MINISTÉRIO DA CIÊNCIA, TECNOLOGIA, INOVAÇÕES E COMUNICAÇÕES
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

sid.inpe.br/mtc-m21c/2019/11.11.10.41-RPQ

SATELLITE SIMULATION DEVELOPER'S GUIDE – ATTITUDE DYNAMICS AND CONTROL OF NONLINEAR SATELLITE SIMULATIONS

Alessandro Gerlinger Romero

URL do documento original:
<http://urlib.net/8JMKD3MGP3W34R/3UCPLUE>

INPE
São José dos Campos
2020

PUBLICADO POR:

Instituto Nacional de Pesquisas Espaciais - INPE
Gabinete do Diretor (GBDIR)
Serviço de Informação e Documentação (SESID)
CEP 12.227-010
São José dos Campos - SP - Brasil
Tel.:(012) 3208-6923/7348
E-mail: pubtc@inpe.br

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA, INOVAÇÕES E COMUNICAÇÕES
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

sid.inpe.br/mtc-m21c/2019/11.11.10.41-RPQ

SATELLITE SIMULATION DEVELOPER'S GUIDE – ATTITUDE DYNAMICS AND CONTROL OF NONLINEAR SATELLITE SIMULATIONS

Alessandro Gerlinger Romero

URL do documento original:
<http://urlib.net/8JMKD3MGP3W34R/3UCPLUE>

INPE
São José dos Campos
2020

ABSTRACT

The satellite attitude and orbit control subsystem (AOCS), that one in charge of the attitude control, can be designed with success by linear control theory if the satellite has slow angular motions and small attitude maneuver. However, for large and fast maneuvers, the linearized models are not able to represent all the perturbations due to the effects of the nonlinear terms present in the dynamics and in the actuators (e.g., saturation) which can damage the system's performance. Therefore, in such cases, it is expected that nonlinear control techniques yield better performance than the linear control techniques, improving the AOCS pointing accuracy without requiring a new set of sensors and actuators. One candidate technique for the design of AOCS control law under a large and fast maneuver is the State-Dependent Riccati Equation (SDRE). SDRE provides an effective algorithm for synthesizing nonlinear feedback control by allowing nonlinearities in the system states while offering great design flexibility through state-dependent weighting matrices. The Brazilian National Institute for Space Research (INPE, in Portuguese) was demanded by the Brazilian government to build remote-sensing satellites, such as the Amazonia-1 and CONASAT mission. In such missions, the AOCS must stabilize the satellite in three-axes so that the optical payload can point to the desired target. Currently, the control laws of AOCS are designed and analyzed using linear control techniques in commercial software. In this work, we report an open-source nonlinear satellite simulator built to analyze control laws and their stability and robustness. This satellite simulator is implemented in Java using Hipparchus (linear algebra library; which was extended in order to support the SDRE technique) and Orekit (flight dynamics framework). The initial results ratify that SDRE yields better performance in the INPE's missions.

SATELLITE SIMULATION GUIA DO DESENVOLVEDOR – DINÂMICA E CONTROLE DE ATITUDE DE SIMULAÇÕES DE SATÉLITE NÃO LINEAR

RESUMO

O subsistema de controle de atitude e órbita (AOCS), aquele responsável pelo controle de atitude, pode ser projetado com sucesso através da teoria de controle linear se o satélite tem movimentos angulares lentos e pequenas manobras de atitude. No entanto, para grandes e rápidas manobras, os modelos linearizados não são capazes de representar todas as perturbações devido aos efeitos dos termos não lineares presentes na dinâmica e nos atuadores (por exemplo, saturação), o que pode comprometer o desempenho do sistema. Portanto, nestes casos, é esperado que técnicas de controle não linear apresentem melhor desempenho que técnicas lineares, melhorando a acurácia de apontamento do AOCS sem necessitar de conjuntos adicionais de sensores e atuadores. Uma técnica candidata para o projeto do controle para grandes e rápidas manobras é a equação de Riccati dependente de estado (*State-Dependent Riccati Equation*; SDRE). SDRE fornece um algoritmo efetivo para a sintetização de controle baseado em realimentação (*feedback control*) permitindo não linearidades nos estados do sistema enquanto oferece grande flexibilidade de projeto. O Instituto Nacional de Pesquisas Espaciais (INPE) é demandado pelo governo Brasileiro para projetar e implantar satélites de sensoriamento remoto, como as missões Amazonia-1 e CONASAT. Em tais missões, o AOCS deve estabilizar o satélite em três eixos de forma que a carga ótica útil possa apontar para o alvo em solo. Atualmente, o controle do AOCS é projetado e analisado usando controle linear em software comercial. Neste trabalho, apresentamos um simulador de satélites não linear projetado para analisar técnicas de controle bem como sua estabilidade e robustez. Este simulador de satélites é implementado em Java usando-se Hipparchus (uma biblioteca de álgebra linear, que foi estendida para suportar a técnica SDRE) e Orekit (um quadro de trabalho para dinâmica de vôo). Os resultados iniciais ratificam que o SDRE oferece melhor performance para as missões do INPE.

LIST OF FIGURES

Page

LIST OF TABLES

Page

CONTENTS

1 INTRODUCTION

The design of a satellite attitude and orbit control subsystem (AOCS), the one in charge of the attitude control, that involves plant uncertainties, large angle maneuvers and fast attitude control following a stringent pointing, requires nonlinear control methods in order to satisfy performance and robustness requirements. An example is a typical mission of the Brazilian National Institute for Space Research (INPE), in which the AOCS must stabilize a satellite in three-axes so that the optical payload can point to the desired target with few arcsecs of pointing accuracy, e.g., Amazonia-1 (ROMERO et al., 2018). Another example is the Nano-satellite Constellation for Environmental Data Collection (CONASAT) (CARVALHO, 2010; B. D. Reis de Mesquita and H. Koiti Kuga and V. Carrara, 2017; ROMERO; SOUZA, 2018), a set of remote sensing CubeSats of the INPE, in which the AOCS must stabilize the satellite in three-axes in order to maximize the receiving of environment data sent by platforms in the Brazilian territory.

One candidate method for a nonlinear AOCS control law is the State-Dependent Riccati Equation (SDRE) method, originally proposed by (PEARSON, 1962) and then explored in detail by (ÇIMEN, 2008; ÇIMEN, 2010; CLOUTIER et al., 1996). SDRE provides an effective algorithm for synthesizing nonlinear feedback control by allowing nonlinearities in the system states while offering great design flexibility through state-dependent weighting matrices. The SDRE can be considered as the nonlinear counterpart of linear-quadratic regulator (LQR) control method (CLOUTIER et al., 1996; ÇIMEN, 2008; ÇIMEN, 2010). SDRE is based on the arrangement of the system model in a form known as state-dependent coefficient (SDC) matrices. Accordingly, a suboptimal control law is carried out by a real-time solution of an algebraic Riccati equation (ARE) using the SDC matrices by means of a numerical algorithm. Therefore, SDRE linearizes the plant about the instantaneous point of operation and produces a constant state-space model of the system. The process is repeated in the next sampling steps, producing and controlling several state dependent linear models out of a nonlinear one.

It is beyond the scope of the present report the following related topics: attitude estimation based on noisy sensor measurements (HUGHES, 1986; SIDI, 2006) and real-time implementation concerns of a SDRE controller based on the Java software (ARMBRUSTER et al., 2007; SHARP et al., 2003) or other software languages as C (MENON et al., 2002).

1.1 Motivation

Problem statement: *National Institute of Space Research (INPE) was demanded by the Brazilian government to build remote-sensing satellites, such as the Amazonia-1 and CONASAT missions. Currently, the control laws of AOCS are designed and analyzed using linear control techniques in commercial software. However, for large and fast maneuvers, the linearized models are not able to represent all the perturbations which can damage the system's performance. One candidate technique for the design of AOCS control law under a large and fast maneuver is the State-Dependent Riccati Equation (SDRE). Moreover, for INPE, the capability to define, to use and to provide an open-source Satellite Simulation has strategic relevance aiming cost reduction and stablishing itself as a open-source layer.*

1.2 Aim

In this report, we present the investigation of a SDRE controller performance by simulations. The simulator is implemented based on Java and related open-source software libraries (Hipparchus - linear algebra library, and Orekit - flight dynamics library), therefore, it can run in a variety of platforms - including an Android operating system in a CubeSat -, it has low cost and it is available to analysis, development and extension. The Hipparchus open-source library was extended in order to solve the ARE equation that is the cornerstone of the SDRE method, a major contribution of the simulator (see Appendix A). Furthermore, the simulator is capable to provide a 3D animation of the spacecraft attitude using Java 3D (JAVA..., 2017).

To the best of our knowledge, neither a Java implementation for an ARE solver nor a Java open-source option for simulation of SDRE controllers are available. Hence, those products constitute major contributions of this work, which can be deployed in an Android operating system in a CubeSat focused on remote sensing. Moreover, since SDC matrices are nonunique, there is no work focused on the optimal arrangement of the SDC for the satellite attitude control stabilization. Such optimal arrangement has the potential to increase performance and enhance robustness. Therefore, another major contribution of the present work is the explicit modeling of the state-space model for a three-axes stabilized attitude-maneuvering satellite using quaternions, Gibbs vector and modified Rodrigues parameters (MRPs). The final, and most important, is the evaluation of which of these models is the optimal factorization of the SDRE technique in an AOCS with nonlinear dynamics for a given Monte Carlo perturbation model based on a set of parameters, initial conditions and references for the controller.

The models are evaluated through Monte Carlo perturbation models in the Satellite Simulation using different linear and nonlinear control laws for an attitude maneuver called the upside-down in the launch and early orbit phase (LEOP). In LEOP, the AOCS must dump the residual angular velocity and point the satellite solar panels towards the Sun.

1.3 Outline

This developer's guide is organized as follows. Chapter 2 explains the procedure to install the software requirements in order to explore, to use and to extend the distributed package containing the Satellite Simulation executable as well as the source files. Chapter 3 presents the dynamics of the attitude and for the sake of completeness a simplified orbit model that supports the simulations. Chapter 4 briefly introduces the SDRE technique and explores the evaluated alternatives for the controller design. Chapter 5 shares some data extracted from the simulations that ratify the better performance exhibited by SDRE controllers. Finally, conclusions are shared in Chapter 6. A list of publications supported by the Satellite Simulation is available in Appendix C.

Using the Distributed Package

Each paragraph of this developer's guide that refers to the distributed package is marked as this example.

---

2 INSTALLING THE DISTRIBUTED PACKAGE

This chapter presents the software requirements, the installation procedure, the contents of the distributed package of Satellite Simulation (ROMERO, 2019a) and how to use it. Moreover, the source code (ROMERO, 2019b) is also available for analysis and extension.

Recall Satellite Simulation (ROMERO, 2019a) is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version. Satellite Simulation is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with Satellite Simulation. If not, see <http://www.gnu.org/licenses/>.

2.1 Software Requirements

The development environment for all the computer-readable material is the "Java Development Kit" (ORACLE, 2017) 1.8, or later, and the software project management maven (FOUNDATION, 2017).

The main runtime library supporting the distributed package is Orekit (OREKIT..., 2017). Orekit, a low level space dynamics library written in Java, has gained recognition since it was released under an open source license in 2008. Orekit aims at providing accurate and efficient low level components for the development of flight dynamics applications. As a library, Orekit provides basic elements (orbits, dates, attitude, frames, ...) and various algorithms to handle them (conversions, propagations, pointing, ...).

Orekit has GeoMagneticField class, which calculates the magnetic field for a given time and position (latitude, longitude and altitude). In order to the GeoMagneticField work, it is need to provide the International Geomagnetic Reference Field (IGRF) files. In particular, the file IGRF.COF must be readable, such file is provided by Geomag(OCEANIC; ADMINISTRATION, 2017) for the version IGRF12.

Furthermore, Orekit is based on Hipparchus (HIPPARCHUS..., 2018). Hipparchus is a libray of lightweight, self-contained mathematics ans statistics components addressing the most common problems not available in Java. In the current context, Hip-

parchus provides the basic mathematical structures for linear algebra and geometry, including the Rotation class (This class implements rotations in a three-dimensional space. Rotations can be represented by several different mathematical entities - matrices, axe and angle, Cardan or Euler angles, quaternions. This class presents an higher level abstraction, more user-oriented and hiding this implementation details. It uses quaternions for the internal representation).

Additionally, libraries for visualization is used, which are: (1) JMathPlot, which provides interactive 2D/3D plot (RICHET, 2017); and, (2) Java3D, which provides a framework for 3D animations (JAVA..., 2017).

2.2 Installation Procedure

This section presents the installation procedure for the software requirements and the distributed package of Satellite Simulation v1.0.

2.2.1 Microsoft Windows

a) Install and configure Java Development Kit 1.8 (ORACLE, 2017);

b) Install and configure Maven (FOUNDATION, 2017);

c) In order to visualize the 3D animations, install and configure Java 3D (JAVA..., 2017);

d) Uncompress the distribution package (ROMERO, 2019a) or the source package (ROMERO, 2019b);

2.3 The Distributed Package and its Content

Once the installation procedure is successfully completed, the local directory should be the one shown in Fig. 2.1. The distributed package can be explained as follows:

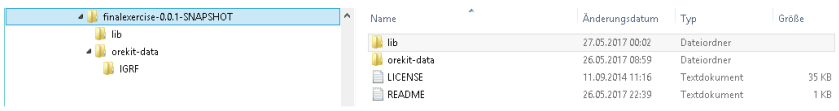


Figure 2.1 - The content of distributed package.

- `lib`

`satellitesimulator-<<version>>.jar` is the main library of the distributed package.

`orekit-<<version>>.jar` is the library of the Orekit (OREKIT..., 2017).

`hipparchus-*-<<version>>.jar` is the library of the Hipparchus (HIPPARCHUS..., 2018).

`JMathPlot-*-<<version>>.jar` is the library of the JMathPlot (RICHET, 2017).

`j3d-core-<<version>>.jar` is the library of the Java3D (JAVA..., 2017).

- `orekit-data` has the necessary files to support the orekit runtime library, including `tai-utc.dat` (OREKIT..., 2017) and `IGRF.COF` (OCEANIC; ADMINISTRATION, 2017).

2.4 Using the Distributed Package

This section presents the common procedures to use the distributed package.

The main command is:

```
java -Dorekit.data.path=./orekit-data -Xms1G -Xmx2G -jar lib\satellitesimulator-0.1
```

The execution of the command generates the console output shown in Fig. 2.2. The console shows the orbit parameters, the mean motion (radians/second) and the

```
SimulationController [Java Application] C:\Program Files\Java\jre1.8.0_74\bin\javaw.exe (03.11.2019 14:57:48)
14:58:01.567 [main] INFO br.inpe.cmc202.simulation.SimulationController  - *********************************
14:58:01.606 [main] INFO br.inpe.cmc202.simulation.SimulationController  - Satellite Simulation
14:58:01.606 [main] INFO br.inpe.cmc202.simulation.SimulationController  - *********************************
14:58:01.607 [main] INFO br.inpe.cmc202.simulation.SimulationController  - Loading configuration...
14:58:01.607 [main] INFO br.inpe.cmc202.simulation.SimulationController  - Reading "simulationcontroller.properties"...
14:58:01.608 [main] INFO br.inpe.cmc202.simulation.SimulationController  - Reading "amazonia1.properties"...
14:58:01.608 [main] INFO br.inpe.cmc202.simulation.SimulationController  - Configuration loaded.
14:58:01.624 [main] INFO br.inpe.cmc202.simulation.SimulationController  - Configuring simulation...
14:58:02.307 [main] INFO br.inpe.cmc202.simulation.SimulationController  - Orbit: keplerian parameters: {a: 7130092.0; e: 0.001111; i: 98.405; pa: 98.405; ra
14:58:02.307 [main] INFO br.inpe.cmc202.simulation.SimulationController  - MeanMotion: 0.001048639438789967 radians/s
14:58:02.308 [main] INFO br.inpe.cmc202.simulation.SimulationController  - Period: 5991.749952137793 s
14:58:02.308 [main] INFO br.inpe.cmc202.simulation.SimulationController  - Simulation Time: 700.0 s (11.682730514317399 % of the orbit period)
14:58:02.311 [main] INFO br.inpe.cmc202.simulation.SimulationController  - Start Time: 2017-06-01T00:00:00.000
14:58:02.312 [main] INFO br.inpe.cmc202.simulation.SimulationController  - End Time: 2017-06-01T00:11:40.000
14:58:02.312 [main] INFO br.inpe.cmc202.simulation.SimulationController  - Fixed simulation step: 0.005 s
14:58:02.318 [main] INFO br.inpe.cmc202.simulation.SimulationController  - Satellite Configuration: Satellite [
 setOfMagnetorquer=null,
 setOfReactionWheels=br.inpe.cmc202.satellite.actuators.SetOfReactionWheels@527e5409,
 controller=ProportionalNonLinearQuaternionSDREController [kinematicsDefinition=GIBBS, Q_sqrt=Array2DRowRealMatrix{{1.0,0.0,0.0,0.0,0.0,0.0,0.0},{0.0,1.0,0.0
 setOfMagnetorquerController=null,
 angularVelocityReference_body={0; 0; 0},
 sunReference_body={1; 0; 0},
 externalTorquesMagnitude=0.0,
 alpha1=0.0]
14:58:02.318 [main] INFO br.inpe.cmc202.simulation.SimulationController  - Initial Attitude - Euler Angles: X 0.0 Y -0.0 Z 180.0
14:58:02.318 [main] INFO br.inpe.cmc202.simulation.SimulationController  - Initial Attitude - Quaternions: Q1 -1.0 Q2 0.0 Q3 0.0 Q4 6.123233995736766E-17
14:58:02.319 [main] INFO br.inpe.cmc202.simulation.SimulationController  - Initial Attitude - Velocity (rad/s): X 0.0 Y 0.0 Z 0.024
14:58:02.319 [main] INFO br.inpe.cmc202.simulation.SimulationController  - Initial Attitude - Acceleration (rad/s^2): X 0.0 Y 0.0 Z 0.0
14:58:02.319 [main] INFO br.inpe.cmc202.simulation.SimulationController  - Simulation configured.
14:58:02.672 [main] INFO br.inpe.cmc202.simulation.StepHandler  - Running the simulation...
1%2%4%5%7%8%9%11%12%14%15%16%18%19%21%22%23%25%26%28%29%30%32%33%35%36%38%39%40%42%43%45%46%47%49%50%52%53%54%56%57%59%60%61%63%64%66%67%69%70%71%73%74%
```

Figure 2.2 - Running the distributed package.

period of the orbit (seconds) computed based on the orbit parameters. Moreover, it shows the simulation time and the integration step (fixed in the satellite simulation). Afterwards, all the computations are done and when finished the graphs are shown.

There are 23 graphs and 1 animation, which are described below:

- Orbital graphs

  True Anomaly - (1)

  Positon Earth Centered Inertial (ECI) - (14)

  Positon Earth Centered Earth Fixed (ECEF) - (15)

  Positon Latitude Longitude Altitude (LLA) - (16)

- Attitude graphs

  Angular velocity of the satellite in respect to the ECI - (2)

  Solar vector in the satellite - (3)

  Euler angles in the ECI - (12)

  Euler angles in the ECEF - (13)

  Rotational part of the kinetic energy - (17)

- Control graphs

  Sensors - a set of sun sensors, a magnetometer and a gyroscope

  Error of the solar vector in the satellite - (4)

  Magnetic vector in the satellite - (8)

  Norm of the magnetic vector in the satellite - (9)

  Angular velocity of the satellite - (2)

  Quaternion of the satellite - (18)

  Quaternion error of the satellite - (19)

  Sun pointing error in the satellite - (20)

  State space quaternion - how the sub state space of vector part of quaternion evolve in time - (22)

  State space angular velocity - how the sub state space of angular velocities evolve in time - (23)

  Actuators - a set of reaction wheels and a set of margnetorque

Reaction wheel control torque - (5)

Reaction wheel angular velocity - (6)

Reaction wheel norm of the angular momentum - (7)

Magnetorque control torque - (10)

Magnetic dipole generated by the magnetorque - (11)

Controller - determinant of controllability matrix (21)

Controller - condition number of controllability matrix (24)

- Attitude animation

  The satellite shape (computed based on the main axes of inertia) and its attitude

A typical attitude graph is shown in Fig. 2.3.

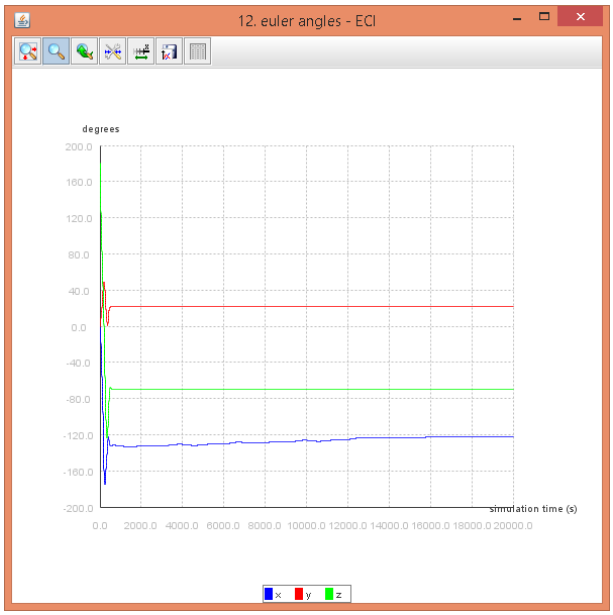


Figure 2.3 - Euler angles in the ECI.

The attitude animation shows the evolution of the attitude of the satellite - viewed from a dextral viewpoint with Z alligned in the direction of the viewer. A typical frame of the attitude animation is shown in Fig. 2.4. The text shares the time of the current attitude and the norm of the control torque of the reaction wheels.

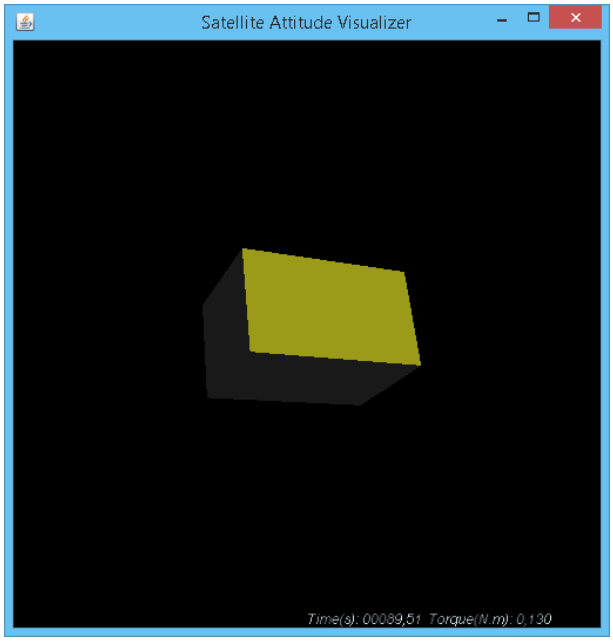


Figure 2.4 - A sample of the satellite shape (computed based on the main axis of inertia) and its attitude.

Using the Distributed Package

There are two major points of changing to evaluate a different satellite configuration:

- `simulation.properties` - changing the start time, initial attitude, orbit parameters (a - semi-major axis (m), e - eccentricity, i - inclination (rad), pa - perigee argument, raan - right ascension of ascending node, anomaly - mean anomaly), step size, simulation time, the sattelite characteristics `<<satellite.properties>>`, and, finally, the reaction wheels controller (*NopeController* - to study the movement of the satellite, a torque free rigid body) as well as the magnetorques controller;

- `<<satellite.properties>>` - changing the reference points (reference vector and reference angular velocity), inertia tensor and configuration of the satellite (if it has magnetorques, if it has reaction wheels).

Additionally, the graphs are plotted by the class `Plotter` and the attitude animation is generated by the class `SatelliteAttitudeVisualizer`. These two classes are commanded by the `StepHandler`, which monitors the simulation storing the values that are used for the graphs and the attitude animation.

