of the MATLAB CubeSat Simulation Library

This library, while conceptually most similar to the SCARS, it lacks some functionalities. For example, for actuators, it provides only general models for perfect and second-order actuators. In SCARS, the actuators are full models, which allows not only for reducing the number of layers of abstraction between the user and the simulation, but also for things like calculation of energy expended by the actuator. Also, this toolbox is sparsely documented - while most functionalities are described within their Simulink block masks, there is no comprehensive guide about how to use them in own models[?].

1.3.2 PrincetonSATELLITE Spacecraft Control Toolbox

PrincetonSATELLITE Spacecraft Control Toolbox is a commercial solution for building spacecraft Simulations. It contains over two thousand functions for attitude and orbit dynamics, simulation, estimation, analysis and design. This is the most robust and comprehensive toolbox available, includes online API, well written documentation and additional modules for unique applications like formation flying, fusion propulsion or solar sails. This would be the best choice for most use-cases, yet it is a paid solution and even the cheapest option - CubeSat Edition - may be out of price range for smaller teams[?].

1.3.3 PROPAT Toolbox

PROPAT is is a small set of functions in Matlab to simulate and propagate orbit and attitude of an Earth's satellite, developed by the single person as an open-source toolbox. Several functions allow to transform between orbit and attitude coordinates and for propagation or rigid body attitude. PROPAT contains only MATLAB scripts, which while useful and can be used as a part of the simulation, do not combine into a model of a whole spacecraft's ADCS subsystem[?].

1.3.4 GAST Toolbox

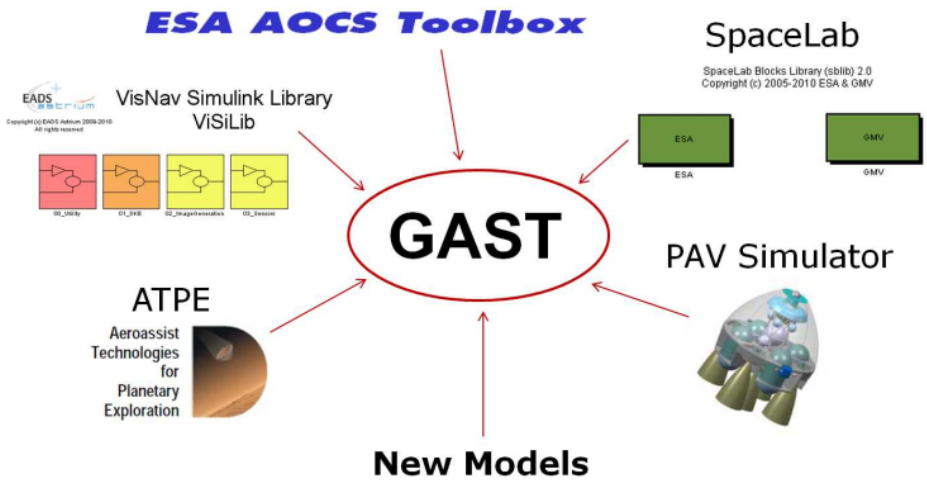


Figure 1.3: Representation of the consolidation of TEC-ECN toolboxes

The GAST toolbox is the result of the consolidation of several toolboxes available in Guidance, Navigation, and Control Systems Section (TEC-ECN) of European Space Research and Technology Centre (ESTEC), such as the AOCS Toolbox, the SpaceLAB library, the ViSiLib library, the ATPE simulator, and the PAV simulator. In addition to consolidating these toolboxes, new models were developed for the GAST toolbox according to the needs of the section. Figure 1.3 shows a pictorial representation of the consolidation of the toolboxes of TEC-ECN. This software was developed in TEC-ECN in 2008, but since it is a product of European Space Agency (ESA), it is not available for use for wider audience[?].

1.3.5 User-created modules available on MathWorks MATLAB Central

As MATLAB is one of the most popular scripting language between engineers, and along with Simulink package it provides tools helpful for simulating mechanical systems, there are many user created modules and packages.

MATLAB Central is a network for asking questions about MATLAB software, discussing solutions and sharing MATLAB and Simulink solutions

and files[?]. On a subsection called File Exchange there are many files available to use within MATLAB framework, some of them relevant to spacecraft design. The most notable examples are listed below.

SAT-LAB

SAT-LAB is a MATLAB-based Graphical User Interface (GUI), developed for simulating and visualizing satellite orbits. The primary purpose of SAT-LAB is to provide a software with a user-friendly interface that can be used for both academic and scientific purposes. While a useful tool, it is only suitable for initial mission planning.[?]

Satellite Orbit Modeling

A collection of MATLAB scripts used for modelling of satellite's perturbed motion with special perturbations approach. While it is very robust, as it can be applied to any problem in celestial mechanics, this module is useful for orbit modeling, not AOCS system design[?].

Smart Nanosatellite Attitude Propagator (SNAP)

The Smart Nanosatellite Attitude Propagator is an attitude propagator for satellites that can be used to analyze the environmental torques affecting a satellite and to design and analyze passive attitude stabilization techniques, such as Passive Magnetic Stabilization, Gravity Gradient Stabilization and Aerodynamic stabilization. This model is the most relevant one for the scope of this thesis, but it lacks possibility to model active attitude stabilization techniques[?].

Satellite Orbits: Models, Methods and Applications

Rather than spacecraft or orbit model, it is a collection of exercises for book *Satellite Orbits: Models, Methods and Applications*. It is interesting from the educational point and refers at least partially to the problem of control system design - mainly GPS sensor. Yet this is not a toolbox by any means[?].

Apollo 11 Moon Landing - 50th Anniversary Model

This example shows how Richard Gran and the other engineers who worked on the Apollo Lunar Module digital autopilot design team could have done it using Simulink, Stateflow, Aerospace Blockset and Simulink 3D Animation if they had been available in 1961. Although it is a very notable example of how MATLAB software family can be used to simulate a whole mission, to

use it for either own spacecraft or educational purposes would require much more reverse engineering and modifications than creating a new model[?].

2 Spacecraft Control Architecture Rapid Simulator (SCARS) Toolbox

This chapter consists of description of the toolbox designed as a part of this thesis work. After the following introduction, in Sections 2.1, 2.2 and 2.3 the objectives of the toolbox and its high level structures are described. After that, one finds theoretical description of satellite mechanics and coordinated systems, with following descriptions of theoretical principles of each major component of the toolbox and their implementation in MATLAB and Simulink software. At the end, methods of visualization of acquired simulations are discussed.

To fullfil the main objective of this thesis, that is to provide a satellite control system prototyping toolbox the community of beginner control engineers, a self-made solution is proposed. This chapter provides the insight into the architecture of Spacecraft Control Architecture Rapid Simulator (SCARS) Toolbox, a software framework created in MATLAB and Simulink. First the main objectives of that solution are stated, then architecture of Spacecraft Control Architecture Rapid Simulator (SCARS) is described, to give the initial description of how the toolbox can be used. In following sections the principles of operation of each major part of the toolbox, and how they were implemented, are presented.

The inputs of SCARS Toolbox - whether used as a parts library and integrated into own project, or as ready-made modular simulation - are parameters of spacecraft hardware, such as for example size of the satellite, trusters operational range, and initial mission parameters, for example time, Keplerian elements or initial body rates. The outputs of the toolbox are performances of each part and simulated behavior of the whole spacecraft, allowing the user to easily test different designs for their satellites.

2.1 Objectives

The toolbox itself covers first two objectives of the the thesis. Following listing further specifies what should be expected of the end product and what features the users should be able to find in SCARS Toolbox:

- A model of orbital dynamics of Earth orbiting satellite;
- Models of most common satellite actuators and sensors and parametrize them, so that the actual hardware can be reproduced in simulation using values from datasheets;

- Modelled sources of environmental forces and torques, modeling most influential sources;
- Several most basic control methods;
- Simulink Custom Library, with all models masked for quick set up;
- Methods of conducting preliminary review of feasibility of used hardware components and control methods;
- Interfaces allowing the user to connect the toolbox with visualization software.

2.2 Choice of software

To fit with the objectives of accessibility and ease of modification MATLAB family of software was chosen. MATLAB is one of the most popular scripting language and with the addition of Simulink software it can become powerful tool with the ability to set up numerical simulations in short time. MATLAB is taught in most technical universities and there is significant number of both courses available online and materials for self-teaching. For one purpose (described in Section 2.12.3) a Python script acting as a dataflow bridge was used, as it was a simplest method to solve a problem described in that chapter. Several other software solutions were used for visualization purposes, with the reasoning described in Section 2.12.

Versatility of MATLAB may be attributed to the number of Add-Ons available for it. SCARS Toolbox uses and requires the following modules:

- Aerospace Toolbox
- Navigation Toolbox
- CubeSat Simulation Library
- Control System Toolbox
- Simulink 3D Animation

2.3 Architecture

SCARS is divided into two parts: 1) Parts Library and 2) Modular Simulation. The Parts Library contains Simulink subsystems, which can be connected to form simulations of various complexity and for multiple scenarios. The other is a Modular Simulation, which can be set up with either MATLAB command line scripts or graphical user interface.

2.3.1 Parts Library

SCARS Parts Library is a ready to use Simulink Custom Library, that is a collection of blocks available to use in Simulink models. All blocks in library downloaded alongside SCARS are parametrized, masked and described to ease the integration of library parts into user simulation. The library is divided into specific sections:

- Satellite Dynamics
- Reference Frames Transformations
- Environment
- Actuators
- Sensors
- Control Algorithms
- Visualization
- Analysis
- Example scenarios

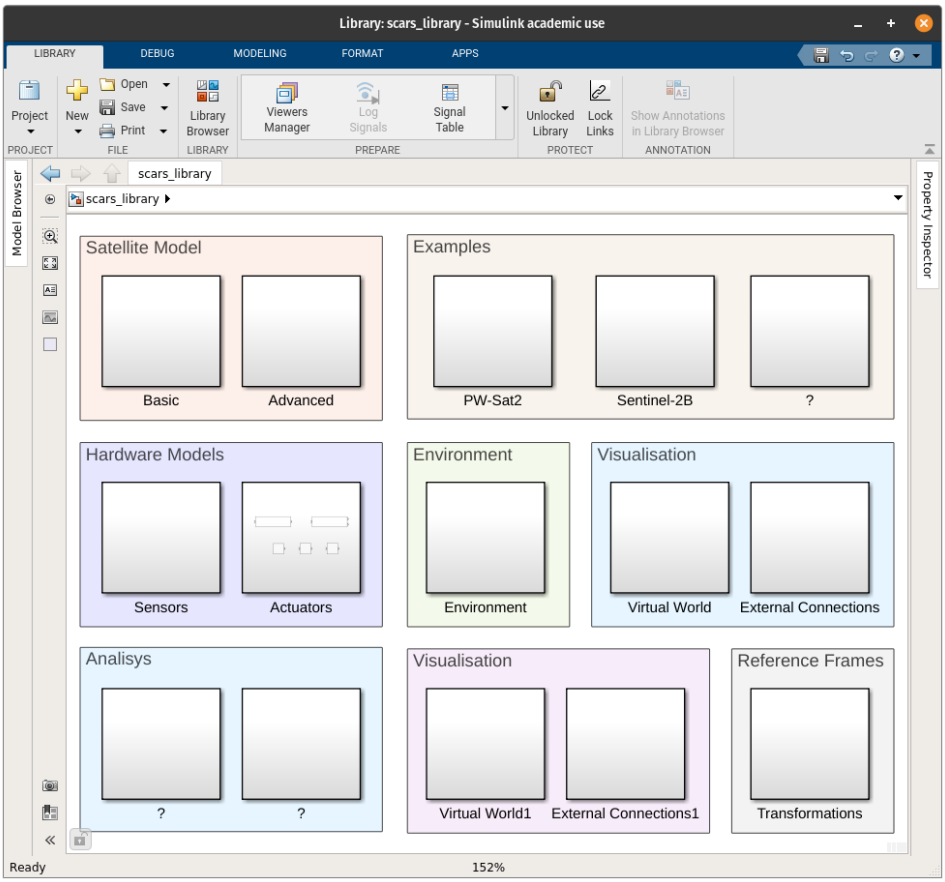


Figure 2.1: SCARS Parts Library screenshot

2.3.2 Modular Simulation

SCARS Modular Simulation is a ready-made simulink model available for setup using prepared scripts and SCARS user interface. The model is a simulation of cube-shaped satellite, which can be set on specified orbit using various initialization methods, such as Keplerian elements in conjunction with Julian date time. (The initialization is further described in Chapter 3). In the same manner, all actuators and sensors available in SCARS library can be chosen. The Modular Simulation makes use of most available blocks, which can be commented out from the model, either by hand or using the user interface, to improve the speed of the simulation.

