DEGREE PROJECT

Simulation and control toolkit for small satellite projects

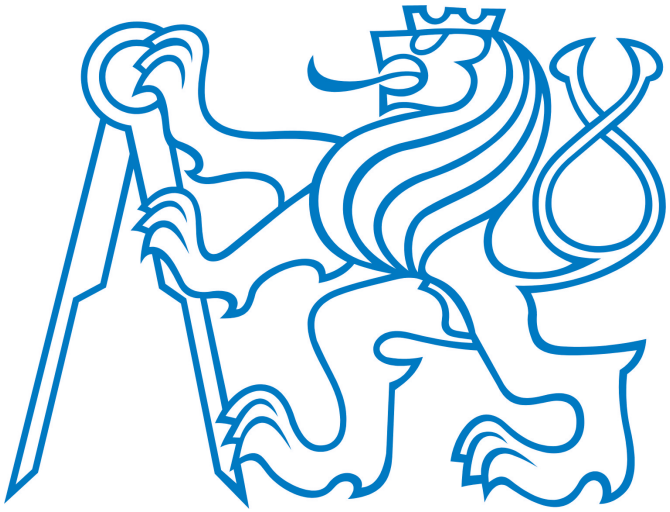
Adam Smialek

Space Engineering, master's level
2020

Luleå University of Technology
Department of Computer Science, Electrical and Space Engineering

LULEÅ UNIVERSITY OF TECHNOLOGY

CZECH TECHNICAL UNIVERSITY IN PRAGUE
FACULTY OF ELECTRICAL ENGINEERING
DEPARTMENT OF CONTROL ENGINEERING



Master Thesis

Simulation and control toolkit for small satellite projects

(Programovy balik pro simulaci a navrh rizeni malych satelitu)

Supervisor: Ing. Martin Hromčík, Ph.D.

Study Program: Cybernetics and Robotics

Specialization: Space Automation and Control

14th of July 2020

MASTER'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Śmiałek Adam** Personal ID number: **492264**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Control Engineering**

Study program: **Cybernetics and Robotics**

Branch of study: **Cybernetics and Robotics**

II. Master's thesis details

Master's thesis title in English:

Simulation and control toolkit for small satellite projects

Master's thesis title in Czech:

Programovy balik pro simulaci a navrh rizeni malych satelitu

Guidelines:

The goal of the project is to propose, implement and demonstrate value of a software framework supporting teams building small satellites - typically CubeSat student projects - during the initial phases of conceptual design, mission planning, and selection and sizing of components. In relation to requirements coming from the expected navigation and flight control functionalities.

1. Make a review of existing related tools. Both from the large satellite business, and the small satellite community. Present a comparative analysis of strengths and weaknesses of existing solutions.
2. Based on the review, prepare a study and the software concept proposal showing specifically the need for this work and expected compatibility of the toolkit with relevant other packages.
3. Propose and implement essential data structures and procedures of the toolkit. Focus on the aspects of conceptual design, mission planning, selection and sizing of components, simulation, design and validation of control laws for ADCS (attitude determination and control system).
4. Demonstrate usefulness of the toolkit on a case study, inspired by selected previous small satellite projects - either your own or described in literature.

Bibliography / sources:

[1] Bryson Jr., Control of Spacecraft and Aircraft, Princeton University Press, 1994.
[2] Blakelock, Automatic Control of Aircraft and Missiles, Wiley, 1991.

Name and workplace of master's thesis supervisor:

doc. Ing. Martin Hromčík, Ph.D., Department of Control Engineering, FEE

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **13.02.2020** Deadline for master's thesis submission: **14.08.2020**

Assignment valid until:
by the end of summer semester 2020/2021

doc. Ing. Martin Hromčík, Ph.D.	prof. Ing. Michael Šebek, DrSc.	prof. Mgr. Petr Páta, Ph.D.
Supervisor's signature	Head of department's signature	Dean's signature

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

....

Place, date Signature

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 into the task of building a small satellite simulation.

Keywords: spacecraft, satellite, AOCS, ADCS, control design, MATLAB, Simulink, toolbox, software, prototyping

Acknowledgement

I would like to give my thanks to Dr. Martin Hromčík, my supervisor from Department of Control Engineering, FEL CTU. Without his guidance this work would be impossible and I especially appreciate all the help in the times of the coronavirus pandemic.

I would also like to thank my girlfriend, Dominika Miszewska, and my family. The former for her infinite patience and excellent proofreading, and the latter for support and understanding that aerospace student can sometimes walk with his head in the clouds.

Finally, I would like to thank the students I have met during my studies at WUT, LTU and CTU. The motivation which comes from seeing as they develop themselves and pursue excellence is invaluable. Especially M.Sc., Eng. Maksymilian Sienkiewicz, as an exemplar of genuine specialist.

Contents

Abbreviations

ADCS	Attitude Determination and Control System
AOCS	Attitude and Orbit Control Systems
CGP	Cold Gas Propulsion
DCM	Direction Cosine Matrix
DLR	German Aerospace Center
ECEF	Earth-Centered, Earth-Fixed
ECI	Earth-Centered Inertial
ESA	European Space Agency
ESTEC	European Space Research and Technology Centre
GNSS	Global Navigation Satellite Systems
GPS	Global Positioning System
iMQT	ISIS Magnetorquer Board
IMU	Inertial Measurement Unit
kRPC	Remote Procedure Call Server for KSP
KSP	Kerbal Space Program
LEO	Low Earth orbit
LQR	Linear Quadratic Regulator
MEMS	Micro-Electromechanical Systems
NASA	National Aeronautics and Space Administration
NED	North East Down
NGA	National Geospatial Intelligence Agency
NIR	Near-infrared
OBC	On Board Computer
PDR	Preliminary Design Review
PID	Proportional-Integral-Derivative controller
REXUS/BEXUS	Rocket and Balloon Experiments for University Students programme
SCARS	Spacecraft Control Architecture Rapid Simulator
SISO	Single Input Single Output
STK	Systems Tool Kit
TEC-ECN	Guidance, Navigation, and Control Systems Section
TLE	Two-line element set
VRML	Virtual Reality Markup Language
WWW	World Wide Web

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 [23]. The design of the control system was based on already existing models. Nevertheless, students with lack of previous practical experience in that field regarded it as a complex task. 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 aforementioned problems, a project life cycle is usually divided into distinct phases, allowing introduction of conducting product reviews within rigid time-frames. An example of such workflow, adopted by most major agencies such as ESA [2] and NASA [3], is a division of the project life cycle into phases, as presented 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, 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 need 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.

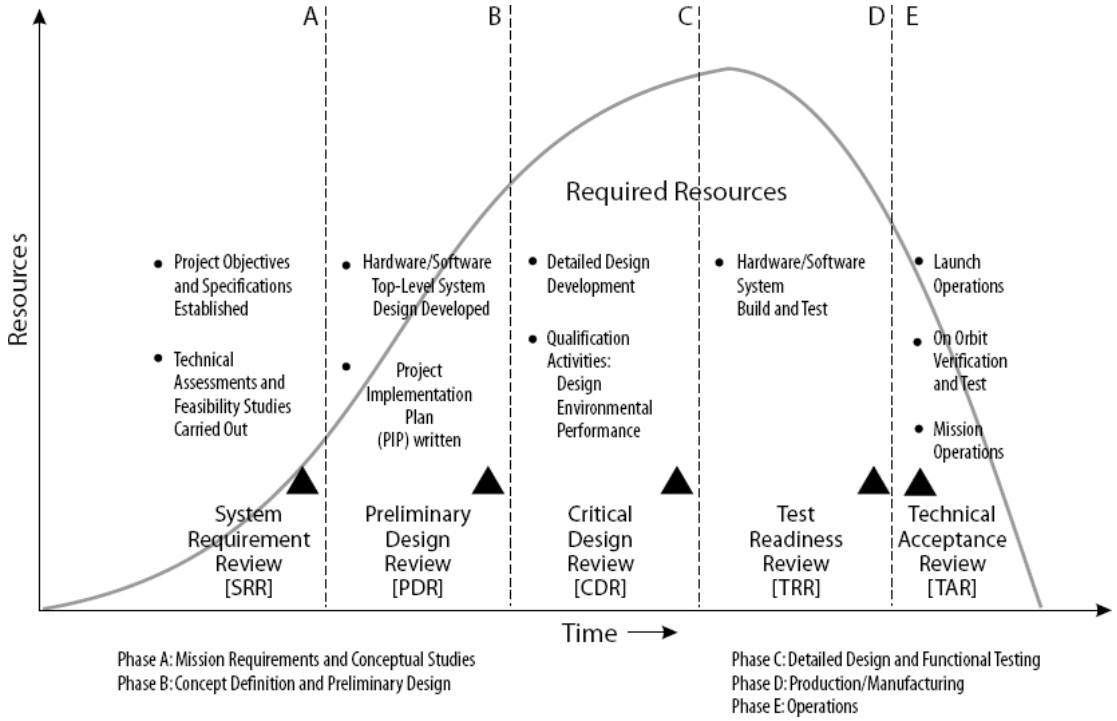


Figure 1.1: Typical space project phases and its life cycle [1]

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 that is a 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 details about the architecture, the features and methods of implementation of a toolbox created to fulfil the aims of the thesis - the Spacecraft Control Architecture Rapid Simulator (SCARS) Toolbox. In addition it includes the description of methods of connecting SCARS with various visualization tools. Chapter 3 explains the documentation and usage of SCARS, while in Chapter 4 contains examples of real life applications of the toolbox. 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 conducting initial design of spacecraft's Attitude Determination and Control System (ADCS), which means that is provides tools 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 cab be removed from the model while the students, for learning purposes, are 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 Section 2.1.

1.3 Already existing tools

The idea for a toolbox for spacecraft mission design and AOCS simulation is not a novelty. Various solutions are 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 despite the fact that the solutions for spacecraft prototyping are available, the need for open-source, easy-to-sue and modify toolbox still remains. Below a list of selected software solutions that fit the most objectives stated in Section 1.2 is stated, with explanation on the details, their features and the features which they lack and 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. This library provides tool for modeling motion and dynamics of CubeSats and nanosatellites. It provides the most basic features, e.g. the simulation of pre-set attitude scenarios, basic actuators and sensors models and integration with MATLAB's Virtual World visualization tools.

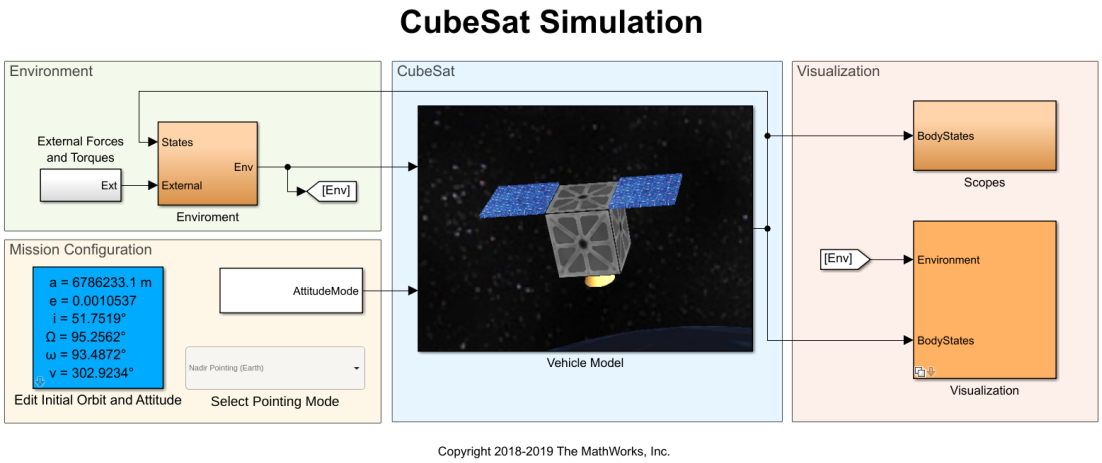


Figure 1.2: Top-level view of the example project of the MATLAB CubeSat Simulation Library

This library, while conceptually most similar to the SCARS, 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 reducing the number of layers of abstraction between the user and the simulation, but also for tasks like calculation of energy expended by the actuator. Moreover, 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[16].

1.3.2 PrincetonSATELLITE Spacecraft Control Toolbox

Figure 1.3: PrincetonSATELLITE Systems logo [22]

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, accessible documentation and additional modules for unique applications like formation flying, fusion propulsion or solar sails. The toolbox is very robust, allowing the user to conduct long term simulations, but is also useful for short term simulations, like maneuver analysis and launch simulation. PrincetonSATELLITE Toolbox is a versatile and comprehensive tool and 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 [17].

1.3.3 PROPAT Toolbox

PROPAT 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 [24].

1.3.4 GAST Toolbox

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

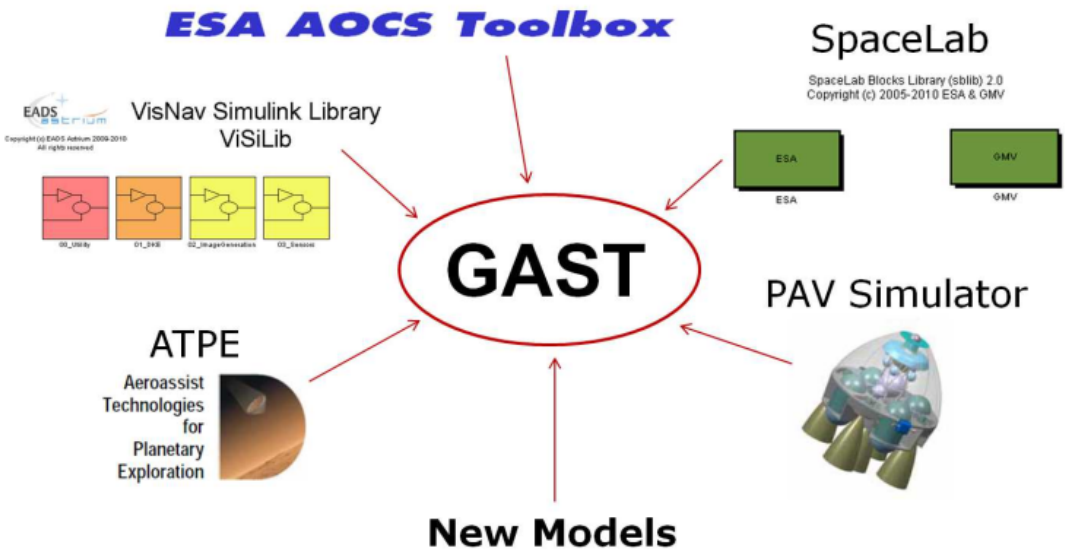


Figure 1.4: Representation of the consolidation of TEC-ECN toolboxes

for the GAST toolbox according to the needs of the section. Figure 1.4 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 wider audience[25].

1.3.5 User-created modules available on MathWorks MATLAB Central

MATLAB is one of the most popular scripting language between engineers, and along with Simulink package it provides tools helpful for simulating mechanical systems. Therefore, numerous modules and packages created by the users can be found online.

MATLAB Central is a network for asking questions about MATLAB software, discussing solutions and sharing MATLAB and Simulink solutions and files[26]. On a subsection called File Exchange many files are 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[27].

Satellite Orbit Modeling

A collection of MATLAB scripts used for modelling of satellite's perturbed motion with special perturbations approach. Despite being very robust, as it can be applied to any problem in celestial mechanics, this module is useful for orbit modeling, not AOCS system design[28].

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[29].

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*. As educational means, the exercises are interesting an refer at least partially to the problem of control system design - mainly GPS sensor. Yet this is not a toolbox by any means[30].

Apollo 11 Moon Landing - 50th Anniversary Model

This example shows how the engineers who worked on the Apollo Lunar Module digital autopilot design could have used 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[31].

1.3.6 Overview

To summarize, software which helps spacecraft control engineers definitely is available. Yet, there is no solution which would have all the requirements that the product of this thesis tries to fulfil. The list of features that could be expected from a tool discussed in Section 1.2 is presented in Table 1.1, with comparison of their inclusion in the examined software.

Feature	**MATLAB CubeSat Simulation Library**	**Princeton-SATELLITE Spacecraft Control Toolbox**	**PROPAT Toolbox**	**GAST Toolbox**	**Smart Nanosatellite Attitude Propagator (SNAP)**
Orbit propagation	Yes	Yes	Yes	Yes	Yes
Mission planing	No	Yes	Partial	No	No
Actuators and sensors model	No	Yes	No	Yes	Only permanent magnets
Sensor fusion	No	Yes	No	Yes	No
Control algorithms	Partially	Yes	No	Yes	No
Environment simulation	No	Partial	No	Yes	Yes
Parts database	No	No	No	Yes	No
Availability	With MATLAB Aerospace Blockset	Fully commercial	Free online	Not available	MATLAB File
Documentation available	Partial	Yes	Yes	Partial	No
Open source	Partially	No	Yes	No	Yes, apart from MATLAB back end

Table 1.1: Comparison of features included in various software

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 fulfill the main objective of this thesis, that is to provide the community of beginner control engineers with a satellite control system prototyping toolbox, a self-made solution is proposed. This chapter provides the insight into the architecture of SCARS Toolbox, a software framework created for purposes of this thesis in MATLAB and Simulink. First the main objectives of that solution are stated, then architecture of 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, for example such as the size of the satellite, thrusters operational range, and initial mission parameters like 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 by itself covers the first two objectives of the the thesis. The following listing further specifies what should be expected of the end product and what features 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;
- Modeled sources of environmental forces and torques, including most sources most relevant for small satellites;
- 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 fulfill the objective of accessibility and ease of modification, MATLAB family of software was chosen as a framework for developed toolbox. MATLAB is one of the most popular scripting language and with the addition of Simulink software it can become a 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 2.11.3) a Python script acting as a dataflow bridge was used, as it was the simplest method to solve a problem described in that chapter. Several other software solutions were used for visualization purposes, with the reasoning presented in Section 2.11.

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 models of various complexity and for multiple scenarios. The latter, a Modular Simulation, can be set up with either MATLAB command line scripts to represent user's spacecraft.

2.3.1 Parts Library

SCARS Parts Library is a ready to use Simulink Custom Library - 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 Models, Control Algorithms, Actuators, Sensors, Environment, Visualization, Example scenarios and Other blocks.

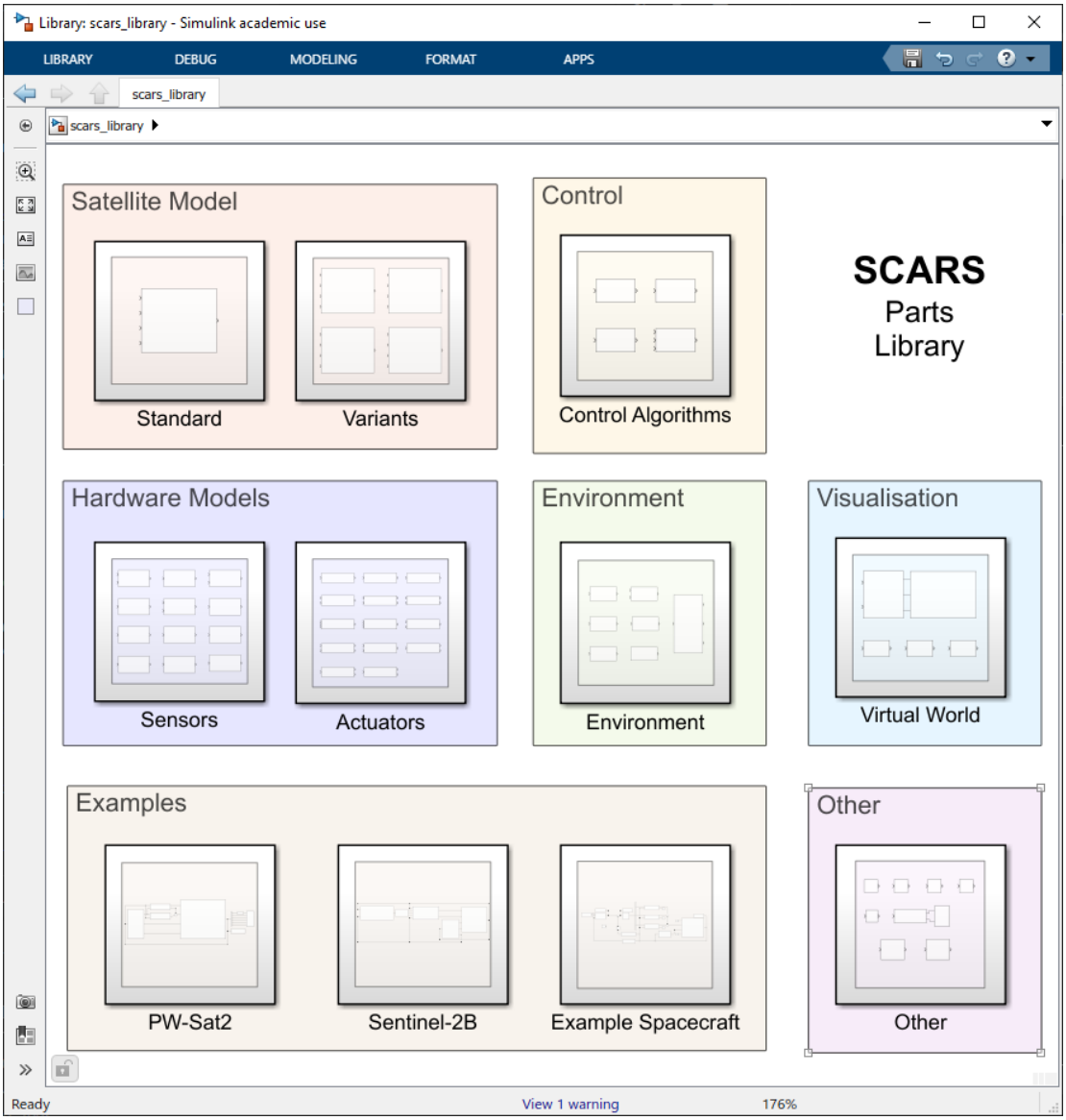


Figure 2.1: SCARS Parts Library screenshot

2.3.2 Modular Simulation

SCARS Modular Simulation is a ready-made Simulink model constructed to provide the user with setup containing complete simulation of the spacecraft. The model can be initialized with prepared script described in Section 3.2. 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 or geographical coordinates with velocity and rates in body axes. In the same manner, all actuators and sensors available in SCARS library can be connected to act on the spacecraft.