It is important to highlight that the Satellite Simulation works with a **fixed step size** for simulation so it is very important to choose the right step size in the `simulation.properties`. Furthermore, such step size is used to the dynamics of the satellite as well as to the controller. Therefore, to increase the applicability of the simulation results, a possible extension is to use a step size for the dynamics and another step size for the control. The heuristics commonly advocated by INPE's engineering staff is to use one second for the satellite control and one hundredth of one second for missiles and aircrafts.

---

3 SATELLITE ATTITUDE DYNAMICS

Orbital dynamics is presented for the sake of completeness since it is not the main object of the current work, nonetheless, it is mandatory for the magnetic actuation. The satellite attitude dynamics is presented in the sequel.

3.1 Orbital Dynamics

Accurate and well-defined time references and cordinate frames are the fundamental basis for orbital dynamics. The next subsections briefly review aspects of time and coordinate frames applied in the Satellite Simulation.

3.1.1 Time

The time references used by the Satellite Simulation are: (1) UTC - Coordinated Universal Time - based on transitions between energy levels in atomic oscillators; and, (2) GMST (Θ) - Greenwich Mean Sideral Time - based on the periodic process associated with the Earth's rotation (ESA, 2017).

Using the Distributed Package

UTC is used in the constructor of the `SimulationController` class, in the following statement: `new AbsoluteDate(2017, 6, 1, 11, 0, 0, TimeScalesFactory.getUTC());` GMST is used in the s `EarthCenteredEarthFixedTransformProvider` class, which computes the angle for a given time to define a rotation around the Z-axis.

3.1.2 Coordinate Frames

The Satellite Ssmulation uses four coordinate frames:

a) Earth-Centered Inertial (ECI) system - is a quasi-inertial reference system, in which the X-axis points in the direction of the mean vernal equinox, the Z-axis points in the direction of the true Earth's rotation at the defined epoch and Y-axis is orthogonal to the former axes. Moreover, it has its origin in the Earth's centre of mass. It is also called Conventional Celestial Reference System (CRS) (ESA, 2017). As a reference system, it has many implementations. True Equator Mean Equinox (TEME) is the implementation used by the Satellite Simulation as the ECI reference frame.

b) Earth-Centered Earth-Fixed (ECEF) system - also called Conventional

Terrestre Reference System (TRS), it is co-rotating with the Earth. It has its origin in the Earth's centre of mass, and the axes are defined as follows: the Z-axis points in the direction of the Earth's rotation, the X-axis is defined as the intersection of the equatorial plane (orthogonal to the Z-axis) with the mean Greenwich meridian, and the Y-axis is orthogonal to both of them (ESA, 2017). In the Satellite Simulation, the transformation from ECI to ECEF is achieved by rotating the ECI around Z-axis by the current GMST angle (neither precession nor nutation nor pole movement are taken into account in this transformation in the present work (ESA, 2017)).

c) Geodetic Coordinates of Latitude, Longitude and Altitude (LLA)(φ, λ, h) system - it is based on the World Geodetic System 1984 (WGS84), an ellipsoid that has its origin in the Earth's centre of mass. The Z-axis is described as the perpendiular axis above the ellipsoid surface, the X-axis pierces the Greenwich meridian (longitude = 0 degrees), and Y-axis pierces the Equator (latitude = 0). In the Satellite Simulation, the transformation from ECEF to LLA is achieved using the ellipsoidal parameters of WGS84 and the closed fomulas presented in Equation 3.1 (AG, 2017).

$$\begin{aligned} \lambda &= \arctan \tfrac{Y}{X} \\ \varphi &= \arctan \tfrac{Z+e'^2 b \sin^3\theta}{p-e^2 a \cos^3\theta} \\ h &= \tfrac{p}{\cos\varphi} - N \\ \\ N &= \tfrac{a}{\sqrt{1-e^2\sin^2\varphi}} \\ p &= \sqrt{X^2+Y^2} \\ \theta &= \arctan \tfrac{Za}{pb} \end{aligned} \tag{3.1}$$

d) North-East-Down (NED) system - also called local tangent plane (LTP), it has as origin the satellite. The X-axis points in the direction of the north, the Y-axis points to the east and Z-axis is orthogonal to the former axes pointing downward. In the Satellite Simulation, the transformation from ECEF to NEED is performed rotating about the Z-axis through longitude and then rotating about the Y-axis by latitude (AG, 2017).

Using the Distributed Package

ECI is defined in the `SimulationController` class using the statement:

`this.eci = new EarthCenteredInertialFrame(FramesFactory.getTEME());`
ECEF is defined by the `EarthCenteredEarthFixedFrame` class, which uses the `EarthCenteredEarthFixedTransformProvider` class as a transform provider. The `EarthCenteredEarthFixedTransformProvider` implements the rotation around the Z-axis by the current GMST.
LLA and NED are used and defined by the `Magnetometer` class, in particular, the method `transformECEFToLLA` transform a given vector described in ECEF to LLA, and the method `transformNEDtoECEF` transform a given vector described in NED to ECEF.

3.1.3 Orbit Definition and Propagation

The Satellite Simulation is based on an elliptical Keplerian orbit (HUGHES, 1986; SIDI, 2006). Such orbits are defined by two parameters: a, the semi-major axis of the orbital ellipse; and, e, the numerical eccentricity of the orbital ellipse. The positioning of these orbits in a given ECI reference frame requires three more parameters, which are: (1) the inclination, is the angle between the orbital plane and the equator; (2) the right ascension of ascending node, is the angle between the ascending node direction and X-axis of ECI; and, (3) the perigee argument, is the angle between the ascending node and perigee directions, measured along the orbital plane (HUGHES, 1986; SIDI, 2006; ESA, 2017).

Equipped with a Keplerian orbit defined by these constant parameters and starting from an initial condition containing an anomaly (in the Satellite Simulation, the mean anomaly $M(t)$) and a date, it is possible to propagate the orbit of a satellite using the following set of Equations 3.2, which defines the eccentric anomaly $E(t)$ and true anomaly $V(t)$.

$$M(t) = \sqrt{\frac{\mu}{a^3}}\, t \tag{3.2a}$$

$$E(t) - e\sin(E(t)) - M(t) = 0 \tag{3.2b}$$

$$V(t) = 2\arctan[\sqrt{\frac{(1+e)}{(1-e)}}\tan\frac{E(t)}{2}] \tag{3.2c}$$

Note perturbations are note considered. Moreover, Satellite Simulation decouples the orbit (translational) and attitude (rotational) dynamics. This decomposition is common in simulations mainly focused on attitude dynamics (CARRARA, 2012), as is the case with Satellite Simulation.

Using the Distributed Package

The Keplerian orbit is defined in the `KeplerianOrbitAroundEarth` class. Moreover, Equation 3.2b is available in the `KeplerEquation` class, which is solved using the Newton-Raphson method provided by Hipparchus (HIPPARCHUS..., 2018).
Finally, the orbit propagation is performed in the method `shiftedBy` of the class `KeplerianOrbitAroundEarth` computing Equation 3.2a for a given Δt.

3.2 Satellite Attitude Dynamics

The Satellite Simulation is based on a parameterizable configuration of a satellite. It is defined to be a three-axes stabilized attitude-maneuvering satellite, therefore, it is basically a *zero-bias-momentum* system. In the sense that, a major control requirement is to remove the unwanted accumulated angular momentum. Therefore, an active control system is needed. Note the existence of angular momentum in the satellite would cause control difficulties when attitude maneuvers in space would be executed since this superfulous momentum would provide the spacecraft with unwanted gyroscopic stability (SIDI, 2006).

Since an active control system is required, it is worthwhile to explore the techniques available to produce torques for the attitude control. There are two types of actuators: (1) the inertial actuators - they change the overall inertial angular momentum of the satellite, in other words, they generate external torques; (2) the momentum exchange actuators - they do not change the inertial angular momentum; or, a symmetrical rotating body produces torque when accelerated about its axis of rotation, since such change in the momentum is internal to the satellite, it transfers the momentum change to the satellite with negative sign (*angular momentum is a conservative quantity*) (SIDI, 2006).

At least, three different ways of producing torque for the attitude control of the satellite are available:

a) Inertial actuators

Earth's magnetic field - magnetorques provide continuous and smooth control, albeit, the low level of the control torques achieved, and, consequently, slow attitude maneuvers (SIDI, 2006)

Reaction force produced by a thruster - no smooth control can be

achieved owing to the inherent impulsive nature of the thrusters (SIDI, 2006), furthermore, there is no straightforwad way to refuel such actuators

b) Momentum exchange actuators

Reaction wheels - for very accurate control and for moderately fast maneuvers, the reaction wheels are preferred since they allow continuous and smooth control with the lowest possible disturbing torques (SIDI, 2006)

Focusing on the sensors, there are two principal types of attitude determination hardware: attitude sensors and angular velocity sensors (SIDI, 2006).

Fig. 3.1 shows the decomposition of the satellite available in the Satellite Simulation.

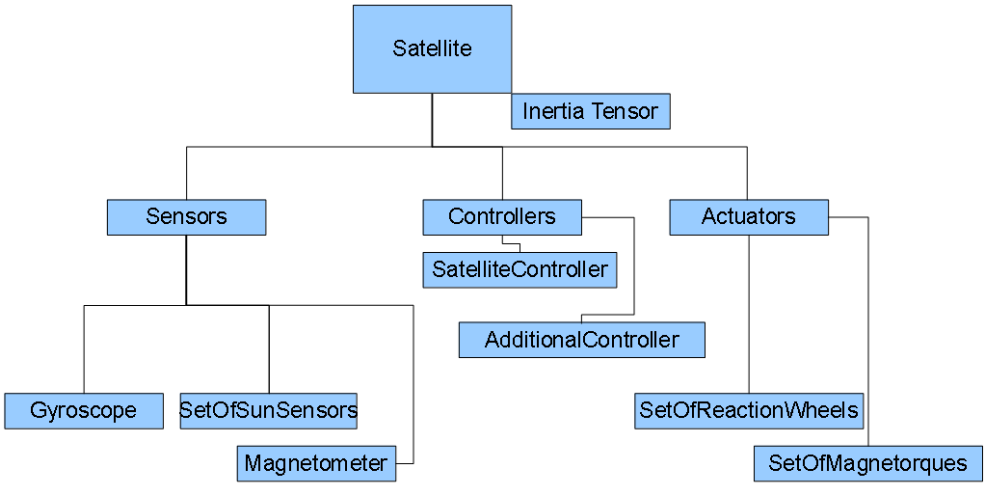


Figure 3.1 - Satellite hierarchical decomposition.

Next subsections explore the kinematics and kinetics of the satellite attitude taking into account the actuators defined. Finally, the sensors and actuators are discussed.

Using the Distributed Package

The satellite hierarchical decomposition is mapped to the Satellite Simulation packages and classes as follows:

a) Inertia Tensor `br.inpe.cmc202.satellite.Satellite.I`

b) Sensors `br.inpe.cmc202.satellite.sensors`

Gyroscope `br.inpe.cmc202.satellite.sensors.Gyroscope`

SetOfSunSensors `br.inpe.cmc202.satellite.sensors.SetOfSunSensors`

Magnetometer `br.inpe.cmc202.satellite.sensors.Magnetometer`

c) Actuators `br.inpe.cmc202.satellite.actuators`

SetOfReactionWheels `br.inpe.cmc202.satellite.actuators.SetOfReactionWheels`

SetOfMagnetorques `br.inpe.cmc202.satellite.actuators.SetOfMagnetorques`

d) Controllers `br.inpe.cmc202.satellite.controllers`

SatelliteController `br.inpe.cmc202.satellite.controllers.NopeController`, `br.inpe.cmc202.satellite.controllers.linear.*`, `br.inpe.cmc202.satellite.controllers.lqr.*` and `br.inpe.cmc202.satellite.controllers.sdre.*`

AdditionalController `br.inpe.cmc202.satellite.controllers.\ SetOfMagnetorquersController`

Finally, the Satellite is mapped to the `br.inpe.cmc202.satellite.Satellite` class.

3.2.1 Kinematics

Given the ECI reference frame ($\mathfrak{F}_i$) and the frame defined in the satellite with origin in its centre of mass (the body-fixed frame, $\mathfrak{F}_b$), then a rotation $R \in SO(3)$ - $SO(3)$ is the set of all attitudes of a rigid body described by 3×3 orthogonal matrices whose determinant is one - represented by an unit quaternion $Q = [q_1\ q_2\ q_3 \mid q_4]^T$ as well as a direction cosine matrix (DCM) C_{body_eci} can define the attitude of the satellite .

Defining the angular velocity $\vec{\omega} = [\omega_1\, \omega_2\, \omega_3]^T$ of $\mathfrak{F}_b$ with respect to $\mathfrak{F}_i$ measured in the $\mathfrak{F}_b$, the kinematics can be described by Equation 3.3 (CARRARA, 2012; HUGHES, 1986).

$$\begin{aligned} \omega^{\times}_{body\_eci} &= -\dot{C_{body\_eci}} C^T_{body\_eci} \\ \dot{C_{body\_eci}} &= C_{body\_eci} \omega^{\times}_{body\_eci} \end{aligned} \tag{3.3}$$

Equation 3.3 allows the prediction of the satellite's attitude if it is available the initial attitude and the history of the change in the angular velocity ($\dot{\theta} = F(\omega, t)$).

Using the Euler angles (3-2-1), the kinematics can also be defined by Equation 3.4 (HUGHES, 1986).

$$\begin{aligned} S = \begin{bmatrix} -\sin\theta_2 & 0 & 1 \\ -\cos\theta_2 \sin\theta_3 & \cos\theta_3 & 0 \\ \cos\theta_2 \cos\theta_3 & -\sin\theta_3 & 0 \end{bmatrix} \\ \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} = S \begin{bmatrix} \dot{\theta_1} \\ \dot{\theta_2} \\ \dot{\theta_3} \end{bmatrix} \end{aligned} \tag{3.4}$$

In order to model how the satellite attitude changes depending on angular velocity, the matrix S is inverted as shown in Equation 3.5 (it has a discontinuity at 90° since

$\cos\frac{\pi}{2} = 0$).

$$S^{-1} = \begin{bmatrix} 0 & \frac{\sin\theta_3}{\cos\theta_2} & \frac{\cos\theta_3}{\cos\theta_2} \\ 0 & \cos\theta_3 & -\sin\theta_3 \\ 1 & \frac{\sin\theta_3 \sin\theta_2}{\cos\theta_2} & \frac{\cos\theta_3 \sin\theta_2}{\cos\theta_2} \end{bmatrix}$$
$$\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix} = S^{-1} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} \tag{3.5}$$

It is common to avoid the singularities that are inherent in the Euler angles using quaternions (HUGHES, 1986; CARRARA, 2012). Equation 3.6 shows the kinematics represented using quaternions $Q = [q_1\ q_2\ q_3 \mid q_4]^T$.

$$\dot{Q} = \frac{1}{2}\Omega(\omega)Q = \frac{1}{2}\Xi(Q)\omega$$

$$\Omega(\omega) \triangleq \begin{bmatrix} 0 & \omega_3 & -\omega_2 & \omega_1 \\ -\omega_3 & 0 & \omega_1 & \omega_2 \\ \omega_2 & -\omega_1 & 0 & \omega_3 \\ -\omega_1 & -\omega_2 & -\omega_3 & 0 \end{bmatrix} \tag{3.6}$$

$$\Xi(Q) \triangleq \begin{bmatrix} q_4 & -q_3 & q_2 \\ q_3 & q_4 & -q_1 \\ -q_2 & q_1 & q_4 \\ -q_1 & -q_2 & -q_3 \end{bmatrix},$$

where the quaternion $Q = [q_1\ q_2\ q_3\ q_4]^T$ satisfies the following identity:

$$q_1^2 + q_2^2 + q_3^2 + q_4^2 = 1 \tag{3.7}$$

Nonetheless, it is worthy to mention that although the definition of the unit quaternion is global in the sense that it can represent all attitudes, each physical attitude $R \in SO(3)$ is represented by a pair of unit quaternions $\pm Q \in \mathbf{S}^3$ (FORTESCUE; SWINERD, 2011). This characteristic can produce undesirable effects as *unwind*, in which the trajectories of the closed-loop system start close to the desired attitude and yet travel a large distance before returning to the desired attitude (FORTESCUE; SWINERD, 2011).

Equation 3.6 can be rewritten to separate terms with q_4 from other elements of the quaternion. Define a vector part of the first three components of the quaternion and denote this by g (Gibbs vector or Rodrigues parameter) as $Q = [g^T|q_4]^T$.

$$\dot{Q} = -\frac{1}{2}\begin{bmatrix}\omega^\times \\ \omega^T\end{bmatrix}\begin{bmatrix}q_1 \\ q_2 \\ q_3\end{bmatrix} + \frac{1}{2}q_4\begin{bmatrix}I_{3\times 3} \\ 0\end{bmatrix}\omega \tag{3.8}$$

Note the Gibbs vector is geometrically singular since it is not defined for 180° of rotation (FORTESCUE; SWINERD, 2011), nonetheless, the Eq. (3.8) is global.

Another option is the modified Rodrigues parameters (MRP) p, which is defined as Equation 3.9 - it is geometrically singular since it is not defined for 360° of rotation (FORTESCUE; SWINERD, 2011), moreover, it is not global since the sole singularity is $q_4 = -1$, what is difficult to occur in numerical simulations due to precision.

$$p = \frac{g}{1+q_4} \tag{3.9}$$

Using Equation 3.9, the kinematics is defined by Equation 3.10 (SHUSTER, 1993).

$$\begin{aligned}\dot{p} = \frac{1}{4}[(1-|p|^2)\omega - 2\omega\times p + 2(\omega.p)p] = \\ \frac{1}{4}(1-|p|^2)\omega + \frac{1}{4}[-2\omega\times p + 2(\omega.p)p] \\ \frac{1}{4}(1-p^Tp)\omega + \frac{1}{4}[-2\omega^\times + 2(\omega^Tp)I_{3\times 3}]p \\ \frac{1}{4}(1-p^Tp)\omega + [-\frac{1}{2}\omega^\times + \frac{1}{2}(\omega^Tp)I_{3\times 3}]p\end{aligned} \tag{3.10}$$

$$\dot{p} = \frac{1}{2}[\frac{1}{2}(1-p^Tp)I_{3\times 3} + p^\times + pp^TI_{3\times 3}]\omega$$

Using the Distributed Package

The kinematics is implemented in the Satellite Simulation by the `KinematicsAttitudeProvider`. It stores the previous attitude defined by the initial attitude firstly and then propagates it using the numerical integration of

Equation 3.6, which leads to, using the explicit Euler method:

$$Q_k = Q_{k-1} + \frac{1}{2}\Omega(\omega_{k-1})Q_{k-1}dt \tag{3.11}$$

A better option would be to use matrix exponential, which would lead to:

$$Q_k = e^{\frac{1}{2}\Omega(\omega_{k-1})dt}Q_{k-1} \tag{3.12}$$

3.2.2 Kinetics

In order to know the history of the change in the angular velocity, it is necessary to understand the history of the change in the angular acceleration ($\dot{\omega} = G(\tau, t)$) of the satellite. According to the Euler-Newton formulation of the rotational motion, angular acceleration is caused by torques, in other words, the change in the angular momentum $\vec{h}$ is equals to the net torques $\vec{g}$ applied in the satellite, see Equation 3.13 (the present subsection is derived based on the centre of mass of the satellite, for the general case, see (CARRARA, 2012; HUGHES, 1986)).

$$\dot{\vec{h}} = \vec{g} \tag{3.13}$$

The angular momentum is also known as the moment of momentum since it defines the moment of a given momentum $\vec{p}$ ($\vec{p} \triangleq m\vec{v}$) about a given point P_{cm}. See Equation 3.14, in which r locates a given point p with respect to P_{cm}.

$$\vec{h} = \vec{r} \times \vec{p} \tag{3.14}$$

Now, taking into account the motion of the body-fixed frame $\mathfrak{F}_b$ with respect to the ECI $\mathfrak{F}_i$ and an angular velocity ω of $\mathfrak{F}_b$ with respect to $\mathfrak{F}_i$ measured in the $\mathfrak{F}_b$, the derivative of the angular momentum in $\mathfrak{F}_b$ is defined by Equation 3.15.

$$\dot{\vec{h}} = \vec{g} - \vec{\omega} \times \vec{h} \tag{3.15}$$

Furthermore, $\vec{h} = \vec{I}.\vec{\omega}$ and $\vec{h} = \vec{I}.\vec{\omega}$, which results in the Equation 3.16.

$$\vec{I}.\dot{\vec{\omega}} = \vec{g} - \vec{\omega} \times (\vec{I}.\vec{\omega}) \tag{3.16}$$

Recall the satellite has a set of 3 reaction wheels, each one aligned with its principal axes of inertia, furthermore, such type of actuator, momentum exchange actuators, does not change the angular momentum of the satellite. Consequentely, it is mandatory to model their influence in the satellite, in particular, the angular momentum, in the scalar form, of the satellite is defined by Equation 3.17.

$$\vec{h} = (I - \sum_{n=1}^{3} I_{n,s} a_n a_n^T)\vec{\omega} + \sum_{n=1}^{3} h_{w,n}\vec{a_n} \tag{3.17}$$

where $I_{n,s}$ is the inertia moment of the reaction wheels in their symmetry axis a_n, $h_{w,n}$ is the angular momentum of the n reaction wheel about its centre of mass ($h_{w,n} = I_{n,s} a_n^T \omega + I_{n,s}\omega_n$) and ω_n is the angular velocity of the n reaction wheel.

One can define I_b using the Equation 3.18.

$$I_b \triangleq I - \sum_{n=1}^{3} I_{n,s} a_n a_n^T \tag{3.18}$$

Using I_b, the motion of the satellite is described by Equation 3.19 (expanded until the version used in the Satellite Simulation).

$$\begin{aligned}
I_b\dot{\omega}^b &= g_{cm} - \omega^\times(I_b\omega + \sum_{n=1}^{3} h_{w,n}a_n) - \sum_{n=1}^{3} g_n a_n \implies \\
\dot{\omega}^b &= I_b^{-1}(g_{cm} - \omega^\times(I_b\omega + \sum_{n=1}^{3} h_{w,n}a_n) - \sum_{n=1}^{3} g_n a_n) = \\
& I_b^{-1}g_{cm} - I_b^{-1}\omega^\times(I_b\omega + \sum_{n=1}^{3} h_{w,n}a_n) - I_b^{-1}\sum_{n=1}^{3} g_n a_n = \\
& I_b^{-1}g_{cm} - I_b^{-1}\omega^\times I_b\omega - I_b^{-1}\omega^\times \sum_{n=1}^{3} h_{w,n}a_n - I_b^{-1}\sum_{n=1}^{3} g_n a_n
\end{aligned} \tag{3.19}$$

where g_{cm} is the net external torque, in the Satellite Simulation, the torque generated by the magnetorques; and g_n are the torques generated by the reactions wheels ($\dot{h_{w,n}} = g_n$).

Using the Distributed Package

The kinetics is implemented in the Satellite Simulation by the `KineticsAttitudeModifier`. Firstly, it propagates the kinetics using the nu-

merical integration of Equation 3.19, which leads to:

$$\omega_k^b = \omega_{k-1}^b + [I_b^{-1} g_{cm_{k-1}} - I_b^{-1}\omega_{k-1}^{\times} I_b \omega_{k-1} - I_b^{-1}\omega_{k-1}^{\times} \sum_{n=1}^{3} h_{w,n_{k-1}} a_n - I_b^{-1} \sum_{n=1}^{3} g_{n_{k-1}} a_n]dt \tag{3.20}$$

then it commands the propagation of the kinematics using `KinematicsAttitudeProvider`, finally, it stores the computed velocity in the attitude stored by the `KinematicsAttitudeProvider`.

3.2.3 Sensors

In the Satellite Simulation, there are three types of sensors: (1) a set of attitude sensors, the set of sun sensors (quite-common on earth-orbiting satellites (SIDI, 2006)); (2) an angular velocity sensor, a gyroscope; and (3) a magnetometer. The sensors, available in the simulator, are ideal and simplified, in the sense that, they can read the physical quantities at any moment with perfect accuracy and no noise. Additionally, the set of sun sensors provides through the entire simulation the same measure of the sun versor $\hat{s}_b$ ($\hat{s}_b = [0.323116\ 0.868285\ 0.376401]^T$) so there is no eclipse, the sun is not moving in the Earth-centered inertial (ECI) reference frame and the sun is always visible by each individual sensor. Indeed, ECI is a quasi-inertial reference frame generally used in AOCS (HUGHES, 1986; SIDI, 2006).