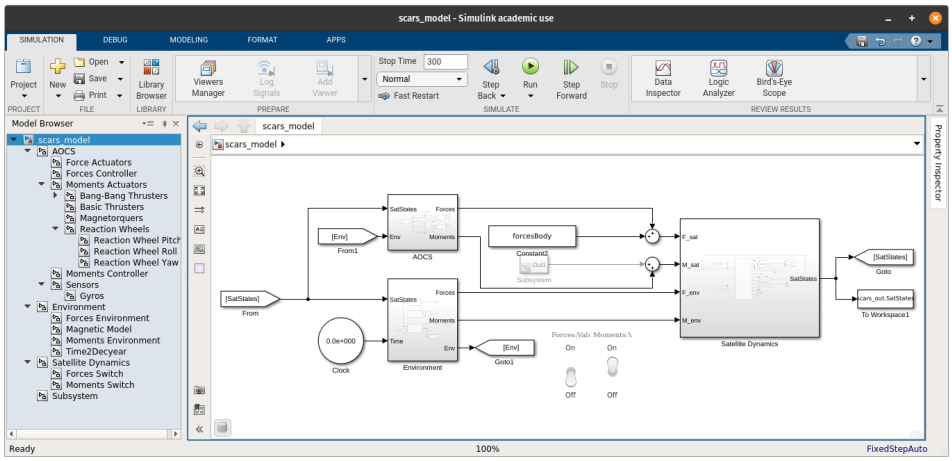


Figure 2.2: SCARS Modular Simulation screenshot

2.4 Orbit dynamics

. . . work in progress. . .

$$\begin{bmatrix} \delta\dot{u} \\ \delta\dot{v} \\ \delta\dot{w} \\ \delta\dot{x} \\ \delta\dot{y} \\ \delta\dot{z} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & -n^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & -n & 2 \\ -n & 0 & 0 & 0 & 0 & 2n^2 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \delta u \\ \delta v \\ \delta w \\ \delta x \\ \delta y \\ \delta z \end{bmatrix} + \begin{bmatrix} T_x/m \\ T_y/m \\ T_z/m \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{1}$$

For small attitude changes of non-spinning spacecraft with respect to inertial frame, equations describing rotational motion can be linearized. The acquired system is then:

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \\ \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & n \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -n & 0 & 0 \end{bmatrix} \begin{bmatrix} p \\ q \\ r \\ \Phi \\ \Theta \\ \Psi \end{bmatrix} + \begin{bmatrix} Q_x/I_x \\ Q_y/I_x \\ Q_z/I_z \\ 0 \\ n \\ 0 \end{bmatrix} \tag{2}$$

Where $n = \sqrt{g/R_E}$ is satellite's orbital angular velocity.

2.5 Spacecraft Mechanics

Spacecraft mechanics are governed by the laws describing the motions of a body under the influence of external and internal forces and torques. Forces acting on a spacecraft influence its translational motion, which in the simplest form can be described as a set of differential equations in a form of $\dot{x} = Ax + Bu$. In this case, x is a state vector built from satellites position vector in three-dimensional Cartesian coordinates and first order derivatives of said vector. For simplified point-on-orbit satellite

One of the goals of this thesis is to create means to design a satellite control system for people without much experience in that field. One of the ways to accomplish that is to eliminate the need to derive complex mathematical models of the systems involved. Nevertheless, there exists a need for such models, for example for LQR control algorithm described in Section 2.11.2. One could try to further develop Equations 2.4 and 2.4 into state-space representation of whole spacecraft system including all actuators and sensors, parametrize it and programmatically modify it, for each configuration, to include only set-up parts. Alternative solution it to use MATLAB Control System Toolbox and its functions to automatically linearize Simulink models. The main advantage of this solution is that it also encompasses all changes from the users to the SCARS Modular Simulation. The process is described in subsection C.

A mathematical model can be used in analytical solutions, but in

A Spacecraft Model block in SCARS is built around Simulink Aerospace Blockset 6DOF ECEF (Quaternion) block.

. . . finish the explanation. . .

2.6 Reference Frames

To find the states of the chosen object, one has to first describe the coordinate system and the reference points used for this definition. Most useful ones from the perspective of the spacecraft AOCS design are described below and their relationship is showed on Figure 2.3.

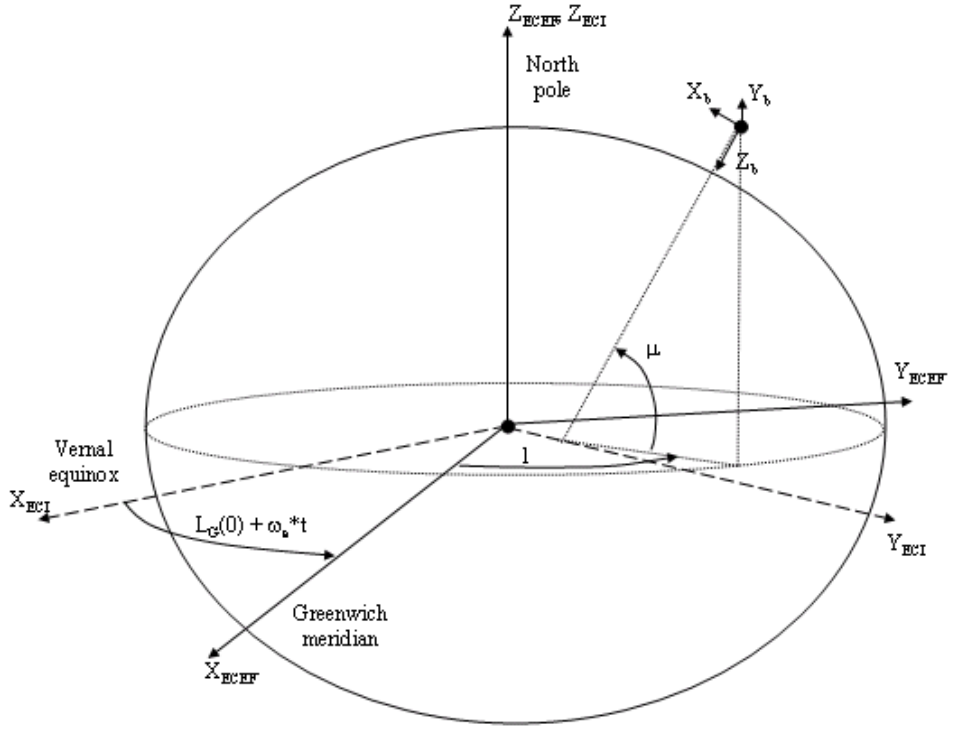


Figure 2.3: Satellite Reference Frames[11]

2.6.1 Satellite Body Frame

Satellite Body Frame has its origin at the center of mass of the spacecraft, with axes directions chosen to fit the design of the spacecraft. For example, for observation missions, most often one of the axes corresponds to the axis of satellite's optical instrument. Position and velocity vectors in this frame will be further noted as $\mathbf{r_B}$ and $\mathbf{v_B}$ respectively.

2.6.2 North east down (NED)

North east down (NED) is a local tangent plane coordinate frame. It is fixed to the body of the satellite, with Z_{NED} axis pointing towards the center of the Earth, X_{NED} axis oriented in the direction of north and Y_{NED} towards east. The most popular application of NED frame is for aircraft and spacecraft, when most objects of interest are below the vehicle, therefore it is convenient to be able to reference the position of the target with positive

value. Position and velocity vectors in this frame will be further noted as $\mathbf{r_{NED}}$ and $\mathbf{v_{NED}}$ respectively.

2.6.3 Earth-Centered Inertial (ECI)

The Earth-Centered Inertial (ECI) frame has its origin located in Earth's center of gravity. It has X axis parallel to and directed as vernal equinox direction, its Z axis is constructed from the vector starting in origin and going through Earth's celestial North pole and Y axis is a vector cross product of the other two. Specific ECI frame used in this thesis is J2000 frame, defined with with the Earth's Mean Equator and Equinox at 12:00 Terrestrial Time on 1 January 2000 [9]. Position and velocity vectors in this frame will be further noted as $\mathbf{r_{ECI}}$ and $\mathbf{v_{ECI}}$ respectively.

2.6.4 Earth-Centered, Earth-Fixed (ECEF)

The Earth-Centered, Earth-Fixed (ECEF), also known as Earth-Centered Rotational (ECR), is a frame of reference can with origin in Earth's center of mass and it's axes are parallel with international reference pole (Z_{ECEF} axis) and international reference meridian (X_{ECEF} axis), with Y_{ECEF} axis being vector cross product of other two. The ECEF frame includes information about rotation of the Earth, hence a point which would be fixed in the ECI frame would be progressing with time in the ECEF frame. Vectors in this frame will be further noted as $\mathbf{r_{ECEF}}$ and $\mathbf{v_{ECEF}}$.

2.7 Coordinates Transformations

Since for different purposes various reference frames and coordinate systems have to be used, it is necessary to have the means to transform vectors between them. As for reference frames the solution is to find a Direction Cosine Matrix (DCM), that is a 3-by-3 matrix which can be used to transform three-dimensional vector $\mathbf{x}$ into another vector $\mathbf{y}$ with the following equation:

$$\mathbf{y} = DCM\mathbf{x} \tag{3}$$

As coordinate transformations are more complex, each one is described in its respective subsection. All transformations are implemented within SCARS Toolbox as the algorithms presented and masked for ease of use.

2.7.1 ECI position and velocity vector to Keplerian elements

As mentioned, Earth orbiting satellite's position can be described by it's position vector $\mathbf{r_{ECI}}$ in Cartesian coordinate system, with center corresponding to Earth's geometric center. That vector in connection with spacecraft's velocity vector $\mathbf{v_{ECI}}$ in same coordinate system can be transformed into Keplerian elements by using following equations:

$$a = \frac{\mu}{2\left(\frac{\mu}{r} - \frac{v^2}{2}\right)} \tag{4}$$

$$i = \cos^{-1}\left(\frac{h_Z}{|\mathbf{h}|}\right) s \tag{5}$$

$$e = \sqrt{\frac{1-h^2}{\mu a}} \tag{6}$$

$$\psi = \cos^{-1}\left(\frac{a - |\mathbf{r_{eci}}|}{ae}\right), \quad where \quad \sin(\psi) = \frac{\mathbf{r_{eci}} \cdot \mathbf{v_{eci}}}{e\sqrt{\mu a}} \tag{7}$$

$$\theta = \sin^{-1}\left[\frac{\sin(\psi\sqrt{1-e^2})}{1 - e\cos(\psi)}\right] \tag{8}$$

$$M = \psi - e\sin(\psi) \tag{9}$$

And Ω and ω can be found from following relations:

$$\sin(\Omega) = \frac{h_X}{\sqrt{h_X^2 + h_Y^2}} \quad and \quad \cos(\Omega) = \frac{h_Y}{\sqrt{h_X^2 + h_Y^2}} \tag{10}$$

$$\sin(\omega + \theta) = \frac{r_Z}{r\sin(i)} \quad and \quad \cos(\omega + \theta) = \frac{r_Z\cos(\Omega) + r_Y\sin(\Omega)}{r} \tag{11}$$

Where terms h_X, h_Y and h_Z are components of $\mathbf{h} = \mathbf{r_{eci}} \times \mathbf{v_{eci}}$ vector and r_X, r_Y and r_Z are components of position vector.