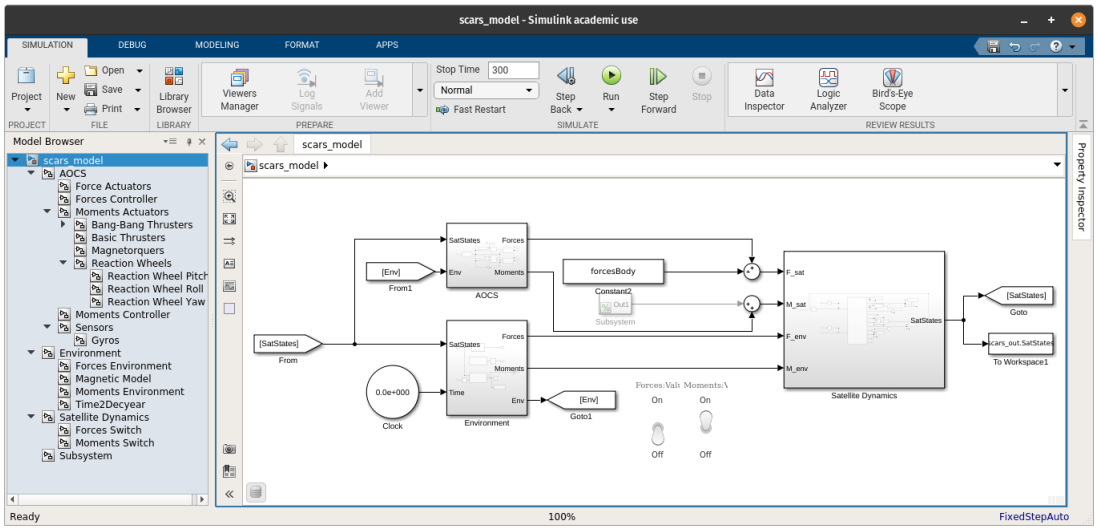


Figure 2.2: SCARS Modular Simulation screenshot

2.3.3 Main Signal Buses

To unify the signals transferred to blocks modeled in SCARS Toolbox a pair of Simulink signal buses is proposed - **SatStates** bus, produced as the output of **Satellite Dynamics** block and **Env** bus, collected as an output of **Environment** block. Since the buses are unified, there is no need for unit or coordinate transformations between, for example, satellite model and sensor model. Following tables present descriptions of signals contained in designed buses. Mentioned reference frames are explained in Section 2.5.

Name	Unit	Description	Size
V_ECEF	m/s	Velocity of the body in ECEF frame, in relation to ECEF reference frame	1x3
X_ECEF	m	Position of the body in ECEF reference frame	1x3
lla	deg, deg, m	Body latitude, longitude and altitude in reference to Earth's Geographical coordinates	1x3
Euler_NED	rad	Body rotation angles in relation to NED reference frame	
DCM_ECI2B	-	Direction Cosine Matrix describing rotation from ECI frame to body frame	3x3
DCM_NED2B	-	Direction Cosine Matrix describing rotation from NED frame to body frame	3x3
DCM_ECEF2NED	-	Direction Cosine Matrix describing rotation from ECEF frame to NED frame	3x3
DCM_ECEF2B	-	Direction Cosine Matrix describing rotation from ECEF frame to body frame	3x3
V_B	m/s	Body acceleration in relation to its inertial reference frame	1x3
Omega_NED	rad/s	Body rotation rate in reference to NED reference frame	1x3
Omega_B	rad/s	Body rotation rate in reference to its inertial reference frame	1x3
Euler_B	rad	Body's rotation angles in relation to its initial position	1x3
dOmega_B/dt	rad/s^2	Derivative of body rotation rate in reference to its inertial reference frame	1x3
A_B	m/s^2	Body acceleration in relation to its inertial reference frame	1x3

Table 2.1: **SatStates** bus signals description

Since **SatStates** bus contains ideal values of satellite states, it is used by environment models and models that include mechanical relations between satellite's frame and their hardware components. For other applications, like creation of control loop, it is advised to use sensor models instead.

Name	**Unit**	**Description**	**Size**
Magnetic Field [nT]	nT	Strength of the magnetic field on Earth's orbit, depending on altitude, position and time	1x3
Environment Force [N]	N	Force acting on a spacecraft - a sum of gravitational pull and atmospheric drag	1x3
Sun's Position [km]	km	Position of the Sun with reference to Earth, in ECEF frame	1x3
Atmosphere Density [km/m3]	kg/m^3	Density of atoms in partial atmosphere at body's altitude	1

Table 2.2: **Env** bus signals description

Env signal bus contains information about parameters of the environment at spacecraft coordinates. Although the values are derived from used models, they should be treated as ideal values. Therefore they are used for applications where the parameter directly influences the bahaviour of the model, like in magnetorquer model described in Section 2.8.4. For comparison, in B-Dot controller described in Section 2.10.3 it is suggested to use a sensor model instead and place it between **Env** block and actuator or On Board Computer (OBC) model.

2.4 Spacecraft Dynamics

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 dynamics following equations can be used:

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

Equation 2.4 describes translational motion of the spacecraft on the orbit, where $n = \sqrt{g/R}$, R is radial distance from attracting center, g is gravitational force per unit mass, m is spacecraft mass and T_x, T_y, T_z are thrust components. On the other hand, for small attitude changes of non-spinning spacecraft with respect to inertial frame, equations describing rotational motion can be linearized. The acquired system is presented on Equation 2.4:

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

Where T_x, T_y, T_z are torque components and I_x, I_y, I_z describe satellite's inertia in given axes.

One of the aims of this thesis is to create means to design a satellite control system for users 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 2.10.2. One could try to further develop Equations 2.4 and 2.4 into state-space representation of the 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 such solution is that it also encompasses all changes from the users to the SCARS Modular Simulation. The process is described in Section C.

To provide the user with an ability to include a on-orbit dynamics model in their simulations, **Satellite Dynamics** block was designed. This module is built around Simulink Aerospace Blockset **6DOF ECEF (Quaternion)** block. SCARS provides a mask for said block, which allows the user to set up the parameters of the spacecraft and initial conditions for the simulation, as showed in Section 4.1.

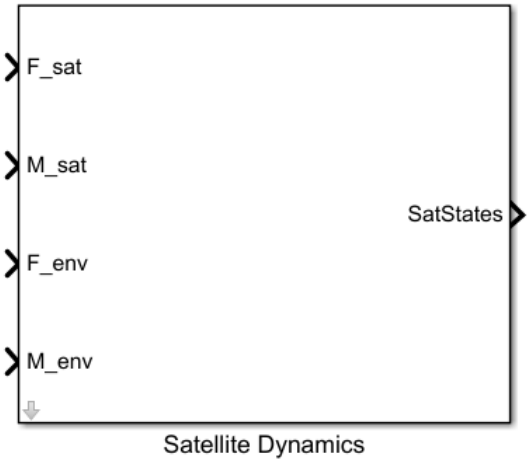


Figure 2.3: **Satellite Dynamics** SCARS block

The inputs to **Satellite Dynamics** block, as seen in Figure 2.3, are `F_sat` and `M_sat`, which respectively correspond to forces and torques acting on the satellite in its body reference frame, and `F_env` and `M_env`, which are in Earth-Centered, Earth-Fixed (ECEF) reference frame. Such setup allows easy connection between this and **Environment** block, described in Section 2.7. The output of this subsystem is a bus signal, described in Section 2.3.3.

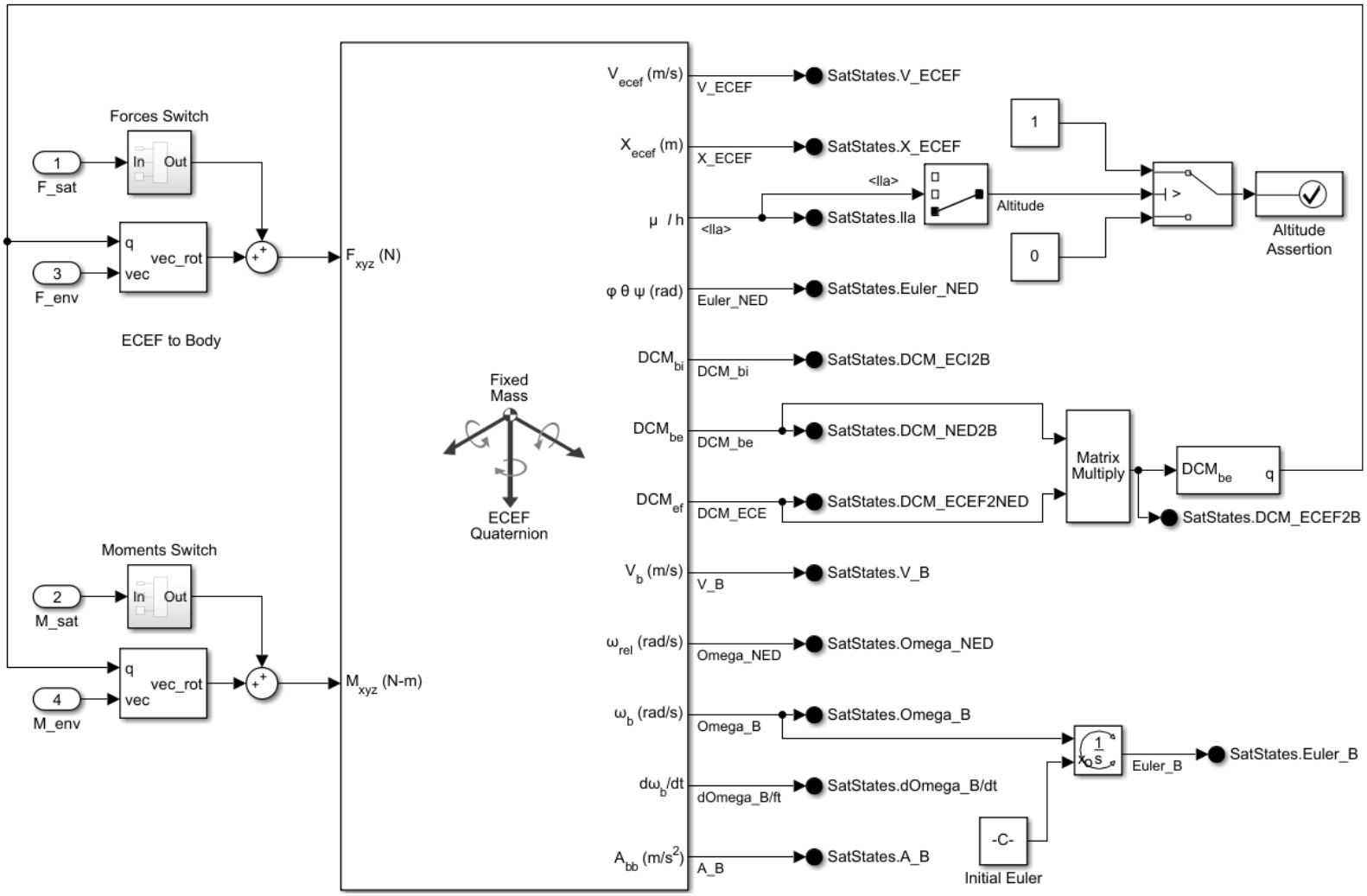


Figure 2.4: Contents of **Satellite Dynamics** SCARS block

2.5 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 presented in Figure 2.5.

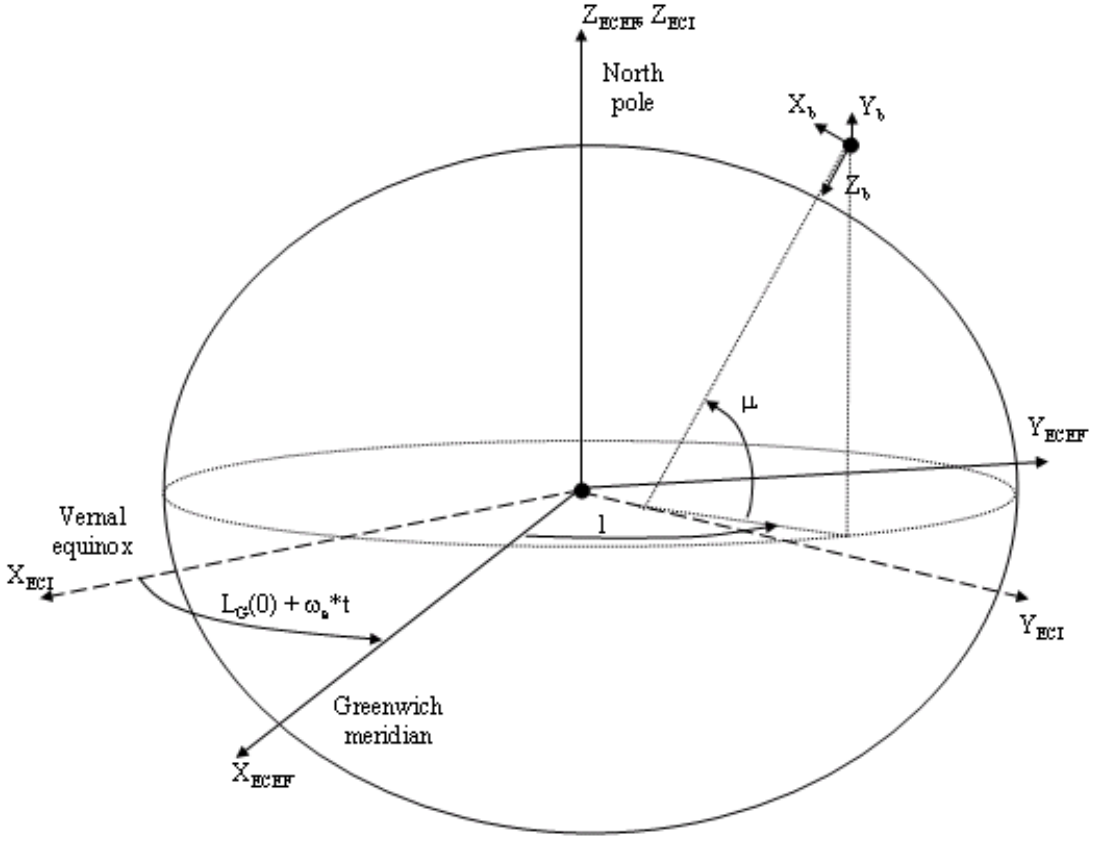


Figure 2.5: Satellite Reference Frames[11]

2.5.1 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.

2.5.2 North, East, Down

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.

2.5.3 Earth-centered Inertial

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.

2.5.4 Earth-centered, Earth-fixed

The ECEF, 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 $\mathbf{v_{ECEF}}$.

2.6 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:

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

Due to complexity of coordinate transformations, 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.

2.6.1 ECI position and velocity vector to Keplerian elements

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.

2.6.2 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:

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

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

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[4].

2.6.3 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:

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

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

2.6.4 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:

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

2.6.5 ECEF to LLA

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

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

Yet 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.

2.7 Environment

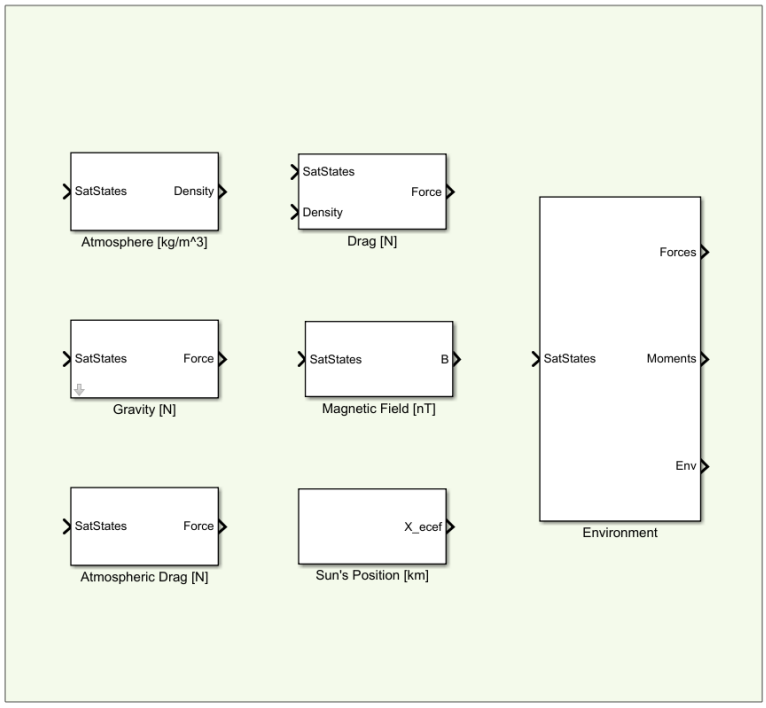


Figure 2.6: SCARS Parts Library Environment module blocks

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. All models available in SCARS Parts Library are also collected in a single, unified block called **Environment**, as seen on Figure 2.7, and use bus described in Section 2.3.3 as an output. These solutions allow convenient implementation of Environment module in user's model, but can negatively affect computing time.

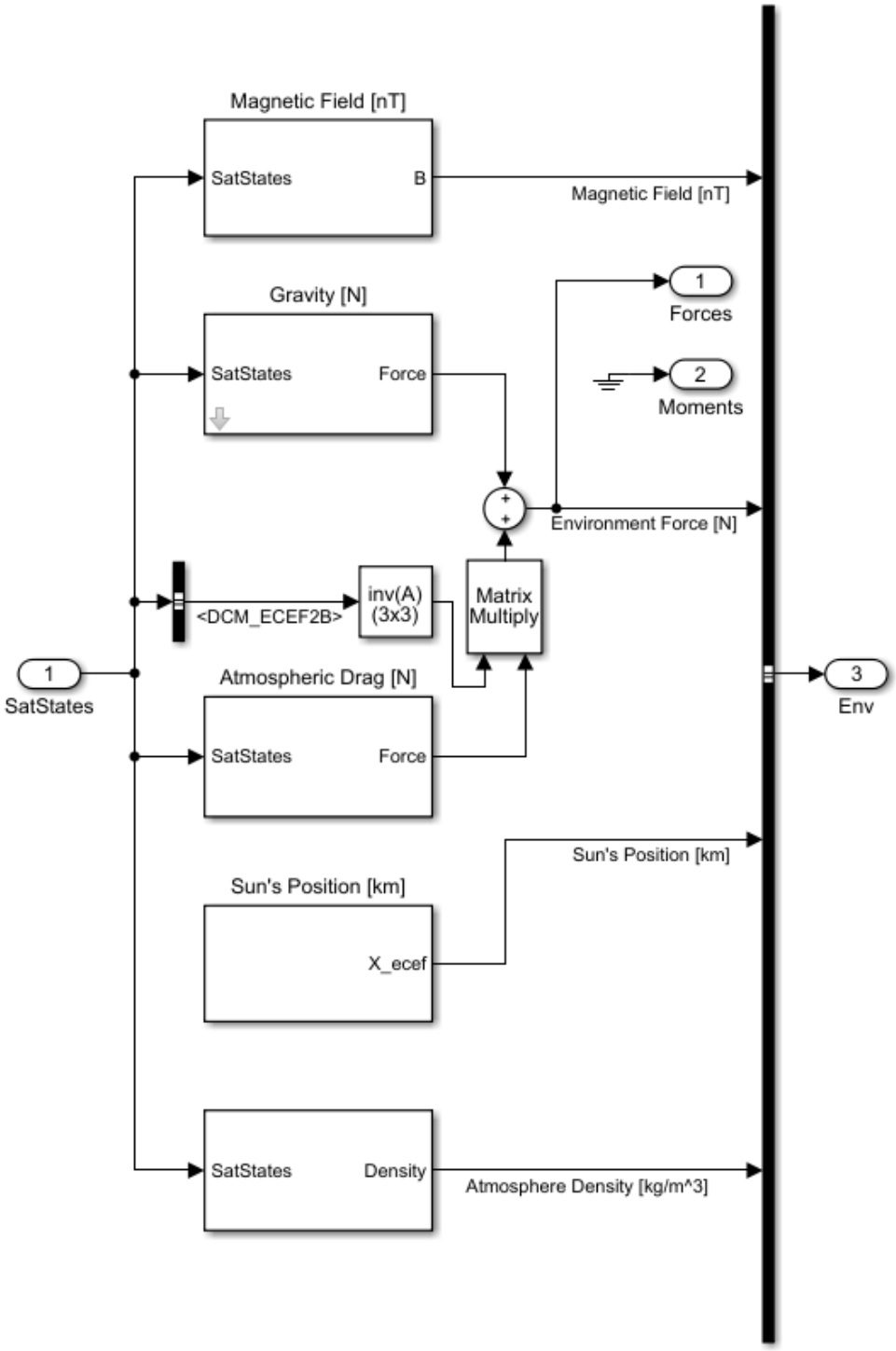


Figure 2.7: Contents of SCARS **Environment** model

2.7.1 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 provide possibly best approximation between observed and calculated orbit. Lagendre polynomials of following form:

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

Are called the zonal terms and Lagendre functions and ones of this form:

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

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 be added together to faithfully model the influence of Earth's gravity field on the spacecraft, it was decided to use a ready model from Simulink Aerospace Blockset - the Spherical Harmonic Gravity Model, with EGM2008 planetary model, due to its accuracy and higher fidelity.

2.7.2 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 dioxide. 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. Yet 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.

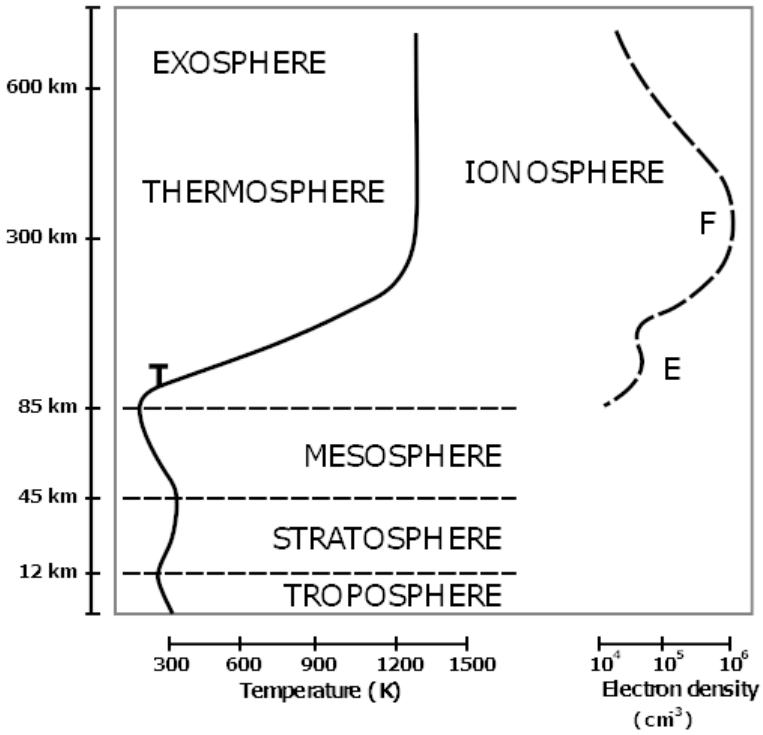


Figure 2.8: Model of Earth's atmosphere layers

The reference atmospheric model used in SCARS is NRLMSISE-00, which takes date and position of the object in geographic coordinate system as input and returns temperature and density of the atmosphere components as output. 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 above this threshold atmospheric forces are set to zero.