3.2.4 Actuators

Two types of actuators are available in Satellite Simulation, which are: (a) a set of reaction wheels, the actuators for the attitude control; and (b) a set of magnetorques, used for the *unloading* of the angular momentum of the reaction wheels.

3.2.4.1 Reaction Wheels

Since an active control system is required, the simulator uses momentum exchange actuators - they do not change the inertial angular momentum; or, a symmetrical rotating body produces torque when accelerated about its axis of rotation, since such change in the momentum is internal to the satellite, it transfers the momentum change to the satellite with negative sign (*angular momentum is a conservative quantity*) (SIDI, 2006).

The type of the momentum exchange actuator used is reaction wheel, a rotating

machine which is commonly applied for very accurate control and for moderately fast maneuvers since it allows continuous and smooth control with the lowest possible disturbing torques (SIDI, 2006). In particular, reaction wheels are often used in satellites that carry optical payloads, as in the previous discussed typical mission of the INPE. For example, a camera-pointing error creates a signal which increases the speed of the wheel, initially at zero. This torque corrects the satellite and leaves the wheel spinning at low speed, until another pointing error speeds the wheel further or decreases its speed. If the error is cyclic during each orbit, the wheel may not approach saturation speed for several orbits.

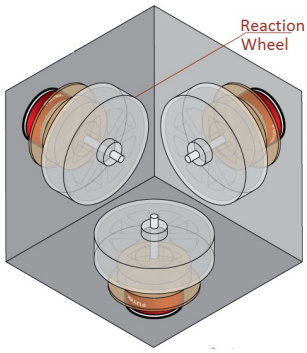


Figure 3.2 - Three reactions wheels mounted in a satellite (adapted from (FUTEK..., 2018)).

Reaction wheels are essentially torque motors with high-inertia rotors. They can spin in either direction and provide one axis of control for each wheel. The basic technical features of a reaction wheel are: maximum achievable torque, maximum momentum capacity (or maximum angular velocity), low torque noise and low coulomb friction (SIDI, 2006). The reaction wheel provided in the simulator models the first two technical features.

3.2.4.2 Magnetorques

Finally, the magnetorques generate magnetic dipole moments M_c whose interact with the earth's magnetic field B producing torques (Equation 3.21).

$$T_c = M_c \times B \tag{3.21}$$

These torques T_c are used, in the Satellite Simulation, for the unloading of the reaction wheel's angular momentum (SIDI, 2006). The basic control equation for

angular momentum unloading of the reaction wheels is Equation 3.22.

$$U_c = -k \sum_{n=1}^{3} h_{w,n} a_n \tag{3.22}$$

in which, k is the unloading control gain. Supposing that the applied M_c happens to be perpendicular to the earth's magnetic field B then M_c is defined by Equation 3.23 (SIDI, 2006).

$$\boldsymbol{M_c} = \frac{1}{B^2}(\boldsymbol{B} \times \boldsymbol{U_c}) \tag{3.23}$$

in which B is the norm of earth's magnetic field vector.

In the Satellite Simulation, magnetorques are available taking into account a maximum magnetic dipole moment, moreover, there is no satellite management to turn off the magnetometer when a magnetorque is turned on and vice-versa during the simulation, the sole option available is the presence or not of such controller.

Using the Distributed Package

The heuristics commonly applied by INPE's engineering staff is: the magnetometer should be turned on 30% of time, whereas the magnetorques should be turned on 70% of the time. It can be implemented in the classes `Magnetometer` and `SetOfMagnetorques`.

---

4 SATELLITE ATTITUDE CONTROL

The satellite attitude control is presented by the following sections.

4.1 Satellite Attitude Control

In a *zero-bias-momentum* system, there are two dynamics states that must be controlled: (1) the attitude (perhaps described by Euler angles θ or unit quaternions Q) and (2) its stability ($\dot{\theta}$, in other words, the angular velocity ω of the satellite). Taking into account Satellite Simulation, the following high-level requirements are: (1) is refined in "the attitude must be stabilized and must follow the sun according to a given sun vector in the satellite" and (2) is refined in "the angular velocity read by the gyroscope must be as close as possible of 0". An additional third (3) requirement is the unloading of the angular momentum of the reaction wheels. These high-level requirements leads to a possible control loop described in Fig.4.1. Note the main

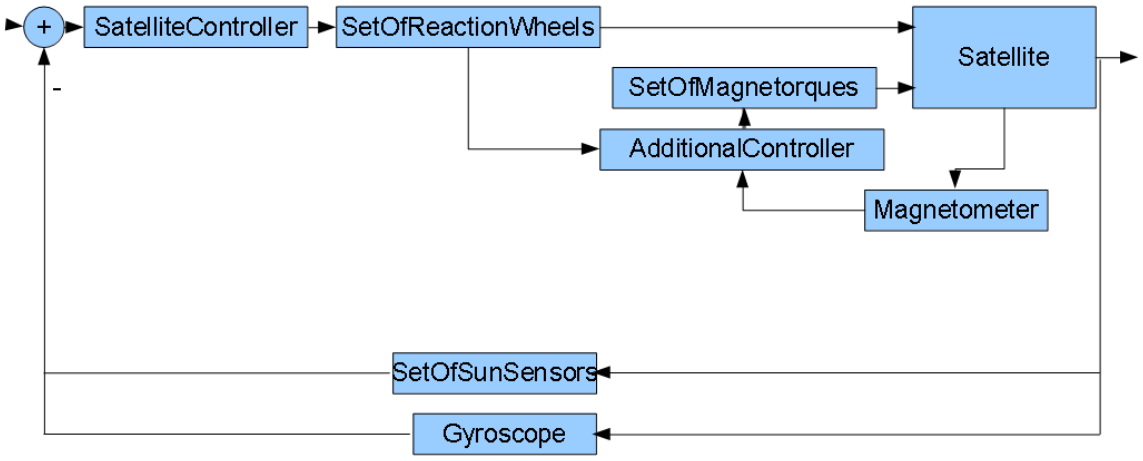


Figure 4.1 - Satellite control.

control loop is exclusively based on the set of reaction wheels as actuators, and an additional control-loop is defined to unload the angular momentum of the reaction wheels. Consequently, the state and the control vectors, for the main control loop,

can be defined by Equation 4.1.

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \theta \\ \omega \end{bmatrix}$$
$$\begin{bmatrix} u_1 \end{bmatrix} = \begin{bmatrix} T_c = \sum_{n=1}^{3} g_n a_n \end{bmatrix} \tag{4.1}$$

The control regulator problem requires that the errors in the attitude and angular velocity must be obtained. The error in the angular velocity is directly obtained from the gyroscope readings, nonetheless, the error in the attitude must be computed.

There are two approaches available in te Satellite Simulation to the computation of the error in the attitude:

- Given two versors, namely (a) the actual sun versor $\hat{s}_b$ in the satellite coordinate frame (constant during the simulation) and (b) the reference versor in the satellite coordinate frame, to compute a rotation (there are many) from the actual sun versor to the reference versor. The computed rotation can be described by an unit quaternion Q.
- Since the previous option can cause undesired discontinuities, the most recommended approach is to use the quaternion error Q_{error}, which is defined as $Q_{error} = Q_{observed}^{-1} * Q_{target}$ (WERTZ; LARSON, 1999). Note Q_{target} is defined in the setup phase of the simulation using the first option once.

Using the Distributed Package

The class `Satellite` has the flag `modeUsingAttitude` to choose the approach for the definition of the error in the attitude.

The following subsections explore the state-space modeling and the controllers' synthesis.

4.2 State-Dependent Riccati Equation (SDRE) Technique

The SDRE technique entails factorization (that is, parametrization) of the nonlinear dynamics into the state vector and the product of a matrix-valued function that depends on the state itself. In doing so, SDRE brings the nonlinear system to a

(nonunique) linear structure having SDC matrices given by Equation 4.2.

$$\begin{aligned} \dot{\vec{x}} &= A(\vec{x})\vec{x} + B(\vec{x})\vec{u} \\ \vec{y} &= C\vec{x}\,, \end{aligned} \tag{4.2}$$

where $\vec{x} \in \mathbb{R}^n$ is the state vector and $\vec{u} \in \mathbb{R}^m$ is the control vector. Notice that the SDC form has the same structure as a linear system, but with the system matrices, A and B, being functions of the state vector. The nonuniqueness of the SDC matrices creates extra degrees of freedom, which can be used to enhance controller performance, however, it poses challenges since not all SDC matrices fulfill the SDRE requirements, e.g., the pair (A,B) must be pointwise stabilizable.

The system model in Equation 4.2 is subject of the cost functional described in Equation 4.3.

$$J(\vec{x_0}, \vec{u}) = \frac{1}{2}\int_0^\infty (\vec{x}^T Q(\vec{x})\vec{x} + \vec{u}^T R(\vec{x})\vec{u})dt\,, \tag{4.3}$$

where $Q(\vec{x}) \in \mathbb{R}^{n\times n}$ and $R(\vec{x}) \in \mathbb{R}^{m\times m}$ are the state-dependent weighting matrices. In order to ensure local stability, $Q(\vec{x})$ is required to be positive semi-definite for all $\vec{x}$ and $R(\vec{x})$ is required to be positive for all $\vec{x}$ (MENON et al., 2002).

The SDRE controller linearizes the plant about the current operating point and creates constant state space matrices so that the LQR method can be used. This process is repeated in all samplings steps, resulting in a pointwise linear model from a non-linear model, so that an ARE is solved and a control law is computed also in each step. Therefore, according to LQR theory and Equation 4.2 and Equation 4.3, the state-feedback control law in each sampling step is $\vec{u} = -K(\vec{x})\vec{x}$ and the state-dependent gain $K(\vec{x})$ is obtained by Equation 4.4 (ÇIMEN, 2010).

$$K(\vec{x}) = R^{-1}(\vec{x})B^T(\vec{x})P(\vec{x})\,, \tag{4.4}$$

where $P(\vec{x})$ is the unique, symmetric, positive-definite solution of the algebraic state-dependent Riccati equation (SDRE) given by Equation 4.5 (ÇIMEN, 2010).

$$P(\vec{x})A(\vec{x}) + A^T(\vec{x})P(\vec{x}) - P(\vec{x})B(\vec{x})R^{-1}(\vec{x})B^T(\vec{x})P(\vec{x}) + Q(\vec{x}) = 0 \tag{4.5}$$

Considering that Equation 4.5 is solved in each sampling step, it is reduced to an ARE. Finally, the conditions for the application of the SDRE technique in a given system model are (ÇIMEN, 2010):

a) $A(\vec{x}) \in C^1(\mathbb{R}^w)$

b) $B(\vec{x}), C(\vec{x}), Q(\vec{x}), R(\vec{x}) \in C^0(\mathbb{R}^w)$

c) $Q(\vec{x})$ is positive semi-definite and $R(\vec{x})$ is positive definite

d) $A(\vec{x})x \implies A(0)0 = 0$, i.e., the origin is an equilibrium point

e) $pair(A, B)$ is pointwise stabilizable (a sufficient test for stabilizability is to check the rank of controllability matrix)

f) $pair(A, Q^{\frac{1}{2}})$ is pointwise detectable (a sufficient test for detectability is to check the rank of observability matrix)

4.2.1 Related Works

A good survey of the SDRE method can be found in (ÇIMEN, 2008) and its systematic application to deal with a nonlinear plant in (ÇIMEN, 2010). The SDRE method was applied by (GONZALES; SOUZA, 2009; STANSBERY; CLOUTIER, 2000; MENON et al., 2002; Di Mauro et al., 2015; Di Mauro et al., 2011) for controlling a nonlinear system similar to the six-degree of freedom satellite model considered in this report. (GONZALES; SOUZA, 2009) defined a simulator using Euler angles based on commercial software, whereas, (Di Mauro et al., 2015) applied quaternions on commercial software. (ROMERO et al., 2018) extended an opensource project and defined an opensource simulator based on Euler angles. Finally, regarding CONASAT, (B. D. Reis de Mesquita and H. Koiti Kuga and V. Carrara, 2017) applied the SDRE as a filter technique together with a PID controller while (ROMERO; SOUZA, 2019d) compared performance and robustness of an SDRE with a PID controller.

The application of SDRE method, and, consequently, the ARE problem that arises, have already been studied in the available literature, e.g., (MENON et al., 2002) investigated the approaches for the ARE solving as well as the resource requirements for such online solving. Recently, (Di Mauro et al., 2015) proposed the usage of differential algebra to reduce the resource requirements for the real-time implementation of SDRE controllers. In fact, the intensive resource requirements for the online ARE solving is the major drawback of SDRE. Nonetheless, the SDRE method has three major advantages: (a) simplicity, (b) numerical tractability and (c) flexibility for the designer, being comparable to the flexibility in the LQR (Di Mauro et al., 2015). Taking into account linear techniques such as LQR to control nonlinear spacecraft systems, (YANG, 2012) argued that a linearized spacecraft model that involves three

components of the quaternion globally stabilizes the nonlinear system, whereas it locally optimizes the spacecraft performance.

4.3 Linear Control

The satellite physical model previous explored is a nonlinear time invariant system. In order to fit such model in a state space model for a linear time invariant (LTI) system described by Equation 4.2, the model must be linearized.

$$\begin{aligned} \dot{\vec{x}} &= A\vec{x} + B\vec{u} \\ \vec{y} &= C\vec{x} \end{aligned} \tag{4.6}$$

4.3.1 Linear Control based on Proportional-Integral-Derivative (PID)

Using control theory, the first alternative for the control of such model, is the usage of a proportional-integral-derivative (PID) controller based on the error of sun versor ($\vec{s}_{be}$) as well as the error of angular velocity ($\vec{\omega}_e$) in the satellite frame. At INPE, a typical controller for the LEOP is shown in Equation 4.7.

$$\begin{aligned} \vec{u_1} &= -(K_p \vec{s_{be}} + K_{pd}\vec{\omega_e} + K_{pi}\int \vec{s_{be}}) \\ \vec{s_{be}} &= [0 \ -\hat{s}_{b3} \ \hat{s}_{b2}]^T \\ \vec{\omega_e} &= \vec{\omega} - \vec{\omega_r} \end{aligned} \tag{4.7}$$

Note this type of controller neither has difficulties in its simulation nor in its numerical real-time implementation.

Using the Distributed Package

The proportional-derivative controler is based on the experience of INPE's engineering and is defined in the class `ProportionalDerivativeLinearSunVectorController`.

`ProportionalDerivativeLinearSunVectorController` uses the current angular velocity and error in the sun vector in the spacecraft to compute the control. It uses two constants `KD` and `KP`, empirically defined.

Finally, the controller algorithm is implemented in the Satellite Simulation by the `ControllerAttitudeModifier`. Firstly, it commands the propagation of the kinetics using `KineticsAttitudeModifier`, then it computes the main and the

additional control torques, afterwards, the actuators are called to actuate (reaction wheels and magnetorques).

4.3.2 Linear Control Based on Linear-Quadratic Regulator (LQR)

A step further in the linear control theory is the usage of LQR, an optimal linear controller, in the sense that the Satellite Simulation operates at minimum cost with such controllers.

4.3.2.1 LQR based on Euler Angles

The assumptions that the angular displacement is small ($\dot{\omega} = 0$ e $\dot{\theta} = \omega$, linearization) and that there are no net external torques ($g_{cm} = 0$) lead to Equation 4.8.

$$\begin{aligned} \begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} &= \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ -I_b^{-1} \end{bmatrix} \begin{bmatrix} u_1 \end{bmatrix} \\ \begin{bmatrix} y \end{bmatrix} &= 1 \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \end{aligned} \tag{4.8}$$

The expansion of the Equation 4.8 generates the Equation 4.9, which defines the constant matrixes A, B e C.

$$\begin{bmatrix} \dot{\theta_1} \\ \dot{\theta_2} \\ \dot{\theta_3} \\ \dot{\omega_1} \\ \dot{\omega_2} \\ \dot{\omega_3} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -0.0032 & 0 & 0 \\ 0 & -0.0028 & 0 \\ 0 & 0 & -0.0019 \end{bmatrix} \begin{bmatrix} u_1 \end{bmatrix} \tag{4.9}$$

$$\begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} = I \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}$$

The constant matrixes A and B are stabilizable, hence, the LQR can be applied.

Using such state space formulation, a linearized proportional controller defined using Linear-Quadratic Regulator (LQR) is available in the Satellite Simulation.

Using the Distributed Package

The proportional optimal linear controler is synthetized in the constructor of the class `ProportionalLinearEulerAnglesLQRController` using the `RiccatiEquationSolver` in Hipparchus.

Finally, the controller algorithm is implemented in the Satellite Simulation by the `ControllerAttitudeModifier`. Firstly, it commands the propagation of the kinetics using `KineticsAttitudeModifier`, then it computes the main and the additional control torques, afterwards, the actuators are called to actuate (reaction wheels and magnetorques).

4.3.2.2 LQR based on full Quaternions

Equation 3.6 and Equation 3.7 can be used to linearize the system around the stationary point ($\omega = 0$ and $Q = [0\,0\,0\,-1]^T$), assuming also that there are no net external torques ($g_{cm} = 0$) lead to Equation 4.10.

$$\begin{bmatrix} x_0 \\ x_2 \end{bmatrix} = \begin{bmatrix} Q \\ \omega \end{bmatrix}$$

$$\begin{bmatrix} \dot{x_0} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} 0 & -\frac{1}{2}I_{3x3} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ -I_b^{-1} \end{bmatrix} \begin{bmatrix} u_1 \end{bmatrix} \tag{4.10}$$

$$\begin{bmatrix} y \end{bmatrix} = 1 \begin{bmatrix} x_0 \\ x_2 \end{bmatrix}$$

The expansion of the Equation 4.10 generates the Equation 4.11, which defines the

constant matrixes A, B e C.

$$\begin{bmatrix} \dot{q_1} \\ \dot{q_2} \\ \dot{q_3} \\ \dot{q_4} \\ \dot{\omega_1} \\ \dot{\omega_2} \\ \dot{\omega_3} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & -\frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -0.0032 & 0 & 0 \\ 0 & -0.0028 & 0 \\ 0 & 0 & -0.0019 \end{bmatrix} \begin{bmatrix} u_1 \end{bmatrix}$$

$$\begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} = I \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} \tag{4.11}$$

However, the constant matrices A and B are not stabilizable, since the linearized model has 7 states but the controllability matrix has 4 as rank. Indeed, (YANG, 2012) shown that **this linearized model with all quaternion components is not stabilizable, meaning that LQR is not applicable.**

Using the Distributed Package

The proportional optimal linear controler is synthetized in the constructor of the class `ProportionalLinearQuaternionFullLQRController` using the `RiccatiEquationSolver`.

Nonetheless, as the pair (A, B) is not stabilizable, such model does not fulfil the requirements for the application of LQR. The `ProportionalLinearQuaternionFullLQRController` is available in order to show the non-controllability result.

4.3.2.3 LQR based on partial Quaternions

Since Equation 3.7 defines a direct method to find q_4, threfore, one option is to model the state of the system without such component of the quaternion (YANG, 2012), which leads to the following equation:

$$\begin{bmatrix} x_3 \\ x_2 \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ \omega \end{bmatrix}$$

$$\begin{bmatrix} \dot{x_3} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} 0 & -\frac{1}{2}I_{3x3} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_3 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ -I_b^{-1} \end{bmatrix} \begin{bmatrix} u_1 \end{bmatrix} \tag{4.12}$$

$$\begin{bmatrix} y \end{bmatrix} = 1 \begin{bmatrix} x_3 \\ x_2 \end{bmatrix}$$

The expansion of the Equation 4.10 generates the Equation 4.13, which defines the constant matrixes A, B e C.

$$\begin{bmatrix} \dot{q_1} \\ \dot{q_2} \\ \dot{q_3} \\ \dot{\omega_1} \\ \dot{\omega_2} \\ \dot{\omega_3} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & -\frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & -\frac{1}{2} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -0.0032 & 0 & 0 \\ 0 & -0.0028 & 0 \\ 0 & 0 & -0.0019 \end{bmatrix} \begin{bmatrix} u_1 \end{bmatrix}$$

$$\begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} = I \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} \tag{4.13}$$

Now, the constant matrices A and B, defined by Equation 4.13, are stabilizable. Using such state space formulation, a linearized proportional controller

defined using LQR is available in the satellite simulator. Note this type of controller neither has difficulties in its simulation nor in its numerical real-time implementation.

Using the Distributed Package

The proportional optimal linear controler is synthetized in the constructor of the class `ProportionalLinearQuaternionPartialLQRController` using the `RiccatiEquationSolver`.

Finally, the controller algorithm is implemented in the Satellite Simulation by the `ControllerAttitudeModifier`. Firstly, it commands the propagation of the kinetics using `KineticsAttitudeModifier`, then it computes the main and the additional control torques, afterwards, the actuators are called to actuate (reaction wheels and magnetorques).

4.4 Nonlinear Control based on SDRE

For small maneuvers a linear controller can be used, however, for large maneuvers the linearized equations do not hold and discontinuities compromise the system (e.g., saturation of the actuators) (SIDI, 2006). In order to avoid linearization, a nonlinear control technique is applied, namely State Dependent Riccati Equation (SDRE).

Assuming that there are no net external torques ($g_{cm} = 0$), kinetics defined by Equation 3.19 can be rearranged as defined by Equation 4.14 using the property $v^{\times}w = -w^{\times}v$.

$$\begin{aligned}
\dot{\omega} &= -I_b^{-1}\omega^{\times}I_b\omega - I_b^{-1}\omega^{\times}\sum_{n=1}^{3} h_{w,n}a_n - I_b^{-1}\sum_{n=1}^{3} g_n a_n \\
&= -I_b^{-1}\omega^{\times}I_b\omega + I_b^{-1}(\sum_{n=1}^{3} h_{w,n}a_n)^{\times}\omega - I_b^{-1}\sum_{n=1}^{3} g_n a_n \\
&= (-I_b^{-1}\omega^{\times}I_b + I_b^{-1}(\sum_{n=1}^{3} h_{w,n}a_n)^{\times})\omega - I_b^{-1}\sum_{n=1}^{3} g_n a_n
\end{aligned} \tag{4.14}$$

4.4.1 Nonlinear Control Based on SDRE and Euler Angles

Taking into account the state and control vectors defined in Equation 4.1, the state space model can be defined using Equations 3.5 and 4.14 in Equation 4.15.

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} 0 & S^{-1} \\ 0 & -I_b^{-1}\omega^{\times} I_b + I_b^{-1}(\sum_{n=1}^{3} h_{w,n} a_n)^{\times} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ -I_b^{-1} \end{bmatrix} \begin{bmatrix} u_1 \end{bmatrix}$$
$$\begin{bmatrix} y \end{bmatrix} = I \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{4.15}$$

Equation 4.15 has been shown to satisfy SDRE conditions. Therefore, the state-dependent gain K can be piecewise computed using Equations 4.4 and 4.5, which in turn is used to compute the control vector $\vec{u_1} = -K(\vec{x})\vec{x}$.

Using the Distributed Package

The proportional nonlinear controler is based on SDRE technique and is defined in the class `ProportionalNonLinearEulerAnglesSDREController` using the `RiccatiEquationSolver` in Hipparchus.

In each step, `ProportionalNonLinearEulerAnglesSDREController` uses the current state to reconstruct the state space model given by Equation 4.15, afterwards, it calls the `RiccatiEquationSolver` to compute the K and, eventually, it uses the K to compute the control torque.

Finally, the controller algorithm is implemented in the Satellite Simulation by the `ControllerAttitudeModifier`. Firstly, it commands the propagation of the kinetics using `KineticsAttitudeModifier`, then it computes the main and the additional control torques, afterwards, the actuators are called to actuate (reaction wheels and magnetorques).