2.7.2 Keplerian elements to ECI position and velocity vector

To quickly obtain position vector from Keplerian elements one may define a coordinate system with x, y axes on orbit's plane with $z = 0$. Then the following equations describe the coordinates:

$$x = a\cos(\psi) - ae \tag{12}$$

$$y = a\sin(\psi)\sqrt{1-e^2} \tag{13}$$

Then the position in Earth's inertial Cartesian coordinate system can be found with following system of equations:

$$\mathbf{r} = \begin{bmatrix} r_X \\ r_Y \\ r_Z \end{bmatrix} = [A_Z(\Omega)]^{-1}[A_X(i)]^{-1}[A_Z(\omega)]^{-1} \begin{bmatrix} x \\ y \\ 0 \end{bmatrix} \tag{14}$$

Where $[A_d(\alpha)]$ stands for transformation matrix about axis d by an α angle.

2.7.3 ECI to ECEF

To transform vectors calculated in inertial frame to Earth-Fixed reference frame one has to multiply the ECI vector by following rotation matrix:

$$DCM_{ECEF}^{ECI} = \begin{bmatrix} \cos\theta_{GMST} & \sin\theta_{GMST} & 0 \\ -\sin\theta_{GMST} & \cos\theta_{GMST} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{15}$$

Where θ_{GMST} is Earth's rotation angle; to be calculated with:

$$\theta_{GMST} = \frac{1}{240} \cdot \mod [24110.54841 + 8640185.812866 \cdot Y + 0.093104 \cdot Y^2 - 6.2 * 10^{-6} \cdot Y^3 + 1.002737909350795\,(3600hh + 60mm + ss)\,, 8640] \tag{16}$$

Where Y is the number or Julian centuries elapsed from the J2000 epoch and $\mod a, b$ is the modulo operator.

2.7.4 ECEF to NED

Implementation of this transformation assumes that the origin of ECEF frame is at the center of the planet, the X_{ECEF} axis intersects the Greenwich meridian and the equator, the Z_{ECEF} axis is the mean spin axis of the planet, positive to the north, and the Y_{ECEF} completes the right-hand system [10]. The following equation shows the DCM for that transformation:

$$DCM_{NED}^{ECEF} = \begin{bmatrix} -\sin\phi\cos\lambda & -\sin\phi\sin\lambda & \cos\phi \\ -\sin\lambda & \cos\lambda & 0 \\ -\cos\phi\cos\lambda & -\cos\phi\sin\lambda - \sin\phi & \end{bmatrix} \tag{17}$$

2.7.5 ECEF to Geodetic Latitude, Longitude, Altitude (LLA)

One can calculate geodetic longitude λ with ease, by following the simple relation:

$$lambda = \arctan(\frac{Y_{ECEF}}{X_{ECEF}}) \tag{18}$$

But to find the geodetic latitude ϕ Bowring's iterative method has to be employed[12]. The calculations are performed inside the SCARS Satellite Model and can be read from the workspace after the simulation is run at least once.

... the explanation...

2.8 Environment

Environment module is responsible for producing environmental parameters such as gravity, magnetic field, atmosphere density, etc., at the position of the simulated spacecraft. The reasoning behind choosing these specific sources are in the description of each sub-module. The main premise was that the source has to be relevant for the choice of actuators.

... description of unified env block and structure

2.8.1 Earths's Gravity Model

Main centripetal force acting on a spacecraft on any orbit is gravity - it is defined by the equation derived from the law formalized by Isaac Newton:

$$\ddot{\mathbf{r}} = -\frac{G(m_1 + m_2)\mathbf{r}}{\|\mathbf{r}\|^2} \tag{19}$$

Where r is the position vector, m_1 and m_2 are the masses of two-body system and G is the universal gravitational constant. Simplified with:

$$m_1 = M_{Earth} \gg m_2 = m_{spacecraft} \tag{20}$$

One can derive the corresponding potential function:

$$u = -\frac{GM_{Earth}}{\mathbf{r}} \tag{21}$$

For a spacecraft on Earth's orbit, this model is a very far-stretched approximation, as it leaves out the influence of Earth's non-ideal shape, changes in density gradient in Earth's interior and perturbations caused by gravitational fields of other bodies. While the influence of other celestial objects is omitted in the SCARS toolbox due to it being mostly designed for lower orbits, one can easily account for Earth's non-spherical mass distribution using function constructed with the use of Lagendre polynomials to calculate

the correction ϵ to potential function (21):

$$\epsilon(r,\theta,\varphi) = \sum_{n=2}^{\infty} \frac{J_n P_n^0(\sin\theta)}{r^{n+1}} + \sum_{n=2}^{\infty}\sum_{m=1}^{n} \frac{P_n^m(\sin\theta)(C_n^m \cos m\varphi + S_n^m \sin m\varphi)}{r^{n+1}} \tag{22}$$

Where the correction is a function of spacecraft's position in spherical coordinate system - r, θ, φ are in order altitude, latitude and longitude. The coefficients J_n, C_n^m and S_n^m are computed to possibly provide best approximation between observed and calculated orbit. Lagendre polynomials of form

$$\frac{P_n^0(\sin\theta)}{r^{n+1}} \tag{23}$$

are called the zonal terms and Lagendre functions

$$\begin{gathered} \frac{P_n^m(\sin\theta)\cos m\varphi}{r^{n+1}} \\ \frac{P_n^m(\sin\theta)\sin m\varphi}{r^{n+1}} \end{gathered} \tag{24}$$

correspond to tesseral terms. The denominating term is the so-called "J_2 term":

$$\frac{J_2\ P_2^0(\sin\theta)}{r^3} = J_2 \frac{1}{r^3}\frac{1}{2}(3\sin^2\theta - 1) = J_2 \frac{1}{r^5}\frac{1}{2}(3r^2 sin^2\theta - r^2) \tag{25}$$

While equations (21) and (22) can added together to faithfully model the influence of Earth's gravity field on the spacecraft, it was decided to use a model from Simulink Aerospace Blockset - the Spherical Harmonic Gravity Model, with EGM2008 planetary model, as it is much more detailed and provides better accuracy.

2.8.2 Partial Atmosphere

Earth's atmosphere is composed of complex layers that are bounded basing on their composition and parameters. Man-made objects on Earth's orbit would be located in thermosphere, if their orbit is at least partially under $600km$ altitude above the surface of the Earth, or exosphere if above it. The former consists mostly of molecular hydrogen and nitrogen, while the latter also of hydrogen, helium ans carbon diaoxide. The main effects of the higher layers of atmosphere on the spacecrafts in Low Earth orbit (LEO) are

drag, degradation of surface materials and spacecraft glow. For the toolbox, the only relevant effect is the first one, resulting in both aerodynamic force and aerodynamic torque acting on the spacecraft.

Aerodynamic forces are created by spacecraft's movement through the atmosphere. The forces acting on the spacecraft are drag, lift and side slip force, but the only one taken into consideration will be the drag, acting on spacecraft's tangential velocity, since the other are of negligible magnitude. To calculate drag force, one has to use the following equation:

$$F_d = -\frac{1}{2}\rho C_d A v^2 \tag{26}$$

Where C_d is the drag coefficient, ρ is atmospheric mass density, A is body area in a cross-section perpendicular to velocity vector and v is the total velocity of the satellite with respect to the atmosphere.

As of now, the effect of aerodynamic torque is omitted in SCARS Toolbox, as to model it with high fidelity one needs to have a 3D model of the specific spacecraft.

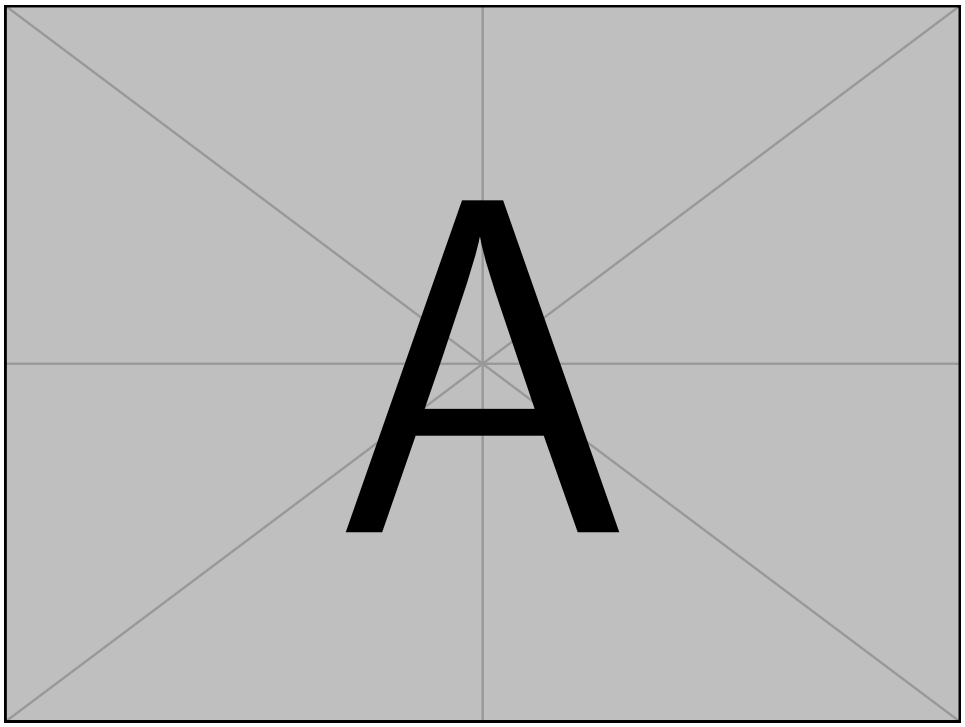


Figure 2.4: Model of Earth's atmosphere layers

The reference atmospheric model used in SCARS is NRLMSISE-00, which takes date and position of the object in geographic coordinate system as inputs and outputs temperature and density of the atmosphere components. As it was built for satellites, it allows for altitudes up to $1000km$. In the toolbox, orbits above that are considered to have negligible impact of the atmosphere and therefore atmospheric forces are set to zero above this threshold.

2.8.3 Sun and Earth Relative Position

The position of Sun in relation to Earth and to satellite is important for simulating mission elements such as spacecraft temperatures or solar panels chargin. times. To acquire Sun's relative position, MATLAB's Aerospace Toolbox function, `planetEphemeris()` was used. By default, the function implements the position based on the DE405 ephemerides in units of km. It was wrapped around SCARS specific function, `getSunPosition()` - a function returning array of Sun's ECI positions every day. The function takes simulation's start time in Julian date format as first parameter and simulation duration, in seconds, as second parameter. It is then implemented in Simulink as a lookup table and whole model is masked for ease of usage.

2.8.4 Earth's Magnetic Model

For precise models of magnetometers and magnetometers it is necessary to include a source of information about Earth's magnetic field. Earth can be approximately modelled as an magnetic dipole, but since the intensity of magnetic field ranges from around 25.000 to $65.000nT$, depending on parameters such as geographic position, altitude, time, and date, hence the need to use a high fidelity model. The choice was to use National Geospatial Intelligence Agency (NGA) World Magnetic Model. This model is already implemented in MATLAB Aerospace Toolbox, so it was just masked for use as a part of SCARS toolbox.

2.9 Actuators

In the following subsections, the descriptions of actuators included in SCARS Toolbox are provided. All of them can be used in a model by themselves or in combination with any other number of actuators. The model linearization method described in subsection C allows for using all provided actuators with all control methods described in subsection 2.11.

2.9.1 Ideal and Simple Actuators

... description of implementation...

2.9.2 Thrusters

One type of actuators that provide the source for external forces and torques acting on a spacecraft are gas thrusters. In case of small satellites, Cold Gas Propulsion (CGP) Systems are the most popular solution, since its simple design leads to smaller actuator mass and low power consumption. A CGP system operates in a process of controlled ejection of compressd liquid or gas propellant.

Spacecraft thrusters can be used for orbit change maneuvers, rapid attitude changes, momentum dumping, nutation and adjusting spin rates. The main advantage of gas propulsion is that the thrust can be controlled with high precision and they can provide high forces and torques. Moreover, there will never be a need for desaturation of a thrusters, in opposition to reaction wheels. Nevertheless the requirement for propellant posses a problem for small satellites, making it a rare method of attitude control in CubeSats and other micro- and nano- satellites.

The key parameters, available for set up in SCARS Toolbox Thruster model are: thrust range, nominal thrust, specific impulse, amount of propellant, total impulse, power consumption, mass and time delay to control. Same as in Simple Actuators, noise sources can be set up.