2.7.3 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 charging 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 ephemeris 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 the first parameter and simulation duration, in seconds, as the second parameter. It is then implemented in Simulink as a lookup table and whole model is masked for ease of usage.

2.7.4 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, it may be necessary to use a high fidelity model. It was decided 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.

2.8 Actuators

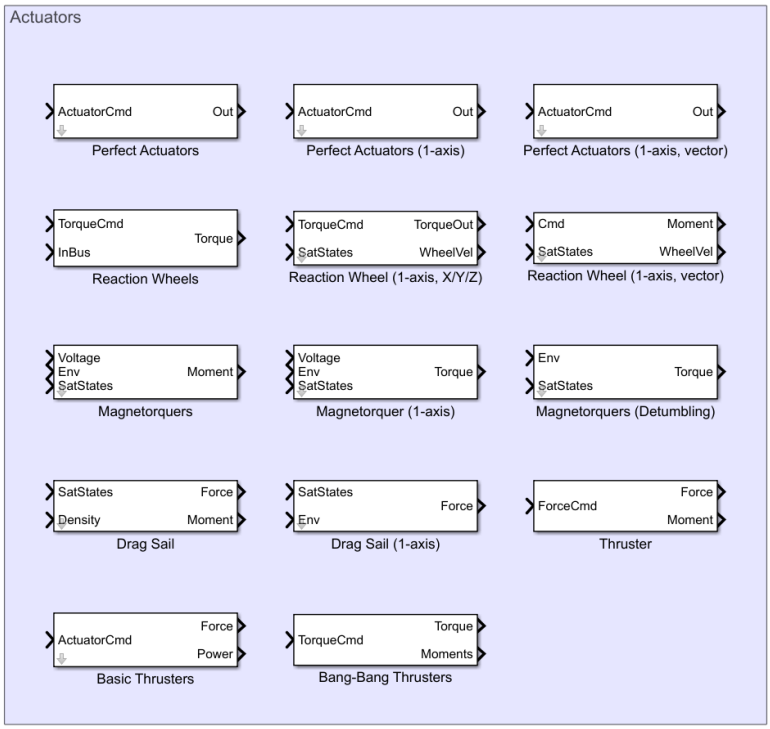


Figure 2.9: All Actuators blocks available in SCARS Parts Library

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 Section C allows for using all provided actuators with all control methods described in Section 2.10.

2.8.1 Ideal and Simple Actuators

Ideal Actuators are simply Simulink subsystems including unit gain. Their purpose is to serve as a placeholder, if other parts of ADCS subsystems are tested and simulation of actuator behavior is not necessary.

Simple Actuators are ought to simulate most generic sources of errors in actuators, for the user to be able to create a more reliable placeholder for actuator not yet available in SCARS Toolbox.

2.8.2 Thrusters

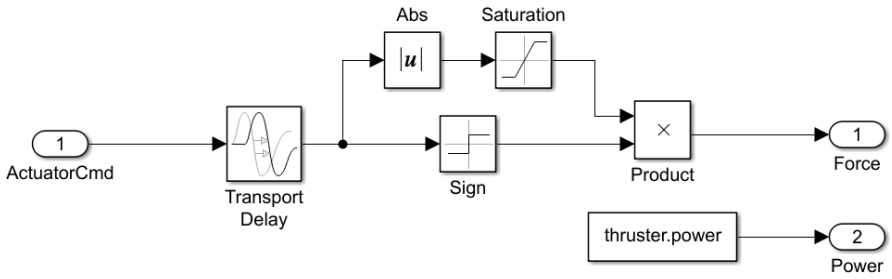


Figure 2.10: Directional Thruster model

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 compressed 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 is no 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 nanosatellites.

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.

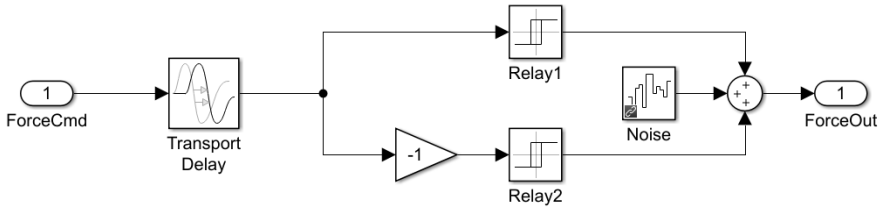


Figure 2.11: Bang-Bang Thruster model

For all thrusters models the only input is the control signal, while the outputs are fuel consumed and either generated force or torque. CGP Systems also have a downside of decreasing thrust profile in relation with time, since thrust is correlated with the pressure of the propellant inside a tank. This property is not yet modeled in SCARS Toolbox.

2.8.3 Reaction Wheels

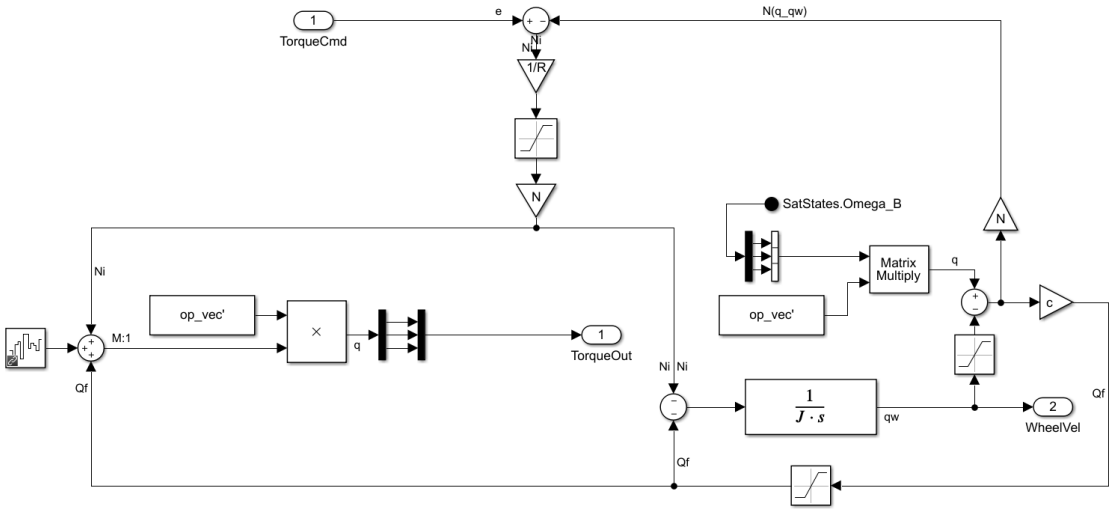


Figure 2.12: SCARS Reaction Wheel model

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 very precise attitude maneuvers, with the possibility to eliminate most disturbance torques. Reaction wheels operate at around 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 around each spacecraft body axis can be considered to be decoupled from motion around 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 as it can be seen on Figure 2.12.

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.

2.8.4 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:

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

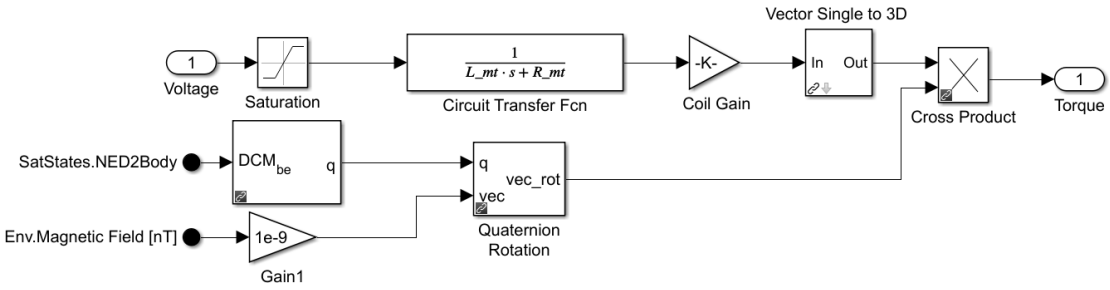


Figure 2.13: SCARS Magnetorquer model

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 Equation 33 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, where L is the solenoid's inductance and R its resistance:

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

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 do not allow precise maneuvering on higher orbits.

2.8.5 Drag Sail

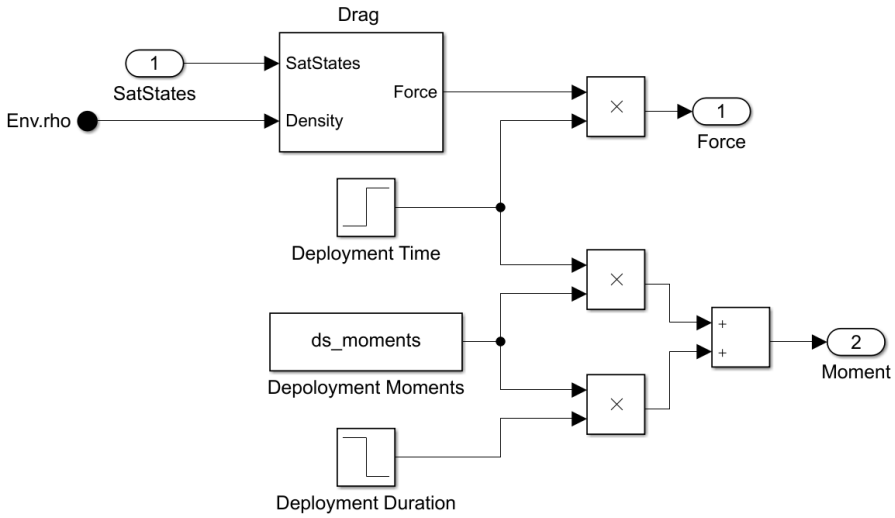


Figure 2.14: SCARS Drag Sail model

Drag sails use the occurrence of partial atmosphere (described in 2.7.2) 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 exclusively 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. Currently the moment of sail's deployment is not simulated in SCARS toolbox.

2.9 Sensors

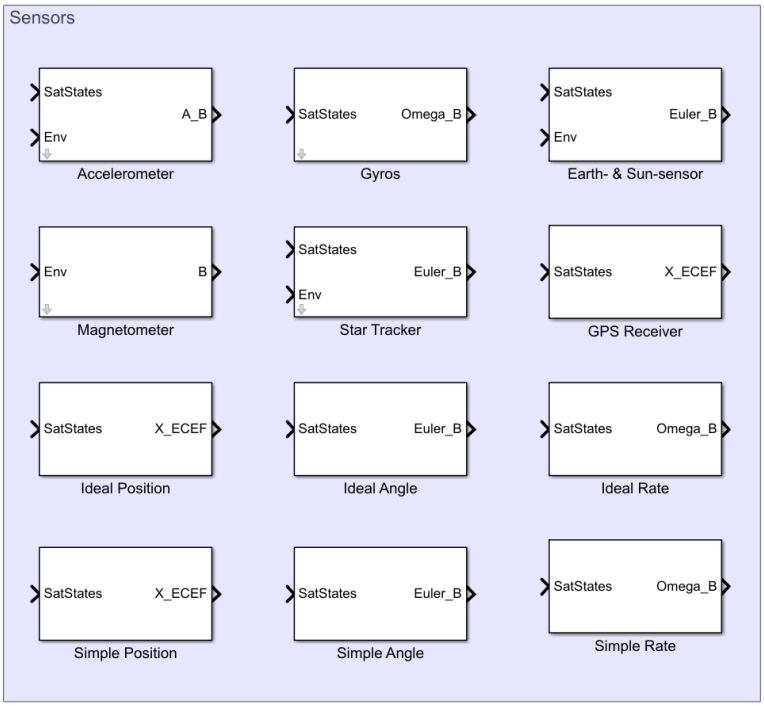


Figure 2.15: All Sensors blocks available in SCARS Parts Library

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.

It is important to mention, that it is possible to model sensors with various degrees of fidelity and different focus. For example, the influence of mechanical parameters on output signal of gyroscope is significant, resulting in a need for modeling it with transfer function describing its properties. On the other hand, in sensors such as star tracker the output is mostly processed in the software, therefore the

focus is put on modeling the influence of spacecraft kinematics on the sensor, such as blinding the camera by the sun.

2.9.1 Ideal and Simple Sensor

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

Simple Sensor does not model any specific type of sensor. It takes most common 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.

2.9.2 GPS Receivers

The Global Positioning System (GPS) is a Global Navigation Satellite Systems (GNSS) owned adn operated by United States government. It allows determining position, velocity and time with 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* [32], allowing 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. To create the model, MATLAB's Navigation Toolbox function, `gpsSensor()`, was nested inside a masked Simulink block. User can set all aforementioned 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.

2.9.3 Accelerometers

Accelerometers are force sensors, most often paired with gyroscopes as a part of Inertial Measurement Unit (IMU) board. To find acceleration three sensors are located with their axes mutually orthogonal. The force measured is external to the

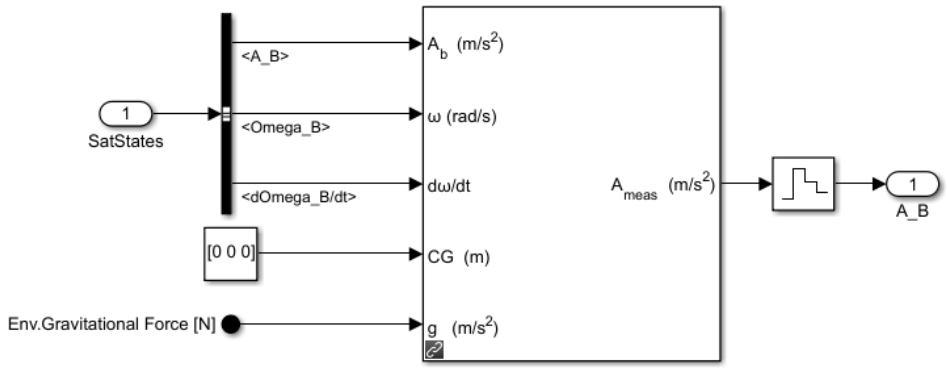


Figure 2.16: SCARS Accelerometer model

board (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.

2.9.4 Magnetometers

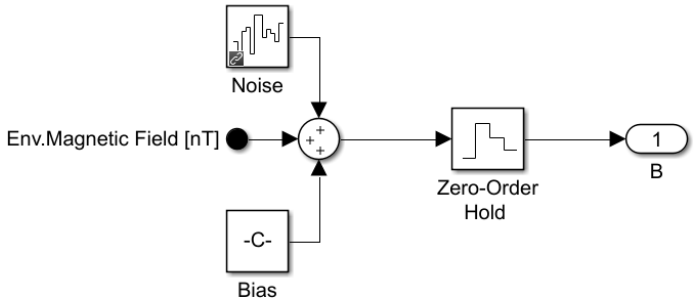


Figure 2.17: SCARS Magnetometer model

To make 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 con-

ducts transformation from measured vector to one of the reference frames used by ADCS subsystem, providing information about its attitude. Due to their light weight, low power consumption and have wide margin for operating temperature, they are a relative sensors to be included on board of a small spacecraft.

The input for this block is magnetic field strength and the output is measured signal, both in nT unit.

2.9.5 Gyroscopes

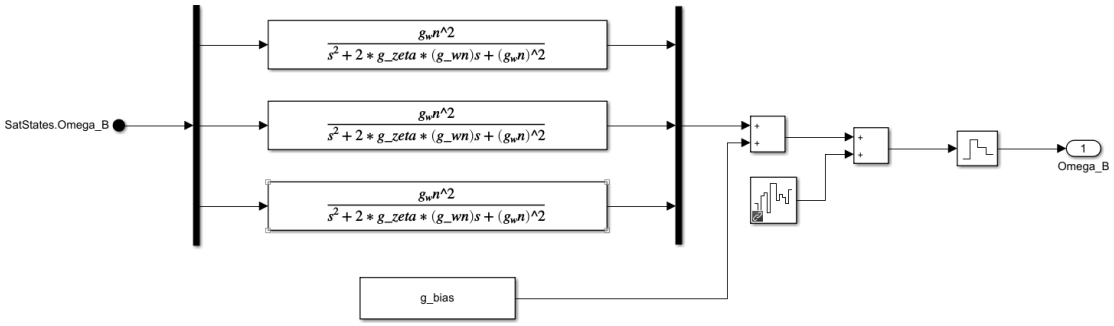


Figure 2.18: SCARS Gyroscope model

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 using 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] In vibrating gyroscope the Coriolis effect is causing 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 advantages 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 compared to pricier alternatives.

In SCARS the gyroscope model, as presented on Figure 2.18, includes a second order transfer function, describing the system using parameters such as natural

frequency, bandwidth, damping ratio, but also it includes gyroscopic bias and noise source, in attempt to transfer ideal gyroscope into real sensor.

2.9.6 Star Tracker

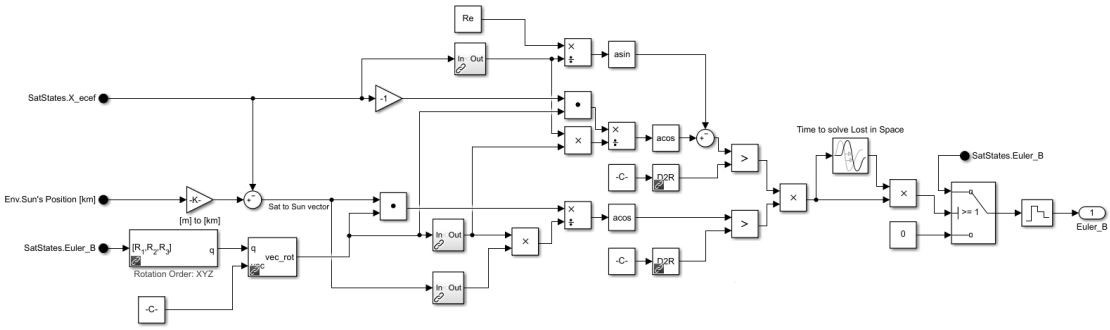


Figure 2.19: SCARS Star Tracker model

A star tracker is a complex attitude sensor, providing the most accurate determination within available commercial solutions. It consist of optical camera and wide array of processing algorithms, which allow to read the position of the stars from captured image and to compare that positions to database of known and visible stars to find the attitude of the spacecraft. Moreover, it can do so without a-priori knowledge, using lost-in-space algorithm [19].

Simulating stars position to achieve high fidelity star tracker model and using parameters such as camera resolution and algorithm accuracy and precision is a complicated computational problem. Therefore it was decided to assume ideal output from star tracker and to focus on mechanical problems, such as blinding the sensitive camera sensor by light from the Sun or reflected by the Earth. The model, presented in Figure 2.19, the model calculates the relative position of Sun and satellite, then Earth and satellite and based on this, and two key parameters input by the user - Sun and Earth exclusive angle, provides the actual output or null value, if the star tracker is blinded. Additionally, if the parameter is non-zero given, the model adds the delay of time it takes the software to solve the lost-in-space problem to the null value output duration.

2.10 Control Methods

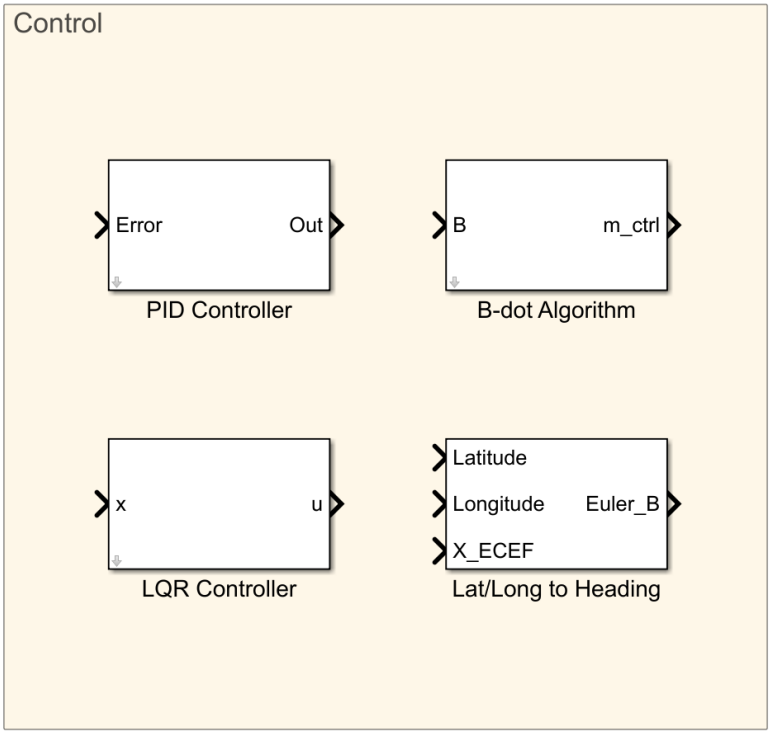


Figure 2.20: All Control blocks available in SCARS Parts Library

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

2.10.1 PID Controoler

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. It can be described with a transfer function in Laplace domain, as seen on Equation 2.10.1

$$L(s) = K_p + \frac{K_i}{s} + K_d s \tag{38}$$

Where K_p, K_i and K_d are, respectively, proportional, integral and derivative gains of the controller. Most relevant application of the PID controller is to minimize the error between reference state and actual state of the plant it controls. Within SCARS, the most common use is to provide input signals for the actuators, whether it is required thrust or moment, or in more specific cases, driving voltage.

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.

2.10.2 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 2.10.2 to generate static gain matrix K.

$$cost = \int x^T Q x + u^T R u \tag{39}$$

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:

$$u = -Kx \tag{40}$$

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 Section 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. The gain matrix is then calculated with MATLAB `lqr` function.

2.10.3 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:

$$M = -k\dot{B} \tag{41}$$

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

2.11 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.