4.4.2 Nonlinear Control Based on SDRE and Quaternions

Taking into account the state and control vectors defined in Equation 4.1, the state space model can be defined using Equation 3.6 (Ω) and Equation 4.14 in Equation 4.16.

$$\begin{bmatrix} x_0 \\ x_2 \end{bmatrix} = \begin{bmatrix} Q \\ \omega \end{bmatrix}$$

$$\begin{bmatrix} \dot{x_0} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\Omega & 0 \\ 0 & -I_b^{-1}\omega^{\times} I_b + I_b^{-1}(\sum_{n=1}^{3} h_{w,n} a_n)^{\times} \end{bmatrix} \begin{bmatrix} x_0 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ -I_b^{-1} \end{bmatrix} \begin{bmatrix} u_1 \end{bmatrix} \tag{4.16}$$

$$\begin{bmatrix} y \end{bmatrix} = I \begin{bmatrix} x_0 \\ x_2 \end{bmatrix}$$

However, the SDC matrices (in Equation 4.16) do not fulfill the SDRE requirements, in particular, the pair (A,B) is not pointwise stabilizable. Another option for the definition of the SDC matrices is to use Equation 3.6 based on Ξ, which leads to Equation 4.17.

$$\begin{bmatrix} x_0 \\ x_2 \end{bmatrix} = \begin{bmatrix} Q \\ \omega \end{bmatrix}$$

$$\begin{bmatrix} \dot{x_0} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{2}\Xi \\ 0 & -I_b^{-1}\omega^{\times} I_b + I_b^{-1}(\sum_{n=1}^{3} h_{w,n} a_n)^{\times} \end{bmatrix} \begin{bmatrix} x_0 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ -I_b^{-1} \end{bmatrix} \begin{bmatrix} u_1 \end{bmatrix} \tag{4.17}$$

$$\begin{bmatrix} y \end{bmatrix} = I \begin{bmatrix} x_0 \\ x_2 \end{bmatrix}$$

Nonetheless, the SDC matrices (in Equation 4.17) do not fulfill the SDRE requirements, in particular, the pair (A,B) is not pointwise stabilizable.

GPL 3.0 GNU Using the Distributed Package

The proportional nonlinear controler is based on SDRE technique and is defined in the class `ProportionalLinearQuaternionFullLQRController` using the `RiccatiEquationSolver`.

Nonetheless, as the pair (A, B) is not pointwise stabilizable, such model does not fulfil the requirements for the application of LQR. The `ProportionalLinearQuaternionFullLQRController` is available in order to show the non-controllability result. The kinematic definition passed to the constructor should be `OMEGA` (in Equation 4.16) or `XI` (in Equation 4.17).

4.4.3 Nonlinear Control Based on SDRE and Gibbs Vector

An alternative option for the definition of the SDC matrices is to use Equation 3.8, which leads to Equation 4.18.

$$\begin{bmatrix} x_0 \\ x_2 \end{bmatrix} = \begin{bmatrix} Q \\ \omega \end{bmatrix}$$

$$\begin{bmatrix} \dot{x_0} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} -\frac{1}{2}\begin{bmatrix} \omega^{\times} \\ \omega^T \end{bmatrix} & 0 & \begin{bmatrix} \frac{1}{2}q_4 I_{3\times 3} \\ 0 \end{bmatrix} \\ 0 & 0 & -I_b^{-1}\omega^{\times} I_b + I_b^{-1}(\sum_{n=1}^{3} h_{w,n} a_n)^{\times} \end{bmatrix} \begin{bmatrix} x_0 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ -I_b^{-1} \end{bmatrix} \begin{bmatrix} u_1 \end{bmatrix}$$

$$\begin{bmatrix} y \end{bmatrix} = 1 \begin{bmatrix} x_0 \\ x_2 \end{bmatrix} \tag{4.18}$$

Equation 4.18 has been shown to satisfy SDRE conditions.

Equation 4.18 can be factored to produce infinity parametrizations. In particular, the kinematics part can be factored out as follows (using that g and ω are vectors so $\omega^T g = g^T \omega$).

$$\begin{aligned} \dot{Q} = \begin{bmatrix} -\frac{1}{2}\omega^{\times} g + \frac{1}{2}q_4 \omega \\ -\frac{1}{2}\omega^T g \end{bmatrix} &= \begin{bmatrix} \frac{1}{2}g^{\times}\omega + \frac{1}{2}\omega q_4 \\ -\frac{1}{2}g^T \omega \end{bmatrix} \\ A_{1_{x_0}}(x_0) &= \begin{bmatrix} -\frac{1}{2}\omega^{\times} & 0 & \frac{1}{2}q_4 I_{3\times 3} \\ -\frac{1}{2}\omega^T & 0 & 0 \end{bmatrix} \\ A_{2_{x_0}}(x_0) &= \begin{bmatrix} 0 & \frac{1}{2}\omega & \frac{1}{2}g^{\times} \\ 0 & 0 & -\frac{1}{2}g^T \end{bmatrix} \end{aligned} \tag{4.19}$$

Now using these two parametrizations can be combined as follows, as α ($0 \leq \alpha \leq 1$) emerges as an additional degree of freedom available providing design flexibility.

$$\begin{aligned} A(x,\alpha) &= \alpha A_1(x) + (1-\alpha)A_2(x) \\ A &= \alpha \begin{bmatrix} -\frac{1}{2}\omega^{\times} & 0 & \frac{1}{2}q_4 I_{3\times 3} \\ -\frac{1}{2}\omega^T & 0 & 0 \end{bmatrix} + (1-\alpha) \begin{bmatrix} 0 & \frac{1}{2}\omega I_{3\times 3} & \frac{1}{2}g^{\times} \\ 0 & 0 & -\frac{1}{2}g^T \end{bmatrix} \end{aligned} \tag{4.20}$$

Using the Distributed Package

The proportional nonlinear controler is based on SDRE technique and is defined in the class `ProportionalNonLinearQuaternionFullSDREController` using the `RiccatiEquationSolver` in Hipparchus. The kinematic definition passed to the constructor should be `GIBBS` (in Equation 4.18), `GIBBS_SECOND` (in Equation 4.19) or `ALPHA` (in Equation 4.20).

In each step, `ProportionalNonLinearQuaternionFullSDREController` uses the current state to reconstruct the state space model given by corresponding equation, afterwards, it calls the `RiccatiEquationSolver` to compute the K and, eventually, it uses the K to compute the control torque.

Finally, the controller algorithm is implemented in the Satellite Simulation by the `ControllerAttitudeModifier`. Firstly, it commands the propagation of the kinetics using `KineticsAttitudeModifier`, then it computes the main and the additional control torques, afterwards, the actuators are called to actuate (reaction wheels and magnetorques).

4.4.4 Nonlinear Control Based on SDRE and Modified Rodrigues Parameters

Another alternative for the definition of the state-space model is to use Equation 3.9 (MRPs), which leads to Equation 4.21.

$$\begin{bmatrix} x_3 \\ x_2 \end{bmatrix} = \begin{bmatrix} p \\ \omega \end{bmatrix}$$

$$\begin{bmatrix} \dot{x_3} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} -\frac{1}{2}\omega^{\times} + \frac{1}{2}(\omega^T p) I_{3\times 3} & \frac{1}{4}(1 - p^T p) I_{3\times 3} \\ 0 & -I_b^{-1}\omega^{\times} I_b + I_b^{-1}(\sum_{n=1}^{3} h_{w,n} a_n)^{\times} \end{bmatrix} \begin{bmatrix} x_3 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ -I_b^{-1} \end{bmatrix} \begin{bmatrix} u_1 \end{bmatrix}$$

$$\begin{bmatrix} y \end{bmatrix} = I \begin{bmatrix} x_3 \\ x_2 \end{bmatrix} \tag{4.21}$$

Equation 4.21 has been shown to satisfy SDRE conditions.

Equation 4.21 can be factored to produce infinity parametrizations. In particular, the kinematics part can be factored out as follows .

$$\dot{p} = -[\frac{1}{2}\omega^{\times} + \frac{1}{2}(\omega^T p)I_{3\times3}]p + [\frac{1}{4}(1 - p^T p)I_{3\times3}]\omega = \frac{1}{2}[\frac{1}{2}(1 - p^T p)I_{3\times3} + p^{\times} + (pp^T)I_{3\times3}]\omega$$

$$\begin{aligned} A_{1_{x_3}}(x_3) &= \begin{bmatrix} -\frac{1}{2}\omega^{\times} + \frac{1}{2}(\omega^T p)I_{3\times3} & \frac{1}{4}(1 - p^T p)I_{3\times3} \end{bmatrix} \\ A_{2_{x_3}}(x_3) &= \begin{bmatrix} 0 & \frac{1}{4}(1 - p^T p)I_{3\times3} + \frac{1}{2}p^{\times} + \frac{1}{2}(pp^T)I_{3\times3} \end{bmatrix} \end{aligned} \tag{4.22}$$

Now using these two parametrizations can be combined as follows, as α $(0 \leq \alpha \leq 1)$ emerges as an additional degree of freedom available providing design flexibility.

$$\begin{aligned} A(x, \alpha) &= \alpha A_1(x) + (1 - \alpha)A_2(x) \\ A &= \alpha \begin{bmatrix} -\frac{1}{2}\omega^{\times} + \frac{1}{2}(\omega^T p)I_{3\times3} & \frac{1}{4}(1 - p^T p)I_{3\times3} \end{bmatrix} + (1 - \alpha) \begin{bmatrix} 0 & \frac{1}{4}(1 - p^T p)I_{3\times3} + \frac{1}{2}p^{\times} + \frac{1}{2}(pp^T)I_{3\times3} \end{bmatrix} \end{aligned} \tag{4.23}$$

Using the Distributed Package

The proportional nonlinear controler is based on SDRE technique and is defined in the class `ProportionalNonLinearMRPSDREController` using the `RiccatiEquationSolver` in Hipparchus. The kinematic definition passed to the constructor should be `FIRST` (in Equation 4.21), `SECOND` (in Equation 4.22) or `ALPHA` (in Equation 4.23).

In each step, `ProportionalNonLinearMRPSDREController` uses the current state to reconstruct the state space model given by corresponding equation, afterwards, it calls the `RiccatiEquationSolver` to compute the K and, eventually, it uses the K to compute the control torque.

Finally, the controller algorithm is implemented in the Satellite Simulation by the `ControllerAttitudeModifier`. Firstly, it commands the propagation of the kinetics using `KineticsAttitudeModifier`, then it computes the main and the additional control torques, afterwards, the actuators are called to actuate (reaction wheels and magnetorques).

---

5 RESULTS

This chapter presents the results obtained from Satellite Simulation that supports the list of publications available in Appendix C. The Satellite Simulation has capabilities: (1) **to simulate** a selected controller regarding initial conditions and satellite characteristics, (2) **to compare** up to three controllers regarding initial conditions subject to a Monte Carlo perturbation model and satellite characteristics, (3) **to tune** one controllers regarding initial conditions, satellite characteristics and modifying alpha subject to a Monte Carlo perturbation model and (4) **to assess stability** one controller regarding initial conditions and satellite characteristics subject to a Monte Carlo perturbation model.

5.1 Simulating Controllers

The discipline CMC-202-4, *Motion of a Rigig Body*, at INPE used the Satellite Simulation to calculate the following data.

Using the Distributed Package

The below data is produced running

```
java -Dorekit.data.path=./orekit-data -Xms1G -Xmx2G -jar lib\satellitesimulator-0.1
```

The satellite characteristics, references and initial conditions can be changed through the modification of the `simulationController.properties`.

Indeed, to reproduce the following data please use the content of the file `simulationcontroller_201706_CMC202.properties`, which points to the `amazonia1.properties`.

The main features of this setup are the usage of: (a) Amazonia-1 characteristics, (b) the controller `ProportionalLinearEulerAnglesLQRController` and (c) the additional controller `SetOfMagnetorquersController`.

5.1.1 Orbital Dynamics

Table 5.1 shows the orbital parameters and initial conditions.

Using simulation time *20000 s (approximately 3 orbits) and fixed step* 0.001, the following graphs are generated by the Satellite Simulation. Namely, True Anomaly - (Fig. 5.1), Positon Earth Centered Inertial (ECI) - (Fig. 5.2(a)), Positon Earth Centered Earth Fixed (ECEF) - (Fig. 5.2(b)) and Positon Latitude Longitude Altitude (LLA) - (Fig. 5.3(a)).

Table 5.1 - Orbit parameters and initial conditions.

Name	Value
Parameters	
semi-major axis (m)	7130.092.000
eccentricity	0.001111
inclination ($degrees$)	98.405
perigee argument ($degrees$)	98.405
right ascension of ascending node ($degrees$)	227.088
Initial conditions	
mean anomaly ($degrees$)	305
date (UTC - YYYY/MM/DD HH:MM:SS)	2017/06/01 11:00:00

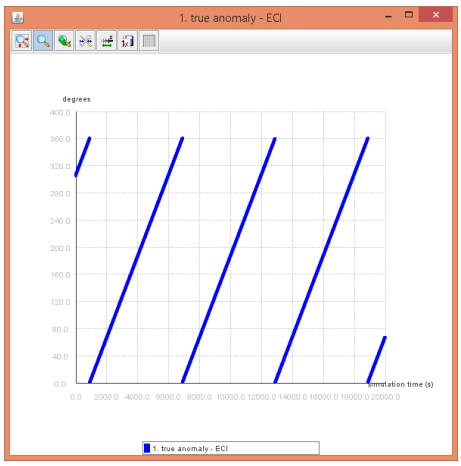


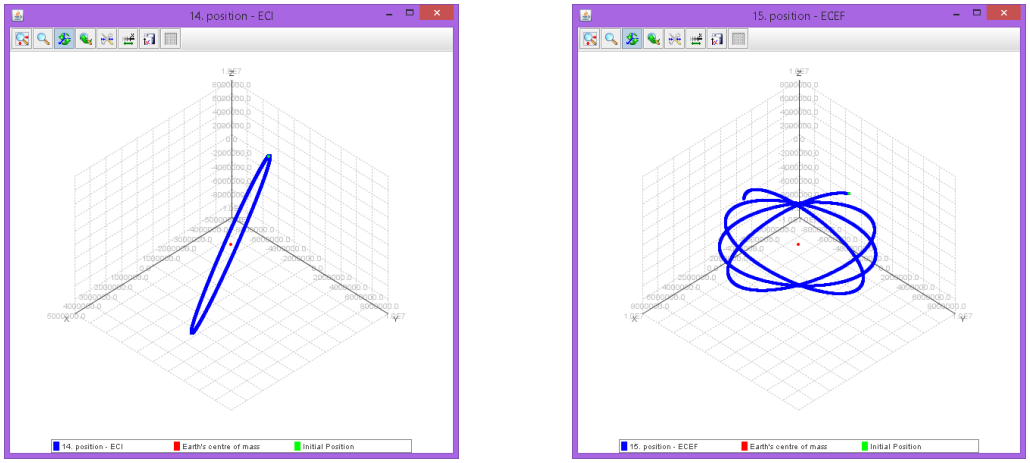
Figure 5.1 - True anomaly.

5.1.2 Satellite Attitude Dynamics

Table 5.2 shows the satellite characteristics and initial conditions.

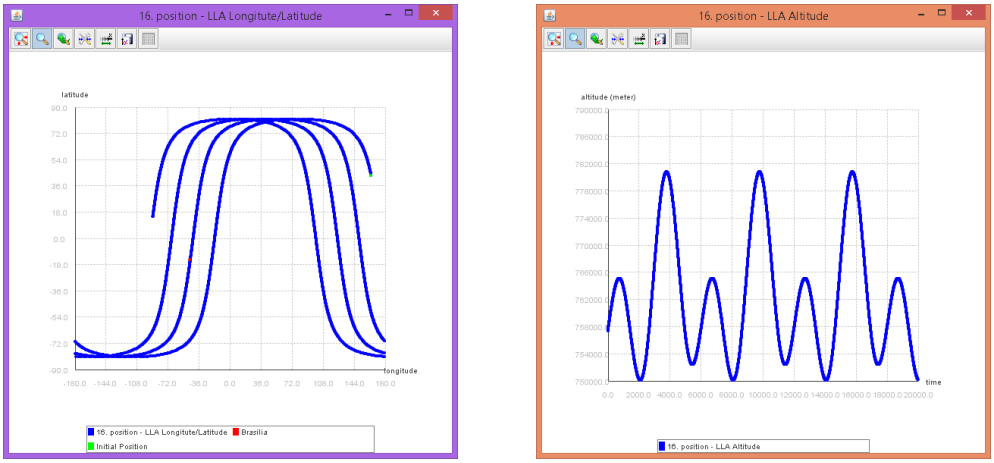
Using simulation time *20000 s, fixed step* 0.001 and *controlled by the ProportionalLinearEulerAnglesLQRController*, the following graphs are generated by the Satellite Simulation. Namely, Angular velocity of the satellite in respect to the ECI - (Fig. 5.4(a)), Solar vector in the satellite - (Fig. 5.4(b)), Euler angles in the ECI - (Fig. 5.5(a)), Euler angles in the ECEF - (Fig. 5.5(b)) and Rotational kinetic energy - (Fig. 5.6).

Moreover, the attitude is visualized through an animation. Fig. 5.7(a) shows an intermediary frame of the visualization while Fig. 5.7(a) shows the final frame of visualization.



(a) Position in ECI (b) Position in ECEF

Figure 5.2 - Position in ECI (a) and position in ECEF (b).

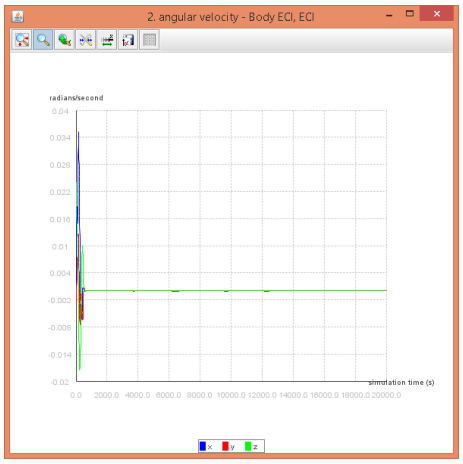


(a) Position in LLA - Longitude/Latitude (b) Position in LLA - Altitude

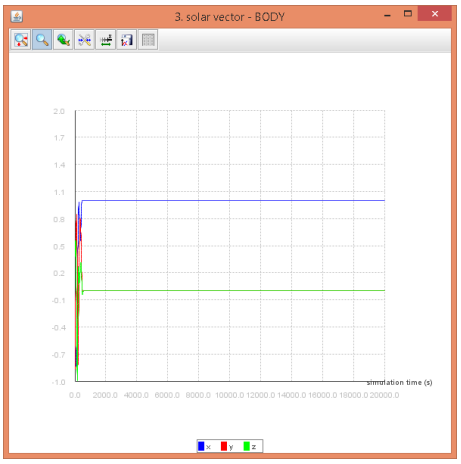
Figure 5.3 - Position in LLA.

Table 5.2 - Satellite characteristics and initial conditions of Amazonia-1.

Name	Value
Characteristics	
inertia tensor ($kg.m^2$)	$\begin{bmatrix} 310.0 & 1.11 & 1.01 \\ 1.11 & 360.0 & -0.35 \\ 1.01 & -0.35 & 530.7 \end{bmatrix}$
Initial conditions	
attitude ($degrees$, XYZ)	$\begin{bmatrix} 0 & 0 & 180 \end{bmatrix}^T$
angular velocity ($radians/second$, XYZ)	$\begin{bmatrix} 0 & 0 & 0.024 \end{bmatrix}^T$

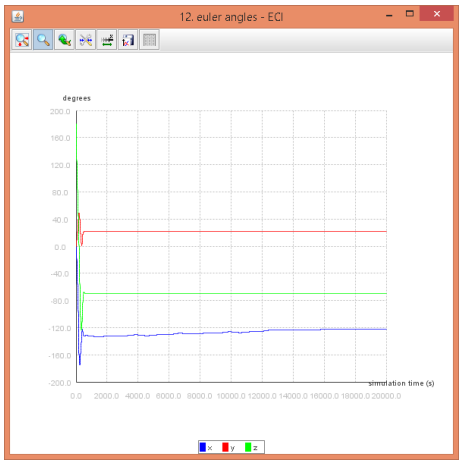


(a) Angular velocity.

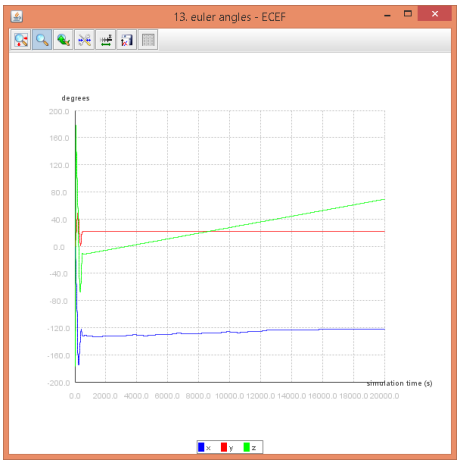


(b) Solar vector in the satellite.

Figure 5.4 - Angular velocity (a) and solar vector in the satellite (b).



(a) Euler angles in ECI.



(b) Euler angles in ECEF.

Figure 5.5 - Euler angles in ECI (a) and Euler angles in ECEF (b).

Using the Distributed Package

In order to analyze the attitude of the satellite without a controller (it can be helpful in some situations), the properties of the file `simulationController.properties` must be changed according to the following lines:

```
simulation.reactionWheelControllerName=NopeController
#simulation.magnetorquersController=
```

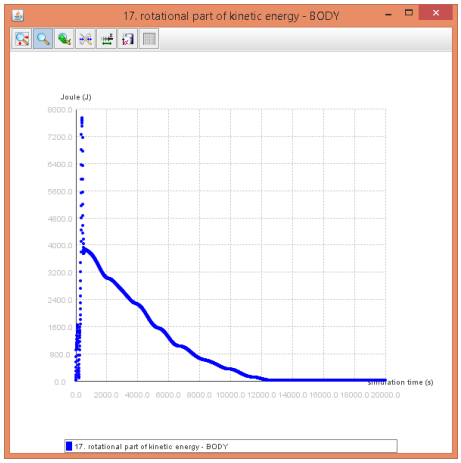
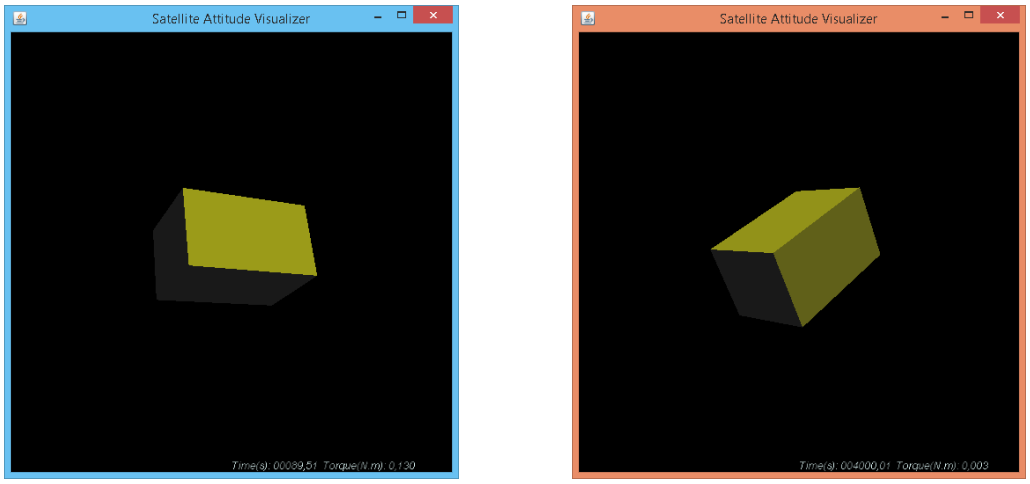


Figure 5.6 - Kinetic energy.



(a) Satellite attitude at a given time t_1.

(b) Satellite attitude at a given time t_2.

Figure 5.7 - Satellite attitude at two different times - (a) - an intermediary time and (b) at the end of simulation).

5.1.3 Satellite Attitude Control

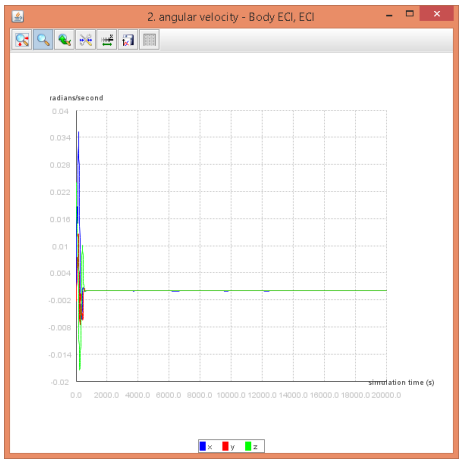
Recall the satellite has 3, ideal and simplified, sensors: a set of sun sensors, a gyroscope and a magnetometer. Moreover, the satellite has 2 set of actuators: 3 magnetorques (each one aligned with the principal axes of inertia) and 3 reaction wheels (each one aligned with the principal axes of inertia). Table 5.3 shows the actuators characteristics taken into account in this work as well as the references for the controller.