In SCARS Parts Library various versions of Thruster are available:

- **Directional Thrusters** - Effective forces are assumed to be located on spacecraft body axes, leading to the lack of external torques, hence this model can be used for orbit corrections and maneuvers.
- **Rotational Thrusters** - Effective forces are assumed to be axisymmetric, therefore there are no forces generated by the thrusters, so this configuration can be used for pure attitude control. Additional parameter required for this model is radial displacement for the thrusters.
- **Bang-Bang Thrusters** - Thrusters that operate in bang-bang control mode, allowing for operation only with no or maximum thrust. Additional parameters required for this model are turn-on and turn-off thresholds in control signal. They can be used either in orbit or attitude control and in respective cases they follow the principles of Directional and Rotational Thrusters.

For all thrusters models the only input is the control signal, while the outputs are fuel consumed and either generated force or torque. CGP Sys-

tems also have a downside of decreasing thurst profile in relation with time, since thrust is corelated with the pressure of the propellant inside a tank. This property is not yet modlled in SCARS Toolbox.

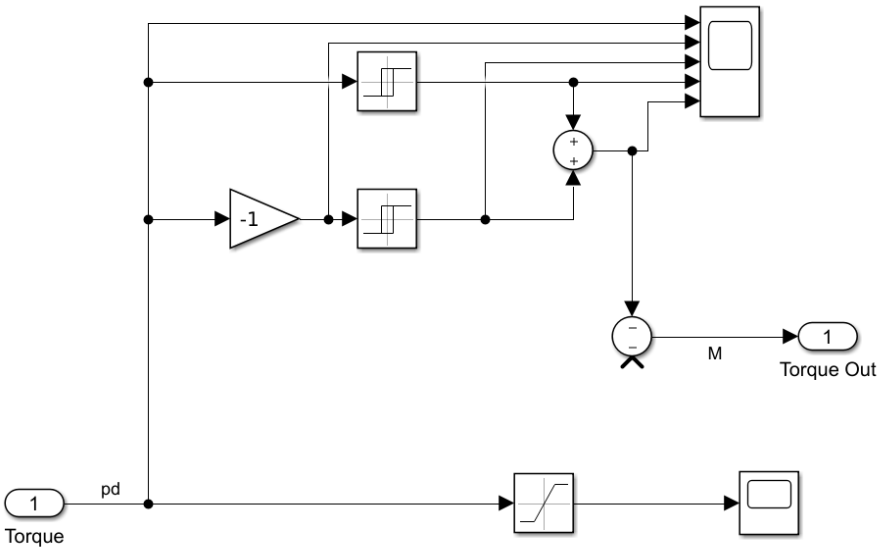


Figure 2.5: Simulink Rotational and Directional Thruster model

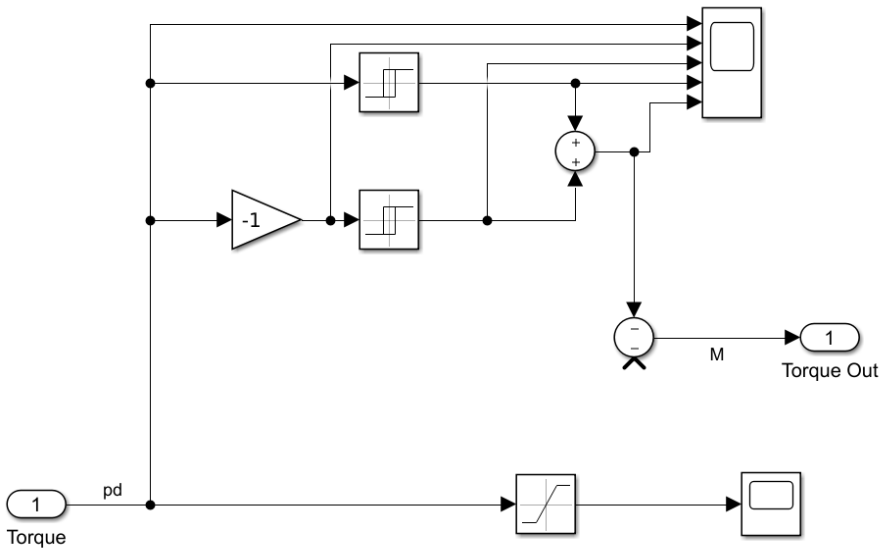


Figure 2.6: Simulink Bang-Bang Thrusters model wrapper

2.9.3 Reaction Wheels

Fast attitude control can be also achieved by the use of reaction wheels - mechanisms consisting of rotating flywheel and proportional electromagnetic torquer, such as DC motor. This allows for very precise attitude maneuvers, with the possibility to eliminate most disturbance torques. Reaction wheels operate around at a non-zero reference speed and change in their angular velocity imposes corresponding torque on the spacecraft. The disadvantage of this solution is that reaction wheels have fixed operating range and to achieve higher angular velocities for the spacecraft, the wheels have to be desaturated using another actuators. In CubeSats, for example, most commonly this would be solved by the addition of magnetorquers.

In fast attitude control the motion about each spacecraft body axis can be considered to be decoupled from motion about two other axes. The equations of motion that describe the influence of reaction wheels angular velocity $\dot{q_w}$ on total angular momentum H are as follows:

$$I_y\dot{q} = Ni + Q_f + Qdy \tag{27}$$
$$\dot{\Theta} = q \tag{28}$$
$$J\dot{q_w} = -Ni - Q_f \tag{29}$$
$$Ri = e - N(q - q_w) \tag{30}$$
$$Q_f = -c(q - q_w) \tag{31}$$
$$H = I_y q + Jq_w \tag{32}$$

Where e, i, R are respectively steering voltage, current in DC motor and armature resistance. N is torque per unit current and c is viscous friction coefficient. Q_f is wheel bearing friction torque and Q_{dy} stands for external disturbance torque. Said equations were modelled in the toolbox with a following diagram:

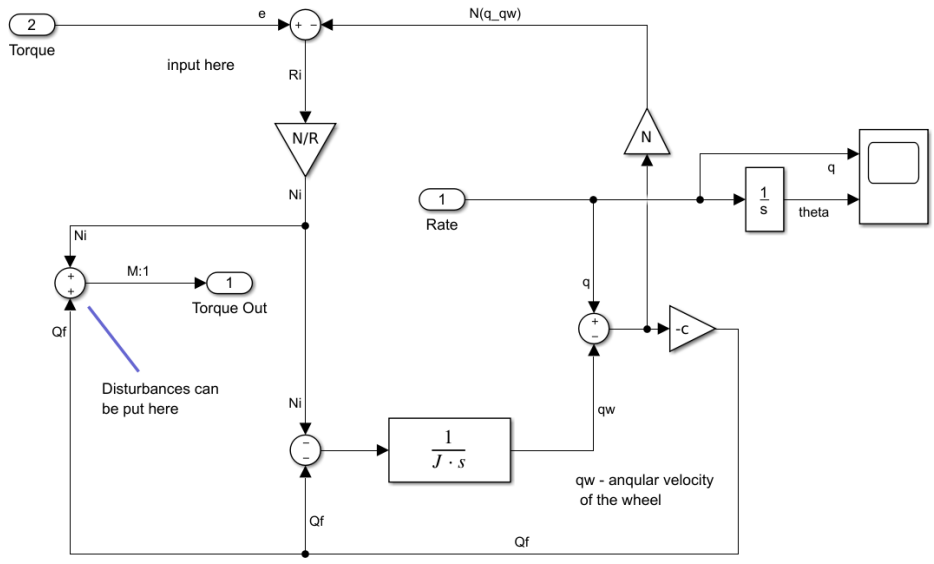


Figure 2.7: Simulink Reaction Wheels model

The problem with modeling off-the-shelf reaction wheels is that datasheets rarely provide the value of viscous friction coefficient c in the DC motor, therefore in SCARS it is considered to be an optional parameter.

2.9.4 Magnetorquers

A magnetorquer is an attitude actuator which uses Earth's geomagnetic field to generate controlling torque. The active part in the magnetorquer is the solenoid, which generates the magnetic dipole moment proportional to the current conducted by the coil. This interaction is described with the following equation:

$$\tau_B = M \times B \tag{33}$$

Where τ_B is mechanical torque acting on the spacecraft, M the generated magnetic moment inside of it and B is the magnetic field density. As the product of a skew-symmetric matrix and a vector it takes a form of:

$$\begin{bmatrix} \tau_{Bx} \\ \tau_{By} \\ \tau_{Bz} \end{bmatrix} \begin{bmatrix} 0 & B_z & -B_y \\ -B_z & 0 & B_x \\ B_y & -B_x & 0 \end{bmatrix} \begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} \tag{34}$$

SCARS Magnetorquer block models a torque rod, a solenoid with a magnetic core. The magnetic moment of a rod magnetorquer is a function of rod current and parameters of the coil, as described in following equation:

$$M = I_M \frac{\pi l w}{4d} \left[\left(\frac{\left[\left(\frac{l}{w} \right) - 1 \right]^{3/2}}{\left(\frac{l}{w} \right) \cosh^{-1} \left(\frac{l}{w} \right) - \left[\left(\frac{l}{w} \right)^2 - 1 \right]^{1/2}} \right) - 1 \right] \tag{35}$$

Where l is the length of magnetic core, w is the width of it and d is the diameter of the wire. I_M is the current flowing through the rod, which can be described with a transfer function:

$$I_M = \frac{V_M}{Ls + R} \tag{36}$$

Where L is the solenoid's inductance and R is its resistance.

The drawback of using magnetorquers for attitude control is that they are unfit for fast maneuvers. Moreover, since Earth's magnetic field density is inversely proportional to cube of distance from Earth's center, then without high grade sensors or on-board models, they don't allow for precise maneuvering on higher orbits.

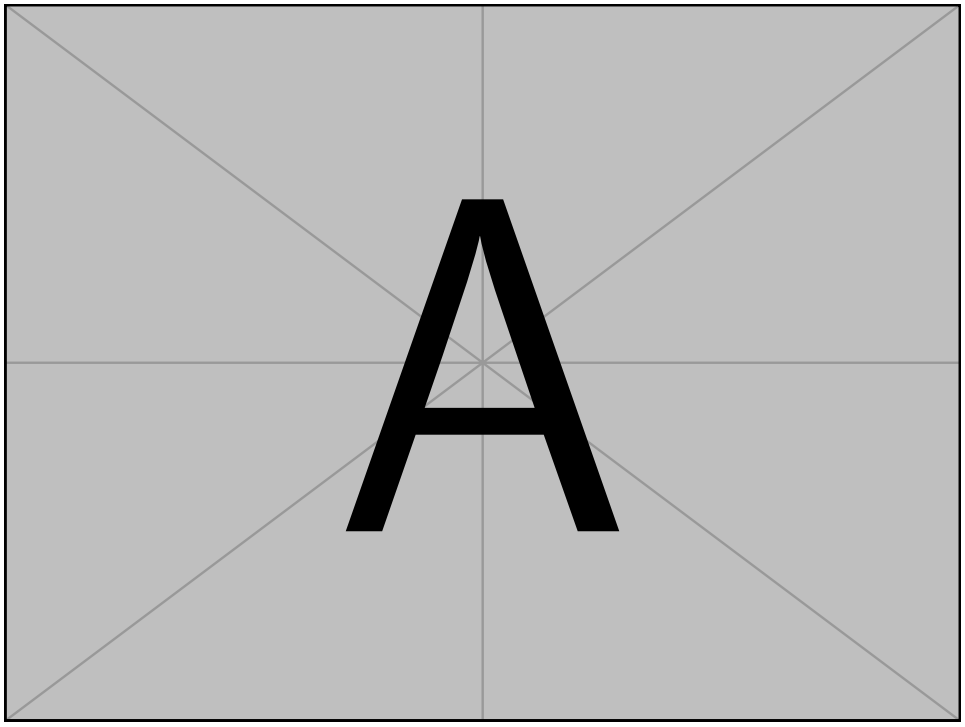


Figure 2.8: Magnetorquers model

2.9.5 Drag Sail

Drag sails use the occurrence of partial atmosphere (described in 2.8.2) to lower satellite's tangential velocity and therefore to quicken the deorbitation of the spacecraft. The premise is to increase area-to-mass-ratio by deploying a large and lightweight structure near the planned end-of-life of the spacecraft. Due to this operating principle, drag sails are only relevant for low and medium mass spacecrafts and are applicable only on LEO. To calculate the perturbing acceleration following equation is used:

$$F = -\frac{1}{2}\rho C_d A v^2 sin\alpha \tag{37}$$

Where *alpha* is the angle between the sail's plane and satellite's velocity vector. For now the moment of sail's deployment is not simulated in SCARS toolbox.