2.11.1 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 control of driving display or animation with MATLAB variables and Simulink signals. Moreover, the toolbox is integrated with VRML viewer and VRML editor, allowing 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:

$$\mathbf{r_{VRML}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \mathbf{r_{ECEF}} \tag{42}$$

To calculate spacecraft's rotation vector **rot** as required by VRML, one has to transform ECEF to Body direction cosine matrix into quaternion $[q_0 \ q_1 \ q_2 \ q_3]$

(scalar first) and then use the following equation:

$$\mathbf{rot} = \begin{bmatrix} q_3 \\ q_1 \\ q_0 \\ 2 * acos(-q_2) \end{bmatrix}^T \tag{43}$$

Figure 2.21 the example of Virtual World visualization is provided. The animation is set up so that the user can move the camera around, or they can choose (visible in top-left corner) a "Sat Cam" viewpoint, which follows the satellite translation and rotation.

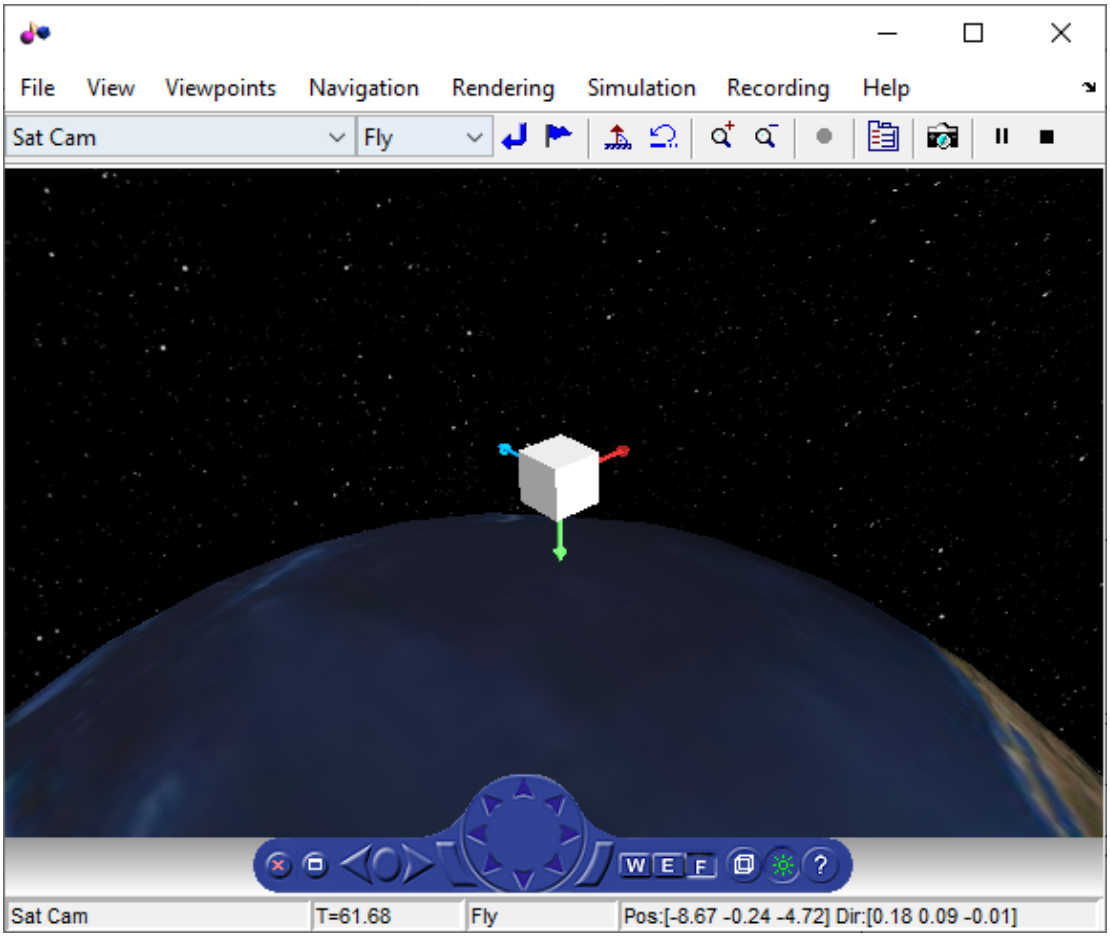


Figure 2.21: Example of Virtual World satellite visualization

2.11.2 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 which allows 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 generates such files for the user, with predetermined format according to STK documentation [33]. 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) they 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> <Y> <P> <R>
```

Where time is formatted in same manner as in ephemeris file and the following parameters consist of yaw, pitch and roll Euler angles given in degrees.

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

Data produced by a simulation of a simple model, consisting only of **Environment** and **Satellite Dynamics** blocks created with SCARS Toolbox was exported into `.e` and `.a`, from which first lines are shown in mentioned Appendixes. The results were imported into STK and can be seen on Figure 2.22 and Figure 2.23.

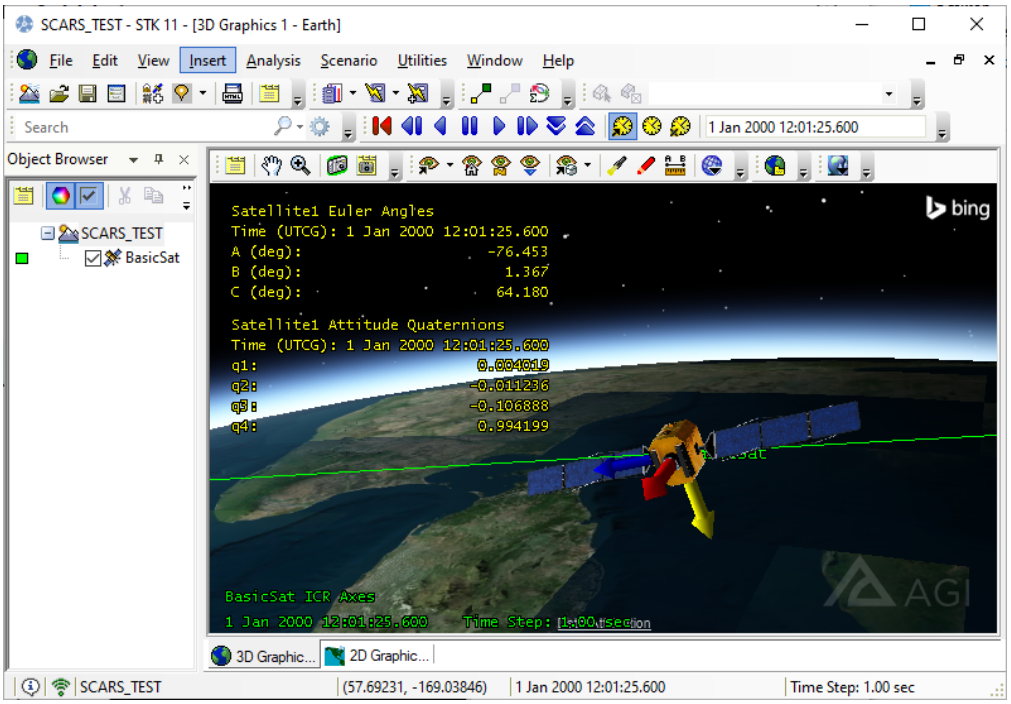


Figure 2.22: Example of STK 3D satellite visualization

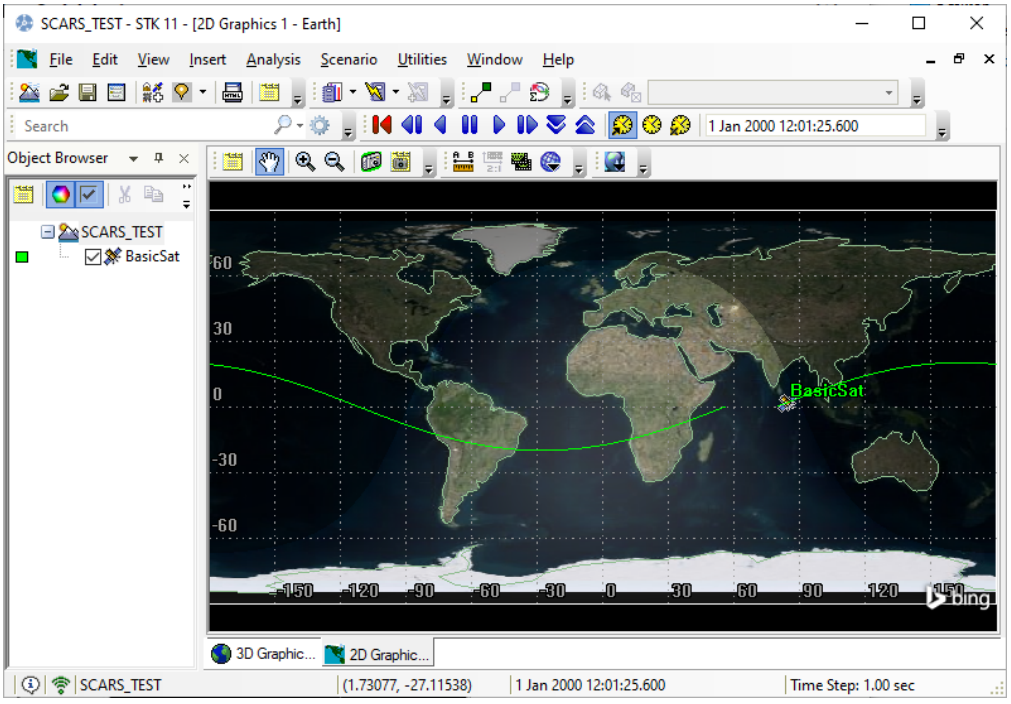


Figure 2.23: Example of STK ground track representation

2.11.3 Kerbal Space Program

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[34].

The connection between MATLAB and KSP is possible because of fanmade Remote Procedure Call Server for KSP (kRPC) mod. It creates an 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 is a difficult task, as MATLAB does not 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.

3 SCARS Documentation

The purpose of this chapter is to provide the user with information about SCARS Toolbox documentation. While most of the structure and architecture of SCARS is described in Section 2.3, the purposes and operation principles of major blocks are shown in Section 2.4, 2.7, 2.8, 2.9, and the instructions on how to use it are provided in Chapter 4, some additional explanation might be necessary.

In following sections contain the descriptions of folder structure in SCARS, the scripts which are shipped along Simulink models and how SCARS uses Simulink model masks to provide easily accessible documentation to the user.

3.1 Folder Structure

Main folder in which SCARS toolbox is downloaded is called `scars_toolbox`. It contains `scarsLibrary.slx` - a file with SCARS Parts Library, `scarsModel.slx` - Simulink SCARS Modular Simulation model and `scarsProject.prj` - a MATLAB project file[38] which allows the user to launch and initialize SCARS Toolbox using just one file.

In addition to that, under `scars_toolbox` following directories are located: `examples` - Directory containing example simulations presented in Chapter 4; `resources` - Directory containing files used by SCARS scripts and Simulink models; `scipts` - Directory containing files described in Section 3.2.

3.2 MATLAB Scripts

Along with Simulink models, SCARS is shipped with few MATLAB scripts. Their aim is to automate menial tasks that have to be performed by the user. Since the scripts guide the user with interactive prompts, there is no need for detailed documentation, just explanation of their purpose, presented in the list below:

- `scarsSetup` - Once run it provides the SCARS Modular Simulation with all necessary variables. Can be copied and edited to initialize the model with parameters chosen by the user.
- `getSunPosition` - Provides **Sun's Position [km]** block with arrays for lookup tables. Has to be run before using said block.
- `createSTKFiles` - Exports data from SCARS simulation into STK ephemeris and attitude files.

- `getSISOSystem` - Changes the system linearized with Simulink Model Linearizer into Single Input Single Output (SISO) system. Walks the user through the process with series of prompts, hence initially it doesn't require any parameters.

3.3 Simulink Models Masks

Each model included in SCARS Parts Library is masked with Simulink mask[35]. This allows the user to open custom interface for selected SCARS block, with editable fields for each parameter used in the setup process of that part. Most importantly, masks contain the description of the block. In case of SCARS, if the block represents a piece of hardware, the description contains short explanation of principle of operation of said part, its purpose within AOCS subsystem, notes about parameters, and description of block's inputs and outputs. In this way, majority of SCARS documentation is attached to the model itself, where it is easily accessible. An example of SCARS block mask can be found in figure Figure 3.1.

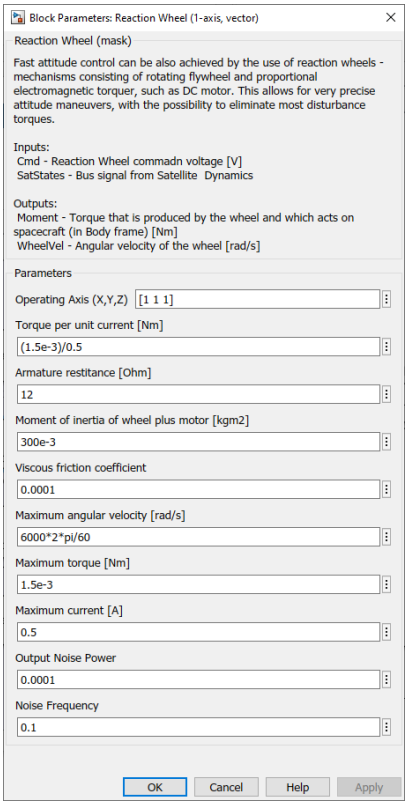


Figure 3.1: SCARS Reaction Wheel block mask

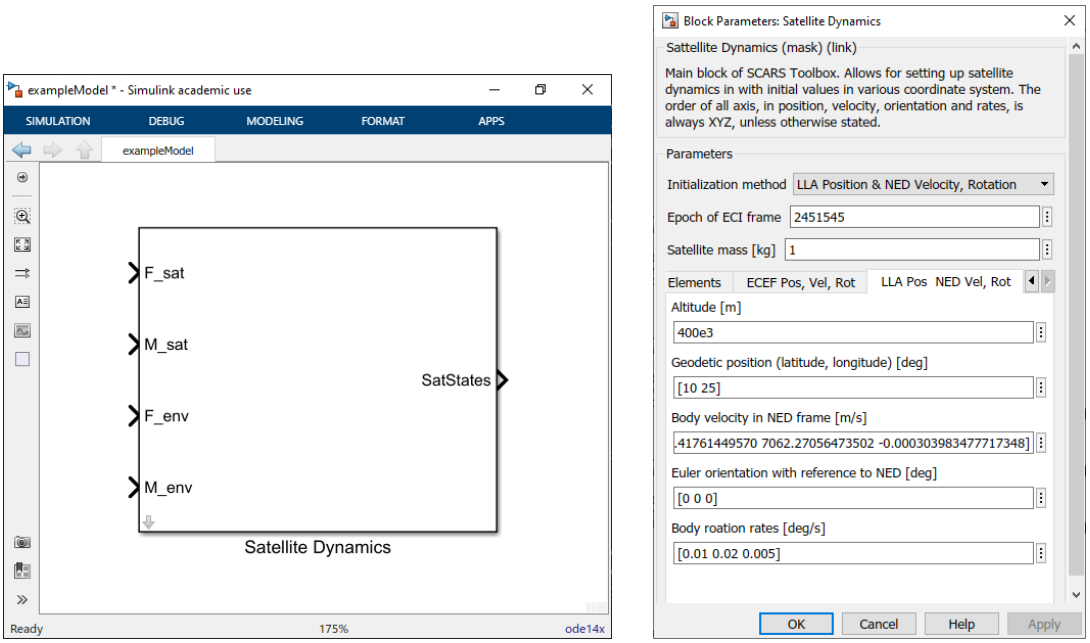
Also, Simulink masks contain pieces of MATLAB code, which is executed after adding or changing the block. This code allows further customization of the initialization process. For example, **Reaction Wheel (1 axis, vector)** model transforms the vector given as axis of operation into its norm, "sanitating" user's input.

4 Case studies

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, of which AOCS Subsystems can be simulated with blocks from SCARS Parts Library. At the end of the chapter examples of control system tests are presented, proving that SCARS can be used for both prototyping and reviewing processes.

4.1 Simple spacecraft example

The nominal usage of SCARS Toolbox is to take a simple objective that designed AOCS subsystem has to fulfil, choose 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 ± 10 arcminutes accuracy during 10s of geographic coordinates tracking**. The model constructed for this example will be further referred to as Example Model.



(a) Example Model (b) Spacecraft Dynamics Mask

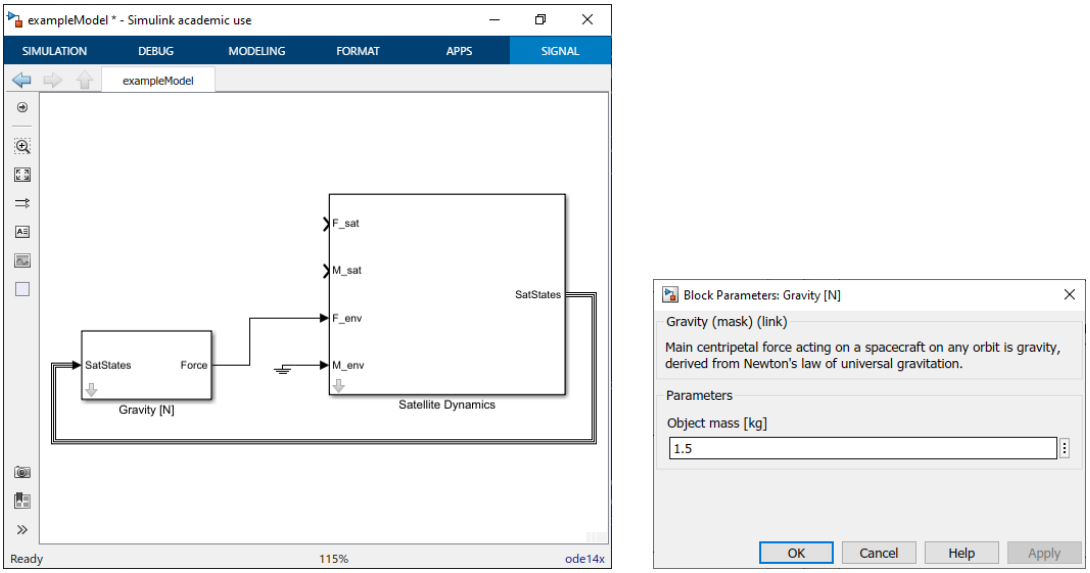
Figure 4.1: Example Model creation: Step 1

Step 1: Initial Spacecraft Dynamics setup

To set up the spacecraft as a point in orbit spacecraft dynamics model the user needs 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. Several initialization methods are available, corresponding to reference frames in which the user can input the data about 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, is presented in Figure 4.1 (b). Initial latitude and longitude were chosen arbitrarily, while still in the neighborhood of the point to be tracked.

Afterwards the user can set up Simulink model solver parameters to **Fixed-step** with **ode14x** solver choice, while leaving the rest with default settings. While not being necessary, it allows larger step-size when using **Derivative** blocks.

Step 2: Environment setup



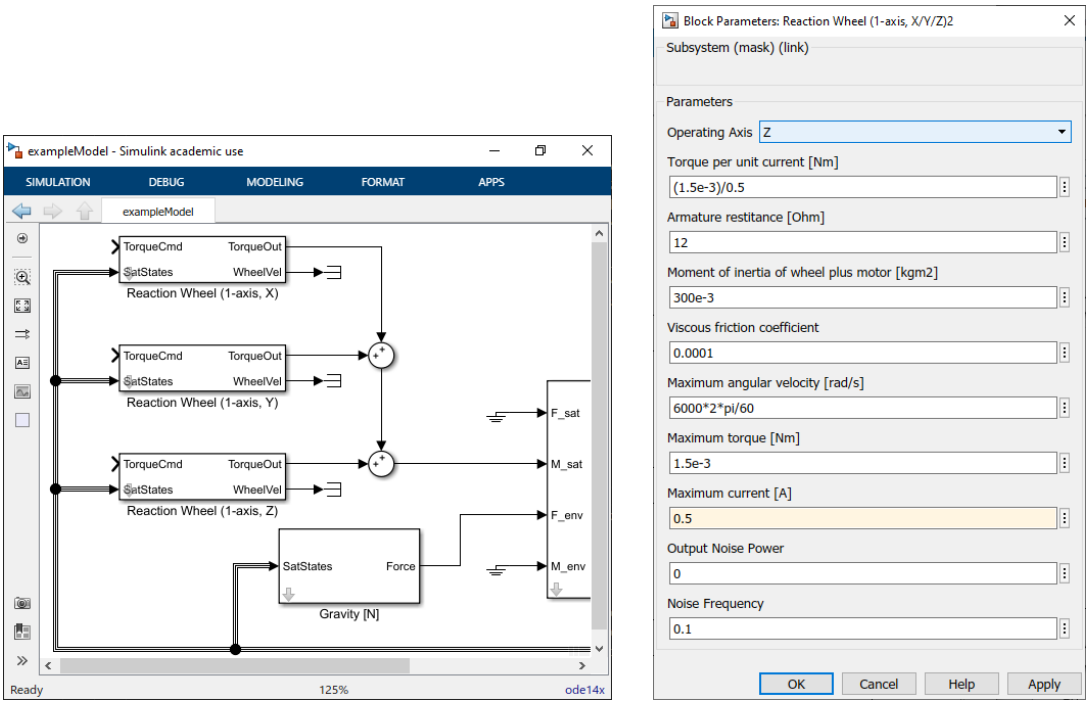
(a) Example Model (b) Gravity Mask

Figure 4.2: Example Model creation: Step 2

This spacecraft does not use magnetorquers nor does not have any major drag-inducing components, therefore the relevant block representing environment's in-

fluence 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

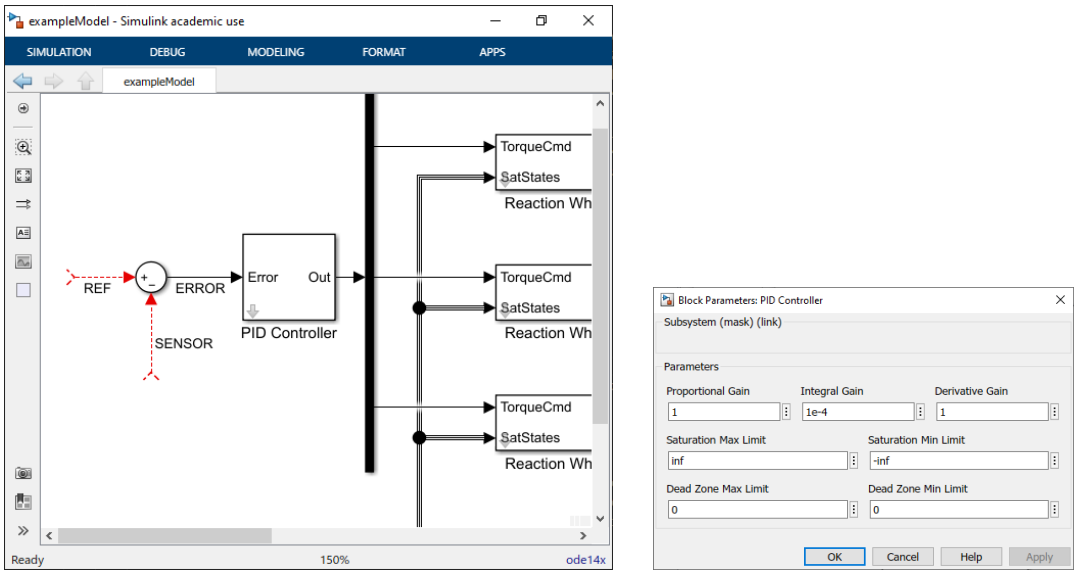


(a) Example Model (b) Reaction Wheel Mask

Figure 4.3: Example Model creation: Step 3

The next step is to implement the choice of actuators into the spacecraft model. In this example the user might 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 Section 2.4), while the outputs are torque and wheel angular rate.

Step 4: Setup of control algorithm



(a) Example Model (b) PID Mask

Figure 4.4: Example Model creation: Step 4

A PID controller was chosen as a control mechanism for reaction wheels. Figure 4.4 (b) represents the initial set up of the parameters of this controller block. Saturation of **PID Controller** output signal was set up in accordance to hardware's maximum voltage.

Step 5: Coordinate transformation and reference signal

SCARS Toolbox also provides a way to speed up the process of building mathematical transformations, allowing the user to conduct initial tests first and to design the software implementation afterwards. This approach leads to significant savings in amount of work to be performed by the control system engineer, as failing solutions 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.

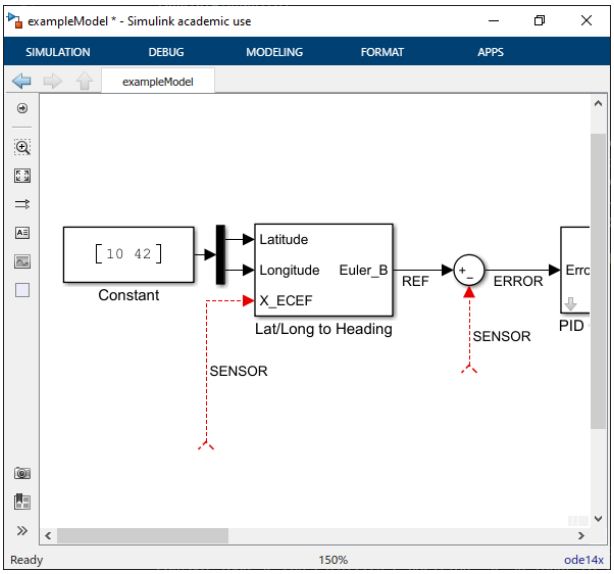
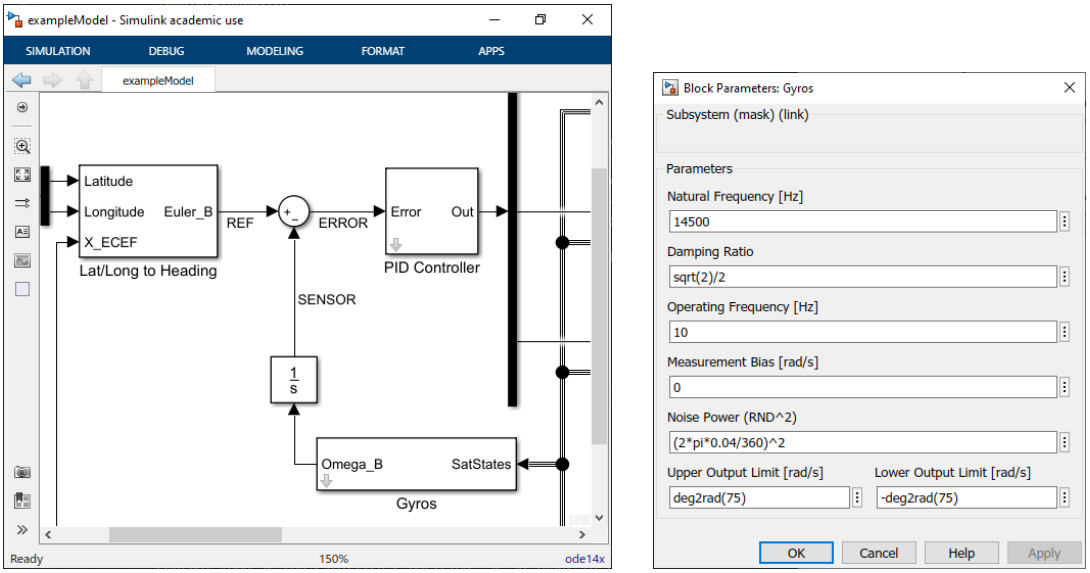


Figure 4.5: Example Model creation: Step 5

Step 6: Sensors choice and setup



(a) Example Model (b) Gyros Mask

Figure 4.6: Example Model creation: Step 6

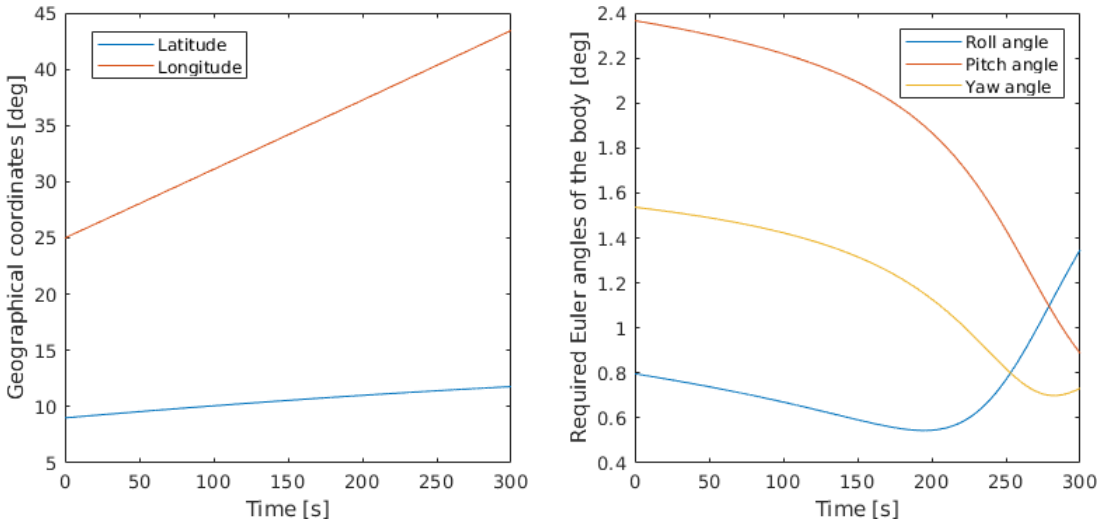
Figure Figure 4.5 shows that the only signals necessary to close the control loop as the satellite's position, as an input to **Lat/Long to Heading**, and Euler angles

in body reference frame. The former is irrelevant to the posed objective of this Example Model, therefore was set up to be measured by the **Ideal Position Sensor (ECEF)** block, and the latter had to be the analysed gyroscope. **Gyros** block was added to the simulation and set up with the parameters of the gyroscope, 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 4.6 (a) and Figure 4.6 (b) respectively.

Step 7: Simulation and verification

Finally, the simulation has to be run by the user and the results can be verified. The model is simulated to track reference geographic position of 10 degrees of latitude and 40 degrees of longitude. The satellite progresses over geographical coordinates as presented on Figure 4.7 (a), which results in calculated reference angle presented on Figure 4.7 (b). As it can be seen, the exposure time happens at around 200s mark, when the satellite is closest to imaging location. Reference angles change rapidly around that point in time.



(a) Geographical position of the satellite

(b) Satellite reference body angle

Figure 4.7: Data used for and produced by tracking subsystem

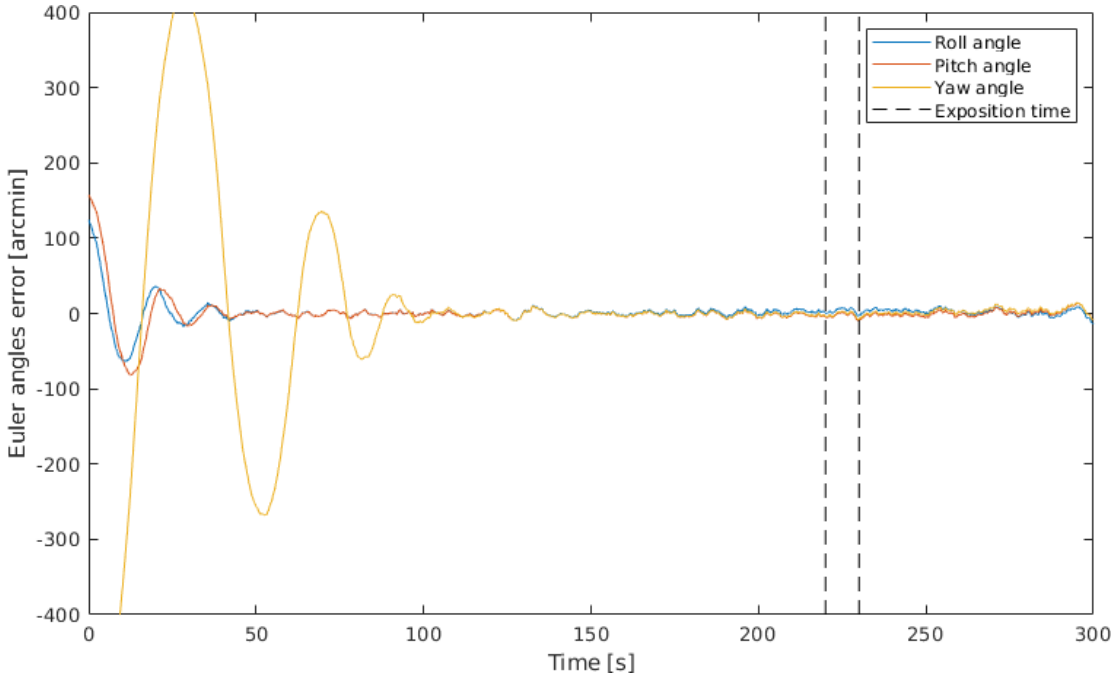