Using simulation time *20000 s, fixed step* 0.001 and *controlled by the Proportion-*

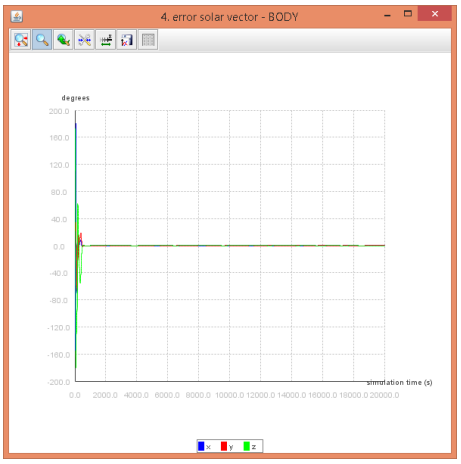
Table 5.3 - Actuators characteristics of Amazonia-1 and references for the controller.

Name	Value
Actuators Characteristics - Reaction Wheels	
inertia ($kg.m^2$)	0.01911
inertia tensor of 3 reaction wheels ($kg.m^2$)	$\begin{bmatrix} 0.01911 & 0 & 0 \\ 0 & 0.01911 & 0 \\ 0 & 0 & 0.01911 \end{bmatrix}$
maximum torque ($N.m$)	0.075
maximum angular velocity (RPM)	6000
Actuators Characteristics - Magnetorques	
Maximum magnetic dipole ($A.m^2$)	30
Norm of earth's magnetic field (nT) in the orbit, B	0.948
References for the controller	
solar vector in the body (XYZ)	$\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T$
angular velocity ($radians/second$, XYZ)	$\begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$

alLinearEulerAnglesLQRController, the following graphs are generated by the Satellite Simulation.



(a) Angular velocity.



(b) Error of the solar vector in the satellite.

Figure 5.8 - Angular velocity (a) and error of the solar vector in the satellite (b).

Using the Distributed Package

It is possible to analyze the attitude of the satellite turning off actuators, e.g., the magnetorque.

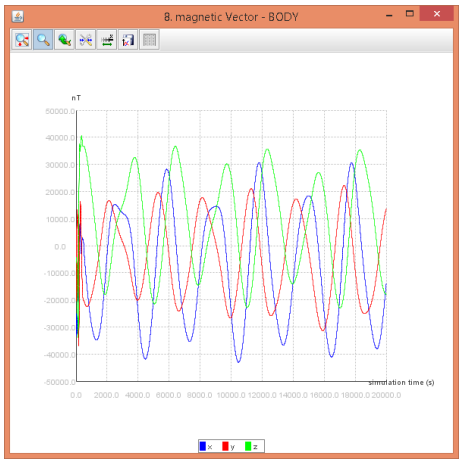
In order to turn off all actuators (any variation is allowed, at least, from the code perspective), the `<<satellite.properties>>` must be changed according to the following lines:

```
reactionWheel=false
magnetorque=false
```

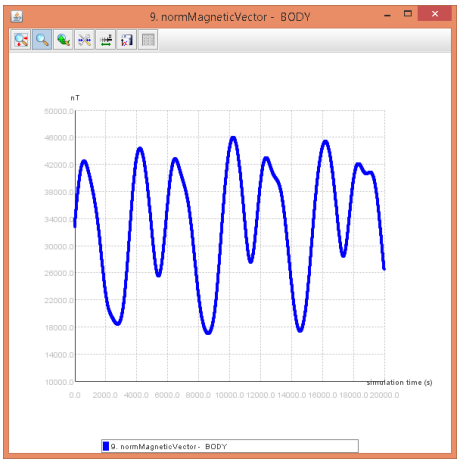
Taking into account the sensors available in the satellite, the ideal gyroscope provides the measurement of the angular velocity of the satellite Fig 5.8(a), whereas based on the reading of the ideal and simplified sun sensors the error is computed (comparing with the reference $\begin{bmatrix}1 & 0 & 0\end{bmatrix}^T$ - see Fig. 5.8(b)).

Furthermore, the magnetometer reads the earth's magnetic field in the satellite. Fig. 5.9(a) shows the field vcetor of the earth's magnetic field and Fig. 5.9(b) shows the computed norm of such vector. The reading of magnetometer should not be used while the magnetorques are turned on, nevertheless, once more a simplification is applied.

Fig. 5.8(a) and Fig. 5.8(b) show that for the given simulation, the satellite is stabilized in the references (angular velocity and error of the solar vector) about 910

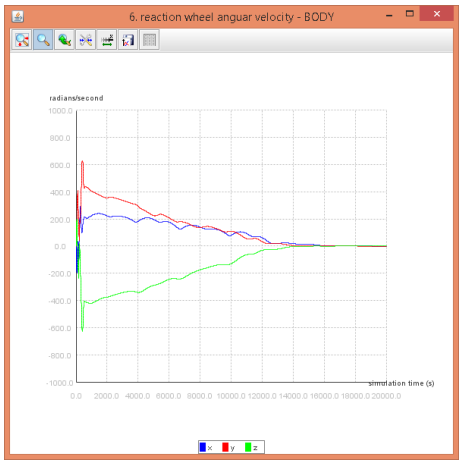


(a) Earth's magnetic field vector.

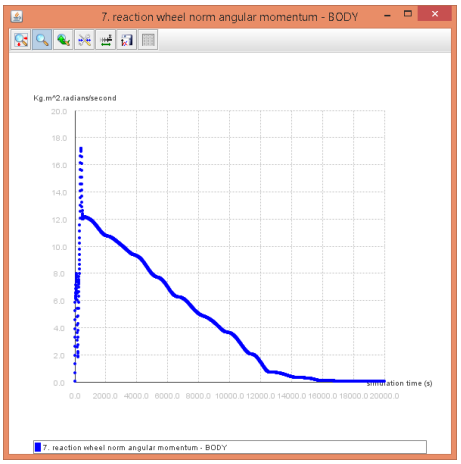


(b) Norm of the earth's magnetic field vector.

Figure 5.9 - The readings of the magnetomer.



(a) Reaction wheels angular velocity.



(b) Norm of reaction wheels angular momentum.

Figure 5.10 - Reaction wheels angular velocity (a) and the norm of reaction wheels angular momentum (b).

seconds.

Focusing on the actuators, the state of the reaction wheels during the simulation is shown in Fig. 5.10(a) and Fig. 5.10(b), besides, the control torque exerted by the reaction wheels in the satellite is shown in Fig. 5.11.

As expected the control torque are high at the beginning of the simulation, see Fig. 5.11. In this period of the simulation, the constraint of the reaction wheels

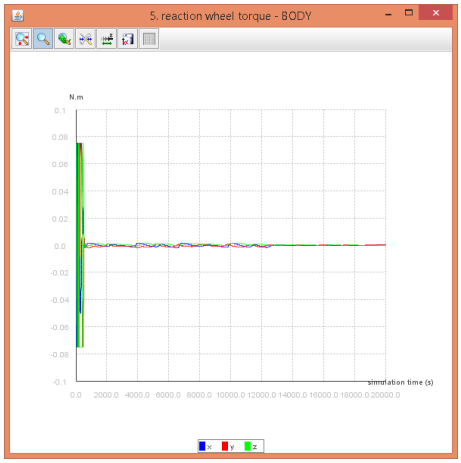
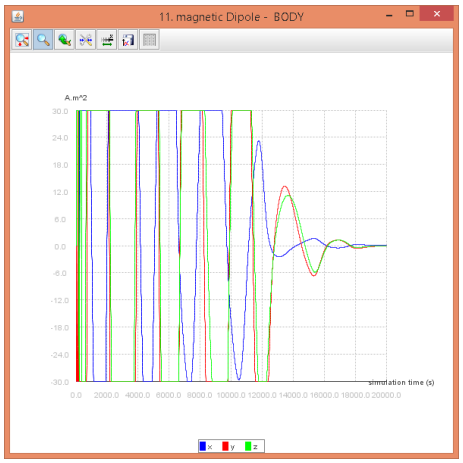
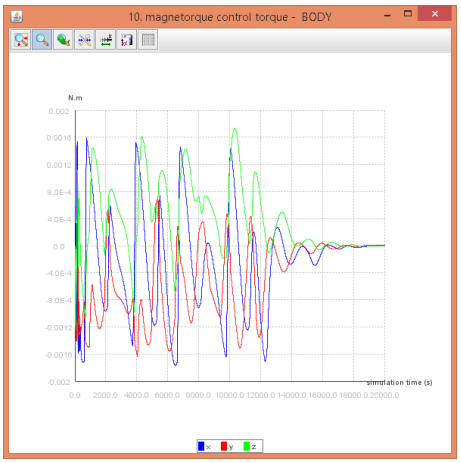


Figure 5.11 - Reaction wheel control torque.



(a) Magnetic dipole generated by the magnetorques.



(b) Magnetorques' control torque.

Figure 5.12 - The state of the magnetorques (a) and the magnetorque control torque (b).

maximum torque defines the amount of torque exerted on the satellite by each reaction wheel.

The additional control requirement of the removal any angular momentum of the reaction wheels is achieved about 20000 seconds by the smooth actuation of the magnetorques. The magnetic dipole generated by each magnetorque is shown in Fig. 5.12(a) and the control torque generated by such magnetic dipoles interacting with the earth's magnetic field are shown in Fig. 5.12(b).

The magnetic dipole generated by the magnetorque respects the constraint imposed

$(30A.m^2)$, moreover, the control torque generated by such magnetic dipoles interacting with the earth's magnetic field are in accordance with the level found in the literature, in the range of 0.001-0.01N.m ((SIDI, 2006); pg. 161).

5.1.4 Orbital and Satellite Attitude Dynamics

The simulation time of 20000s is selected since it is the sufficient time for the satisfaction of control requirements, however, for the analisys of the selected orbit it is only 3.33 of an orbit. Therefore, the result of another simulation is shown. Using simulation time *130000 s (approximately 36 hours or 21 orbits) and fixed step* 0.1, the following graphs are generated by the Satellite Simulation. Namely, True Anomaly - (Fig. 5.13), Positon Earth Centered Inertial (ECI) - (Fig. 5.14(a)), Positon Earth Centered Earth Fixed (ECEF) - (Fig. 5.14(b)) and Positon Latitude Longitude Altitude (LLA) - (Fig. 5.15).

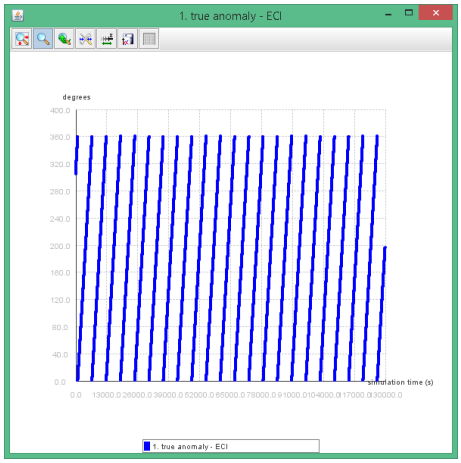
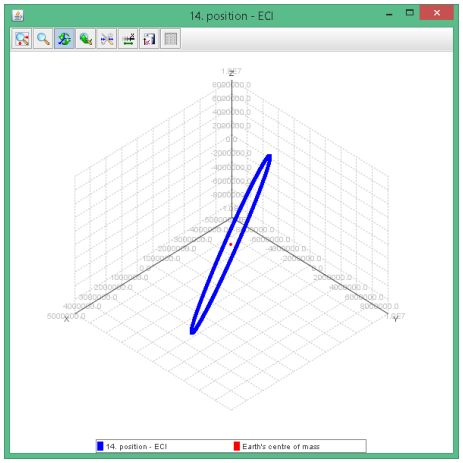


Figure 5.13 - True anomaly.

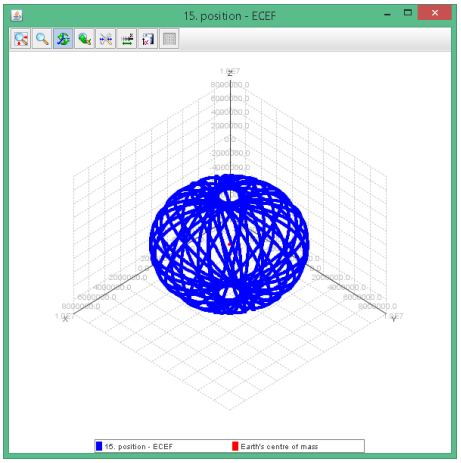
An interesting aspect of the interaction of the orbit dynamics and the satellite attitude is the behavior of the euler angles in the ECI and ECEF. Fig. 5.16(a) shows the euler angles in the ECI during 21 orbits, as expected, there is no change in the angles. Nonetheless, as ECEF is turning around the Z axis of the ECI, the euler angles in the ECEF shows that Z is changing (see Fig. 5.16(b)). Indeed, for an observer in the earth, the satellite is slowly turning around its Z axis.

Using the Distributed Package

In order to analyze the attitude of the satellite through a longer period of time, the

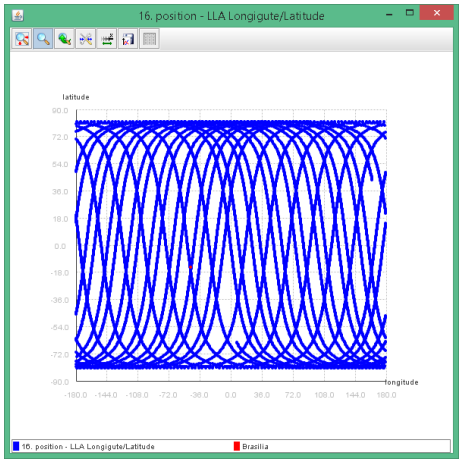


(a) Position in ECI

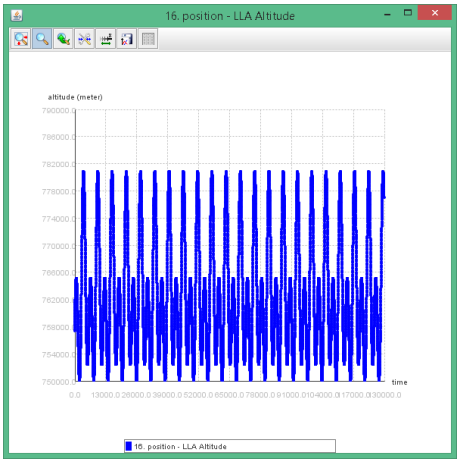


(b) Position in ECEF.

Figure 5.14 - Position in ECI (a) and position in ECEF - simulation time 130000s.

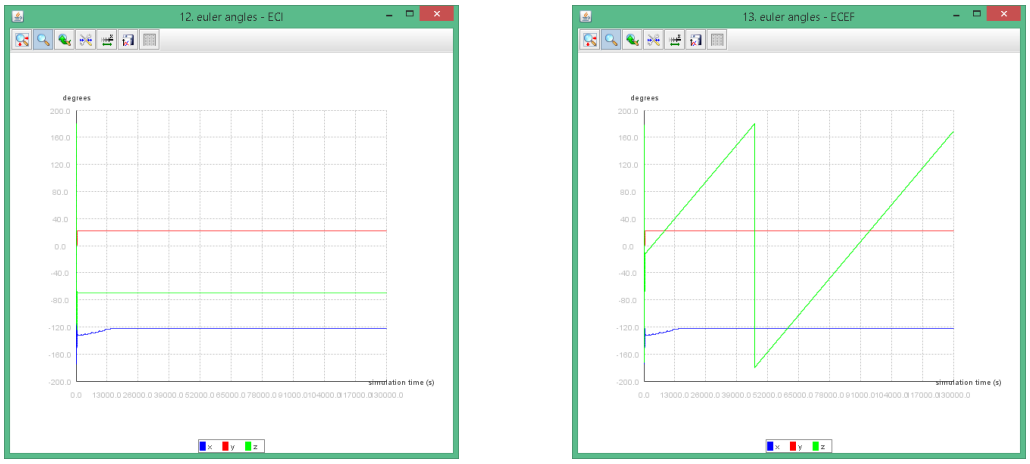


(a) Position Longitude/Latitude.



(b) Position Altitude.

Figure 5.15 - Position in LLA - simulation time 130000s.



(a) Euler angles in the ECI. (b) Euler angles in the ECEF

Figure 5.16 - Euler angles in the ECI (a) and ECEF (b) - simulation time 130000s.

properties of the file `simulationController.properties` must be changed according to the following lines:

```
simulation.time=130000d
simulation.step=.1d=
simulation.intervalToStore=100
```

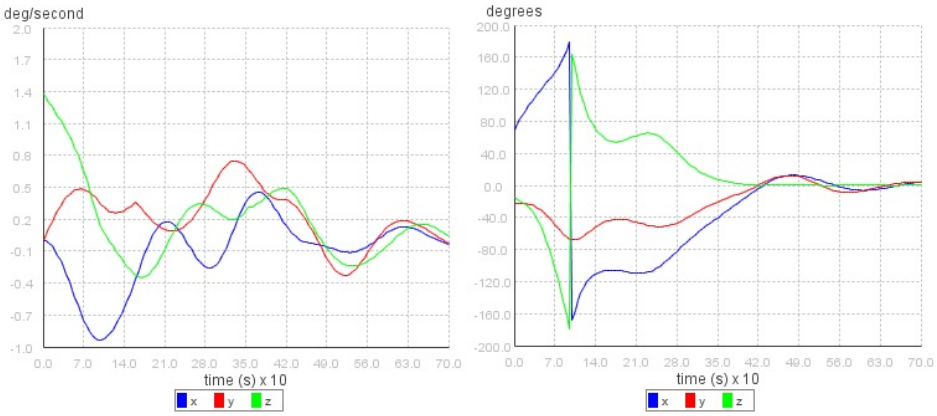
5.2 Comparing Controllers

5.2.1 Comparing PID and SDRE through Simulation

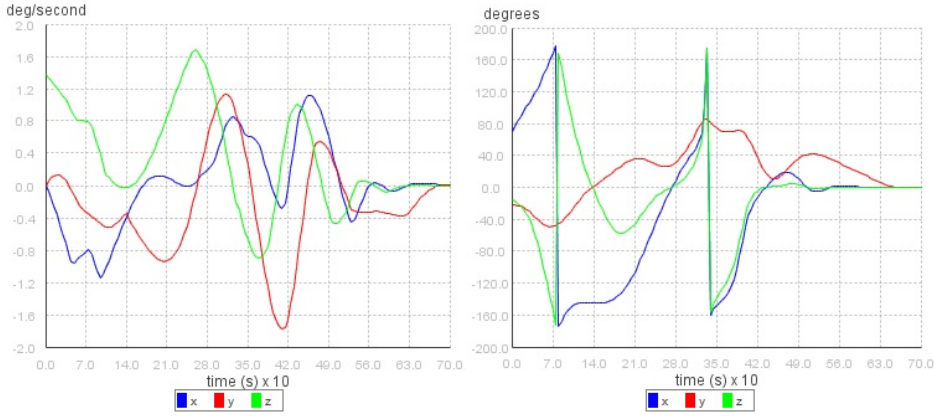
(ROMERO; SOUZA, 2019c) compared the performance of the linear PID (Equation 4.7) and nonlinear SDRE based on Euler Angles (Equation 4.15) using Satellite Simulation.

Fig. 5.17 shows that the SDRE controller has a better performance then a PID controller ($K_p = 0.4$, $K_{pd} = diag(-4, -4, -4)$, $K_{pi} = 0$), saving ate least 30 seconds for the stabilization, at cost of more control effort (visible in the angular velocity of the satellite).

The simulation results are in accordance with the literature (ÇIMEN, 2008; ÇIMEN, 2010; CLOUTIER et al., 1996; GONZALES; SOUZA, 2009) since a SDRE controller has better performance in large maneuvers than the linear controllers. Nonetheless, it is worth to highlight that the performance of both controllers can be better since tunning can be performed in the PID controller, e.g., synthesizing an optimal linear



(a) Results from PID Controller (angular velocity and error of the solar vector).



(b) Results from SDRE Controller (angular velocity and error of the solar vector).

Figure 5.17 - Comparison between controllers performance.

controller using linear-quadratic regulator (LQR) method, as well as in the SDRE controller by choosing better matrices for R and Q or making them function of the state (state-dependent weighting).

Using the Distributed Package

The above data is produced running

```
java -Dorekit.data.path=./orekit-data -Xms1G -Xmx2G -jar lib\satellitesimulator-0.1
```

The satellite characteristics, references and initial conditions can be changed through the modification of the `simulationController.properties`.

Indeed, to reproduce the following data please use the content of the file `simulationcontroller_201706_CMC202.properties`, which points to the `amazonia1.properties`.

The main features of this setup are the usage of: (a) Amazonia-1 characteristics, (b) the controllers `ProportionalDerivativeLinearSunVectorController` and

`ProportionalNonLinearEulerAnglesSDREController`; and (c) no additional controller.

5.2.2 Comparing PID, LQR and SDRE through Monte Carlo

(ROMERO et al., 2018) compared the performance of the linear PID (Equation 4.7), linear LQR based on partial quaternions (Equation 4.13) and nonlinear SDRE based on Gibbs Vector (Equation 4.18) using Satellite Simulation.

In order to compare the performance of the controllers, a simulation test was conducted with the full Monte Carlo perturbation model described as follows: (1) the initial Euler angle errors of the nonlinear spacecraft system are randomly selected using independent normal distributions ($\mu = 0$, $\sigma = 1$) multiplied by 90°; and (2) the initial angular velocity errors are randomly selected using independent normal distributions ($\mu = 0$, $\sigma = 1$) multiplied by 0.01 rad/s.

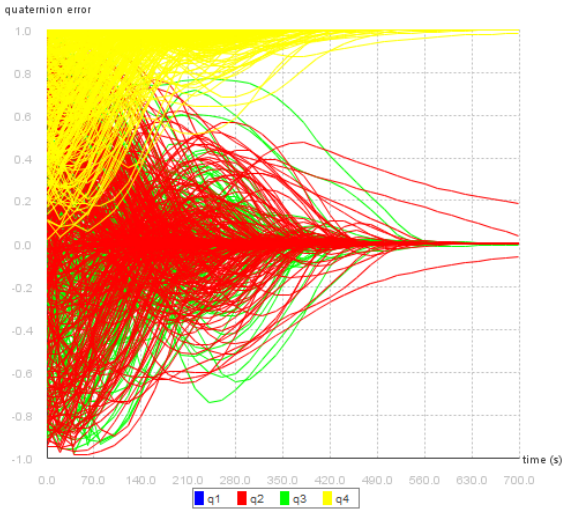
The Monte Carlo model ran 300 times and in each time one simulation of the three controllers was executed. Such executions used simulation time 700 seconds, fixed step 0.01 seconds, the data presented in Table 5.2, Table 5.3 and the following controllers: (1) PID controller ($K_p = 1$, $K_{pd} = diag(-24, -26, -32)$, $K_{pi} = 0$) defined by Equation 4.7; (2) LQR controller ($R = 1$ and $Q = 1$) defined by Equation 4.13; and (3) SDRE controller ($R = 1$ and $Q = 1$) defined by Equation 4.18.

Each graph in Fig. 5.18 shows the respective collection of all quaternion errors computed during simulations for a given controller. The PID controller is not able to control all runs in the specified simulation time. In addition to the lack of control of all runs, the LQR controller does not exhibit asymptotically stability. The SDRE controller has the best performance since is exhibits asymptotically stability for all runs in at most 630 seconds.