2.10 Sensors

For precise orbit, or attitude, determination both sensors and mathematical models have to be used. Spacecraft sensors can be divided into two types, based on the nature of the performed measurement. One type, inertial sensors reflect the rate of change, therefore any other source of measurement is needed, for initial value acquisition and integration error correction. On the other side there are reference sensors, providing absolute measurements. Sensors of this type measure external parameters, such as Sun's position or Earth's magnetic field intensity, which when compared against mathematical or empirical models can bare the information about satellite's position or attitude. This division is visible in SCARS models, as inertial sensors require input of satellite states, while reference sensors need input from environment model.

2.10.1 Ideal and Simple Sensor

Ideal Sensor is a Simulink subsystem block with unit gain inside, used for testing satellite behavior when sensor errors are not necessary to be taken into consideration.

Simple Sensor is not modelling any specific type of sensor. It can take most parameters used to transform generic ideal sensor into model which corresponds to real hardware, that is: sampling frequency, measurement range and most common sources of errors.

... description of implementation...

2.10.2 GPS Receivers

The Global Positioning System (GPS) is a global navigation satellite systems (GNSS) owned adn operated by United States government. It allows for determining position, velocity and time using data taken from at least four GPS satellites.

Previously using GPS receivers in LEO was burdened with technical challenges, as off-the-shelf components were mostly designed for terrestrial operations, not encompassing for example for large variations in the received signal Doppler frequencies. Recently smaller GPS receivers became available, even for CubeSat use, such as *Venus838FLPx GPS Receiver*[?], allowing for real time orbit determination using GPS navigation in smaller satellite projects.[8] When choosing a GPS receiver one must take several parameters into consideration: update rate, horizontal position accuracy, vertical position accuracy, velocity accuracy and failure rate.

All listed parameters are set up in SCARS *GPS Receiver* part, but rather than designing a model from scratch, MATLAB's Navigation Toolbox function, `gpsSensor()`, was nested inside a masked Simulink block. User can set all beforementioned parameters by editing GPS model's mask fields. Inputs of the model are satellite's true position and true velocity, and the outputs are position and velocity as computed by GPS receiver.

2.10.3 Accelerometers

Accelerometers are force sensors, most often paired with gyroscopes as a part of Inertial Measurement Unit (IMU) board. To measure acceleration three sensors are located with their axes mutually orthogonal and the force external to the board is measured (with the exception of the gravitational force, as it likewise influences the proof mass of the sensor). These measurements are integrated once to obtain the velocity of the spacecraft with respect to the inertial space, or twice to calculate estimated position.

Accelerometer model in SCARS is based on Three-axis Accelerometer from MATLAB Aerospace Blockset. It is masked to be easily integrated with any model produced with SCARS Toolbox.

... description of implementation...

2.10.4 Magnetometers

Making use of implemented Magnetic Field Model, a model of a set of magnetometers is available as a part of SCARS Toolbox. From magnetometer sensors the measurements of direction and magnitude of magnetic field can be acquired. After comparison with Earth's magnetic model spacecraft's on board software conducts transformation from measured vector to one of the reference frames used by ADCS subsystem, providing information about its attitude. Magnetometers are reliable choice of sensors, as they are lightweight, consume low amounts of power, operate in wide temperature ranges.

... description of implementation...

2.10.5 Gyroscopes

Gyroscopes, which fall under category of inertial sensors, measure angular rate around fixed axis. In smaller spacecraft, which is in great deal of SCARS toolbox use-cases, the conventional spinning mass gyroscopes are rarely used, due to limitations in mass and size. Recent developments allow for use of much smaller and cheaper micro-electromechanical systems

(MEMS) gyroscopes, which are vibrating angular rate sensors. They were chosen to model for the toolbox, as of popularity in projects with highly restricted budget.[5] Inside of vibrating gyroscope the Coriolis effect is a cause the vibrating core to produce a force acting on its support. The measurement of the force is used to determine the rate of rotation of the body around gyroscope axis. MEMS gyros are similar to integrated circuits, which use the miniaturized version of mechanisms based on principles of operation of either vibrating wheels, tuning forks, resonant solids or similar common designs.[6] While, besides previously mentioned qualities, the upsides of MEMS gyroscopes are availability of both analog and digital outputs, low power consumption and commercial availability. On the other hand, MEMS gyros have shorter lifetime and lower performance when compare to pricier alternatives.

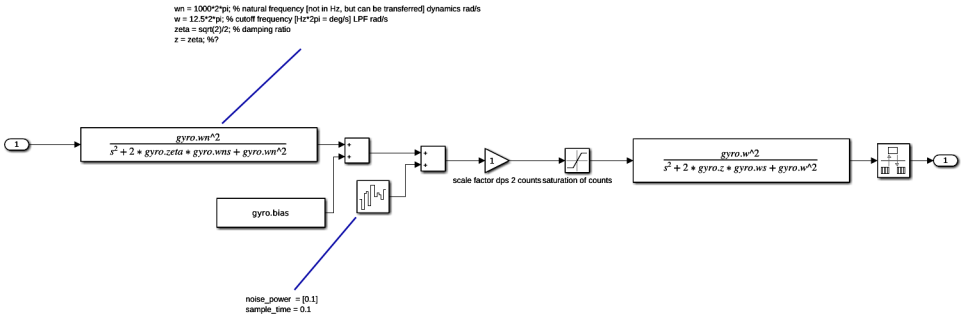


Figure 2.9: Gyroscope model

In the toolbox the following sources of gyroscope errors are modeled:

- Bias offset:
- Scale factor:
- Angle/Velocity Random Walk (A/VRW): A high frequency noise term that has a correlation time much shorter than the sample time. Can be defined as white noise component on the sensor output. Specified either in $\frac{deg}{\sqrt{h}}$ or, as a power spectral density, in $\frac{(deg/h)^2}{Hz}$.
- Quantization Error: An error which is caused by the digital quantization of output signal, obtained when sampling analog input.
- ...

. . . work in progress. . .

2.11 Control Methods

In following sections all control methods implemented in SCARS are described, along with their implementation. Furthermore, the tools available in the toolbox are presented.

2.11.1 PID Controoler

... finish description...

Proportional-integral-derivative controller (PID) is a feedback control loop method, widely used in most industrial applications where the simplicity of the design in of importance.

The controller can be set up to be only proportional, integral or derivative controller, or any combination of these modes. In that case, the gain values for unused modes have to be set to zero.

In SCARS the input of PID Controller block is error signal and the output is control signal.

Attitude vs velocity control

... description... s

2.11.2 LQR

Linear Quadratic Regulator (LQR) is an optimal control method that uses a solution which in simplest form minimizes the quadratic cost function presented in Equation 2.11.2 to generate static gain matrix K.

$$cost = \int x^T Q x + u^T R u \tag{38}$$

LQR method requires the state (Q) and control (R) weighting matrices, which respectively correspond to state and input vectors of the system. They describe the control effort that the controller puts on either minimizing the error in each state or magnitude of each input. Both Q and R matrices are diagonal, and most often are chosen arbitrarily and tuned in iterative process to achieve required controller behavior. Once calculated, the static gain matrix K is used in a feedback control law:

$$u = -Kx \tag{39}$$

To use LQR method in SCARS Toolbox, the state-space system of the spacecraft model has to be found first. As mentioned before, this is done by following the linearization process described in subsection C.

Implementing LQR Controller in SCARS toolbox automates the process for the user, asking only to input Q and R matrices as block's mask parameters.

... create the block and describe the implementation...

2.11.3 B-dot Algorithm

B-dot algorithm is popularly used for spacecraft detumbling. In its principle, magnetorquers are used to generate a torque that dampens the initial rotation of the spacecraft. The required magnetic moment is proportional to the change of magnetic field around the spacecraft. The required magnetic dipole M is calculated from the following equation:

$$M = -k\dot{B} \tag{40}$$

[13] Where k is the tunable control constant and B is the magnetic field intensity in satellite body frame.

2.11.4 Quaternion Feedback Control

... description...

2.11.5 Analisys Tools

... description...

2.12 Visualization Tools

While analytical approach may provide all necessary information to conduct a technical review of a control system, it might be convenient to present the behavior of the spacecraft in visual form. In these chapters three software solutions are described: one directly implemented in SCARS Toolbox and other two can use simulation outputs to show how modelled satellite performs.

2.12.1 MATLAB Virtual Reallity Toolbox

Virtual Reality Toolbox is an extension for MATLAB which allows creating and interacting with 3D virtual reality models of dynamic systems. In its core it uses Virtual Reality Markup Language (VRML), a language created in the early days of World Wide Web (WWW) to display 3D objects and animations. This toolbox provides a way for implementing the VRML

models inside MATLAB script or Simulink simulation, and allows for control of driving display or animation with MATLAB variables and Simulink signals. Moreover, the toolbox is integrated with VRML viewer and VRML editor, allowing for building and displaying models directly from MATLAB environment.

Virtual Reality Toolbox was used in SCARS as most core method of visualization. The VRML model is set up with 3 objects: Satellite, Earth and Sun, as they can be considered most useful when observing the effects of the simulation. Satellite model also includes objects representing antennas' range or optical instrument's field of view, if set up in simulation. This feature can be useful for analysis of imaging capabilities.

The transformations required to process the data generated by SCARS' Vehicle Dynamics block into VRML parameters are as follows:

$$r_{VRML} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} r_{ECEF} \tag{41}$$

... rest of transformations descriptions and example of visualization...

2.12.2 Systems Tool Kit

Systems Tool Kit (STK), formerly named Satellite Tool Kit, is a platform for analyzing and visualizing variety of ground, sea and space platforms missions. STK is a commercial software solution used by most major organizations and companies such as National Aeronautics and Space Administration (NASA), ESA, German Aerospace Center (DLR), Boeing, ICEYE. Features most relevant to the topic of this thesis are the graphical engine allowing for displaying the position and attitude of the satellite, and the set of analytical tools, such as ground station connection time calculator, allowing for fine-tuning of mission details.

To visualize SCARS simulation results with STK, one must generate timestamped ephemeris and attitude files. SCARS can generate such files for the user, with predetermined format according to STK documentation[?]. Both files contain the preamble specifying parameters such as scenario epoch time, central body, coordinate system, distance unit and format of the file. As SCARS relies mostly on ECEF reference frame, it is also chosen for ephemeris and attitude files. After the preamble, the file contains the lines for each data point. In case of ephemeris file (`.e` file) the have a format of:

```
<TimeInSeconds> <X> <Y> <Z> <xDot> <yDot> <zDot>
```

Where the unit of time is seconds and relative to defined scenario epoch and following parameters are ECEF vectors in m, m/s and m/s^2 respectively. For the attitude file (`.a` file), the format is:

```
<TimeInSeconds> <q1> <q2> <q3> <q4>
```

Where time is formatted in same manner as in ephemeris file and the following parameters build a quaternion vector, with fourth element being scalar component.

Example `.e` and `.a` files can be found in subsection A and subsection B respectively.