Figure 4.8: Plot of Euler angle error against reference angles, derived from geographical coordinates and satellite position

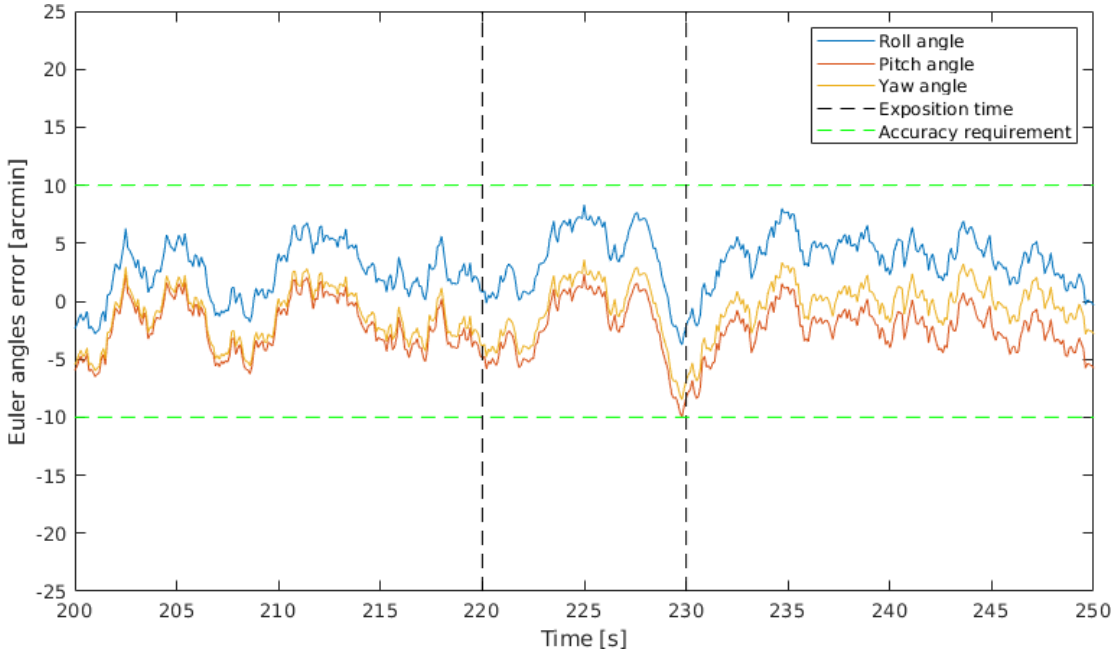


Figure 4.9: Plot of Euler angle error against reference angles, derived from geographical coordinates and satellite position, in focus on time between 200 and 250 seconds

Figure 4.8 shows the results from Example Model simulation. It is assumed that the satellite has attained attitude close to required during previous orbit, to mitigate the possibility of high body rates achieved during reference tracking rise time. During the first minute the satellite reaches accuracy under 1 degree and the error decreases.

The resulting plot is magnified to focus on simulation time between 200 and 250 seconds. It can be observed that for 10 seconds of marked timing the accuracy of the control system maintains the angle within ± 10 arcminutes. While this can be considered satisfactory and concludes this example, it is possible to improve the response of the system using methods presented in Section 4.4.

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." [15]

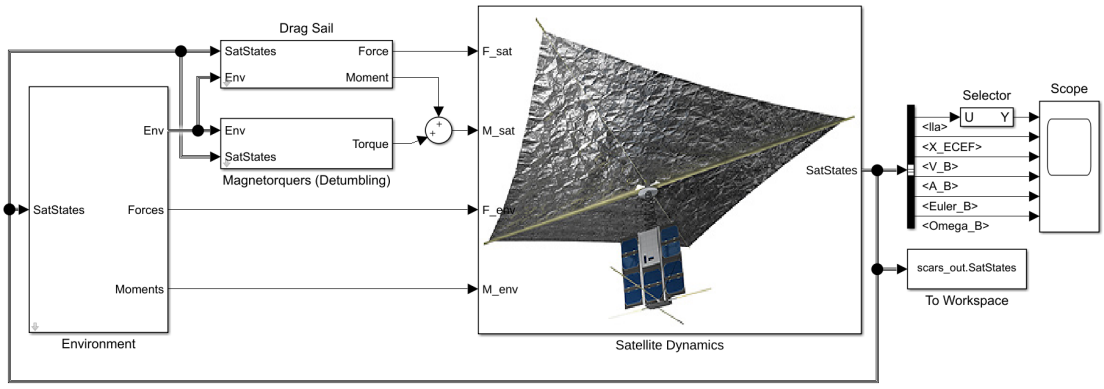
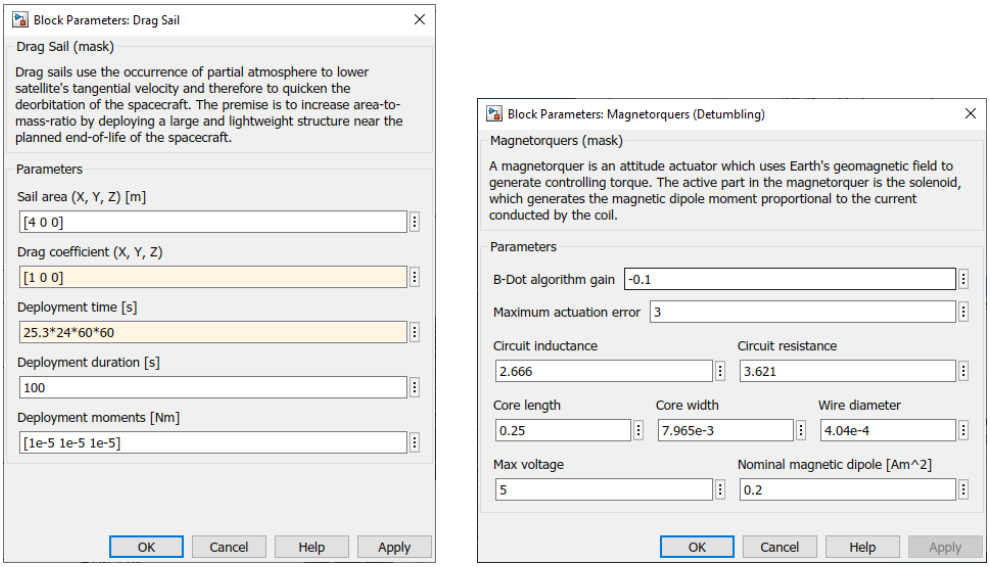


Figure 4.10: PW-Sat2 model created with components from SCARS Parts Library

As it can be seen on Figure 4.10, PW-Sat2 Simulink model is build exclusively from parts available SCARS toolbox, save for connecting blocks. It contains **Satellite Dynamics** block as the core of the simulation and it is set up using Keplerian elements taken from PW-Sat2 first Two-line element set (TLE) frame. Apart from that, full SCARS **Environment** model is included and connected

to two actuators: **Drag Sail** and **Magnetorquers (Detumbling)**. Figure 4.11 shows the setup of actuators' parameters - sources of these values are described in following sections.



(a) Drag Sail block mask (b) Magnetorquers (Detumbling) block mask

Figure 4.11: Parameters of SCARS PW-Sat2 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 crucial parameters used by SCARS model of PW-Sat2 are listed in Table 4.1.[36] Exact values are taken directly from the datasheet of ISIS Magnetorquer Board (iMQT)[37], as it was the magnetorquer used by PW-Sat2. Since SCARS contains only model for magnetorquer rods, the air core magnetorquer was assumed to be another torque rod. Also, since not every parameter of magnetorquer could be found in the datasheet, the missing fields were filled with data from similar ones.

Parameter	Value
Nominal magnetic dipole	$0.2Am^2$
Maximum actuation envelope error	$3\mu T$
Power consumption during actuation	$1.2W$
Maximum operating voltage	$5V$
Mass	196g

Table 4.1: Parameters gather from magnetorquer board installed on PW-Sat2

As it can be seen on Figure 4.12, detumbling was mostly successful, with small angular rates remaining on each axis. Such result is possible, since change in magnetic field on magnetorquers is proportional to the angular rate of the spacecraft, so when spacecraft is rotating slowly the torque generated by the actuator is also minor. Given enough time the satellite should approach near-zero rotation rate, but resulting value should be enough for good connection with the ground station to transfer into more active control mode.

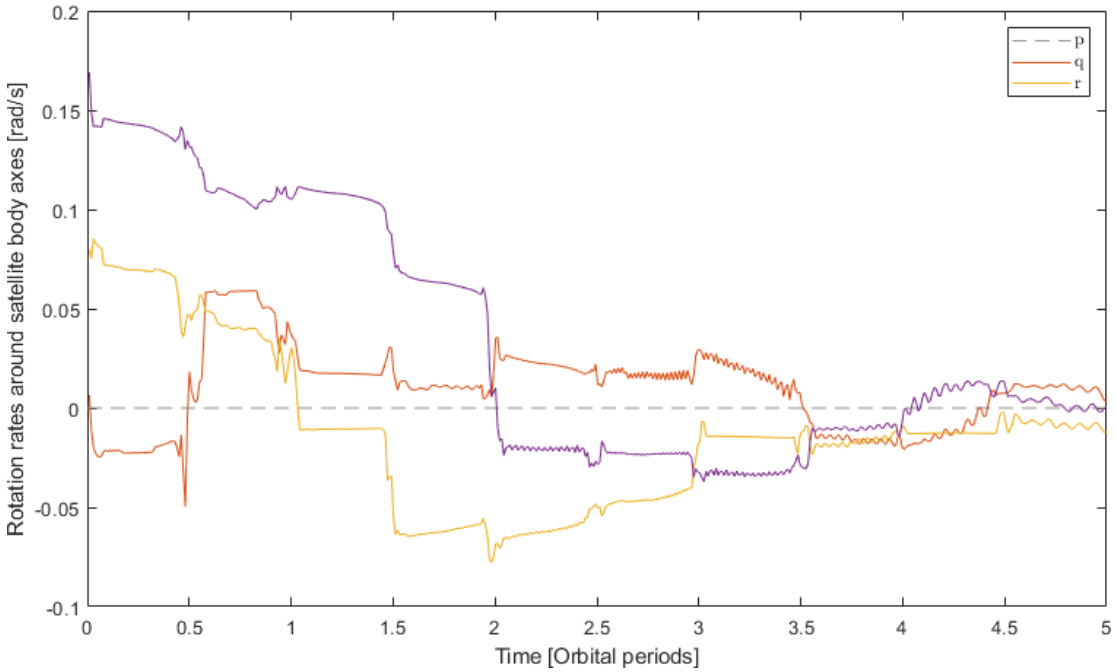


Figure 4.12: Results from SCARS simulation, with magnetorquers set up for detumbling

4.2.2 Deorbitation with drag sail

One of the main objectives of PW-Sat2 mission was to deploy and test the effectiveness of its drag sail in deorbitation maneuver. The sail was 2x2m square made from aluminized polyester boPET film[18].

In this example, SCARS toolbox was tested against data points derived from NORAD measurements. The simulation was run with the drag sail set up to be deployed around 25th day of the mission.

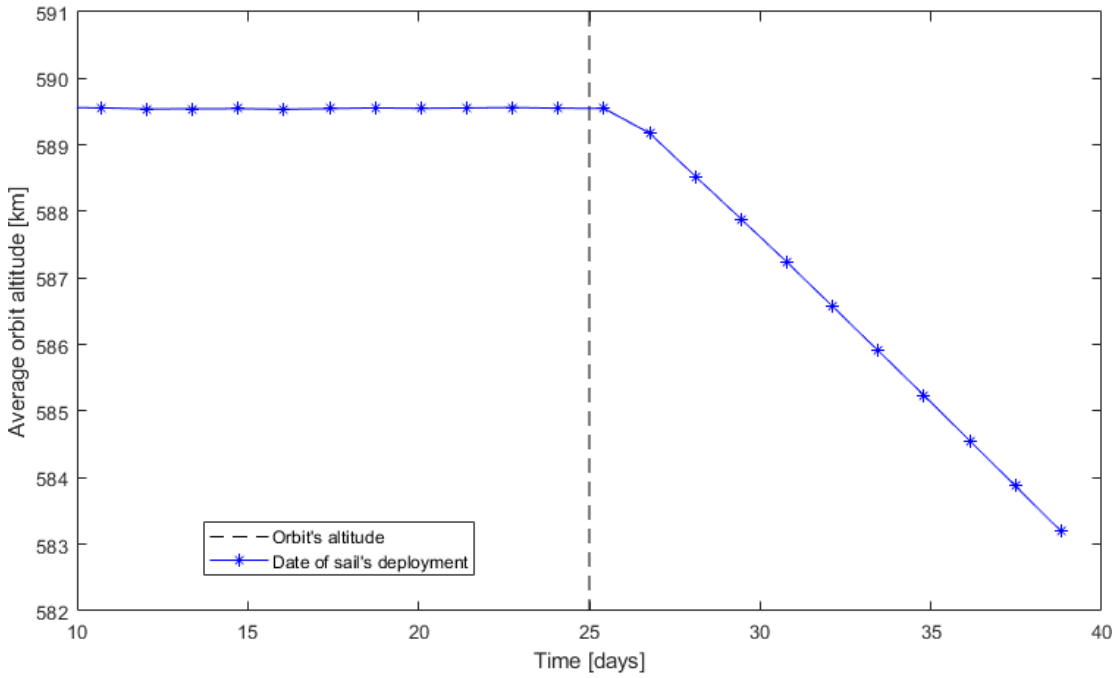


Figure 4.13: Results from SCARS simulation, with drag sail included

Since sail's deployment the aforementioned decrease of satellite's attitude is presented in Figure 4.14, showing the proof of deceleration caused by the sail. When comparing results from SCARS simulation with data collected from PW-Sat2 TLEs, visible on Figure 4.14 it is apparent that attitude changes are much more brisk in SCARS model than in real life. Such results were to be expected, as in PW-Sat2, shortly after deployment the sail has torn, therefore the effective drag was much lower than simulated value[21]. Also, since TLE data is a mixture of propagation and observation, the rate of descent is not accurate for the first few days after sail's deployment. In addition to that, simulated results may seem much smoother, as they the data points are averaged over duration of 10 orbits, when TLE data points are located arbitrarily.

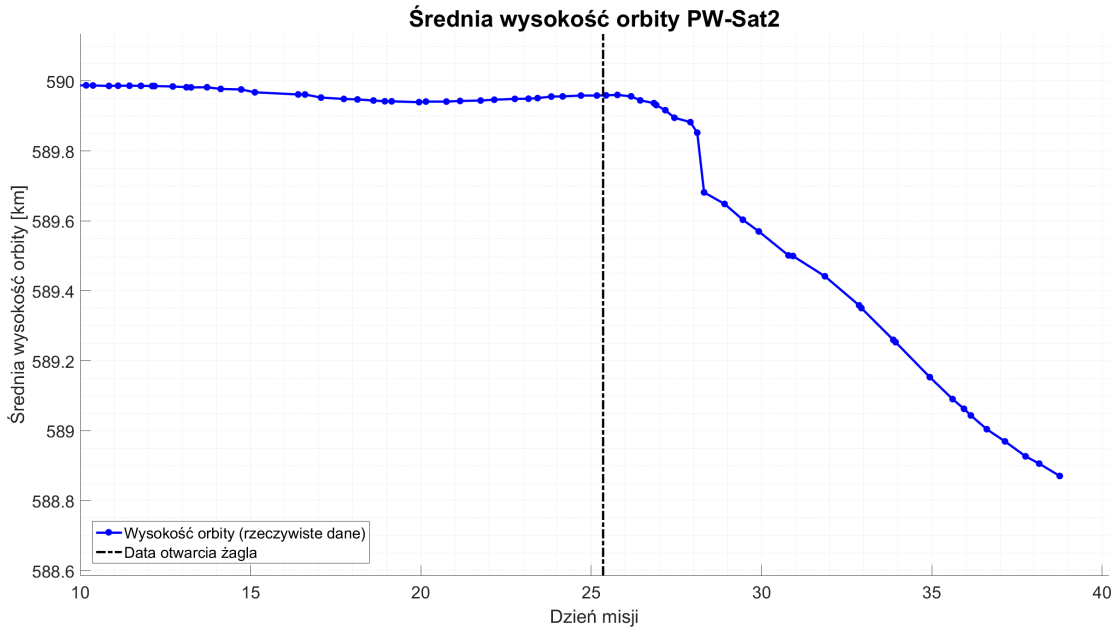


Figure 4.14: Average altitude of PW-Sat2 satellite as taken from North American Aerospace Defense Command measurements. On X axis there is mission time in days, on Y axis there is an average altitude in km [20]

Simulation of PW-Sat2 deorbitation showcases the SCARS Toolbox' ability of performing long-term simulation.

4.3 Sentinel-2

Sentinel-2 is an 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 its 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 [40].

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 [39]. The actuators and sensors on board of Sentinel-2 are described in Table 4.2.

Using that data, the model was created using SCARS Parts Library for demonstration purposes. It ought to prove that this toolbox can be used not only for

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

Table 4.2: Actuators and sensors on board of Sentinel-2 spacecraft [39]

small educational satellite missions, but also for purposes of larger scientific and commercial satellites. As one can see in Figure 4.15, the connections between **Satellite Dynamic** model, **Environment** block, **Sensors** and **Actuators** subsystems are solved with signal buses described in Section 2.3.3.

The subsystems can be explored, showing the setup of SCARS block responsible for sensors in Figure 4.16 and actuators in Figure 4.17. Since the model serves only as an example, it is not fully functional. The on board computer and state machine responsible for sensor fusion and for fault detection and isolation are not recreated in the model - reproducing algorithms behind this could be a topic for another thesis. Also, one can notice that Sentinel-2 model includes a sun sensor model. In current version of SCARS it could be only implemented by using **Ideal Sensor** block.

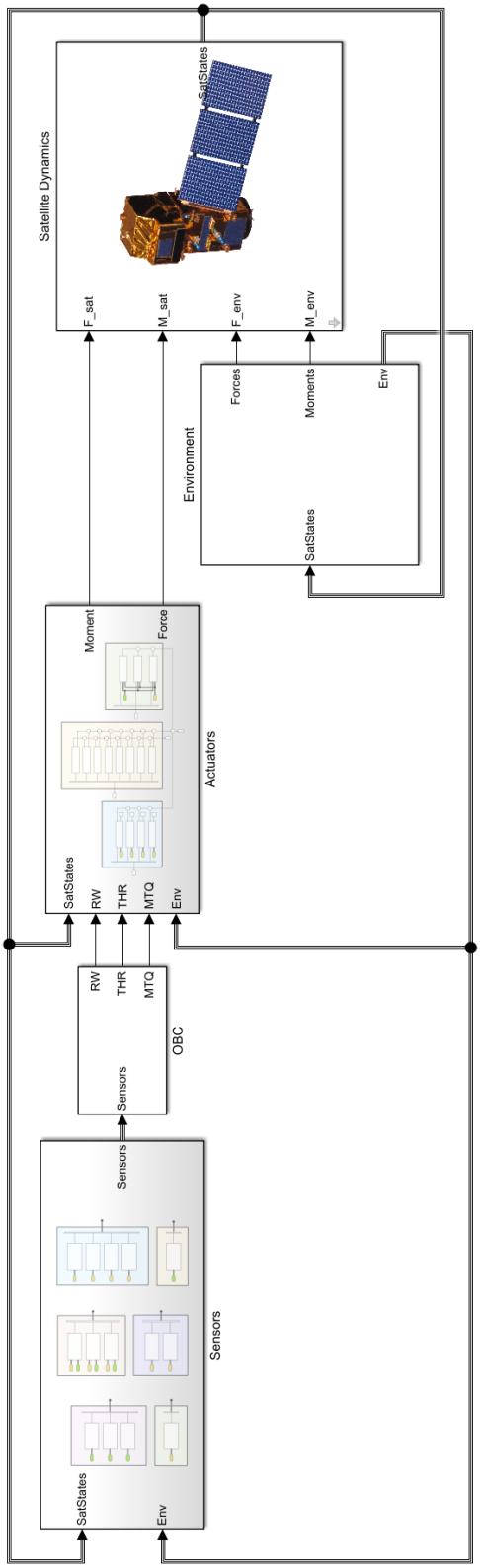


Figure 4.15: Sentinel-2 satellite ADCS model top-level view

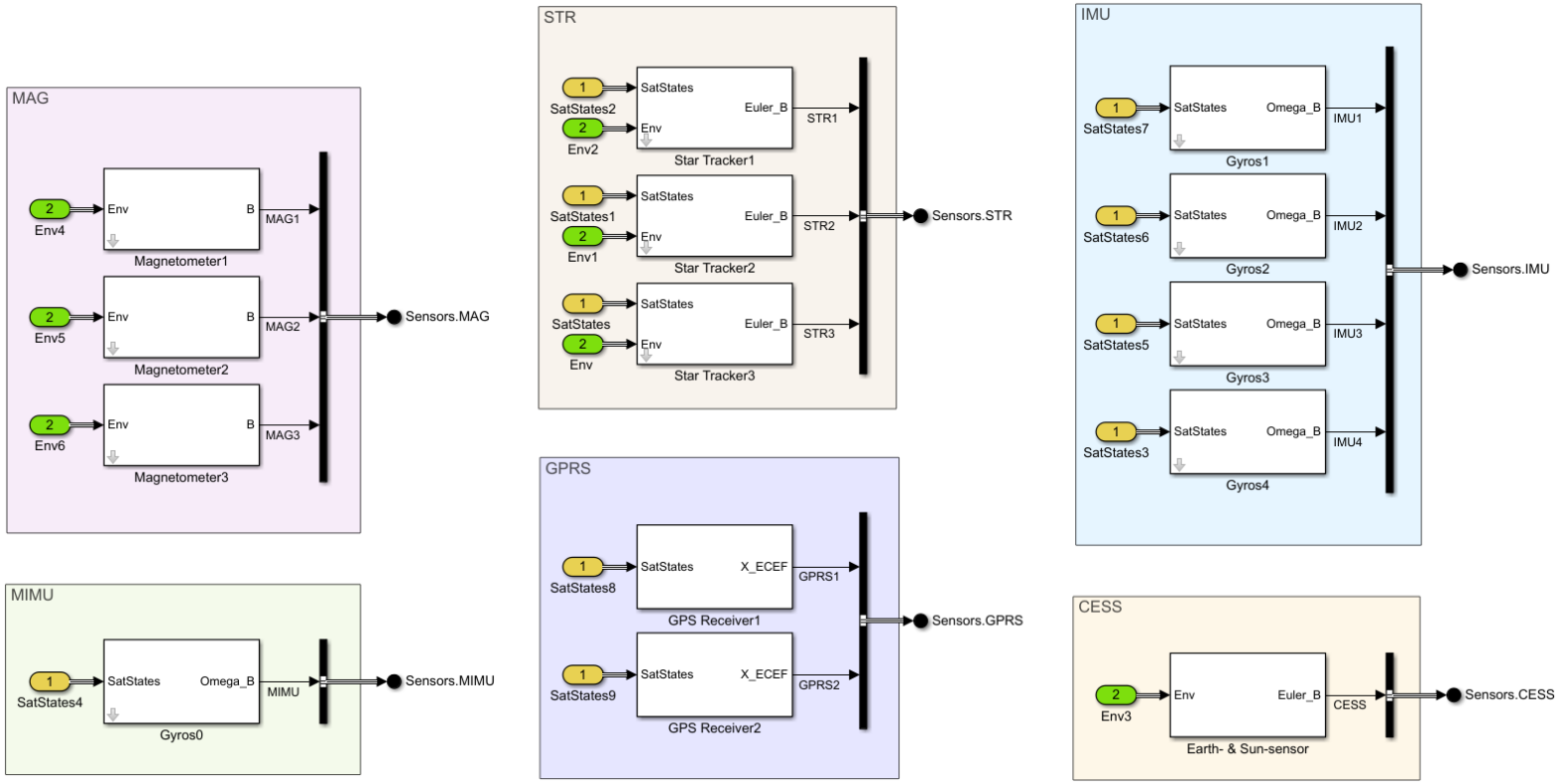


Figure 4.16: Sentinel-2 satellite ADCS model, Sensors subsystem

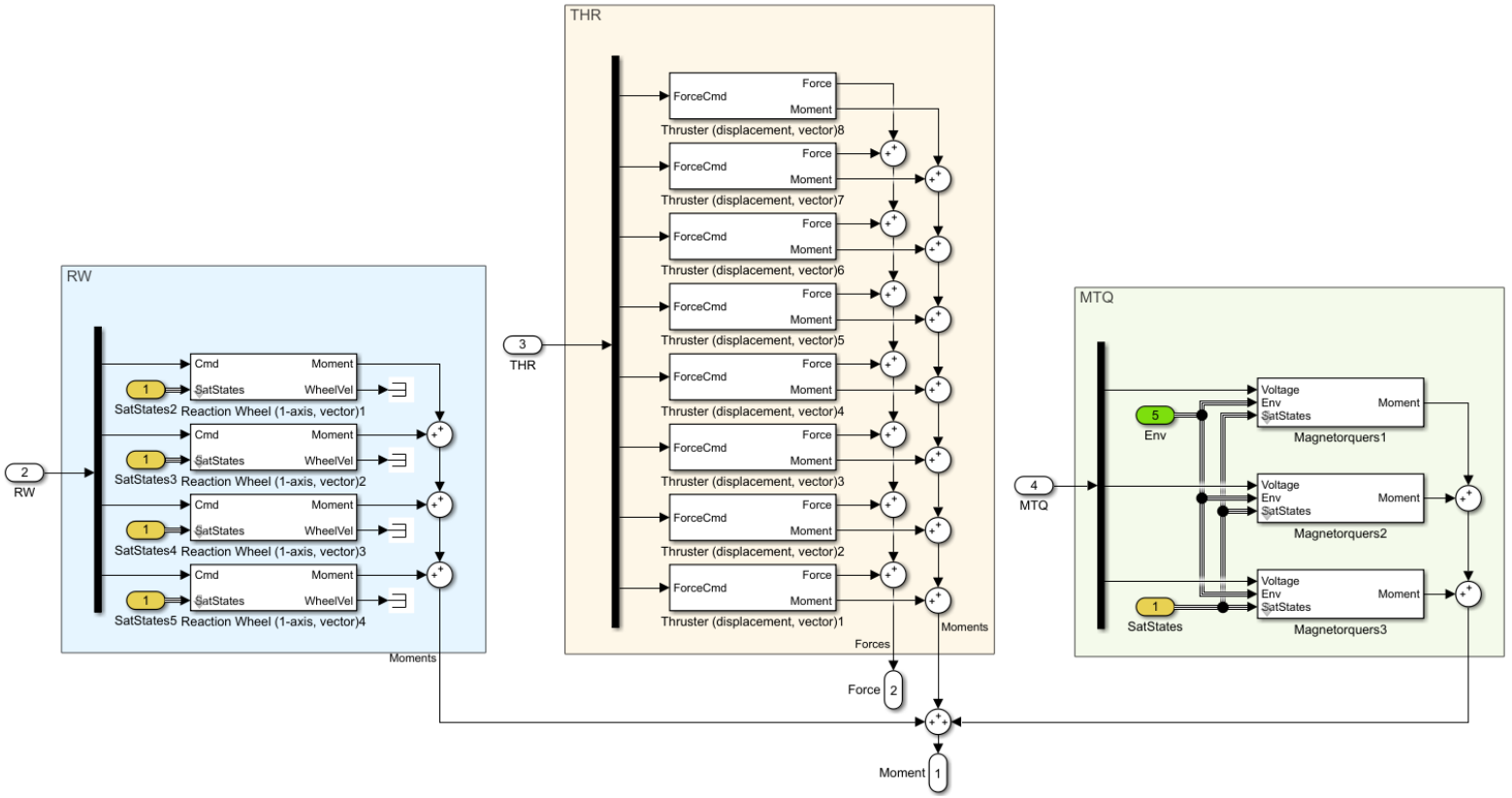


Figure 4.17: Sentinel-2 satellite ADCS model, Actuators subsystem

4.4 Other application of SCARS Toolbox

While previous sections provide examples for which whole spacecrafts models had to be constructed, the following examples show how SCARS Toolbox can be used to conduct further tests on already prepared models.

4.4.1 Controller design using linearized model

In Section 4.1, the gains in PID controller for reaction wheels were set up with empirical analysis, rather than on any tuning method. Alternative to that would be to use the linearization method described in Section C and Control System Designer, which is available as a part of MATLAB Control System Toolbox. To showcase this possibility Example Model from previous chapter was linearized according to appendix, with resulting state-space representation:

$$A = \begin{bmatrix} 2.5e\text{-}6 & 0 & 0 & 7.5e\text{-}7 & 0 & 0 & 0 & 0 & 0 \\ 0 & -2.5e\text{-}6 & 0 & 0 & 7.5e\text{-}7 & 0 & 0 & 0 & 0 \\ 0 & 0 & -2.5e\text{-}6 & 0 & 0 & 7.5e\text{-}7 & 0 & 0 & 0 \\ 2.5e\text{-}6 & 0 & 0 & -7.5e\text{-}7 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2.5e\text{-}6 & 0 & 0 & -7.5e\text{-}7 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2.5e\text{-}6 & 0 & 0 & -7.5e\text{-}7 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{44}$$

$$B = \begin{bmatrix} \text{-}0.00025 & 0 & 0 \\ 0 & \text{-}0.00025 & 0 \\ 0 & 0 & \text{-}0.00025 \\ 0.00025 & 0 & 0 \\ 0 & 0.00025 & 0 \\ 0 & 0 & 0.00025 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{45}$$

Where rows from 1 to 3 represent reaction wheels angular rates, from 4 to 6 are satellite angular rates and from 7 to 9 - body angles. Said system can be put into controller-plant feedback loop in figure Figure 4.18 in form $G = Ax + Bu$.

The minimal realization of a transfer function of this system can be approximated to $G(s) = \frac{2.5 \cdot 10^{-4}}{s(s+3.25 \cdot 10^{-6})}$. The response is very close to an ideal double integrator, with some damping coming from losses inside the reaction wheels, coming from simulated losses on the circuits and mechanical components.

In the loop presented on Figure 4.18, the controller function acquired from Control System Designer is put in place of C block.

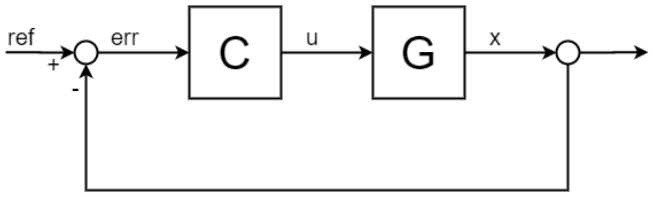


Figure 4.18: Feedback loop diagram

As Control System Designer only works with SISO systems, the next step is to choose which output should be analysed. In case of Example Model it does not make a difference which axis is chosen. This task can be performed with `getSISOSystem` described in Section 3.2.

After setting it up in Control System Designer, it shows Bode plots, Root Locus diagram and Step Response plot for the system. Using provided tools one can set up desired form of the controller, and edit Bode plots and Root Locus diagram until desired response is achieved. It was assumed that the controller needs a complex pole and a real zero, to be able to follow reference signal during tracking. Since the noise in gyroscopes might be causing the plant to drift too much, the desired system has to have a larger real part of the pole pole than zero. After setting it up and using Bode Editor to get high enough gain and phase margin, as seen on Figure 4.19.

The acquired controller has a transfer function of form:

$$C(s) = 10200 \frac{1 + 10s}{1 + 0.33s} \tag{46}$$

Which turned out to be a damped controller, with step response presented on Figure 4.20.

Figures 4.21 and 4.22 represent the performance of the new controller, comparable with the one designed by empirical methods, with slight advantage of using the one tuned with Control System Designer. Figures 4.23 and 4.24 contain, for comparison, Bode and Step Response plots generated for controller from Section 4.1.

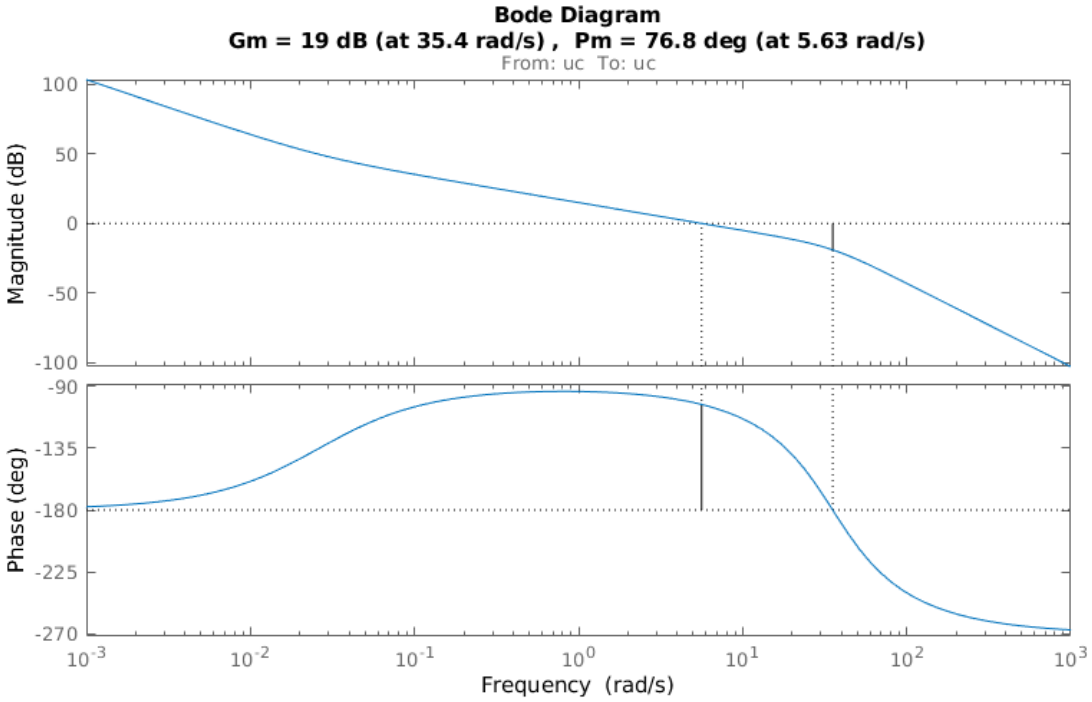


Figure 4.19: Bode plot of tuned system

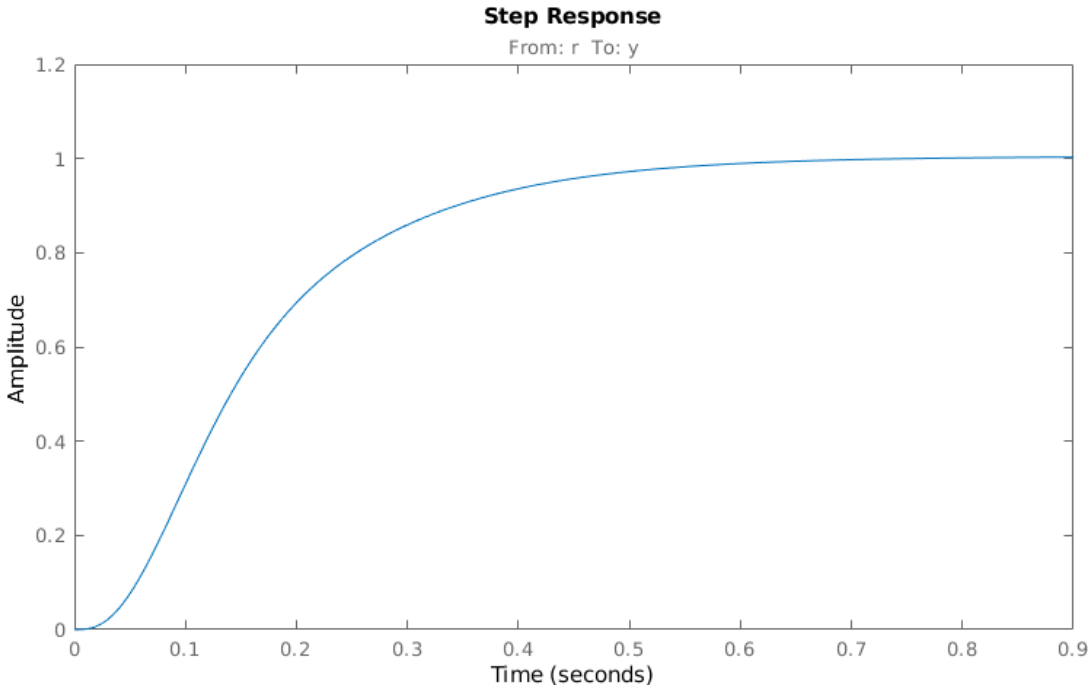