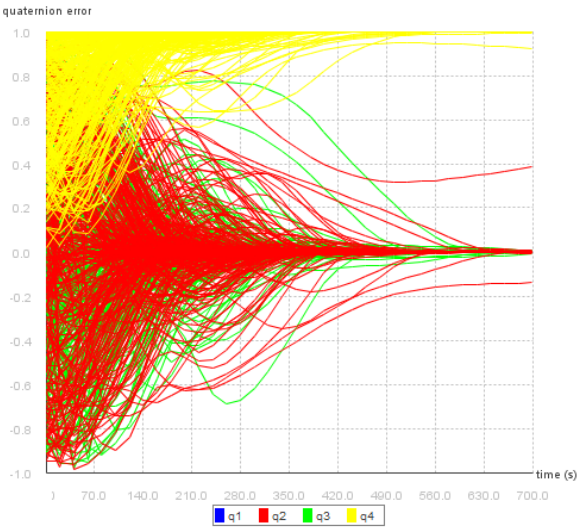
The simulation results are in accordance with the literature (ÇIMEN, 2008; ÇIMEN, 2010; CLOUTIER et al., 1996; GONZALES; SOUZA, 2009) since a SDRE controller has better performance than the linear controllers. Nonetheless, the LQR controller does not exhibit asymptotically stability as prescribed by (YANG, 2012).

Using the Distributed Package

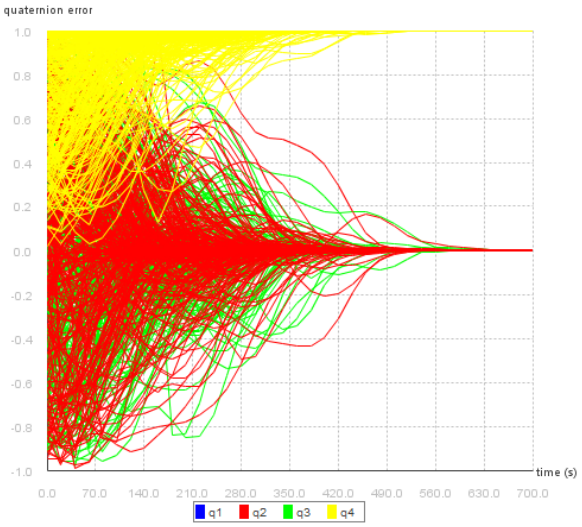
This simulation is produced running the class `MultiSimulationController` and method `main`, with the following parameters:



(a) PID Controller.



(b) LQR Controller.



(c) SDRE Controller.

Figure 5.18 - Comparison between controllers performance.

Table 5.4 - Satellite characteristics of CONASAT and references.

Name	Value
Satellite Characteristics	
inertia tensor ($kg.m^2$)	$\begin{bmatrix} 0.0547 & 0 & 0 \\ 0 & 0.0519 & 0 \\ 0 & 0 & 0.0574 \end{bmatrix}$
Actuators Characteristics - Reaction Wheels	
inertia ($kg.m^2$)	0.00015
maximum torque ($N.m$)	0.000625
maximum angular velocity (RPM)	750
References for the controller	
solar vector in the body (XYZ)	$\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T$
angular velocity ($radians/second$, XYZ)	$\begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$

```
new MultiSimulationController(300).run();
```

The satellite characteristics, references and initial conditions can be changed through the modification of the `simulationController.properties`.
Indeed, to reproduce the following data please use the content of the file `simulationcontroller_201806_SPACEOPS.properties`, which points to the `amazonia1.properties` ensuring that the additional controller is turned off (no desaturation of reaction wheels).

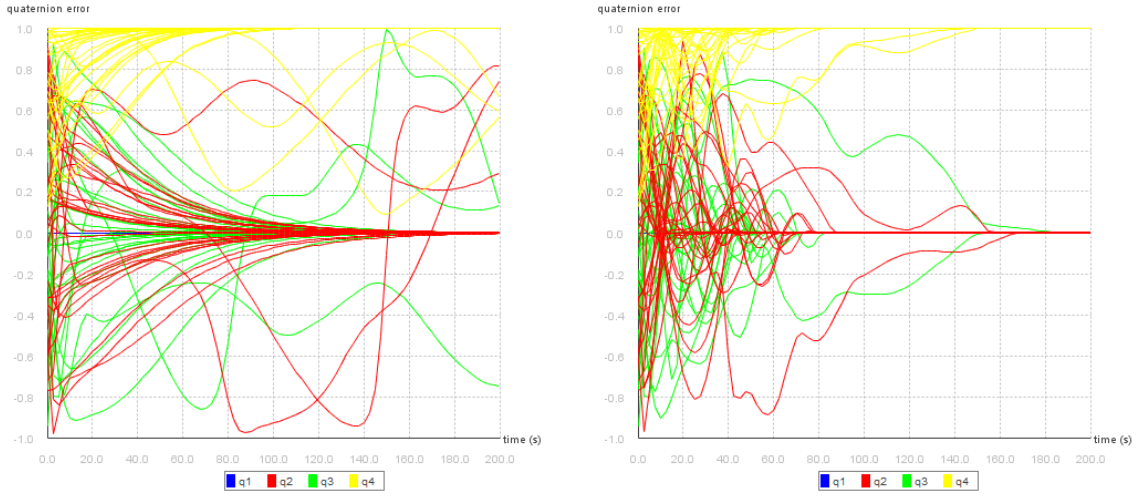
The usage of a Monte Carlo perturbation model based on normal distributions to generate the initial conditions for the attitude is the option applied in the the class `MultiSimulationController`. A better alternative would be to construct a probability density function (pdf) of a random attitude (SHUSTER, 2003).

5.2.3 Comparing PID and SDRE through Monte Carlo

(ROMERO; SOUZA, 2019d) and (ROMERO; SOUZA, 2018) compared the performance of the linear PID (Equation 4.7) and nonlinear SDRE based on Gibbs Vector (Equation 4.18) using Satellite Simulation for a CubeSat, the CONASAT (B. D. Reis de Mesquita and H. Koiti Kuga and V. Carrara, 2017).

Table 5.4 shows the satellite characteristics (B. D. Reis de Mesquita and H. Koiti Kuga and V. Carrara, 2017) and references used in the simulation results.

In order to compare the performance and robustness of the controllers, a simula-



(a) (b)

Figure 5.19 - Quaternion error comparison between controllers: (a) PID and (b) SDRE.

tion test was conducted with the full Monte Carlo perturbation model described as follows: (1) the initial Euler angle errors of the nonlinear spacecraft system are randomly selected using three independent normal distributions ($\mu = 0$, $\sigma = 1$) multiplied by 90°; and (2) the initial angular velocity errors are randomly selected using three independent normal distributions ($\mu = 0$, $\sigma = 1$) multiplied by 0.1 rad/s.

The Monte Carlo model ran 30 times and in each time one simulation of the two controllers was executed. Such executions used simulation time 200 seconds, fixed step 0.005 seconds, the data presented in Table 5.4 and the following controllers: (1) PID controller ($K_p = 1$, $K_{pd} = diag(-24, -26, -32)$, $K_{pi} = 0$) defined by Equation 4.7; and (2) SDRE controller ($R = 1$ and $Q = 1$) defined by Equation 4.18.

Each graph in Fig. 5.19 shows the respective collection of all quaternion errors computed during simulations for a given controller. Fig. 5.19 shows that the scalar part of quaternion (q_4) converges to 1 ($q_4 \rightarrow 1$) as well as that the vectorial part of the quaternion converges to 0 ($[q_1\, q_2\, q_3]^T \rightarrow [0\,0\,0]^T$) in the sense that the target attitude of the satellite was reached. The PID controller, in Fig. 5.19 (a), is not able to control all runs in the specified simulation time. On the other hand, the SDRE controller, in Fig. 5.19 (b), has the best performance since is exhibits asymptotically stability for all runs in at most 180 seconds.

The simulation results are in accordance with the literature (ÇIMEN, 2008; ÇIMEN, 2010; CLOUTIER et al., 1996; GONZALES; SOUZA, 2009) since a SDRE controller has better performance and robustness than the linear controllers.

Using the Distributed Package

This simulation is produced running the class `MultiSimulationController` and method `main`, with the following parameters:

```
new MultiSimulationController(30).run();
```

The satellite characteristics, references and initial conditions can be changed through the modification of the `simulationController.properties`.

Indeed, to reproduce the following data please use the content of the file `simulationcontroller_201902_SPIE.properties`, which points to the `conasat.properties`.

5.2.4 Comparing SDRE through Monte Carlo and Determinant of Controllability

(ROMERO; SOUZA, 2019a) and (ROMERO; SOUZA, 2019b) compared the performance of the nonlinear SDRE based on Gibbs Vector (Equation 4.18) and nonlinear SDRE based on MRPs (Equation 4.21) using Satellite Simulation.

According to (ÇIMEN, 2010), an effective approach for selecting the optimal state-space model for the SDRE is to attempt to maximize the pointwise stabilizability of the possible models, since pointwise control effort can be directly linked with controllability. Controllability criterion requires the value of determinant of the controllability matrix to be different from zero, therefore, a graphical comparison of the absolute value of the determinant of controllability matrix can be used to reveal when pointwise controllability is maximized. For multi-input systems, as the one studied in the present paper, the controllability matrix is nonsquare, then the controllability matrix multiplied by its transpose is used to evaluate the determinant.

Numerical simulations were performed to determine which of the equations (4.18 or 4.21) maximizes the controllability of the system for the given satellite characteristics, initial conditions and references in Table 5.2 and Table 5.3. In order to compare the controllability, a full Monte Carlo perturbation model was conducted with two independent systems each one defined by equations 4.18 and 4.21 applying a SDRE controller ($Q = 1$ and $R = 1$). The Monte Carlo perturbation model is described as follows: (1) the initial Euler angle errors of the nonlinear spacecraft system are randomly selected using independent normal distributions ($\mu = 0°$, $\sigma = 0.001°$); and (2) the initial angular velocity errors are randomly selected using independent normal distributions ($\mu = 0.001\ rad/s$, $\sigma = 0.009\ rad/s$).

The Monte Carlo model ran 20 times and in each time one simulation of the two systems was executed. Such executions used simulation time 700 seconds, fixed step 0.01 seconds and the data presented in Table 5.2 and Table 5.3.

Fig. 5.20(a) shows the controllability of the state-space model defined by Equation 4.18 (using quaternions and the Gibbs vector), whereas Fig. 5.20(b) shows the controllability of the state-space model defined by Equation 4.21 (MRPs).

As controllability matrix is based on $A(x)$ and B (a constant and sparse matrix), and $A(x)$ is formed by the quaternions and angular velocities (which regulator problem drives to zero, except $q4$) the determinant of the controllability matrix is expected to be small.

Based on Fig. 5.20, it is possible to conclude that MPRs maximizes the controllability through the simulations since, using the same scale, MRPs provides the greatest determinant.

In order to confirm that controllability of MRPs is better than the one provided by Gibbs, the condition number of the controllability matrix is evaluated. Fig. 5.21 shown that the condition number of the controllability based on MRPs is the smallest, furthermore, it is much smaller. Therefore, the MRPs provide better controllability.

Using the Distributed Package

This simulation is produced running the class `MultiSimulationController` and method `main`, with the following parameters:

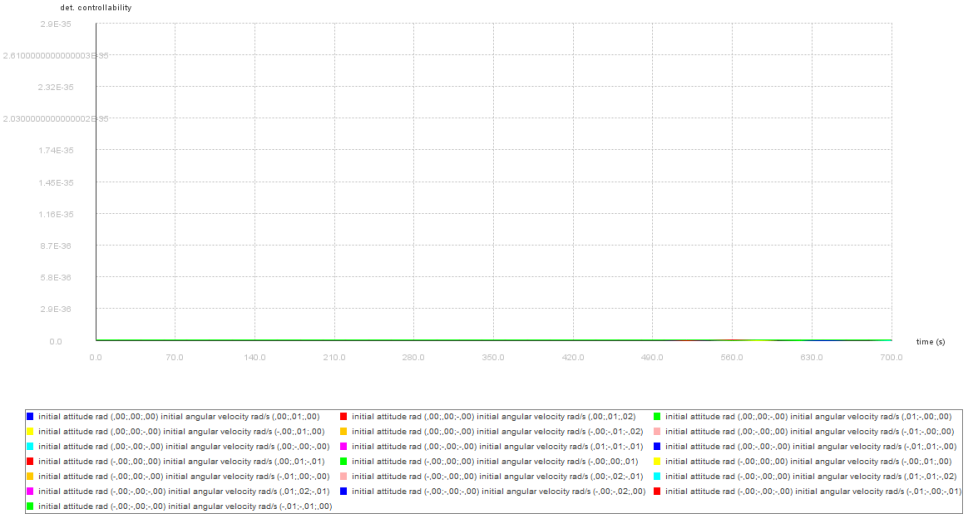
```
new MultiSimulationController(20).run();
```

The satellite characteristics, references and initial conditions can be changed through the modification of the `simulationController.properties`.

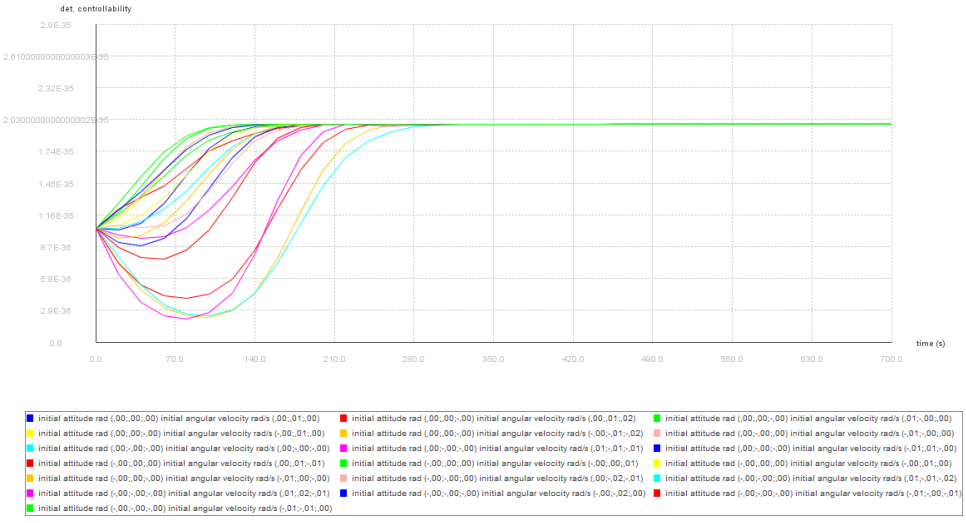
Indeed, to reproduce the following data please use the content of the file `simulationcontroller_201901_ICEDyn.properties`, which points to the `amazonia1.properties` ensuring that the additional controller is turned off (no desaturation of reaction wheels).

5.3 Tunning Controllers

(ROMERO; SOUZA, 2019a) and (ROMERO; SOUZA, 2019b) tunned nonlinear SDRE based on MRPs (Equation 4.21) using Satellite Simulation.

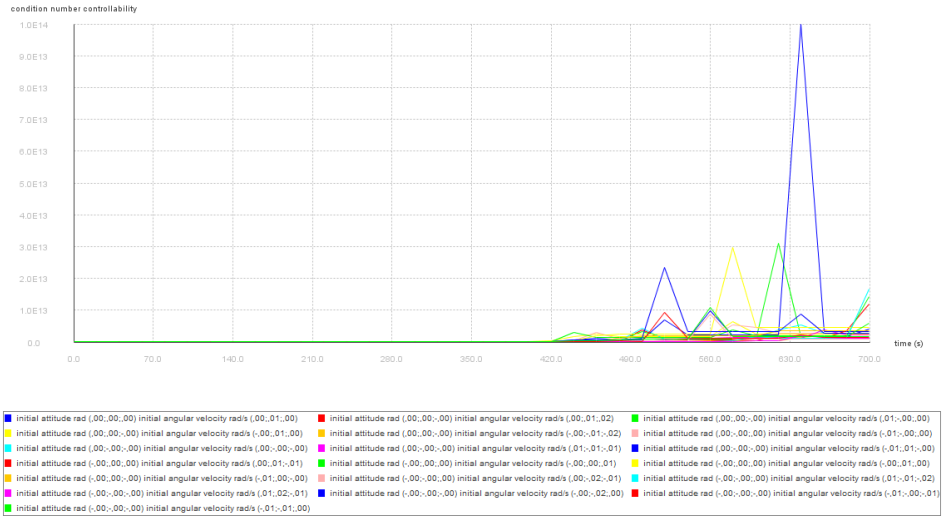


(a) Controllability of state-space model defined using quaternions with Gibbs vector (Equation 4.18).

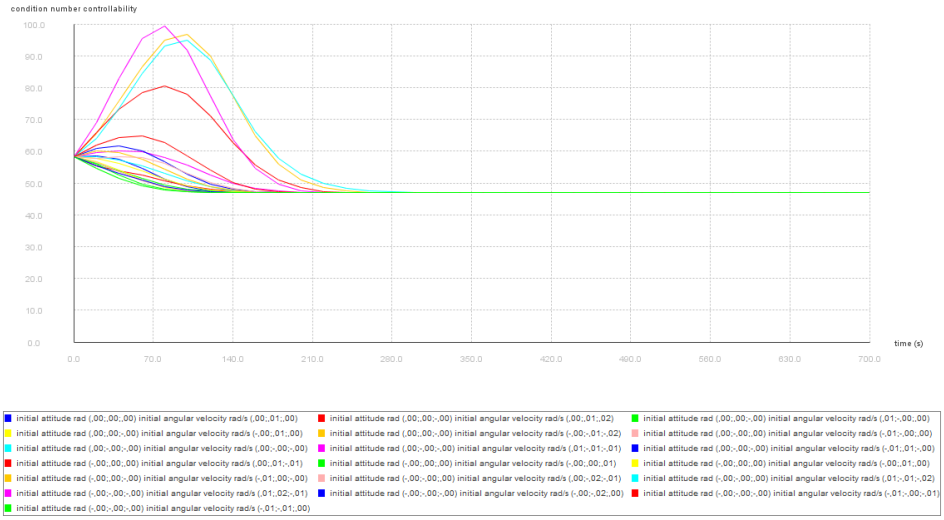


(b) Controllability of state-space model defined using MRPs (Equation 4.21).

Figure 5.20 - Comparison (determinant of controllability) between state-space models.



(a) Controllability of state-space model defined using quaternions with Gibbs vector (Equation 4.18).



(b) Controllability of state-space model defined using MRPs (Equation 4.21).

Figure 5.21 - Comparison (condition number of Controllability) between state-space models.

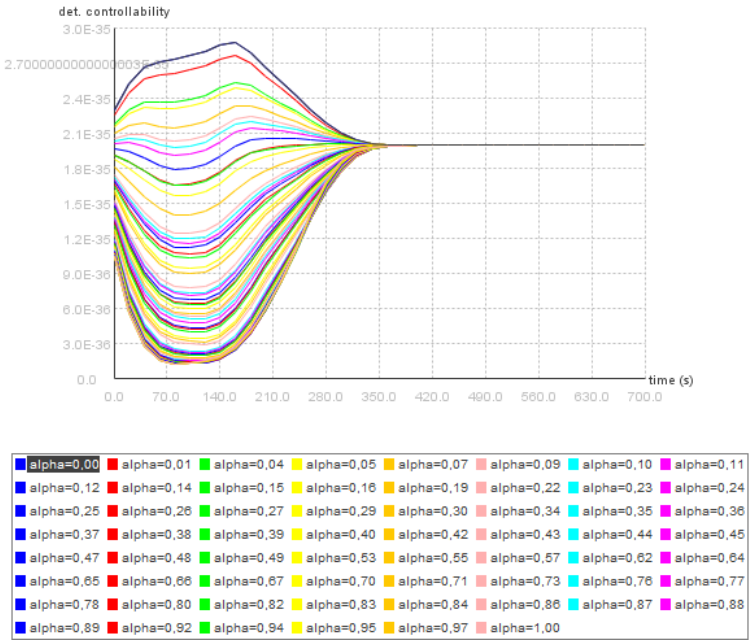


Figure 5.22 - Controllability of state-space model defined using MRPs (Equation 4.23).

For multivariable state-space models, as the one studied in the present paper, given two distinct SDC matrices $A_1(x)$ and $A_2(x)$ then there is an infinite number of SDC parametrizations (ÇIMEN, 2010). Such infinite parametrizations can be constructed using Equation 4.23 for nonlinear SDRE based on MRPs. Equation 4.23 can be used to evaluate if the combination of the two components ($A_1(x)$ and $A_2(x)$) provides better controllability.

Applying parametrizations defined in Equation 4.23, a simulation test was conducted with a full Monte Carlo perturbation model, in which α was randomly selected in the interval $0 \leq \alpha \leq 1$. The goal was to evaluate the impact of α based on a given satellite characteristics, initial conditions and references in Table 5.2 and Table 5.3.

The Monte Carlo model ran 90 times and in each time one simulation using a different α was performed for the same data in Table 5.2 and Table 5.3. Fig. 5.22 shown the resulting controllability of each run.

It is possible to conclude that parametrization defined by Equation 4.23 is the optimal since the controllability is the highest through the entire simulation when $\alpha = 0$ (highlighted in the legend, when only A_2 in Equation 4.23 is *active*). Such conclusion is based on the characteristics of the satellite, the initial conditions, the references

for the controller in Table 5.2 and the Monte Carlo perturbation model so it neither valid for the general case nor even for a different initial condition out of the range of the Monte Carlo perturbation model.

Using the Distributed Package

This simulation is produced running the class `MultiAlphaSimulationController` and method `main`, with the following parameters:

```
new MultiAlphaSimulationController(90).run();
```

The satellite characteristics, references and initial conditions can be changed through the modification of the `simulationController.properties`.

Indeed, to reproduce the following data please use the content of the file `simulationcontroller_201901_ICEDyn.properties`, which points to the `amazonia1.properties` ensuring that the additional controller is turned off (no desaturation of reaction wheels).

5.4 Assessing Stability of Controllers

Robust stability can be evaluated in the presence of model uncertainty using Satellite Simulation.

For example, the Monte Carlo model ran 20 times and in each time one simulation using MRPs (Equation 4.21) was executed, in such simulations the inertia tensor is changed by $\pm$ 7% in each component, in order to evaluate robust stability. Fig. 5.23(a) and Fig. 5.23(b) shown, respectively, the quaternion errors and angular velocity errors.

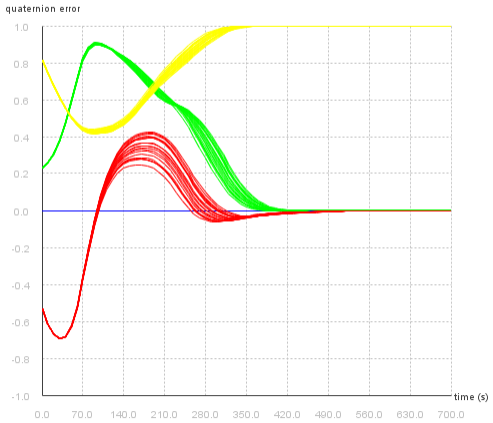
Such simulations results are neither valid for the general case nor for parametric uncertainty out of the range of the Monte Carlo perturbation models due to the underlining nonlinear dynamics.

Using the Distributed Package

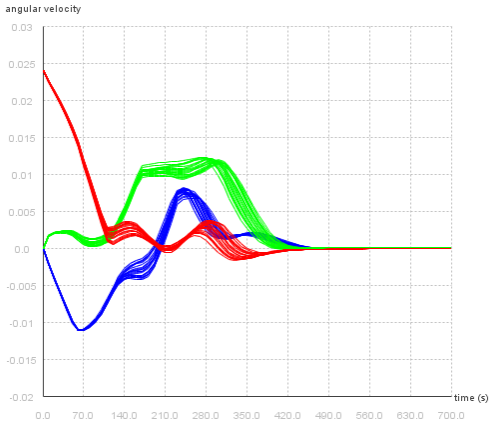
This simulation is produced running the class `MultiSimulationParametricUncertaintyController` and method `main`, with the following parameters:

```
new MultiSimulationParametricUncertaintyController(20).run();
```

The satellite characteristics, references and initial conditions can be changed



(a) Quaternion error - parametric uncertainty (Equation 4.21).



(b) Angular velocity error - parametric uncertainty (Equation 4.21).

through the modification of the `simulationController.properties`.
Indeed, to reproduce the following data please use the content of the file `simulationcontroller_201901_MSSP.properties`, which points to the `amazonia1.properties` ensuring that the additional controller is turned off (no desaturation of reaction wheels).

