POLITECNICO DI TORINO

Master degree course in Aerospace Engineering

Master Degree Thesis

Control of Rendez-vous and Docking phases of two CubeSats, by means of artificial intelligence algorithms.

Supervisors:
Prof. Sabrina Corpino
Prof. Fabrizio Stesina

Candidates
Agnese D'Acunto
Matricola: 233417

Anno accademico 2018-2019

Summary

On-orbit proximity operations require advanced guidance, navigation and control systems to obtain high pointing accuracy and very precise manoeuvring. The design of an attitude and orbit control is of primary importance in such delicate operations, since a spacecraft is subjected to many disturbance forces and torques, due to many sources, that lead to orbit variation (e.g. decay) and undesired angular accelerations. For missions with proximity operation, innovative Attitude Determination and Control Subsystem (ADCS) and a Guidance, Navigation and Control Subsystem (G, N &S) are of paramount importance. Orbit and attitude control is carried out thanks to the proper interactions of several elements, such as sensors, controller and actuators. The sensors determine the position and orientation of the body in space, but also its translational speed and rotational velocity around its centre of mass with respect to local and inertial reference systems. The control system is governed by sophisticated algorithms that aim at optimize the number and quality of manoeuvres and fuel, determining the commands to actuators, in order to correct the spacecraft trajectory and orientation and match the desired values. The complexity of this system increases in the small satellite field, where the technology is not completely mature and these kind of missions are never been performed.

The aim of this thesis is the definition of a control strategy for proximity manoeuvres between two 6U CubeSats. In particular, it was analysed the control of the last phases of the rendez-vous, from the last hold point up to the mating, the rendez-vous and docking operations, through the definition of the mathematical model of the problem (i.e. relative translational and rotational dynamics and kinematics, environmental effects on the spacecraft movements), its implementation and simulation in Matlab®/Simulink®, and the assessment of the results, considering different simulation conditions (e.g. final accuracy, execution time, initial position and orientation). The choice of the control strategy to adopt in this work fell on machine learning algorithms; more precisely, a direct artificial neural network control was developed, in order to explore the capability . Artificial neural networks are a framework of many different machine learning algorithms and they

are a mathematical and simplified representation of a biological neural system. Their main characteristic is the learning capability from a training input/output set or a mathematical function: the neural network, once learnt its task, can reproduce the desired output, even if the inputs change. The neural network was trained with an input/output set of values acquired by the simulation of the same model in which the control was exerted through a Proportional-Integral-Derivative controller.

The ability of neural network algorithms to adapt to different and sometimes unpredictable conditions could be very useful in spacecraft control problems, and has been studied in this work. In particular, a basic comparison between two controller, one based on neural networks and one Proportional Integrative and Derivative controller, was made demonstrating the more adaptability and efficiency of the neural network method over PID control, when external conditions change.

Acknowledgements

Contents

Summary III

Acknowledgements V

1 Introduction 1
- 1.1 Mission description . . . 2
  - 1.1.1 CubSat . . . 2
- 1.2 Space Environment . . . 4
- 1.3 Attitude and orbit Control . . . 6
- 1.4 Thesis Organization . . . 7

2 Mathematical model 9
- 2.1 Reference frames . . . 9
- 2.2 Satellite translational dynamics and kinematics . . . 10
  - 2.2.1 Equation of motion of a body in an inertial frame . . . 10
  - 2.2.2 Keplerian orbit and orbital parameters . . . 13
  - 2.2.3 Euler-Hill equations . . . 15
- 2.3 Satellite rotational dynamics and kinematics . . . 18
  - 2.3.1 Rotational kinetics of a rigid body . . . 18
  - 2.3.2 Quaternions . . . 18
  - 2.3.3 Rotational dynamics of a rigid body: Euler's moment equation 20
- 2.4 Perturbing Forces and Torques . . . 22
  - 2.4.1 Aerodynamic Drag . . . 22
  - 2.4.2 Solar radiation pressure . . . 22
  - 2.4.3 Gravity gradient . . . 23
  - 2.4.4 Earth magnetic field . . . 24