Figure 4.20: Step response of tuned system

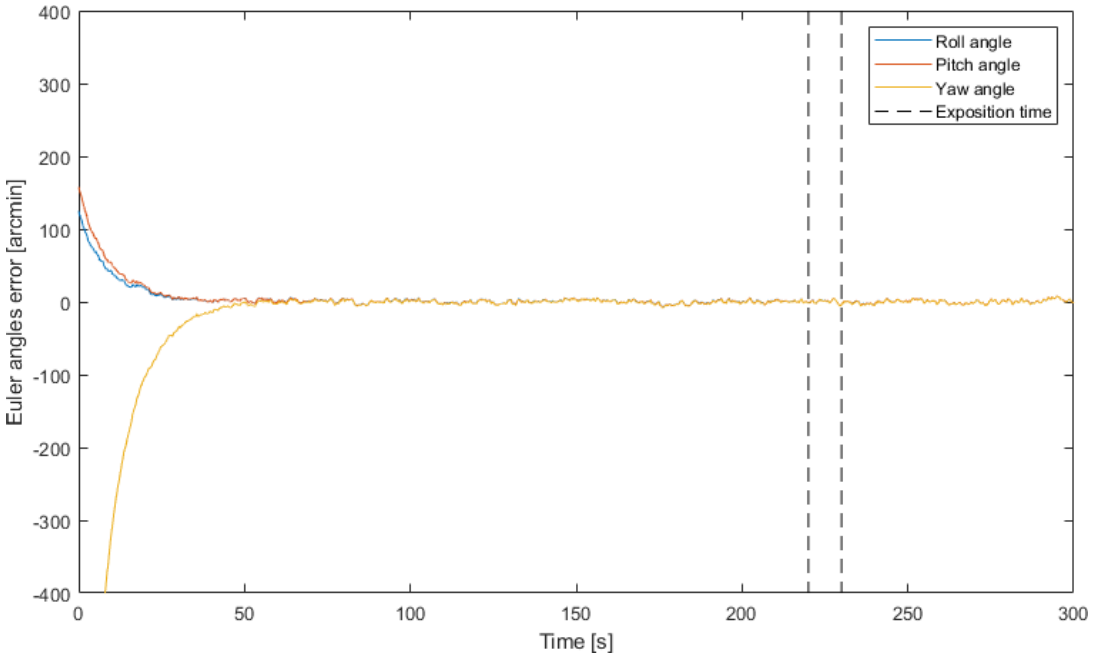


Figure 4.21: Plot of Euler angle error against reference angles, derived from geographical coordinates and satellite position

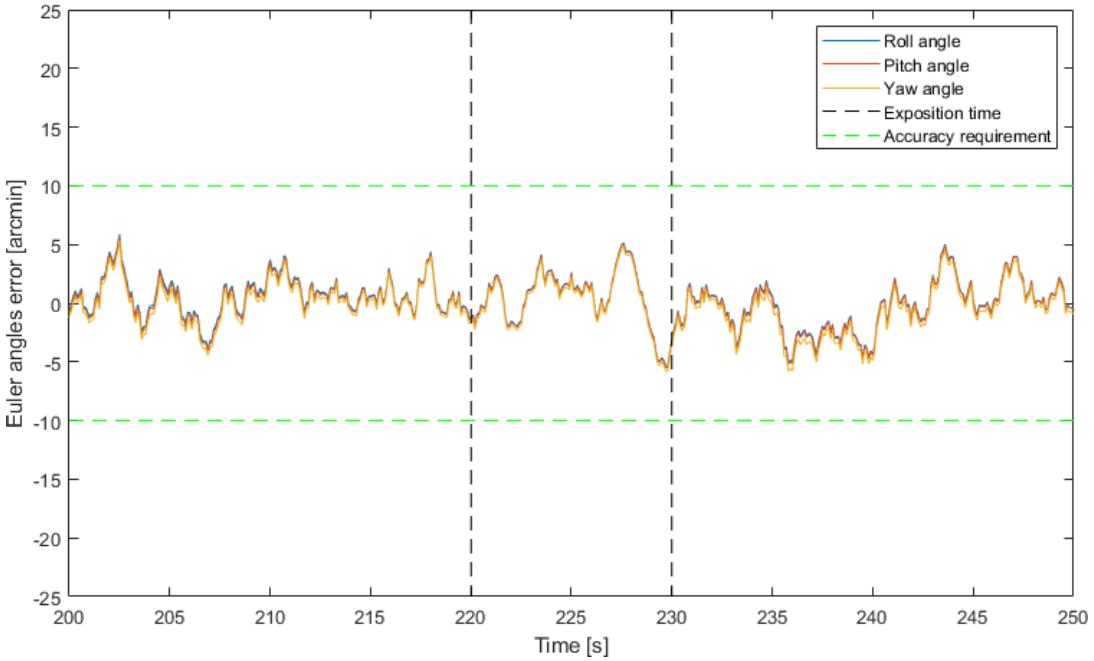


Figure 4.22: Plot of Euler angle error against reference angles, derived from geographical coordinates and satellite position, in focus on time between 200 and 250 seconds

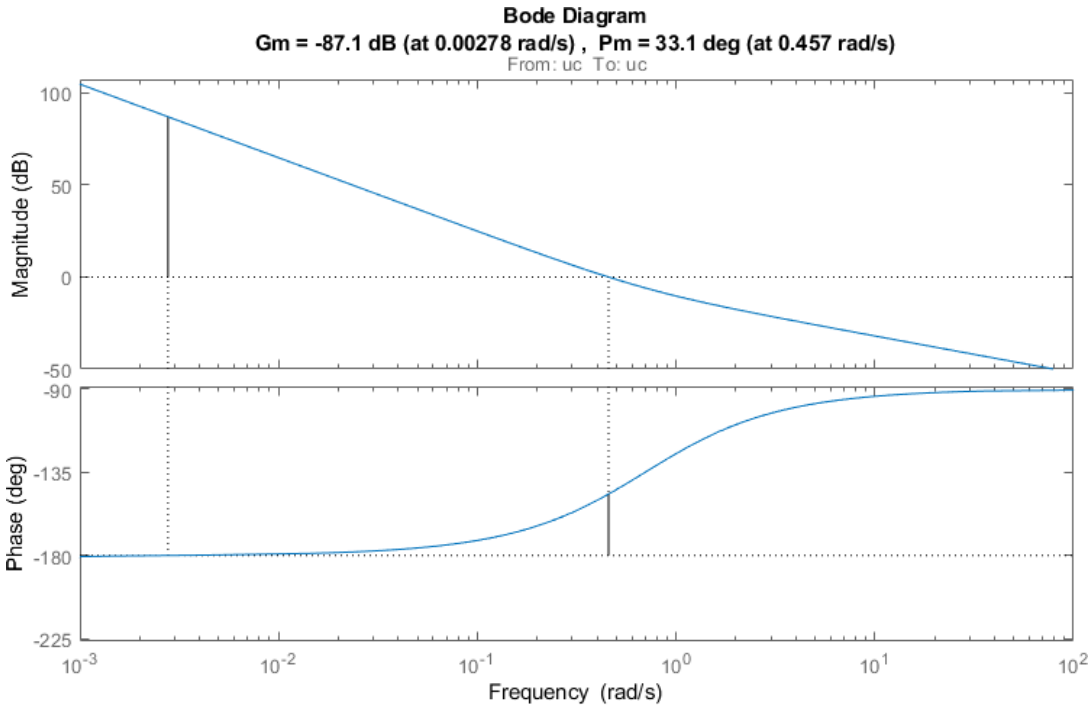


Figure 4.23: Bode plot of PID controller from Section 4.1

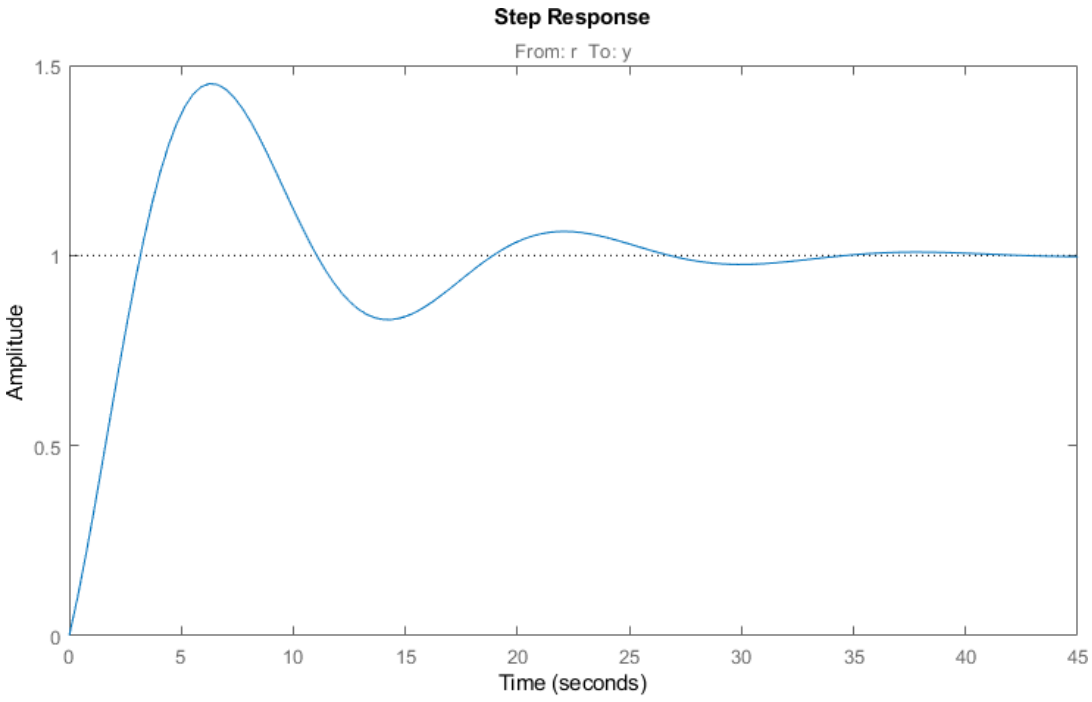


Figure 4.24: Step Response plot of PID controller from Section 4.1

4.4.2 Contingency scenarios simulation

SCARS provides an easy ways of testing various contingency scenarios, which means it is possible to quickly adapt the simulation to represent redundant structures, by swapping one sensor or actuator for another, or just by modifying existing elements. Scenario and Example Model from Section 4.1 can be considered good example. In this case, Nan-oTorque GSW-600 reaction wheels set contains four reaction wheels, one for each Cartesian axis and one located on direction vector $\mathbf{r} = [1, 1, 1]$. This setup makes it possible for the satellite to have 3 degrees of freedom even if one actuator is not responding.

To adapt Example Model to this scenario, one must only change one reaction wheel block from **Reaction Wheel (1-axis, X/Y/Z)** to **Reaction Wheel (1-axis, vector)** and set the **Operating Axis** parameter to [1 1 1] value (the norm is calculated from the vector, so its length is not relevant in this case). Then similar tests as in Section 4.1 can be performed to measure whether the new setup can satisfy the mission's requirements.

5 Conclusions

The aims behind this work were stated in Section 1.2, with description. SCARS Toolbox was designed as a part of a project destined to fulfil these objectives. Below said list is repeated, with description of proposed solutions and discussion on the degree of the success in regards to fulfillment of these aims.

- **Conduct a review of existing tools for preliminary spacecraft design**: In Section 1.3 a comprehensive review was presented, with reasons why there is still a neef for a toolbox such as SCARS and why any available solution do not fill this niche.
- **Create a spacecraft dynamics and AOCS model**: For purposes of the thesis a spacecraft dynamics model was created, basing on tools available in MATLAB and Simulink software family. The product, **Satellite Dynamics** block described in Section 2.4, was mostly based on review and implementation of existing solutions, not designed from the ground up. On the other hand, the actions taken to build the toolbox fulfilled an objective of creating a library of models, from which an ADCS subsystem can be built. The resulsts of that were presented in both a simple spacecraft case in Section 4.1 and also in Section 4.3, with advanced set of sensors and actuators. However during the creation of these models some problems were encountered, such as lack of detailed listing of hardware parameters in available datasheets, or, as it can be seen in Section 4.2, it was necessary to assume or estimate certain parameters of the magnetometers, as the only provided ones were on lower level of complexity than SCARS magnetometer model.
- **Assemble a library of models**: SCARS Parts Library was created, and with techniques described in Section 2.3.3 the components can be easily connected with each other. Moreover, since models available as a part of SCARS Toolbox are composed from basic elements and niche Simulink toolboxes were avoided, it is possible to include them in unrelated models with next to no set up. To have a complete list of most crucial building blocks for control system design, SCARS Toolbox is only lacking a way to perform sensor fusion such as, for example, Kalman filter implementation.
- **Provide a documentation of the toolbox**: This objective is mostly fulfilled by the contents of this thesis, specifically Chapter 2, Chapter 3 and Chapter 4. In addition to that, the blocks available as a part of SCARS Parts Library contain descriptions leading the users to methods of their implementation.
- **Share the toolbox to be available online**: As SCARS Toolbox is al-

ready a usable product, there are directions in which it could be improved (as described in last paragraph of this chapter). Release of this software on MATLAB Central is planned, along with reaching out to CubeSat communities to present the toolbox. For now, it can be found on author's public GitHub repository under a following address: `https://github.com/asmialek/SCARS-Toolbox`.

Apart from that, SCARS Toolbox itself was created with much more specific goals in mind, most of them described in Section 2.1. It can be declared that all these objectives were met and it was proven so in the scope of this thesis.

It can be argued that SCARS Toolbox could be further expanded, mitigating some problems described in this chapter. For example, to avoid the problem of having a datasheet with too few details, the models could be prepared with various options for initial set up, or with multiple version of the model, operating with different sets of parameters. To solve the problem with lacking a model of certain actuator, more models can be designed to be used as a part of SCARS Parts Library. However in its base structure, SCARS Toolbox fulfils the objectives for which it was designed. Similarly to most large software projects, there is always room for improvement, but it can be safely assumed that SCARS can prove itself to be an useful tool for both unexperienced student teams and for control systems engineers searching for a way to quickly design a prototype model.

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.

[15] Warsaw University of Technology Students' Space Association. *"What is PW-Sat2", PW-Sat2 Project Website*, (accessed July 30, 2020). `https://pw-sat.pl/en/mission/`.

[16] MathWorks Aerospace Products Team. *Aerospace Blockset CubeSat Simulation Library MathWorks File Exchange Webpage*, (accessed July 30, 2020). `https://www.mathworks.com/matlabcentral/fileexchange/70030-aerospace-blockset-cubesat-simulation-library`.

[17] Princeton Satellite Systems. *Princeton Satellite Systems, Spacecraft Control Toolbox Webpage*, (accessed July 02, 2020). `http://www.psatellite.com/products/sct/`.

[18] Warsaw University of Technology Students' Space Association. *PW-Sat2 Critical Design Review, Deployment Team*, (accessed June 10, 2020). `https://pw-sat.pl/wp-content/uploads/2014/07/PW-Sat2-C-05.00-DT-CDR.pdf`.

[19] Tjorven Delabie. Star tracker algorithms and a low-cost attitude determination and control system for space missions. 2016.

[20] Warsaw University of Technology Students' Space Association. *PW-Sat2 Twitter Page*, (accessed June 10, 2020). `https://twitter.com/PWSat2/status/1085604280118185985`.

[21] Space24 Pawel Ziemnicki. *PW-Sat2 Pekniecia na powloce kosmicznego zagla*, (accessed June 13, 2020). `https://www.space24.pl/pw-sat2-pekniecia-na-powloce-kosmicznego-zagla`.

[22] Princeton Satellite Systems. *Princeton Satellite Systems Website*, (accessed May 27, 2020). `http://www.psatellite.com/`.

[23] IRISC Team. *IRISC Student Experiment Documentation*, (accessed May 27, 2020). `http://rexusbexus.net/wp-content/uploads/2020/03/BX28_IRISC_SED_v5-1_26Feb.pdf`.

[24] Valdemir Carrara. An open source satellite attitude and orbit simulator toolbox for matlab. In *Proceedings of the 17th International Symposium on Dynamic Problems of Mechanics*, 2015.

[25] Louise Lindblad. Modelling and simulation of gnc/aocs systemsfor conceptual studies, 2013.

[26] MathWorks. *MATLAB Central Website*, (accessed May 27, 2020). `https://www.mathworks.com/matlabcentral/?s_tid=srchbrcm`.

[27] Dimitrios Piretzidis. *SAT-LAB: A MATLAB Graphical User Interface for simulating and visualizing Keplerian satellite orbits, MATLAB Central File Exchange*, (accessed August 11, 2020). `https://www.mathworks.com/matlabcentral/fileexchange/63344-sat-lab-a-matlab-graphical-user-interface-for-simulating-and-visualizing-keplerian-satellite-orbits`.

[28] Meysam Mahooti. *Satellite Orbit Modeling, MATLAB Central File Exchange*, (accessed August 11, 2020). `https://www.mathworks.com/matlabcentral/fileexchange/54877-satellite-orbit-modeling`.

[29] Samir Rawashdeh. *Smart Nanosatellite Attitude Propagator (SNAP), MATLAB Central File Exchange*, (accessed August 11, 2020). `https://www.mathworks.com/matlabcentral/fileexchange/68652-smart-nanosatellite-attitude-propagator-snap`.

[30] Meysam Mahooti. *Satellite Orbits: Models, Methods and Applications, MATLAB Central File Exchange*, (accessed August 11, 2020). `https://www.mathworks.com/matlabcentral/fileexchange/54840-satellite-orbits-models-methods-and-applications`.

[31] MathWorks Aerospace Products Team. *Apollo 11 Moon Landing - 50th Anniversary Model, MATLAB Central File Exchange*, (accessed August 11, 2020). `https://www.mathworks.com/matlabcentral/fileexchange/72127-apollo-11-moon-landing-50th-anniversary-model`.

[32] Inc. SkyTraq Technology. *Venus838FLPx GPS Receiver, Datasheet*, (accessed August 11, 2020). `http://navspark.mybigcommerce.com/content/Venus838FLPx-SPC_DS_v4.pdf`.

[33] Systems Tool Kit (STK). *Systems ToolKit Ephemeris File Format (*.e) Documentation Page*, (accessed August 11, 2020). `https://help.agi.com/stk/11.0.1/Content/stk/importfiles-02.htm`.

[34] Inc. Take-Two Interactive. *Kerbal Space Program Website*, (accessed August 11, 2020). https://www.kerbalspaceprogram.com/.

[35] MathWorks Simulink Documentation. *Create Block Masks*, (accessed August 11, 2020). https://www.mathworks.com/help/simulink/block-masks.html.

[36] Warsaw University of Technology Students' Space Association. *PW-Sat2 Critical Design Review, Attitude Determination and Control System*, (accessed June 10, 2020). https://pw-sat.pl/wp-content/uploads/2014/07/PW-Sat2-C-01.00-ADCS-CDR.pdf.

[37] ISIS Innovative Solutions In Space B.V. *ISIS Magnetorquer Board (iMTQ) Brochure*, (accessed June 27, 2020). https://www.isispace.nl/wp-content/uploads/2016/02/iMTQ-Brochure-v1.pdf.

[38] MathWorks Software Developement Tools. *MATLAB Projects Documentation*, (accessed August 14, 2020). https://www.mathworks.com/help/matlab/projects.html.

[39] G Wiedermann, W Gockel, S Winkler, JM Rieberm, B Kraft, and D Reggio. The sentinel-2 satellite attitude control system–challanges and solutions. In *presented at 9th International ESA Conference on Guidance, Navigation & Control Systems*, 2014.

[40] ESA Sentinel. User handbook. *ESA Standard Document*, 64, 2.

List of Figures

List of Tables

Appendices

A Example STK Ephemeris File