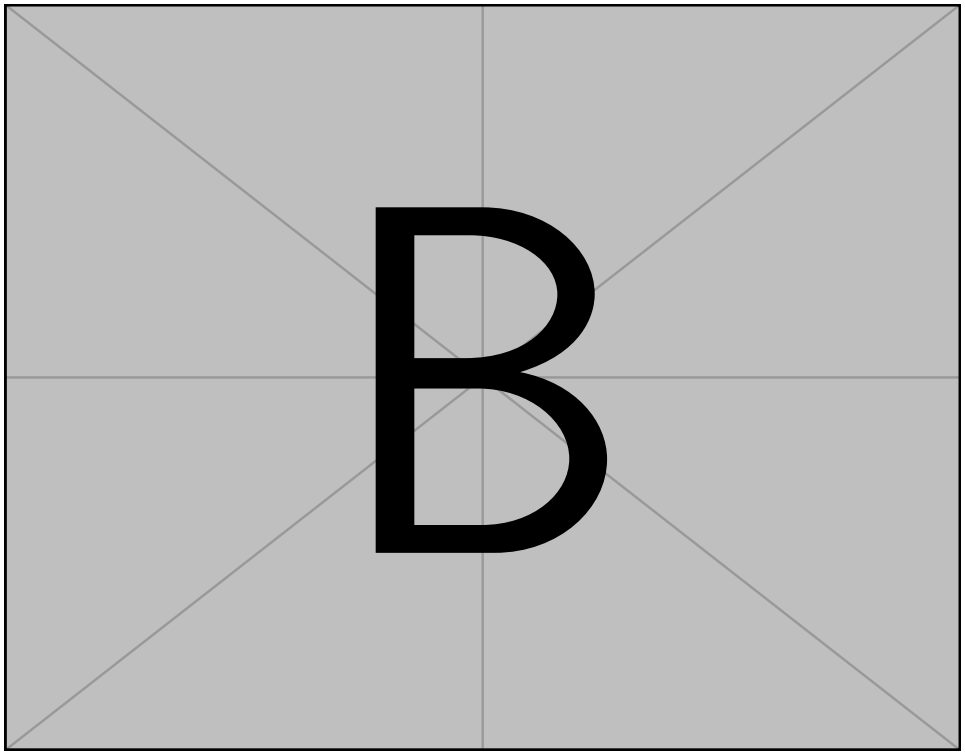


Figure 2.10: Example of STK 3D satellite visualization

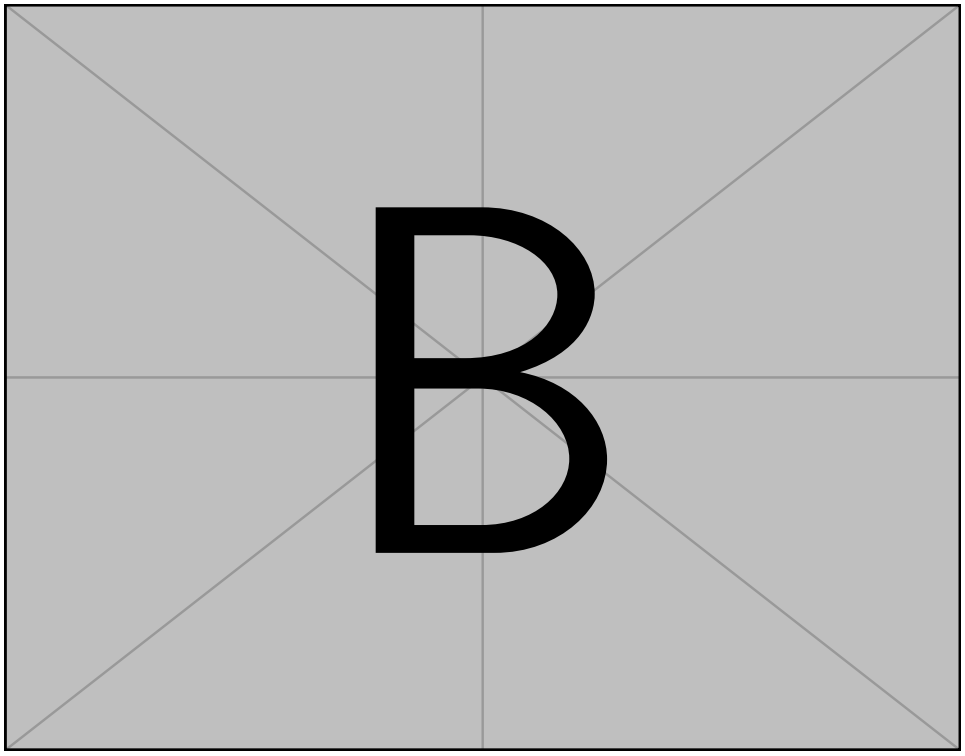


Figure 2.11: Example of STK ground track representation

2.12.3 Kerbal Space Program

Finally, Kerbal Space Program (KSP), a space flight simulation video game, can be used as a nonconventional method to visualize the results of SCARS Toolbox simulation. In KSP the player directs a developing space program originated on fictional Earth-like planet Kerbin. The game provides the tools for the players to design and fly rockets, probes, satellites, spaceplanes, rovers, and other spacecraft from a library of components.[?] The aim of this visualization method was to build a sample satellite in KSP, simulate it in SCARS and execute a Hohmann Transfer within a game, using simulation outputs as game inputs.

The connection between MATLAB and KSP is possible because of fan-made Remote Procedure Call Server for KSP (kRPC) mod. It creates a API server running alongside the game, with which calls can be made using already written clients in most popular languages, like C++, Python, Lua, Java, etc. Integrating it with MATLAB has proved to be a difficult task,

as MATLAB doesn't provide simple means for threading, which means that inputs for the game have to be precalculated to work in real time. Moreover, there is no kRPC library written directly for MATLAB, therefore a simple Python bridge was written to parse the data taken from the game, compare them with pre-generated SCARS simulation scenario outputs and send them to KSP as in-game AOCS subsystem inputs.

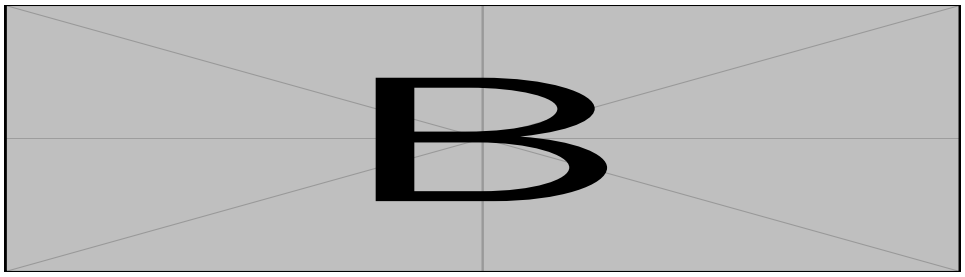


Figure 2.12: Example of kRPC Python client code

. . . rest description and screenshots of visualization. . .

3 SCARS Documentation

. . . implementation. . .

3.1 Folder Structure

. . . description. . .

3.2 MATLAB Live Script

. . . description. . .

3.3 Simulink Models Masks and Help

. . . description. . .

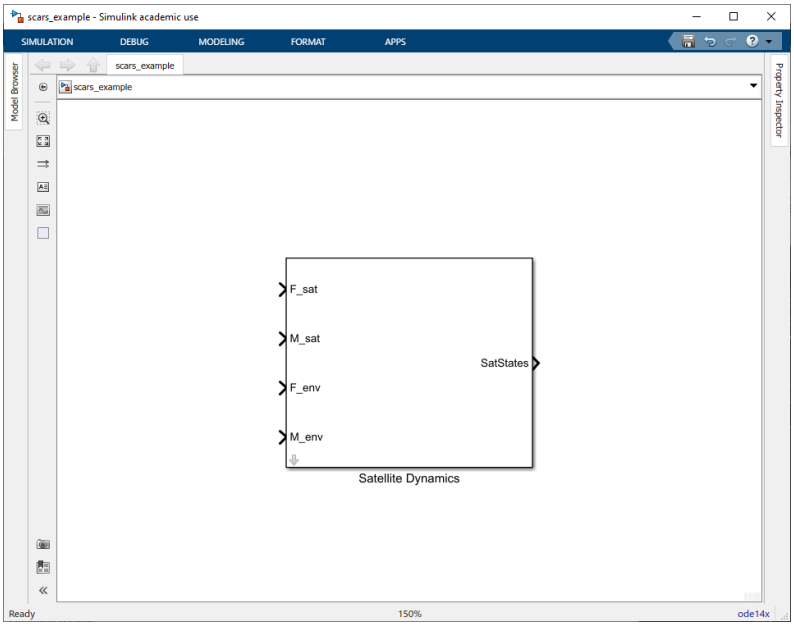
4 Examples of usage

The following chapters shows typical use-cases of SCARS Toolbox. First two sections can serve the purpose of teaching with step-by-step instructions how to set-up a simple project. Following parts showcase real life example spacecrafts, which AOCS Subsystems can be simulated as accordingly set up SCARS Modular Simulations. Finally, examples of control system tests are shown, proving that SCARS can be used for both prototyping and reviewing processes.

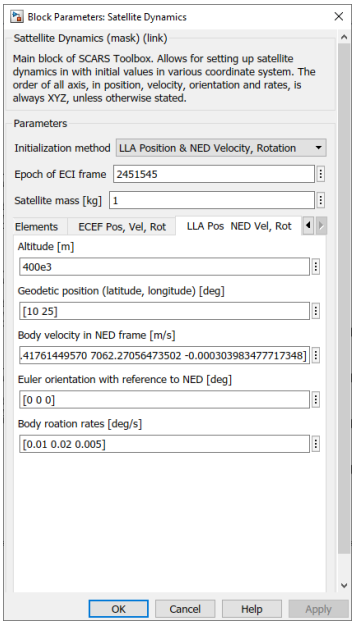
4.1 Simple spacecraft example

The nominal usage of SCARS Toolbox is take a simple objective that designed AOCS subsystem has to fulfil, chose on board hardware and model the spacecraft accordingly, using only necessary components. To showcase the basic workflow below are presented the steps describing a process to **check whether chosen set of reaction wheels and gyroscopes can provide 20 arcsecond accuracy during 5s of geographic coordinates tracking**. The model constructed for this example will be further referred to as Example Model.

Step 1: Initial Spacecraft Dynamics setup



(a) Example Model



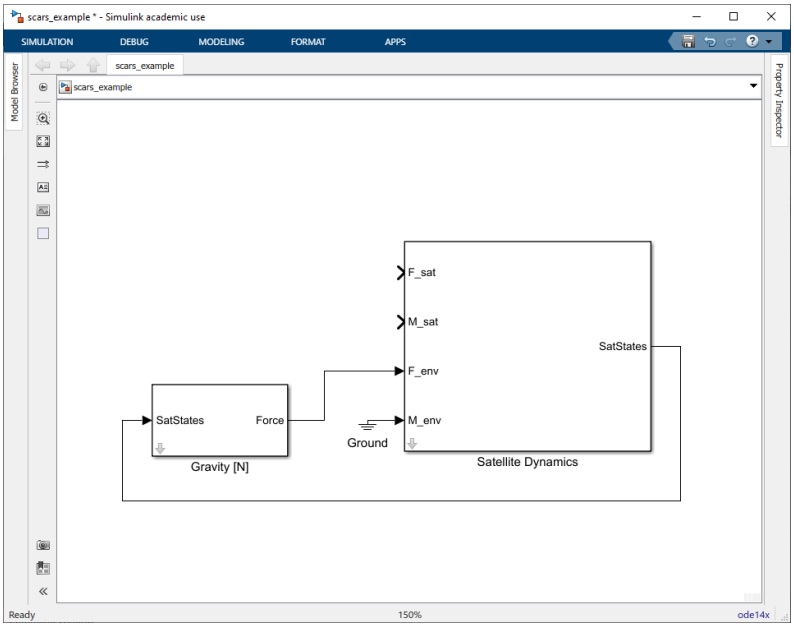
(b) Spacecraft Dynamics Mask

Figure 4.1: Step 1

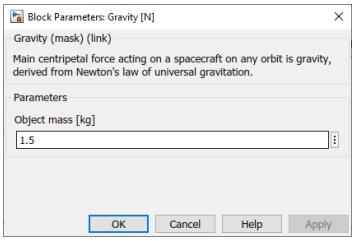
To set up the spacecraft as a point in orbit spacecraft dynamics model everything that needs to be done is to add **Spacecraft Dynamics** block from SCARS Parts Library, as seen on Figure 4.1 (a) and to input parameters describing the simulated spacecraft into object's mask. Available are several initialization methods, corresponding to reference frames in which the user can input the starting point. In this case, since the objective is to track geographic coordinates, the method of choice is **LLA Position & NED Velocity, Rotation**. The choice of parameters, corresponding to average CubeSat, can be seen on Figure Figure 4.1 (b). Initial latitude and longitude were chosen arbitrarily and are in the neighborhood of the tracking point, which will be set up later.

Afterwards the user can set up Simulink model solver parameters to **Fixed-step** with **ode14x** solver choice, with the rest left to default settings. It is not necessary, but allows for larger step-size when using **Derivative** blocks.