---

6 CONCLUSIONS

To the best of our knowledge, the proof of concept of the open-source satellite simulator for SDRE controllers is original (ROMERO et al., 2018; ROMERO; SOUZA, 2018; ROMERO; SOUZA, 2019c; ROMERO; SOUZA, 2019a; ROMERO; SOUZA, 2019b; ROMERO; SOUZA, 2019d) (see Apendix C for the list of publications directly related to the Satellite Simulation). Moreover, the simulator can run in a variety of platforms - including an Android operating system in a remote sensing CubeSat - as well as it has low cost. It is worth to mention that 39,33% of the cost of CONASAT was selected for the acquisition of software license (page 12, (CARVALHO, 2010)) whereas the open-source options, like the one proposed here, would reduce such cost to zero.

Furthermore, to the best of our knowledge, this is an original contribution for the optimal arrangement of the SDC for a three-axis stabilized satellite model. The results shown that different SDCs can produce extremely different results ranging from non-applicability of the SDRE technique to differences in the controllability and, consequently, in the performance and robustness of the system. Unfortunately, the optimal factorization found is neither valid for the general case nor even for different initial conditions out of the range of the Monte Carlo perturbation model due to the underlining nonlinear dynamics. However, the procedure applied can provide guidance for engineers.

Another contribution is the kinematical part of the state-space models in equations 4.20, 4.23, since they can be used in any system that exhibit rotational motion, e.g., airplanes.

Regarding the discussion whether the SDRE technique and its SDC factorization in AOCS can yield gains in the missions developed by INPE, since performance in the LEOP is critical to the success of a mission and the simulation results show that the performance and robustness of SDRE controllers can be enhanced by optimal factorizations (in particular, with kinematics based on MRPs); SDRE can yield gains. Nonetheless, its implementation requires more computing resources and tend to exhibit difficulties for verification. Therefore, it is too early to draw a definitive conclusion.

In conclusion, the open-source satellite simulator fills a gap in the open-source community, the availability of options to run SDRE controllers and filters in Java - including an Android operating system in a remote sensing CubeSat. The optimal factorization of SDC in the SDRE technique is of utmost importance for perfor-

mance and robustness of nonlinear systems controlled by such technique. Moreover, the simulation results indicate that SDRE can yield gains in the performance of INPE missions, in particular, Amazonia-1 and CONASAT.

6.1 Outlook

The following items are part of the outlook:

- Evaluate the Monte Carlo perturbation models in a large number of executions;
- Mathematical Modeling and evaluation of saturation in the actuators (ÇIMEN, 2010);
- Analytical evaluation of the stability through second method of Lyapunov or region of attraction;
- Torque free motion of rigid body - Investigation of stability (see Appendix B);
- Hybrid systems convergence and sliding mode control.

REFERENCES

AG uBlox. **Datum Transformation of GPS Positions**. 2017. https://microem.ru/files/2012/08/GPS.G1-X-00006.pdf. Access date: 01.Apr.2017. 14

ARMBRUSTER, A.; BAKER, J.; CUNEI, A.; FLACK, C.; HOLMES, D.; PIZLO, F.; PLA, E.; PROCHAZKA, M.; VITEK, J. A real-time java virtual machine with applications in avionics. **ACM Trans. Embed. Comput. Syst.**, ACM, New York, NY, USA, v. 7, n. 1, p. 5:1–5:49, dec. 2007. ISSN 1539-9087. Available from: <http://doi.acm.org/10.1145/1324969.1324974>. 1

B. D. Reis de Mesquita and H. Koiti Kuga and V. Carrara. Estimation and Attitude Control in CONASAT Nominal Operation Mode: An Approach for SDRE Filter and PID Control. **IEEE Latin America Transactions**, v. 15, n. 5, p. 835–842, 2017. 1, 30, 58

CARRARA, V. **Cinemática e dinâmica da atitude de satélites artificiais**. São José dos Campos: Instituto Nacional de Pesquisas Espaciais, 2012. 111 p. Available from: <http://urlib.net/sid.inpe.br/mtc-m19/2012/01.26.19.13>. Access in: 02.May.2017. 15, 19, 20, 22

CARVALHO, M. J. M. de. **Estudo de uma Missão Espacial para Coleta de Dados Ambientais Baseada em Nano Satélites**. 2010. Access date: 21.Oct.2018. 1, 69

ÇIMEN, T. State-Dependent Riccati Equation (SDRE) control: A survey. **IFAC Proceedings Volumes (IFAC-PapersOnline)**, v. 17, n. 1 PART 1, p. 3761–3775, 2008. ISSN 14746670. 1, 30, 54, 56, 59, 77

_____. Systematic and effective design of nonlinear feedback controllers via the state-dependent Riccati equation (SDRE) method. **Annual Reviews in Control**, v. 34, n. 1, p. 32–51, 2010. ISSN 13675788. 1, 29, 30, 54, 56, 59, 60, 64, 70, 77

CLOUTIER, J. R.; D'SOUZA, C. N.; MRACEK, C. P. Nonlinear regulation and nonlinear H-infinity control via the state-dependent Riccati equation technique. **Conference on Nonlinear Problems in Aviation**, n. September, 1996. 1, 54, 56, 59, 77

Di Mauro, G.; Di Lizia, P.; LAVAGNA, M. Control of Relative Motion via State-Dependent Riccati Equation Technique. In: **AAS/IAAA Astrodynamics Specialist Conference, Vol 142, Univelt, San Diego, Ca**. [S.l.: s.n.], 2011. 30

Di Mauro, G.; SCHLOTTERER, M.; THEIL, S.; LAVAGNA, M. Nonlinear Control for Proximity Operations Based on Differential Algebra. **Journal of Guidance, Control, and Dynamics**, American Institute of Aeronautics and Astronautics, v. 38, n. 11, p. 2173–2187, apr 2015. ISSN 0731-5090. Available from: <https://doi.org/10.2514/1.G000842>. 30, 77, 78

ESA. **GNSS Data Processing - Volume I: Fundamentals and Algoritms**. 2017. http://www.navipedia.net/GNSS_Book/ESA_GNSS-Book_TM-23_Vol_I.pdf. Access date: 01.Apr.2017. 13, 14, 15

FORTESCUE, P. W.; SWINERD, G. G. Attitude Control. **Spacecraft Systems Engineering**, n. June, p. 289–326, 2011. Available from: <http://doi.wiley.com/10.1002/9781119971009.ch9>. 20, 21

FOUNDATION, A. **Maven 3**. 2017. https://maven.apache.org/. Access date: 26.May.2017. 5, 6

FUTEK: Torque sensor - Application 317. 2018. Access date: 13.Apr.2018. Available from: <http://www.futek.com/application/torque-sensor/Satellite-Reaction-Wheel-Torque>. vii, 25

GONZALES, R. G.; SOUZA, L. C. G. d. Application of the sdre method to design a attitude control system simulator. **Advances in the Astronautical Sciences**, v. 134, n. Part 1-3, p. 2251–2258, 2009. ISSN 0065-3438. Setores de Atividade: Educação. Access in: 11 mar. 2018. 30, 54, 56, 59

HIPPARCHUS (version 1.3-SNAPSHOT). 2018. Access date: 28.Jan.2018. Available from: <https://github.com/Hipparchus-Math/hipparchus>. 5, 7, 16, 77

HUGHES, P. C. **Spacecraft Attitude Dynamics**. [S.l.]: New York, 1986. viii, 1, 15, 19, 20, 22, 24, 79, 80

JAVA 3D API (Version 1.5.1). 2017. Access date: 26.May.2017. Available from: <http://www.oracle.com/technetwork/articles/javase/index-jsp-138252.html>. 2, 6, 7

LIU Y.; CHEN, L. **Chaos in Attitude Dynamics of Spacecraft**. [S.l.]: Springer, 2013. 80, 82, 83, 84

MENON, P. K.; LAM, T.; CRAWFORD, L. S.; CHENG, V. H. Real-time computational methods for SDRE nonlinear control of missiles. **Proceedings of the American Control Conference**, v. 1, p. 232–237, 2002. ISSN 07431619. 1, 29, 30, 77, 78

NAYFEH, A. H. Applied Nonlinear Dynamics. 1995. Available from: <http://doi.wiley.com/10.1002/9783527617548>. 82

OCEANIC, N.; ADMINISTRATION, A. **Geomag 7.0 - Windows**. 2017. https://www.ngdc.noaa.gov/IAGA/vmod/igrf.html. Access date: 01.Apr.2017. 5, 7

ORACLE. **Java Development Kit 1.8**. 2017. http://www.oracle.com/technetwork/java/javase/overview/index.html. Access date: 26.May.2017. 5, 6

OREKIT (version 8.0). 2017. Access date: 01.Apr.2017. Available from: <https://www.orekit.org>. 5, 7

PEARSON, J. D. Approximation Methods in Optimal Control I. Sub-optimal Control. **Journal of Electronics and Control**, Taylor & Francis, v. 13, n. 5, p. 453–469, 1962. Available from: <https://doi.org/10.1080/00207216208937454>. 1

RICHET, Y. **JMath Plot v1**. 2017. https://github.com/yannrichet/jmathplot. Access date: 26.May.2017. 6, 7

ROMERO, A. G. **Hipparchus Contributions**. Hipparchus.org, 2017. Access date: 03.Nov.2019. Available from: <https://github.com/Hipparchus-Math/hipparchus/commit/b9d19a715993bb82b9711e7910a4dd238d7bb91b>. 78

_____. **Satellite Simulation v0.1.0 - Executable**. National Institute for Space Research, 2019. Access date: 03.Nov.2019. Available from: <http://urlib.net/rep/8JMKD3MGP3W34R/3SMTKRH>. 5, 6

_____. **Satellite Simulation v0.1.0 - Sources**. National Institute for Space Research, 2019. Access date: 03.Nov.2019. Available from: <http://urlib.net/rep/8JMKD3MGP3W34R/3SNSQ7L>. 5, 6

ROMERO, A. G.; SOUZA, L. C. G. Application of the sdre technique based on java in a cubesat attitude and orbit control subsystem. In: **IAA Proceedings of the 3rd IAA Latin American CubeSat - IAALACW**. [S.l.: s.n.], 2018. 1, 58, 69, 87

ROMERO, A. G.; SOUZA, L. C. G.; CHAGAS, R. A. Application of the sdre technique in the satellite attitude and orbit control system with nonlinear dynamics. In: _____. **2018 SpaceOps Conference**. [s.n.], 2018. Available from: <https://arc.aiaa.org/doi/abs/10.2514/6.2018-2536>. 1, 30, 56, 69, 88

ROMERO, A. G.; SOUZA, L. C. G. de. Application of a new optimal factorization of the sdre method in the satellite attitude and orbit control system design with nonlinear dynamics. In: **Proceedings of the Eleventh International Conference on Advances in Satellite and Space Communications SPACOMM 2019**. Valencia, Spain: [s.n.], 2019. 60, 61, 69, 90

_____. Optimal factorization of the state-dependent riccati equation technique in a satellite attitude and orbit control system. In: **Proceedings of the International Conference on Structural Engineering Dynamics (ICEDyn 2019)**. Viana do Castelo, Portugal: [s.n.], 2019. 60, 61, 69, 89

_____. Satellite controller system based on reaction wheels using the state-dependent riccati equation (sdre) on java. In: CAVALCA, K. L.; WEBER, H. I. (Ed.). **Proceedings of the 10th International Conference on Rotor Dynamics – IFToMM**. Cham: Springer International Publishing, 2019. p. 547–561. ISBN 978-3-319-99268-6. 54, 69, 91

_____. State-dependent Riccati equation controller using Java in remote sensing CubeSats. **Journal of Applied Remote Sensing**, SPIE, v. 13, n. 3, p. 1 – 10, 2019. ISSN 19313195. Available from: <https://doi.org/10.1117/1.JRS.13.032509>. 30, 58, 69, 87

ROSENSTEIN, M. T.; COLLINS, J. J.; De Luca, C. J. **A practical method for calculating largest Lyapunov exponents from small data sets**. [S.l.], 1992. 82

SHARP, D. C.; PLA, E.; LUECKE, K. R.; HASSAN, R. J. Evaluating real-time java for mission-critical large-scale embedded systems. In: **The 9th IEEE Real-Time and Embedded Technology and Applications Symposium, 2003. Proceedings.** [S.l.: s.n.], 2003. p. 30–36. ISSN 1545-3421. 1

SHIVARAMA, R.; FAHRENTHOLD, E. P.; SHIVARAMA, R.; FAHRENTHOLD, E. P. Chapter 3 Hamilton ' S Equations With Euler Parameters for Rigid Body Dynamics Modeling Hamilton ' S Equations With Euler Parameters. n. 512, 2018. 82

SHUSTER, M. D. **A Survey of attitude representations**. 1993. 439–517 p. 21

_____. Uniform attitude probability distributions. **Journal of the Astronautical Sciences**, v. 51, n. 4, p. 451–475, 2003. ISSN 00219142. 58

SIDI, M. J. **Spacecraft Dynamics and Control - A Practical Engineering Approach**. [S.l.]: Cambridge University press, 2006. 1, 15, 16, 17, 24, 25, 26, 36, 52

STANSBERY, D. T.; CLOUTIER, J. R. Position and attitude control of a spacecraft using the state-dependent riccati equation technique. In: **Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No.00CH36334)**. [S.l.: s.n.], 2000. v. 3, p. 1867–1871 vol.3. ISSN 0743-1619. 30

TALBOR, M. **Chaos and integrability in nonlinear dynamics : an introduction**. [S.l.]: Wiley-Interscience; 1 edition (January 4, 1989), 1989. 384 p. 82

WERTZ, J. R.; LARSON, W. **Space Mission Analysis and Design**. [S.l.: s.n.], 1999. 976 p. ISBN 9780792359012. 28

YANG, Y. Analytic LQR Design for Spacecraft Control System Based on Quaternion Model. **Journal of Aerospace Engineering**, v. 25, n. 3, p. 448–453, 2012. ISSN 0893-1321. Available from: <http://ascelibrary.org/doi/abs/10.1061/{%}28ASCE{%}29AS.1943-5525.0000142>. 30, 34, 35, 56

APPENDIX A - ALGEBRAIC RICCATI EQUATION SOLVING

The algebraic Riccati equation (ARE) numerically solving is the cornerstone of the SDRE since a closed-form solution of Equation 4.5 is awkward except for a few simple dynamic systems (ÇIMEN, 2008; ÇIMEN, 2010; CLOUTIER et al., 1996; Di Mauro et al., 2015). The numerical techniques for ARE solving can be divided into two groups: (1) direct and (2) iterative methods.

The *RiccatiEquationSolverImpl*, the major contribution to the Hipparchus (HIPPARCHUS..., 2018), applies direct method to compute a first estimate for P_0 (stable), afterwards, such initial estimate is used to approximate the final P using an iterative method. For the sake of completeness, a brief description of the *RiccatiEquationSolverImpl* is shared.

Considering the direct method applied in the *RiccatiEquationSolverImpl*, the solution P of an ARE of order n can be obtained in terms of the solution of a linear Hamiltonian equation of order $2n$ (MENON et al., 2002) as shown in Equation A.1.

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \end{bmatrix} = H \begin{bmatrix} X \\ Y \end{bmatrix} \tag{A.1}$$
$$H = \begin{bmatrix} A & -BR^{-1}B^T \\ -Q & -A^T \end{bmatrix}$$

where H is the *Hamiltonian matrix* corresponding to the ARE.

Complementarily, H can be decomposed using an ordered complex eigen decomposition (additional contributions to Hipparchus, *OrderedComplexEigenDecomposition* and *ComplexEigenDecomposition*) as given by Equation A.2 (a better numerical alternative would be an ordered Schur decomposition (MENON et al., 2002)).

$$H = U\Lambda U^{-1} \tag{A.2}$$
$$U = \begin{bmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{bmatrix} \tag{A.3}$$

Finally, assumming that H has no purely imaginary eigenvalues, consequently, if a real λ is an eigenvalue of H, then so is $-\lambda$. Therefore, the eigenvectors U_{11} and U_{21} can be used to define the unique, symmetric, positive-definite solution P_0 as $P_0 = U_{21}U_{11}^{-1}$ (MENON et al., 2002).

Equipped with P_0 as initial estimate, an iterative method, the Kleinman algorithm (MENON et al., 2002; Di Mauro et al., 2015), is executed in *RiccatiEquationSolverImpl*. Let $K_0 = -P_0 R^{-1} B^T$ be such that closed-loop system $(A + BK_0^T)$ has all eigenvalues with negative real parts (stable), then P and K are recursively defined as shown in Equation A.4.

$$P_i = vec^{-1}[((A + BK_i^T)^T \otimes 1 + 1 \otimes (A + BK_i^T)^T)^{-1} \, vec(-K_i R K_i^T - Q)]$$
$$K_{i+1} = -P_i B R^{-1} \qquad \text{(A.4)}$$

where *vec* is the vectorization unary operator that is a linear transformation which converts a matrix into a column vector and $\otimes$ is the Kronecker product of two matrices. Then $P_i \geq P_{i+1}$ and $\lim_{i \to \infty} P_i = P$ (MENON et al., 2002).

Using the Distributed Package

The contributions (*ComplexComparator*, Kronecker product in the *Array2DRowRealMatrix*, *ComplexEigenDecomposition*, *OrderedComplexEigenDecomposition*, *OrderedEigenDecomposition*, *RiccatiEquationSolver* and *RiccatiEquationSolverImpl* as well as the respective automated tests) to the Hipparchus library are available at (ROMERO, 2017).

APPENDIX B - STABILITY INVESTIGATION

B.1 Introduction

This appendix reports initial results focused on the investigation about stability of a rigid body in a torque free motion.

Firstly, the classical results, e.g., (HUGHES, 1986) (page 114), define ω-stability, attitude-stability and directional-stability regarding the so called "pure-spins". These results are based on the magnitude of the inertia component of each axis in the inertia tensor. Taking into account three axes, (1) the major-axis is the one that has the greatest component in the diagonal of the inertia tensor, (2) the minor-axis is the one that has the lesser component in the diagonal of the inertia tensor and (3) the intermediary-axis spin is the other.

Regarding *ω-stability* (sub-state-space covering only ω), taking into account the "pure-spins": **(1) the major-axis spin is Lyapunov stable, (2) the minor-axis spin is Lyapunov stable and (3) the intermediary-axis spin is not Lyapunov stable** ((HUGHES, 1986); page 114).

None of the three relative equilibria (pure-spin solutions) is *attitude stable* in the sense of Lyapunov. A lesser form of attitude stability is called directional stability. *Directional stability* is based on the fact that while the angular momentum vector $\vec{h}$ cannot resist angular rate perturbations about itself, it can bring about stability with respect to perturbations perpendicular to itself. Thus, under certain conditions, a body-fixed axis aligned in the reference motion $\vec{h}$ will deviate arbitrarily slightly from this direction, provided the inertial disturbances are sufficiently small. In other words, the direction of this body-fixed axis is Lyapunov stable. In these terms, **the major axis is directionally stable for a major-axis spin and that the minor axis is similarly directionally stable for a minor-axis spin**.

It is another classical result, the geometrical interpretation (HUGHES, 1986) in which the motion in ω-state-space is represented by the tip of $\vec{\omega}$ traveling on a curve, called a polhode, defined by the intersection of two ellipsoids: the energy ellipsoid $\mathscr{T}$ and the momentum ellipsoid $\mathscr{H}$, as illustrated in Fig. B.1. Therefore, regarding the ω-state-space, the motion, if it exists, is always periodic.

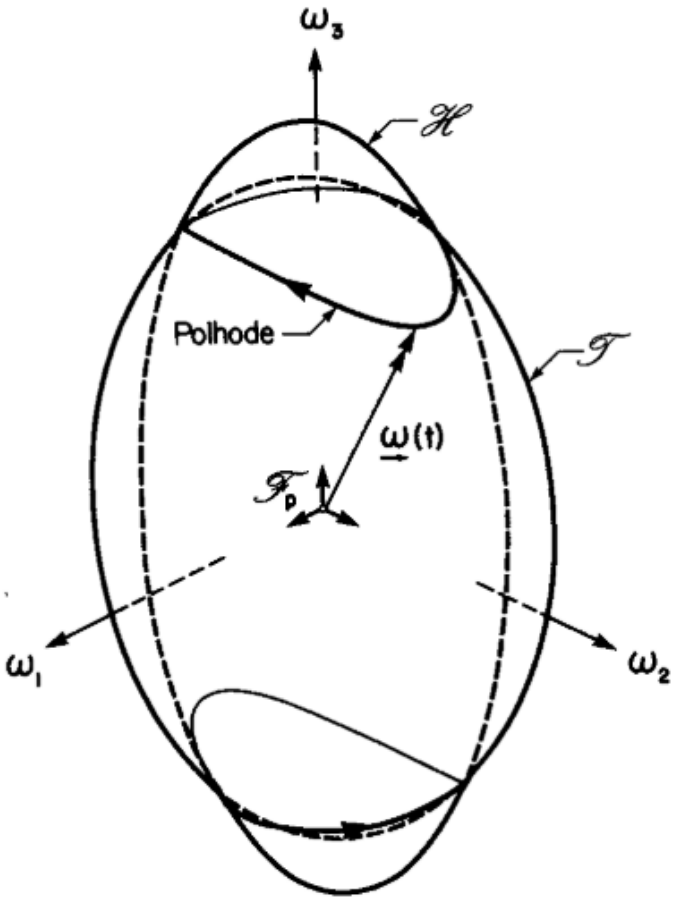


Figure B.1 - The geometrical interpretation of $\vec{\omega}$ (adapted from (HUGHES, 1986) (page 108)).

B.1.1 Problem Statement

The Satellite Simulation is used to investigate these classical results regarding ω-stabillity as well as the motions in which the intermediary axis is spinning, "pure-spins" and "not-pure-spins". A relevant question, open to the best of our knowledge, is whether the spin in the intermediary-axis generates a periodic motion, a quasiperiodic motion or a chaotic motion concerning the whole state-space, a seven-dimensional state-space (the three-dimensional for vectorial quaternion components $[q1\, q2\, q3]^T$ plus the one-dimensional for real quaternion component $q4$, plus the three-dimensional for the angular velocities $[\omega_1\, \omega_2\, \omega_3]^T$). Recall a chaotic motion can be practically defined as a bounded steady-state response that is not an equilibrium state or a periodic motion, or a quasiperiodic motion (LIU Y.; CHEN, 2013).

Taking into account Table 5.2, the major-axis spin in Amazonia-1 is Z, the minor-axis spin is X and the intermediary-axis spin is Y. Moreover, the following simu-

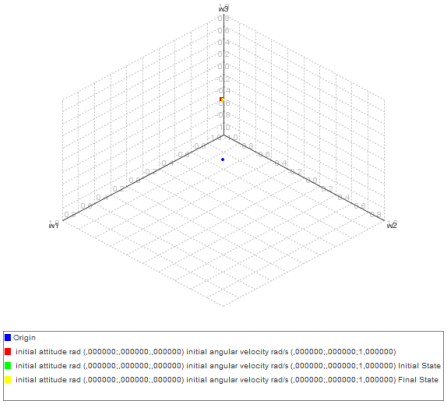
lations were performed using Statellite Simulation with fixed step equals to 0.0001 seconds and simulation time equals to 10000 seconds. Finaly, all velocities are measured in radians/second unit.

B.2 ω-stable, in the sense of Lyapunov, "pure-spins"

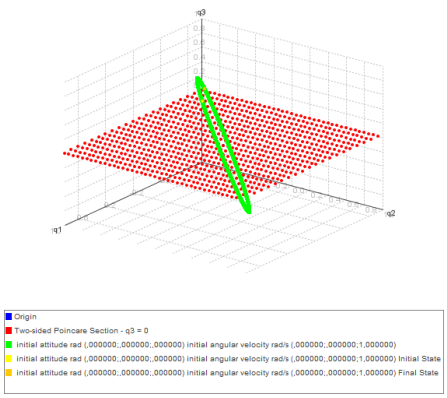
Fig. B.2 is one simulation for the initial angular velocity $[0\,0\,1]^T$. Regarding ω-state-space, the angular velocity in Z is "fixed" during the simulation resulting in small variations in the other axes angular velocities (at most 0.01), exhibiting a periodic motion as predicted by the literature (next section shares numerical results of this periodic motion). Complementarily, the vector components of quaternion-state-space define a closed-curve for that "pure-spin". Such closed-curve in the quaternion-state-space is dectected in the two-sided Poincaré section (defined by $q3 = 0$) as fixed points.