```
stk.v.4.3
BEGIN Ephemeris
NumberOfEphemerisPoints 1201
ScenarioEpoch           1 Jan 2000 12:0:0.0000
InterpolationOrder      5
DistanceUnit            Meters
CentralBody             Earth
CoordinateSystem        Fixed
EphemerisTimePosVel
0    1229340.480 6658465.510 -182.453 -6604.570 1219.461 2623.985
0.1  1228680.016 6658587.418   79.945 -6604.710 1218.705 2623.985
0.2  1228019.538 6658709.251  342.343 -6604.849 1217.949 2623.985
0.3  1227359.046 6658831.008  604.742 -6604.989 1217.192 2623.985
0.4  1226698.540 6658952.689  867.140 -6605.128 1216.436 2623.985
0.5  1226038.020 6659074.295 1129.539 -6605.267 1215.680 2623.985
0.6  1225377.486 6659195.825 1391.937 -6605.407 1214.923 2623.984
0.7  1224716.939 6659317.280 1654.336 -6605.546 1214.167 2623.984
0.8  1224056.377 6659438.659 1916.734 -6605.685 1213.411 2623.984
0.9  1223395.802 6659559.962 2179.133 -6605.824 1212.654 2623.984
1    1222735.212 6659681.190 2441.531 -6605.963 1211.898 2623.983
1.1  1222074.609 6659802.342 2703.930 -6606.102 1211.141 2623.983
1.2  1221413.992 6659923.418 2966.328 -6606.240 1210.385 2623.983
1.3  1220753.361 6660044.419 3228.726 -6606.379 1209.628 2623.982
1.4  1220092.716 6660165.344 3491.124 -6606.518 1208.872 2623.982
1.5  1219432.057 6660286.193 3753.523 -6606.656 1208.115 2623.981
1.6  1218771.385 6660406.967 4015.921 -6606.795 1207.359 2623.981
1.7  1218110.698 6660527.665 4278.319 -6606.933 1206.602 2623.980
1.8  1217449.998 6660648.288 4540.717 -6607.072 1205.846 2623.980
1.9  1216789.284 6660768.835 4803.115 -6607.210 1205.089 2623.979
2    1216128.556 6660889.306 5065.513 -6607.348 1204.333 2623.978
2.1  1215467.814 6661009.701 5327.911 -6607.486 1203.576 2623.978
2.2  1214807.059 6661130.021 5590.308 -6607.624 1202.820 2623.977
2.3  1214146.289 6661250.265 5852.706 -6607.762 1202.063 2623.976
2.4  1213485.506 6661370.434 6115.104 -6607.900 1201.306 2623.976
...
```

B Example STK Attitude File