Step 2: Environment setup



(a) Example Model

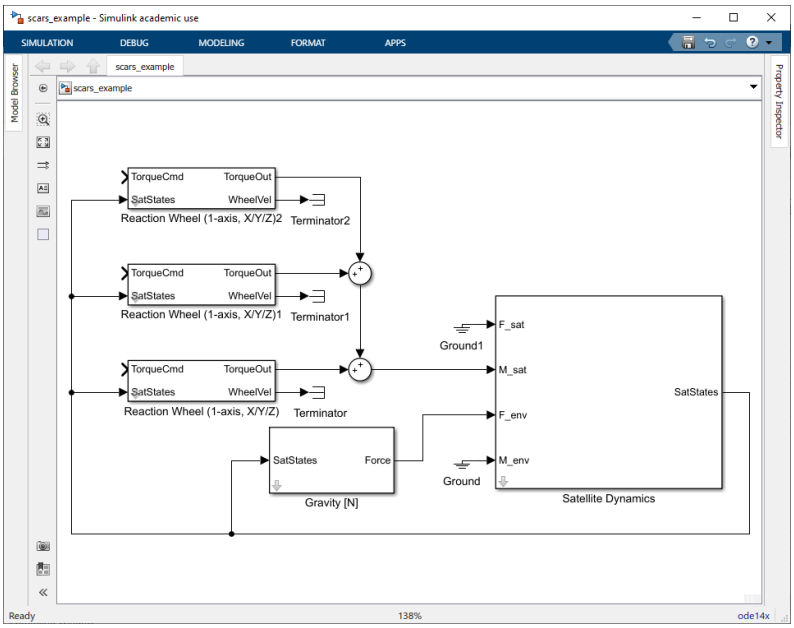


(b) Gravity Mask

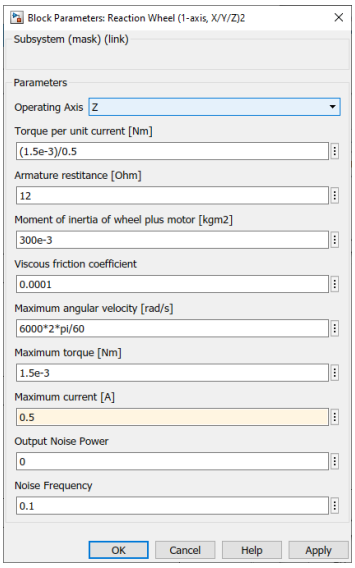
Figure 4.2: Step 2

As this spacecraft does not use magnetorquers nor does not have any major drag-inducing components, the only relevant block representing environment's influence on the satellite is the **Gravity [N]** block. The only parameter to input is spacecraft's mass, as seen on Figure 4.2 (a).

Step 3: Actuators choice and setup



(a) Example Model

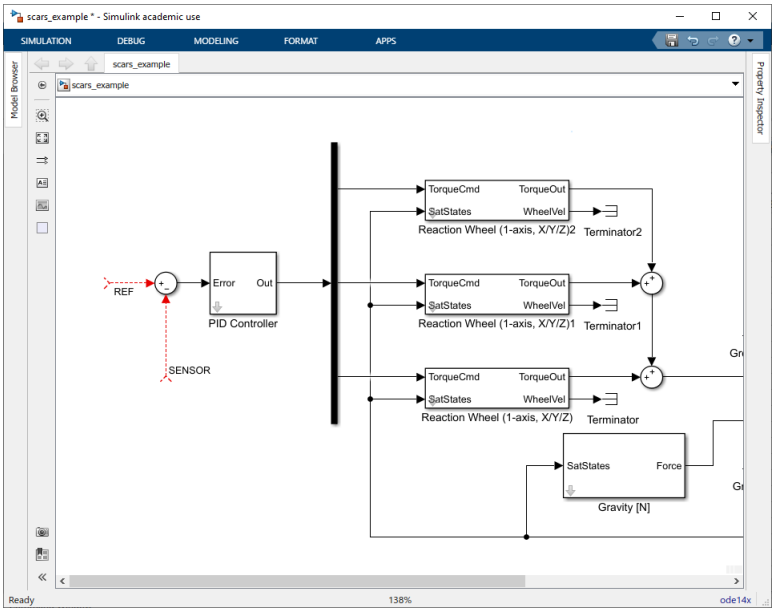


(b) Reaction Wheel Mask

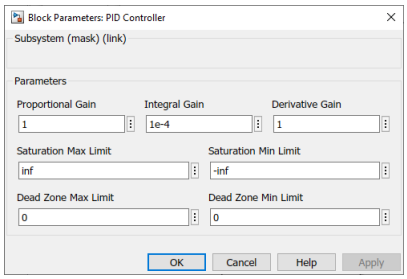
Figure 4.3: Step 3

The next step would be to implement the choice of actuators into the spacecraft model. In this example the user could want to test the NanoTorque GSW-600 reaction wheels, in nominal configuration of one wheel for each spacecraft body axis, from GomSpace manufacturer. The list of relevant parameters, compiled from the actuator's datasheet, can be seen on Figure 4.3 (b). They were put into as parameters of **Reaction Wheel (1 axis X,Y,Z)** block from SCARS Parts Library and added to Example Model. The required inputs are the control signal and **SatStates** bus signal (described in subsection 2.5), while the outputs are torque and wheel angular rate.

Step 4: Setup of control algorithm



(a) Example Model

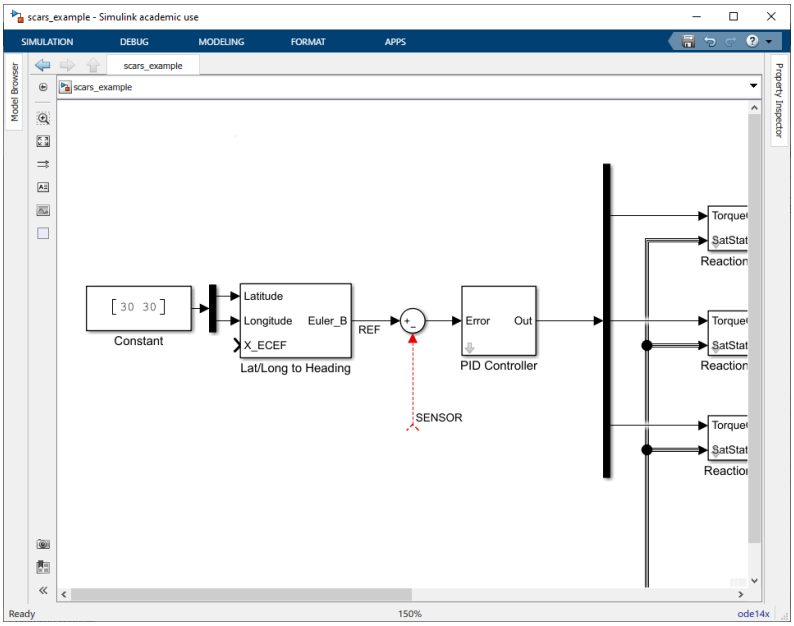


(b) PID Mask

Figure 4.4: Step 4

A PID controller was chosen as a control mechanism as the source of input signal in reaction wheels. The parameters were initially set as can be seen on Figure 4.4 (b). Saturation of **PID Controller** block output signal was set up in accordance to hardware's maximum voltage.

Step 5: Coordinate transformation and reference signal

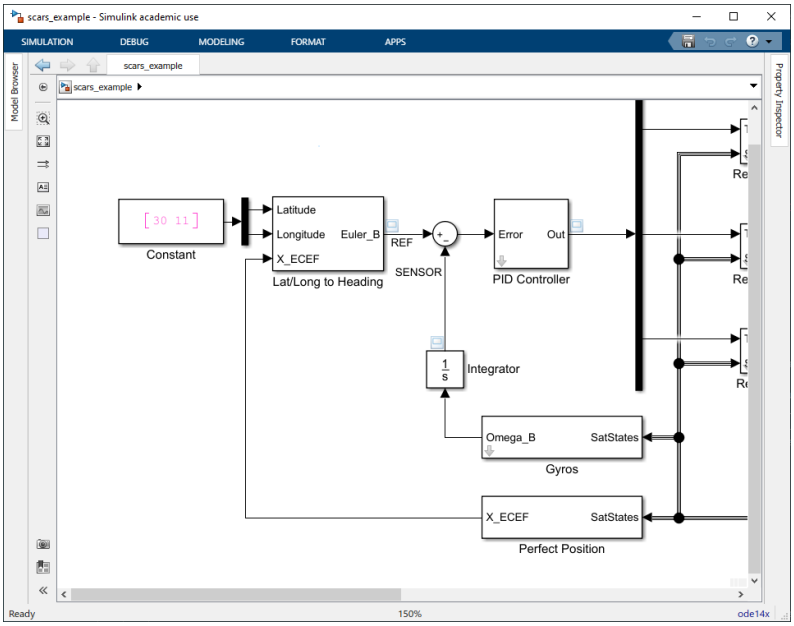


(a) Example Model

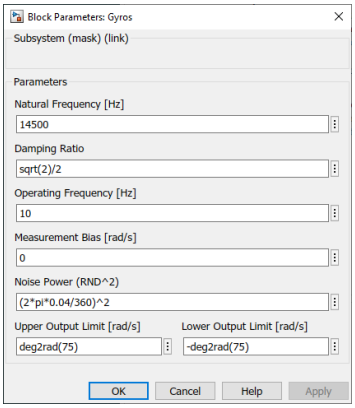
Figure 4.5: Step 5

SCARS Toolbox also provides a way to speed up the process of building mathematical transformations, allowing the user conduct initial tests first and only later think about software implementation. This approach may lead to significant savings in manhours, as failing approaches can be rejected without spending time on setting up algorithms from scratch. In this case, **Lat/Long to Heading** block was used, without the need for any further setup. As first two inputs are the desired geographical coordinates and the last one is the position vector of the satellite, in ECEF reference system.

Step 6: Sensors choice and setup



(a) Example Model



(b) Gyros Mask

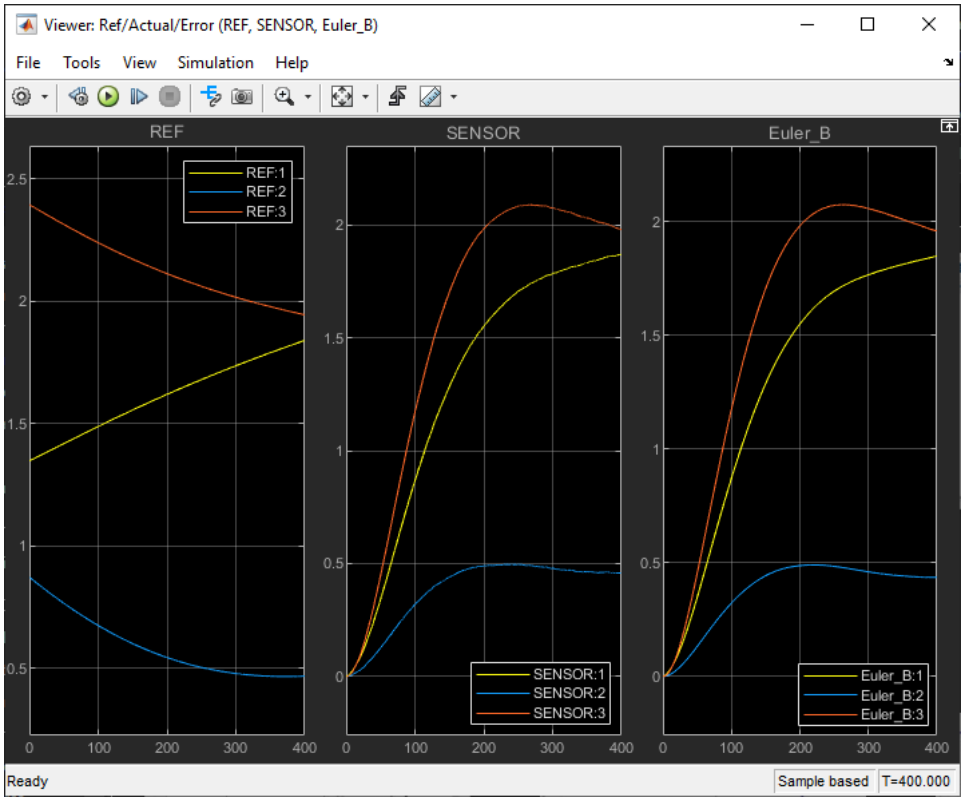
Figure 4.6: Step 6

As one can see on Figure ..., the only signals necessary to close the control loop are satellite's position, as an input to **Lat/Long to Heading** and Euler angles in body reference frame. The former was not relevant to the

posed objective of this Example Model, therefore was set up to be measured by the **Ideal Position Sensor (ECEF)** block, but the latter had to be a gyroscope. **Gyros** block was added to the simulation and set up with the parameters of . . . gyro, chosen by the fictitious user to be ADXRS614 MEMS Gyroscope, as proposed by Li et al.[14] The extract from the datasheet and it's representation as SCARS' block parameters can be found on Figure Figure 4.7 (a) and Figure 4.7 (b) respectively.