Fig. B.3 is one simulation for the initial angular velocity $[1\,0\,,0]^T$. Regarding ω-state-space, the angular velocity in X is "fixed" during the simulation resulting in small variations in the other axes angular velocities (at most 0.01), exhibiting a periodic motion as predicted by the literature. Complementarily, the vector components of quaternion-state-space define a closed-curve for that "pure-spin". Once again, such closed-curve in the quaternion-state-space is dectected in the two-sided Poincaré section (defined by $q3 = 0$) as fixed points.

In each case, two periodic motions are found, therefore, in the seven-dimensional

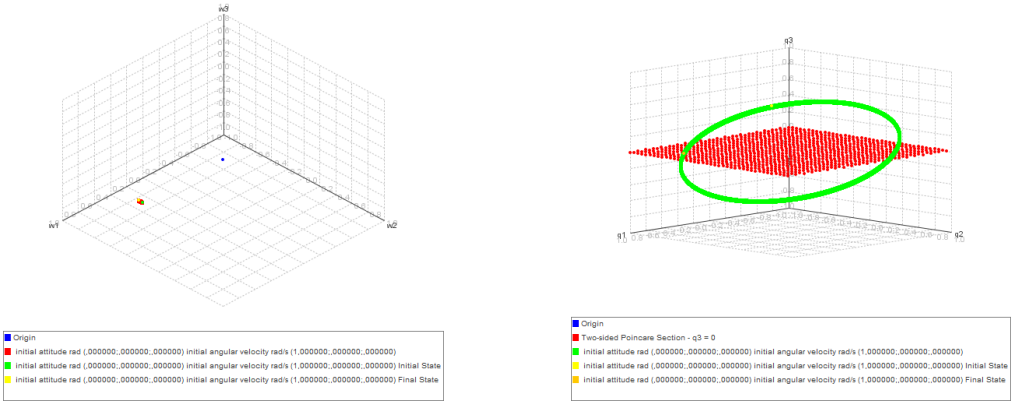


(a) ω-state-space



(b) vector components of quaternion-state-space

Figure B.2 - The major-axis spin in Amazonia-1 is Z.



(a) ω-state-space

(b) vector components of quaternion-state-space

Figure B.3 - The minor-axis spin in Amazonia-1 is X.

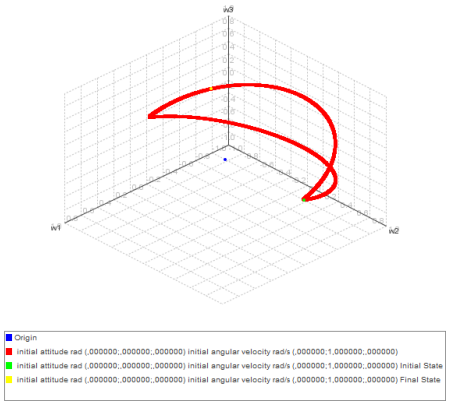
state-space, the trajectories form a 2-torus since there are two Lyapunov exponents with value 0 (each one unveiled by the periodic motions) (LIU Y.; CHEN, 2013).

It is well-known that the applied mathematical model can be defined usig the Hamilton's equations (SHIVARAMA et al., 2018). Moreover, a fundamental result states that the state-space trajectories of a Hamiltonian system with n degrees of freedom and possessing n integrals of motion lie on an n-dimensional manifold which is topologically equivalent to an n-tori (TALBOR, 1989) (pages 71-74). Therefore, the numerical results are in accordance with the literature, furthermore, the two **integrals of motion** are well-known, the physical conservative quantities: kinetic energy and angular momentum.

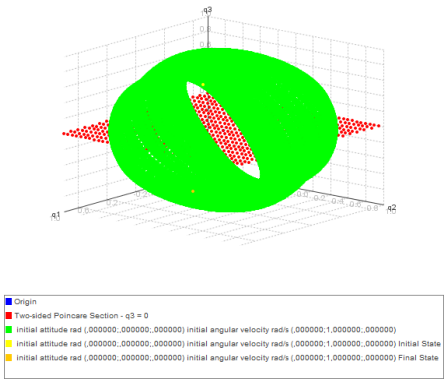
B.3 Unstable, in the sense of Lyapunov, spins

In order to investigate the unstable spins, the current work uses three tools: (1) phase portraits for vectorial part of the quaternion components and for the angular velocities, (2) a two-sided Poincaré section in the former phase portraits and (3) the evaluation of the distance between two different, but closed, trajectories (NAYFEH, 1995)) in the former phase portrait as well as the approximation of the largest Lyapunov exponent (ROSENSTEIN et al., 1992).

Fig. B.4 is one simulation for the initial angular velocity $[0\,1\,0]^T$. Regarding ω-state-space, a closed-curve emerges. Complementarily, the vector components of quaternion-state-space define a quasiperiodic motion for that "pure-spin".

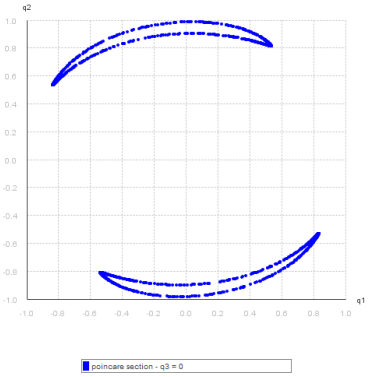


(a) ω-state-space

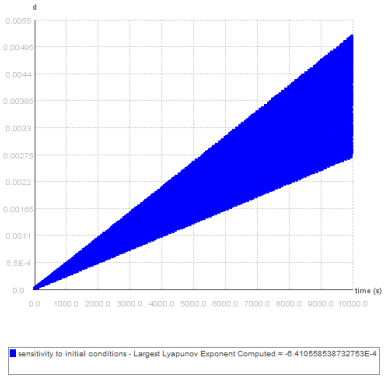


(b) vector components of quaternion-state-space

Figure B.4 - The intermediary-axis spin in Amazonia-1 is Y.



(a) Poincaré section

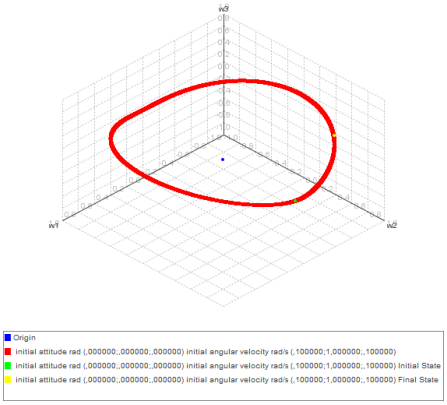


(b) sensitivity to initial conditions

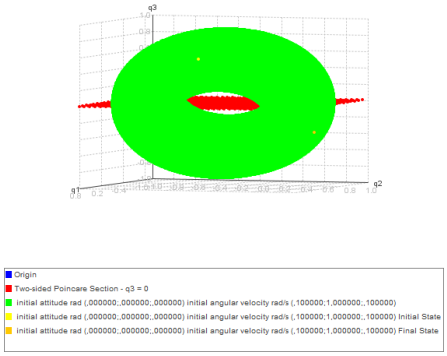
Figure B.5 - The intermediary-axis spin in Amazonia-1 is Y.

Indeed, such quasiperiodic motion is unveiled by the defined two-sided Poincaré section, as shown in Fig. B.5, since limit cycles are observed in such section what is a clear mark of a 2-torus. Furthermore, using d_0 as $10^{-6}\,radians/second$ the distance between different, but closed, trajectories are measured, which unveiled that this is not a chaotic motion since such measured distance increases linearly with time. Finally, the largest Lyapunov exponent was approximated as 0, once again, ratifying the quasiperiodic motion with two zeros as Lyapunov exponents, a 2-torus (LIU Y.; CHEN, 2013).

Fig. B.6 is a simulation for the initial angular velocities $[.1\ 1\ .1]^T$. Regarding ω-



(a) ω-state-space



(b) vector components of quaternion-state-space

Figure B.6 - A compound motion including intermediary-axis spin in Amazonia-1 is Y.

state-space, a closed-curve emerges as described by the polhode. Complementarily, the vector components of quaternion-state-space defines a quasiperiodic motion.

The quasiperiodic motion is clearly observed in the Poincaré section shown in Fig. B.7

Fig. B.8 shown that the manifolds exhibit a topological isomorphism independent of the changes in the angular velocities since the initial angular velocities are $[.2\ 1\ .2]^T$.

In conclusion, one periodic motion and one quasiperiodic motion are found, therefore, in the seven-dimensional state-space, the trajectories form a 3-torus since there are three Lyapunov exponents with value zero (LIU Y.; CHEN, 2013).

B.4 Initial Conclusions

The application of the concept of Lyapunov stability is well-established for the evaluation of stability of equilibrium points, or fixed points in the state-space, usually at origin.

Nonetheless, the nature of a rigid body in a torque free motion is not well-suited for Lyapunov stability since, at least, the attitude, or rotation, of the rigid body is constrained by two integrals of motion: conservation of rotational kinetic energy and conservation of angular momentum. Therefore, it naturally emerges a “periodic” motion.

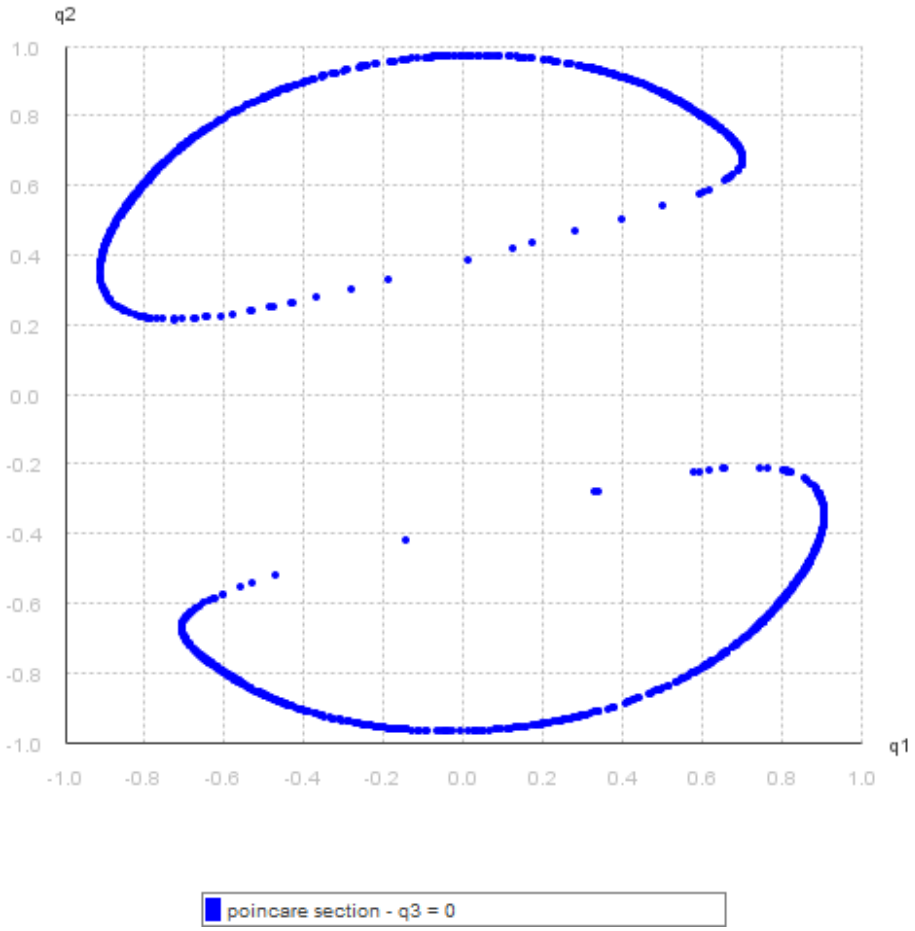


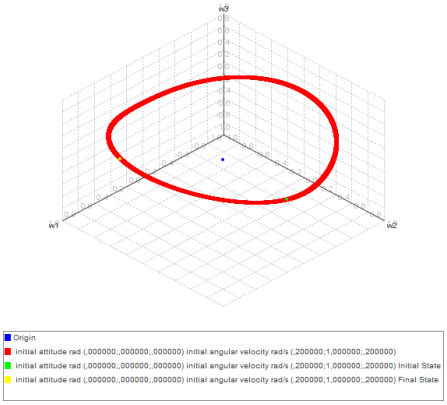
Figure B.7 - Poincaré section for the vector components of quaternion-state-space in Fig. B.4

The present initial investigation shows that the general description of a rigid body in a torque free motion is, at least, a 3-torus, unveiled by phase portraits, Poincaré sections and Lyapunov Exponents.

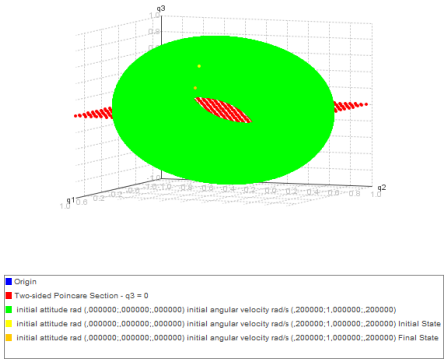
We believe that the understanding of the rigid body in a torque free motion should enable better procedures for the controller design, analysis and synthesis.

Using the Distributed Package

This simulation is produced running the class `MultiSimulationTorqueFreeInvestigation` and method `main`, with the fol-



(a) ω-state-space



(b) vector components of quaternion-state-space

Figure B.8 - A compound motion including intermediary-axis spin in Amazonia-1 is Y.

lowing parameters:

`new MultiSimulationTorqueFreeInvestigation().run();`

The satellite characteristics, references and initial conditions can be changed through the modification of the `simulationController.properties`.

Indeed, to reproduce the following data please use the content of the file `simulationcontroller_201905_STATESPACE.properties`, which points to the `amazonia1.properties` ensuring that all controllers are turned off.

APPENDIX C - LIST OF PUBLICATIONS

C.1 Space Engineering

C.1.1 2019

State-dependent Riccati equation controller using Java in remote sensing CubeSats (ROMERO; SOUZA, 2019d)
Qualis: B1 (Engineering III, Quadrennium 2013-2016)
Abstract: STRaND and PhoneSat programs have attracted the attention of the aerospace community by applying, in CubeSats, commercial off-the-shelf smartphones based on Google's Android. In Android, the development commonly applies Java hence this language is a natural candidate for the attitude and orbit control subsystem (AOCS). Moreover, such AOCS can be designed with success by linear control theory; however, the linearized models are not able to represent all the effects of the nonlinear terms present in the dynamics. Therefore, nonlinear control techniques can yield better performance. An example is the Nano-Satellite Constellation for Environmental Data Collection, used as the reference in this work, a set of remote sensing CubeSats from the Brazilian National Institute for Space Research, in which the AOCS must stabilize the satellite in three-axes. We present the investigation of a state-dependent Riccati equation (SDRE) controller, a nonlinear controller, based on attitude errors given by quaternions. The investigation uses Java, accordingly, it can run on an Android operating system in a CubeSat, and it has low cost. Two controllers (linear and SDRE) were evaluated using a Monte Carlo perturbation model. The initial results show that the SDRE controller provides better performance.

C.1.2 2018

Application of the SDRE technique based on Java in a Cubesat Attitude and Orbit Control Subsystem (ROMERO; SOUZA, 2018)
Abstract: In 2013, the STRaND (University of Surrey and Surrey Satellite Technology Ltd) and the PhoneSat (NASA) programs attracted attention of the aerospace community applying commercial off-the-shelf smartphones in CubeSats. Both programs deployed CubeSats using smartphones based on Google's Android, in which application development is mainly based on Java programming language. Some of these CubeSats had actuators, e.g., STRaND-1 had three reaction wheels mounted in an orthogonal configuration to provide three-axis control, whereas PhoneSat 2.0 beta had magnetorquers to de-tumble the spacecraft. Taking into account a CubeSat that runs Android operating system (based on a smartphone), it is natural to

evaluate the attitude and orbit control subsystem (AOCS) based on Java. Moreover, such AOCS can be designed with success by linear control theory, if the satellite has slow angular motions and small attitude maneuver. However, the linearized models are not able to represent all the perturbations due to the effects of th e nonlinear terms present in the dynamics and in the actuators (e.g., saturation) which can damage the system's performance. Therefore, it is expected that nonlinear control techniques yield better performance than the linear control techniques, improving the AOCS pointing accuracy. One nonlinear candidate technique for the design of AOCS control law is the State-Dependent Riccati Equation (SDRE). SDRE provides an effective algorithm for synthesizing nonlinear feedback control by allowing nonlinearities in the system states while offering great design flexibility through statedependent weighting matrices. In this paper, we present a simulator and the investigation of a SDRE control law based on attitude errors given by quaternion error. The simulator is based on Java and related open-source software libraries (Hipparchus - linear algebra library, and Orekit - flight dynamics library), therefore, it can run on a variety of platforms - including an Android operating system in a CubeSat - and it has low cost. The Java open-source libraries were extended in order to solve the optimization problem that is the cornerstone of the SDRE method. Two control laws (a linear and a SDRE based) were simulated using a Monte Carlo perturbation model. The Nano satellite Constellation for Environmental Data Collection (CONASAT), a CubeSat from the Brazilian National Institute for Space Research (INPE), provided the parameters for the simulations. The initial results of the simulations shown that the SDRE-based controller provides better performance.

Application of the SDRE Technique in the Satellite Attitude and Orbit Control System with Nonlinear Dynamics (ROMERO et al., 2018)

Abstract: The satellite attitude and orbit control subsystem (AOCS) can be designed with success by linear control theory if the satellite has slow angular motions and small attitude maneuver. However, for large and fast maneuvers, the linearized models are not able to represent all the perturbations due to the effects of the nonlinear terms present in the dynamics and in the actuators (e.g., saturation) which can damage the system's performance. Therefore, in such cases, it is expected that nonlinear control techniques yield better performance than the linear control techniques, improving the AOCS pointing accuracy without requiring a new set of sensors and actuators. One candidate technique for the design of AOCS control law under a large and fast maneuver is the State-Dependent Riccati Equation (SDRE). SDRE provides an effective algorithm for synthesizing nonlinear feedback control by allowing nonlinearities in the system states while offering great design flexibility

through state-dependent weighting matrices. The Brazilian National Institute for Space Research (INPE, in Portuguese) was demanded by the Brazilian government to build remote-sensing satellites, such as the Amazonia-1 mission. In such missions, the AOCS must stabilize the satellite in three-axes so that the optical payload can point to the desired target. Currently, the control laws of AOCS are designed and analyzed using linear control techniques in commercial software. In this paper, we discuss whether the application of the SDRE technique in the AOCS design can yield gains in the missions developed by INPE. Moreover, we report a proof of concept of an open-source satellite simulator built to analyze control laws based on SDRE. This satellite simulator is implemented in Java using Hipparchus (linear algebra library; which was extended in order to support the SDRE technique) and Orekit (flight dynamics framework).

C.2 Control Engineering

C.2.1 2019

Optimal Factorization of the State-Dependent Riccati Equation Technique in a Satellite Attitude and Orbit Control System (ROMERO; SOUZA, 2019b)

Abstract: The satellite attitude and orbit control system (AOCS) can be designed with success by linear control theory if the satellite has slow angular motions and small attitude maneuver. However, for large and fast maneuvers, the linearized models are not able to represent all the perturbations due to the effects of the nonlinear terms present in the dynamics and in the actuators (e.g., saturation). Therefore, in such cases, it is expected that nonlinear control techniques yield better performance than the linear control techniques. One candidate technique for the design of AOCS control law under a large maneuver is the State-Dependent Riccati Equation (SDRE). SDRE entails factorization (that is, parameterization) of the nonlinear dynamics into the state vector and the product of a matrix-valued function that depends on the state itself. In doing so, SDRE brings the nonlinear system to a (nonunique) linear structure having state-dependent coefficient (SDC) matrices and then it minimizes a nonlinear performance index having a quadratic-like structure. The nonuniqueness of the SDC matrices creates extra degrees of freedom, which can be used to enhance controller performance, however, it poses challenges since not all SDC matrices fulfill the SDRE requirements. Moreover, regarding the satellite's kinematics, there is a plethora of options, e.g., Euler angles, Gibbs vector, modified Rodrigues parameters (MRPs), quaternions, etc. Once again, some kinematics

formulation of the AOCS do not fulfill the SDRE requirements. In this paper, we evaluate the factorization options (SDC matrices) for the AOCS exploring the requirements of the SDRE technique. Considering a Brazilian National Institute for Space Research (INPE) typical mission, in which the AOCS must stabilize a satellite in three-axis, the application of the SDRE technique equipped with the optimal SDC matrices can yield gains in the missions. The initial results show that MRPs for kinematics provides an optimal SDC matrix.

Application of a New Optimal Factorization of the SDRE Method in the Satellite Attitude and Orbit Control System Design with Nonlinear Dynamics (ROMERO; SOUZA, 2019a)

Abstract: The satellite Attitude and Orbit Control System (AOCS) can be designed with success by linear control theory if the satellite has slow angular motions and small attitude maneuver. However, for large and fast maneuvers, the linearized models are not able to represent all the perturbations due to the effects of the nonlinear terms present in the dynamics and in the actuators. Therefore, in such cases, it is expected that nonlinear control techniques yield better performance than the linear control techniques. One candidate technique for the design of AOCS control law under a large maneuver is the State-Dependent Riccati Equation (SDRE). SDRE entails factorization (that is, parameterization) of the nonlinear dynamics into the state vector and the product of a matrix-valued function that depends on the state itself. In doing so, SDRE brings the nonlinear system to a (not unique) linear structure having State-Dependent Coefficient (SDC) matrices and then it minimizes a nonlinear performance index having a quadratic-like structure. The non uniqueness of the SDC matrices creates extra degrees of freedom, which can be used to enhance controller performance; however, it poses challenges since not all SDC matrices fulfill the SDRE requirements. Moreover, regarding the satellite's kinematics, there is a plethora of options, e.g., Euler angles, Gibbs vector, Modified Rodrigues Parameters (MRPs), quaternions, etc. Once again, some kinematics formulations of the AOCS do not fulfill the SDRE requirements. In this paper, we evaluate the factorization options of SDC matrices for the AOCS exploring the requirements of the SDRE technique. Considering a Brazilian National Institute for Space Research (INPE) typical mission, in which the AOCS must stabilize a satellite in three-axis, the application of the SDRE technique equipped with the optimal SDC matrices can yield gains in the missions. The initial results show that MRPs for kinematics provides an optimal SDC matrix.

Satellite Controller System Based on Reaction Wheels Using the State-

Dependent Riccati Equation (SDRE) on Java: Vol. 2 (ROMERO; SOUZA, 2019c)

Abstract: Complex space missions involving large angle maneuvers and fast attitude control require nonlinear control methods to design the Satellite Controller System (SCS) in order to satisfy robustness and performance requirements. One candidate method for a nonlinear SCS control law is the State-Dependent Riccati Equation (SDRE). SDRE provides an effective algorithm for synthesizing nonlinear feedback control by allowing nonlinearities in the system states while offering great design flexibility through state-dependent weighting matrices. In that context, analysis by simulation of nonlinear control methods can save money and time. Although, commercial 3D simulators exist that can accommodate various satellites components including the controllers, in this paper, we present a 3D simulator and the investigation of a SDRE control law performance by simulations. The simulator is implemented based on Java and related open-source software libraries (Hipparchus - linear algebra library, and Orekit - flight dynamics library), therefore, it can run in a variety of platforms and it has low cost. These open-source libraries were extended in order to solve the optimization problem that is the cornerstone of the SDRE method, a major contribution of the simulator. The simulator is evaluated taking into account a typical mission of the Brazilian National Institute for Space Research (INPE), in which the SCS must stabilize a satellite in three-axis using reaction wheels so that the optical payload can point to the desired target. Two SCS control laws (a linear and a SDRE based) were simulated for an attitude maneuver in the launch and early orbit phase (LEOP), the upside-down maneuver. The results of simulations shown that SDRE-based controller provides better performance.