```
stk.v.4.3
BEGIN Attitude
NumberOfAttitudePoints 1201
ScenarioEpoch           1 Jan 2000 12:0:0.0000
InterpolationOrder      5
CentralBody             Earth
CoordinateSystem        Fixed
AttitudeTimeEulerAngles
0    -6.3611e-15 5.0193e-15 -2.8123e-15
0.1  1.466e-06   3.9918e-06 0.075
0.2  5.8778e-06  1.5962e-05 0.15
0.3  1.3256e-05  3.5903e-05 0.225
0.4  2.3623e-05  6.3808e-05 0.3
0.5  3.6997e-05  9.9667e-05 0.375
0.6  5.3401e-05  0.00014347 0.45
0.7  7.2855e-05  0.00019522 0.52501
0.8  9.538e-05   0.0002549  0.60001
0.9  0.000121    0.0003225  0.67501
1    0.00014973  0.00039802 0.75001
1.1  0.00018159  0.00048144 0.82501
1.2  0.0002166   0.00057277 0.90002
1.3  0.0002548   0.00067198 0.97502
1.4  0.00029618  0.00077908 1.05
1.5  0.00034079  0.00089406 1.125
1.6  0.00038862  0.0010169  1.2
1.7  0.00043972  0.0011476  1.275
1.8  0.0004941   0.0012861  1.35
1.9  0.00055177  0.0014325  1.425
2    0.00061276  0.0015868  1.5
2.1  0.0006771   0.0017488  1.5751
2.2  0.00074479  0.0019187  1.6501
2.3  0.00081587  0.0020964  1.7251
2.4  0.00089034  0.0022818  1.8001
...
```

C Model Linearization with Control System Toolbox

This appendix contains the step-by-step instructions to linearize the chosen SCARS model. In the case of this example, the aim is to acquire linear model of satellite with added actuators, to use it for control system design. The relevant block in this SCARS model are **Satellite Dynamics** as a core of simulation, one **Reaction Wheel (1-axis, X/Y/Z)** for each major axis. Figure C.1 presents the whole model used in this example.

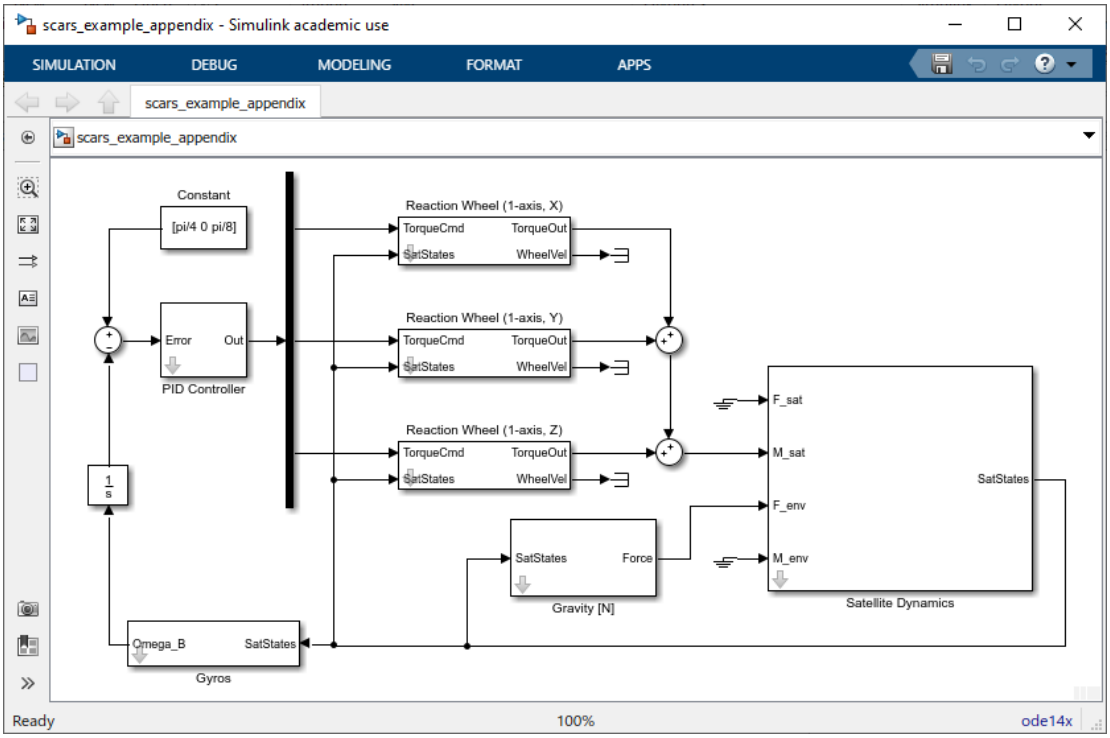


Figure C.1: Satellite model built with SCARS for purposes of linearization presentation

Step 1: Create output signals

To linearize the SCARS model one has to determine inputs and outputs which will be represented by the acquired linearized model. While the model inputs are same as the actuators blocks inputs (or **Demux** outputs), the outputs have to be extracted from **SatStates** bus. To do that, the user has to add **Bus Selector** block and choose, in case of this example, **Euler_B** signal, as seen on Figure C.2.

It has to be noted that if any controller is set up to provide input signal, like **PID Controller** in this example, it has to be commented out from the model before linearization.

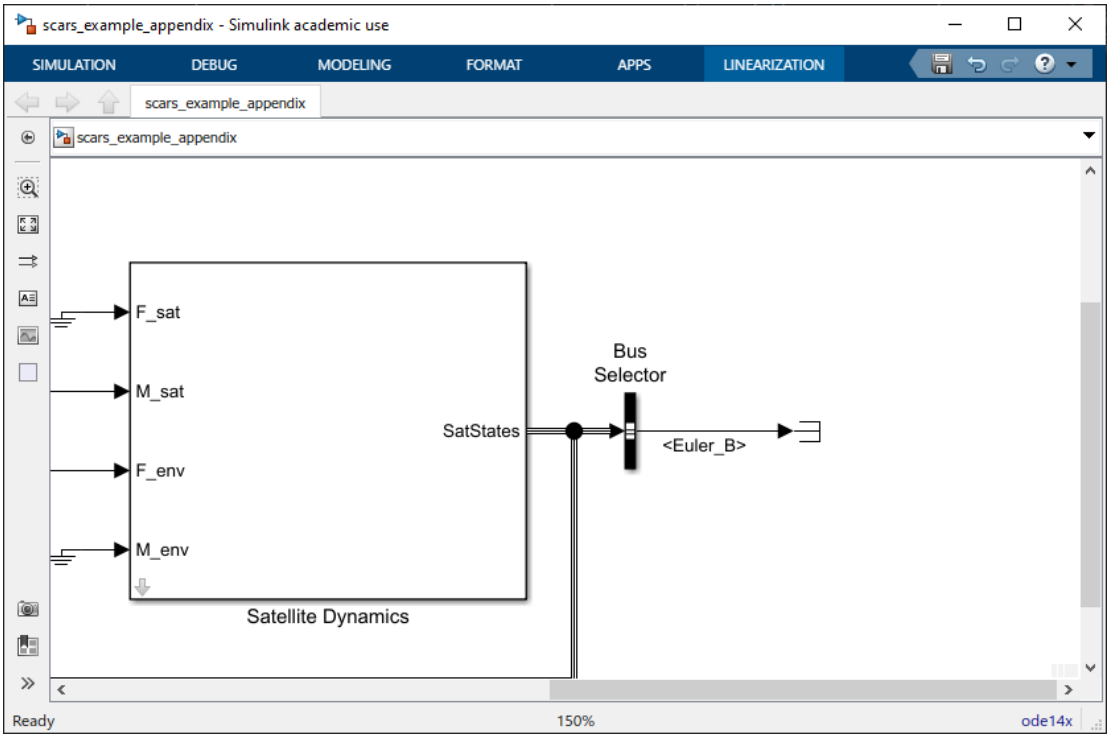
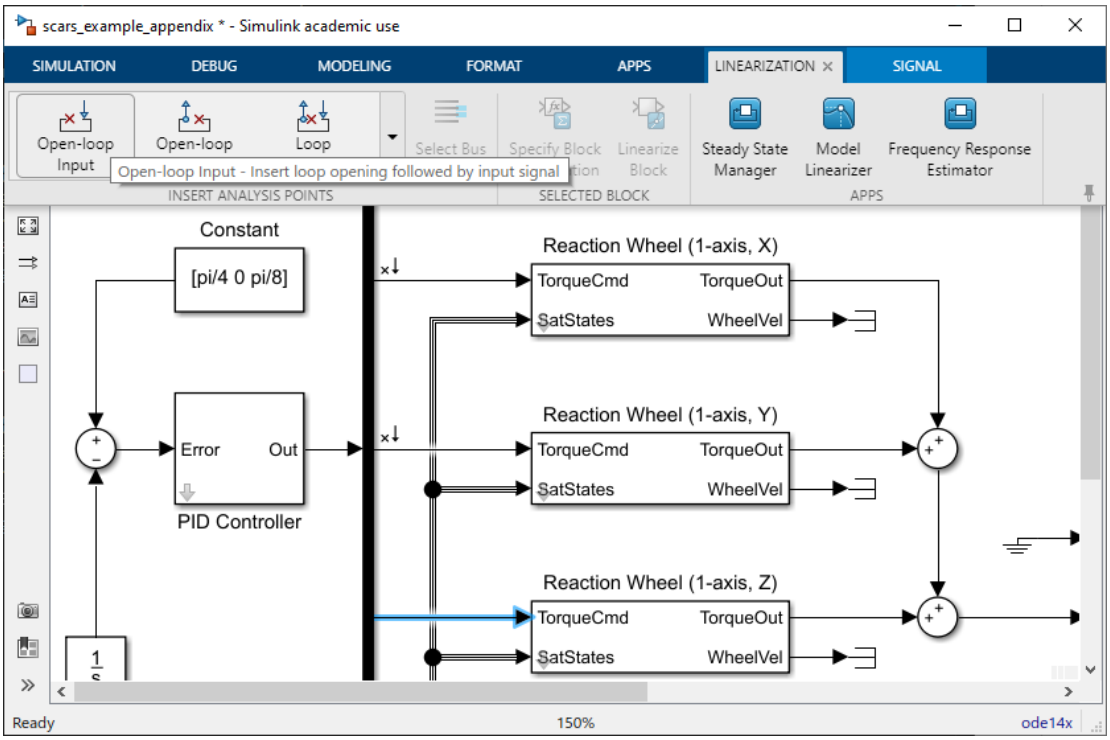


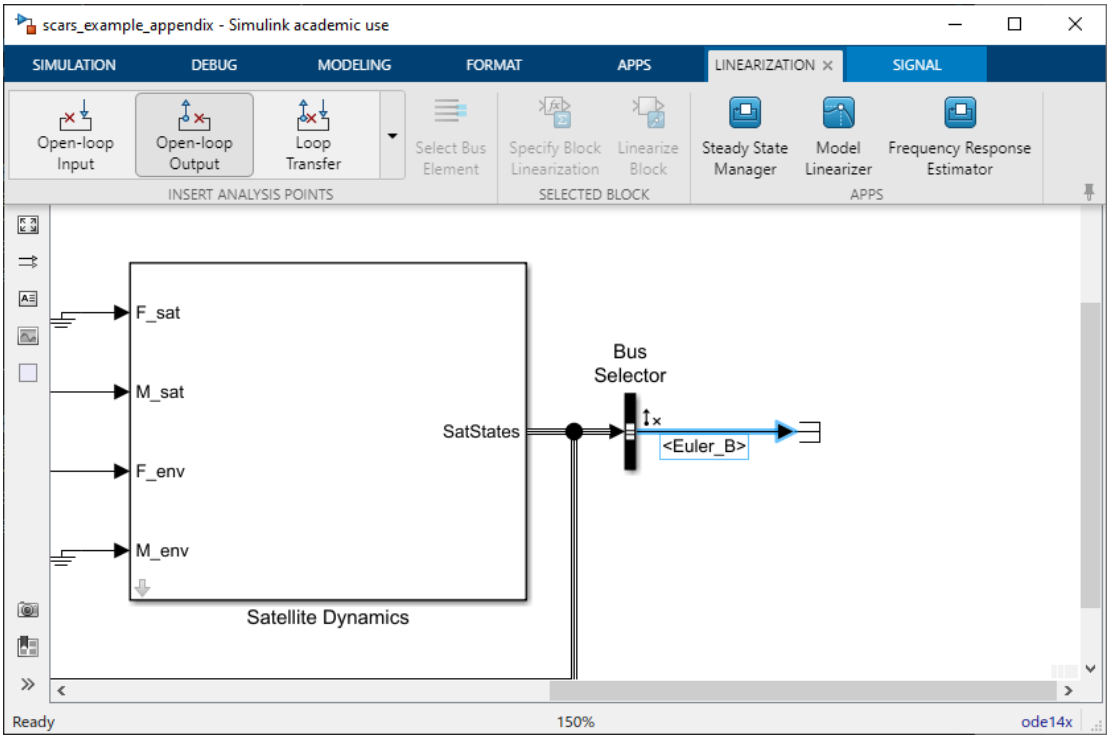
Figure C.2: Extraction of output signal from bus port

Step 2: Select input and output signals

Once all signals are available in the model, it is necessary to properly mark them for Simulink Linear Analysis Tool. To do that, the user has to open *Linearization Manager* from *Apps* tab in Simulink model editor. Then, a signal has to be chosen and in *Linearization* tab a correct type of signal has to be applied. In case of input it has to be *Open-loop Input*, as seen on Figure C.3 (a), and in case of output it has to be *Open-loop Output*, as seen on Figure C.3 (b). Small arrow icons over the signal route show that the I/O ports are properly set up.



(a) Open loop input setup



(b) Open loop output setup

Figure C.3: Step 2: Open loop I/O setup

Step 3: Trim the model

After setting up the model I/O signals, the user can open *Linear Analysis Tool* by choosing *Model Linearizer* from either *Apps* or *Linearization* tabs. Once loaded, the initial conditions have to be set up for the linearization process. To do that, the user can trim the model by choosing *Trim Model...* from *Model Initial Condition* in *Setup* section of *Linear Analysis* tab.

For this example, the only checkboxes that need to be marked in opened dialog are *Steady State* for all three states of both `Satellite Dynamics/6DOF ECEF (Quaternion)/p,q,r` and `Satellite Dynamics/Euler_B`, as can be seen on Figure C.4. After that the user has to click the *Start trimming* button.

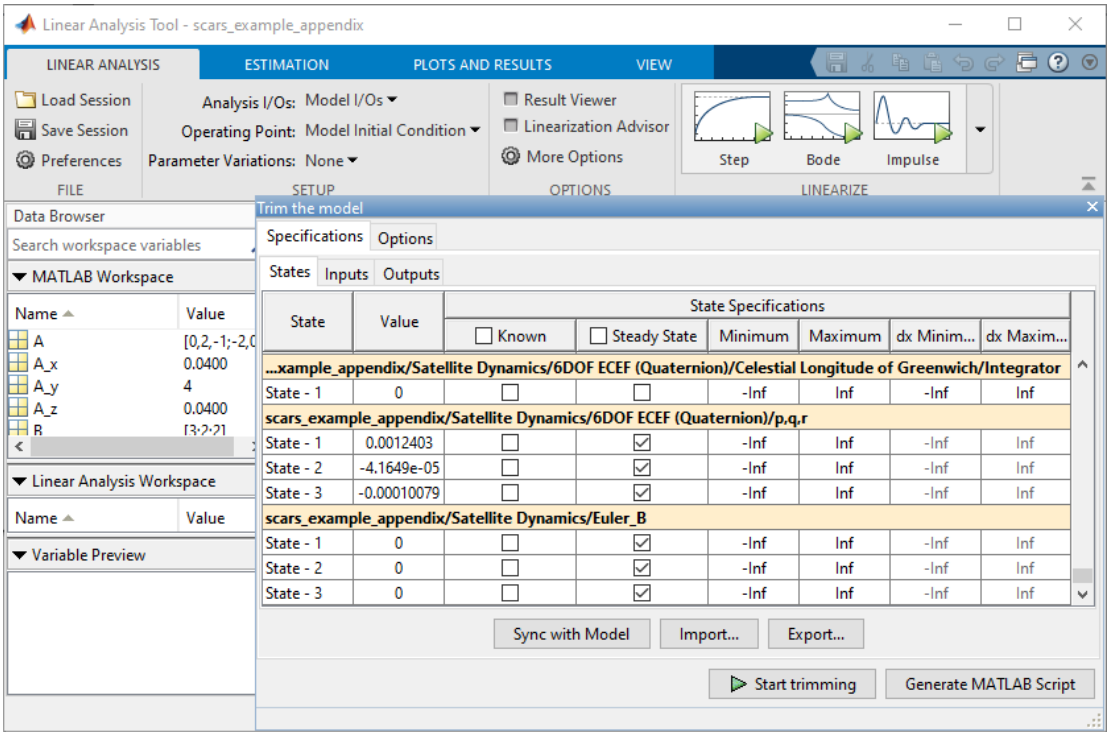


Figure C.4: Model trimming

Step 4: Linearization of the model

Once the model has been trimmed the initial conditions are automatically set. Then, the user can launch linearization process from *Linearize* section of *Linear Analysis* tab, by choosing any plot or option there. This should result in a plot (if chosen) and linearized model available in *Linear Analysis Workspace*, under the

name `linsys1` or similar. The user can then move it it *MATLAB Workspace* and use it for various purposes, such as control system design as presented in example in Section 4.4.1.

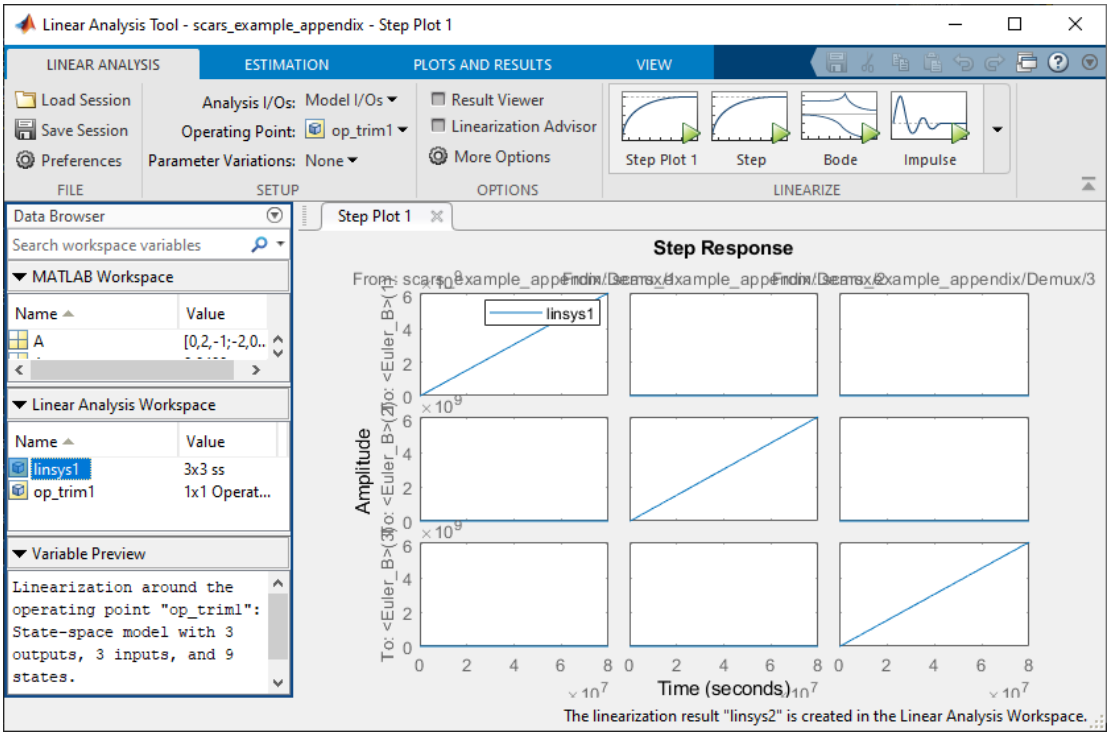


Figure C.5: Linear Analysis Tool window after successful linearization