Step 7: Simulation and verification

Finally, the simulation can be run by the user and the results can be verified.



(a) Example Model

Figure 4.7: Step 7

. . . *finish description*. . .

4.2 PW-Sat2

As written on its website, "PW-Sat2 is a student satellite project started in 2013 at Warsaw University of Technology by the Students Space Association members. Its main technical goal is to test new deorbitation technology in form of a large deorbitation sail whereas the project purpose is to educate a group of new space engineers. In February 2018 PW-Sat2 became fully integrated and was being prepared to the launch into orbit planned for the second half of 2018." [?]

... screenshots of model...

4.2.1 Detumbling

One of two modes of control that PW-Sat2 operates in (with the other one being Sun Pointing Mode) is Detumbling Control Mode. Detumbling maneuver is performed after deployment of the spacecraft from a carrier rocket. As the satellites are separated from the deployment mechanism, they are burdened by non-zero initial angular rates. To counteract that and stabilize a satellite PW-Sat2 is equipped with a set of two perpendicular magnetorquer rods and one air core, in total one coil acting along each of satellite's body axis. The most important parameters used by SCARS model of PW-Sat2 are listed in Table 4.2.1. [?] Exact values are taken directly from the datasheet of ISIS Magnetorquer Board (iMQT) [?].

Parameter	**Value**
Nominal magnetic dipole	$0.2Am^2$
Maximum actuation envelope error	$3\mu T$
Power consumption during actuation	$1.2W$
Mass	196g

... plots with description...

4.2.2 Deorbitation with drag sail

One of the main objectives of PW-Sat2 mission was to deploy and test the effectiveness of it's drag sail in deorbitation maneuver. The sail was $2x2m$ square made from aluminized polyester boPET film. [?]

In this example, SCARS toolbox was tested against data points derived from NORAD measurements. The simulation was run in two cases: only with drag sail set up and with drag sail and magnetorquers on to keep sail's plane perpendicular to spacecraft's orbit tangent vector.

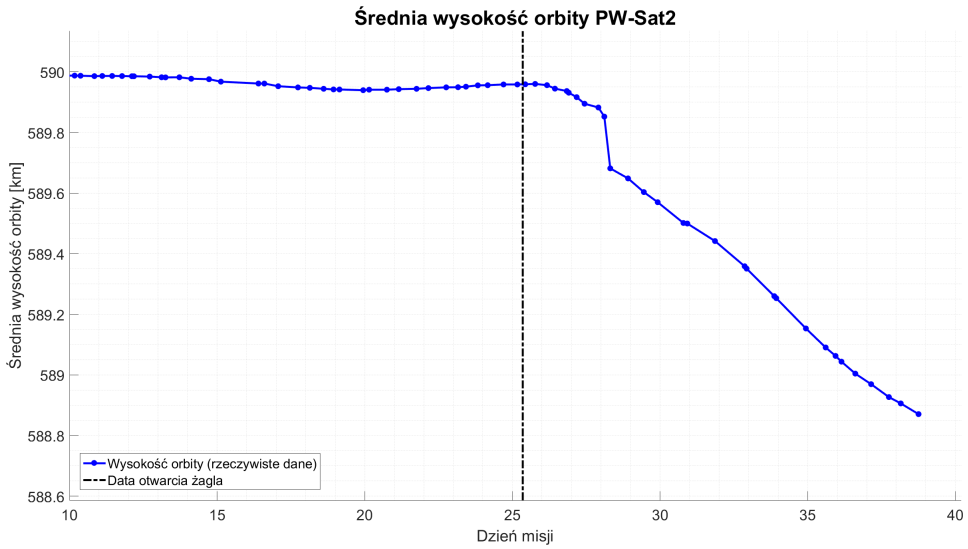


Figure 4.8: Average altitude of PW-Sat2 satellite as taken from NORAD measurements. On X axis there is mission time in days, on Y axis there is average altitude in km.

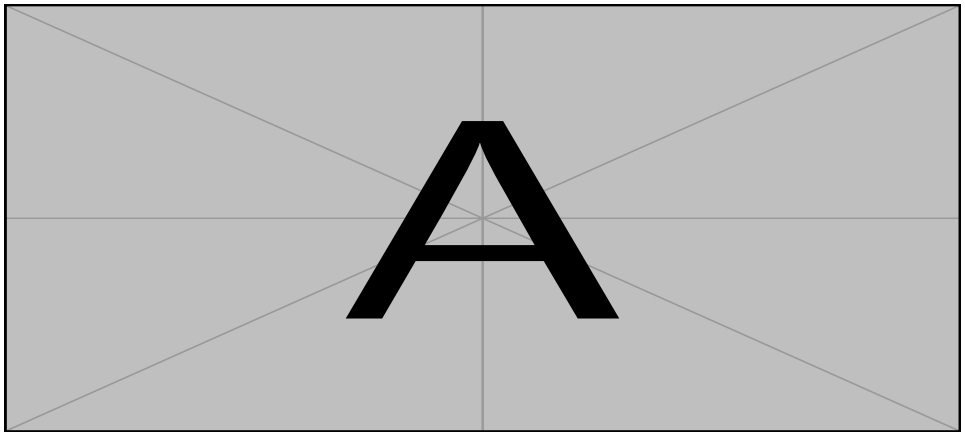


Figure 4.9: Results from SCARS simulation, with only drag sail included

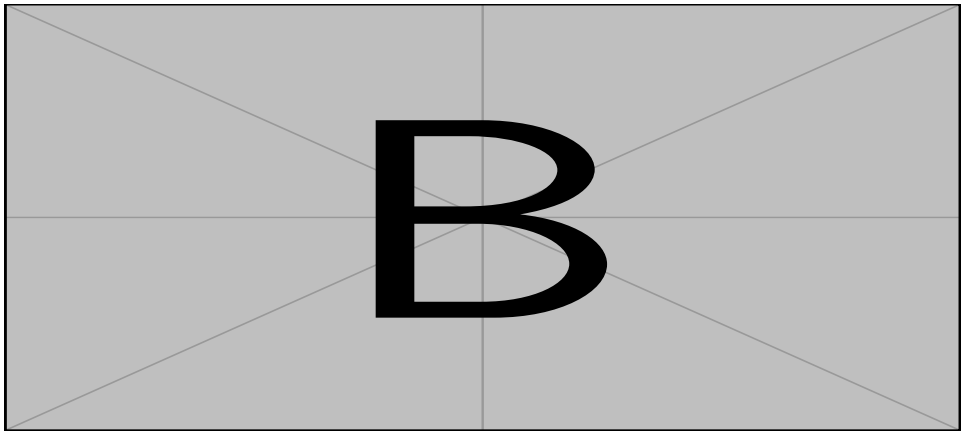


Figure 4.10: Results from SCARS simulation, with magnetorquer-driven attitude correction

4.3 Sentinel-2

Sentinel-2 is a European polar satellite mission carried out by ESA as a part of Copernicus Programme. It consists of constellation of twin polar orbit satellites, Sentinel-2A and Sentinel-2B and it's aim is to deliver Earth observation data to broad public, providing wide range of services such as natural emergency management, agricultural monitoring or water classification[?].

As per document describing Sentinel-2 ADCS subsystem, the satellites operate on a sun-synchronous orbit, with $786km$ mean altitude and $10:30$ local time of descending node. They maintain Earth-oriented attitude in all operational modes. The required pointing performance is moderate, but the main design driver is the need for precise geo-location of the images[?].

The actuators on board of Sentinel-2 and simulated with SCARS toolbox for demonstration purposes are described in Table 4.3.

No.	**Unit**	**Type**	**Supplier**	**Name**
3	MAG	3-axis fluxgate magnetometer	ZARM Technik	FGM-A-75
2	GPRS	2 band GPS receiver	RUAG	-
3	STR	Active pixel sensor star tracker	Jena Optronik	Astro APS
4	IMU	High performance fibre optical gyro	Astrium	ASTRIX 200
3	MTQ	$140Am^2$ magnetic torquer	ZARM Technik	MT140-2
4	RW	$18Nms$ reaction wheel	Honeywell	HR12
8	THR	$1N$ monopropellant thruster	EADS ST	CHTIN-6

... earth-pointing test: plots and description...

4.4 Examples of tests possible with SCARS

... introduction...

4.4.1 Control system robusntess analisys

... plots and description...

4.4.2 Long term simulation

... plots and description...

4.4.3 Contingency scenarios

... plots and description...

5 Conclusions

... introduction...

5.1 Problems encountered

... work in progress...

It was a mistake to try to build one simulation for both fast maneuvers and long term mission planning, as it is difficult to set up all simulation parts to work with varying time-step.

While a set of sensors is modelled to be ready to use as a part of SCARS Toolbox, there are no sensor fusion nor filtering algorithms included. The results are presented with perfect sensors.

Currently the mission designer module is a very simple solution, leaving up to the user to create signals which would turn the control systems on and off, and create reference signals.

References

[1] Nghi M Nguyen et al. Effective space project management. In *Proc. of the Project Management Institute Annual Seminars & Symp.(Houston, Texas, USA,)*, 2000.

[2] ECSS Secretariat. Ecss-m-30-01a space project management. organization and conduct of reviews, 1999.

[3] Stephen J Kapurch. *NASA systems engineering handbook*. Diane Publishing, 2010.

[4] David A Vallado. *Fundamentals of astrodynamics and applications*, volume 12. Springer Science & Business Media, 2001.

[5] Mario N Armenise, Caterina Ciminelli, Francesco Dell'Olio, and Vittorio MN Passaro. *Advances in gyroscope technologies*. Springer Science & Business Media, 2010.

[6] Jonathan Bernstein et al. An overview of mems inertial sensing technology. *Sensors*, 20(2):14–21, 2003.

[7] Ebrahim H Kapeel and Ahmed M Kamel. Modeling and simulation of low cost mems gyroscope using matlab (simulink) for uav autopilot design. 2019.

[8] Vivian M Gomes, Helio K Kuga, and Ana Paula M Chiaradia. Real time orbit determination using gps navigation solution. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, 29(3):274–278, 2007.

[9] B. Schutz, B. Tapley, and G.H. Born. *Statistical Orbit Determination*. Elsevier Science, 2004.

[10] MathWorks Aerospace Blockset Documentation. *6DOF ECEF (Quaternion)*, (accessed July 13, 2020). `https://www.mathworks.com/help/aeroblks/6dofecefquaternion.html`.

[11] MathWorks Aerospace Blockset Documentation. *About Aerospace Coordinate Systems*, (accessed July 13, 2020). `https://www.mathworks.com/help/aeroblks/about-aerospace-coordinate-systems.html`.

[12] George P Gerdan and Rodney E Deakin. Transforming cartesian coordinates x, y, z to geographical coordinates ϕ, λ, h. *Australian surveyor*, 44(1):55–63, 1999.

[13] Pedro A Capo-Lugo, John Rakoczy, and Devon Sanders. The b-dot earth average magnetic field. *Acta Astronautica*, 95:92–100, 2014.

[14] Junquan Li, Mark Post, Thomas Wright, and Regina Lee. Design of attitude control systems for cubesat-class nanosatellite. *Journal of Control Science and Engineering*, 2013, 2013.

Appendices

A Example STK Ephemeris File

. . . code. . .

B Example STK Attitude File

. . . code. . .

C Model Linearization with Control System Toolbox

. . . explanation. . .