3 Control Theory 25
- 3.1 Dynamic system control . . . 25
  - 3.1.1 Dynamical system . . . 25

3.2 Dynamical system control . . . 26
3.3 Neural control . . . 27
3.3.1 Neural Network . . . 27
3.3.2 Training the neural network . . . 29
3.3.3 The backpropagation algorithm . . . 30
3.3.4 Direct neural control . . . 34
3.3.5 Feed-forward neural network and Levenberg-Marquardt algorithm . . . 35
3.4 PID control . . . 36

4 Simulation model 39
4.1 Satellite Dynamics Model . . . 42
4.1.1 Translational Dynamics model . . . 42
4.1.2 Rotational dynamics model . . . 44
4.2 PD-controller . . . 47
4.3 Neural Network Controller . . . 50
4.3.1 Regression and Mean Squared Error . . . 53
4.4 Training the Neural Network with PD-control . . . 53
4.4.1 Translational motion NN control . . . 53
4.4.2 Rotational motion NN control . . . 59

5 Simulation results 63
5.1 Translational motion control: nominal conditions . . . 65
5.2 Translational motion control: changing initial conditions . . . 69
5.2.1 Simulation 1 . . . 69
5.2.2 Simulation 2 . . . 72
5.2.3 Simulation 3 . . . 75
5.3 Attitude control: nominal conditions . . . 78
5.4 Attitude control: changing initial conditions . . . 82
5.4.1 Simulation 1 . . . 82
5.4.2 Simulation 2 . . . 84
5.5 Attitude control: adding sensor . . . 86
5.5.1 Simulation 1 . . . 86
5.5.2 Simulation 2 . . . 89
5.6 Discussion . . . 91

6 Conclusion 93

Bibliography 95

List of Tables

5.1 Chaser nominal properties. 63
5.2 Orbital parameters. 63
5.3 Trajectory PID-controller gains. 64
5.4 Attitude PID-controller gains. 64
5.5 Initial conditions. 65
5.6 Initial conditions. Simulation 1 69
5.7 Initial conditions. Simulation 2 72
5.8 Initial conditions. Simulation 3 75
5.9 Initial conditions. 78
5.10 Attitude PID-controller gains. 78
5.11 Initial conditions. 82
5.12 Initial conditions. 84
5.13 Initial conditions.Simulation 1 86
5.14 Initial conditions. Simulation 2 89

List of Figures

1.1 Rendez-vous and docking phases. s_0, s_1, s_2, s_3 and s_4 are key points of the last approaching phases. . . . 3
1.2 CubeSat. . . . 3
1.3 6U CubSat. . . . 4
1.4 Low-Earth Orbit. . . . 5
2.1 ECEF reference frame. . . . 9
2.2 Spacecraft Local Orbital frame. . . . 10
2.3 Spacecraft Body Fixed Reference frame. . . . 11
2.4 Two-body problem. . . . 11
2.5 Conic sections. . . . 13
2.6 Types of orbits. . . . 15
2.7 Orbital parameters. . . . 15
2.8 Relative position of Chaser and Target satellites. . . . 16
2.9 Non-inertial coordinate reference frame. . . . 17
2.10 Euler angles. . . . 19
2.11 Earth magnetic field. . . . 24
3.1 Open-loop control 27
3.2 Closed-loop control 27
3.3 Scheme of the artificial neuron. . . . 28
3.4 Two-layer neural network. . . . 29
3.5 Two-layer neural network. . . . 32
3.6 Direct neural control scheme. . . . 34
3.7 Multi-layer feed-forward neural network. . . . 35
3.8 PID control scheme. . . . 37
4.1 First level Simulink scheme. . . . 40
4.3 Relative Translational dynamics block. . . . 43
4.4 Chaser Rotational dynamics block. . . . 45
4.5 Kinetics. . . . 46
4.6 Kinematics. . . . 47
4.7 PID-controller for translational motion. . . . 48
4.8 PID-controller for rotational motion. . . . 49

4.9 Neural Network controller. 50
4.10 Sigmoid activation function. 52
4.11 Regression plot after every training session. 55
4.12 Regression plot after every training session. 56
4.13 From the first up on the left, clockwise: Training regression plot, Validation regression plot, Test regression plot, All data regression plot. 57
4.14 Regression plot after every training session. 60
4.15 From the first up on the left, clockwise: Training regression plot, Validation regression plot, Test regression plot, All data regression plot. 61
5.1 Neural Network architecture for trajectory control. 65
5.2 Chaser-Target relative position, nominal conditions. 66
5.3 Chaser-Target relative velocity, nominal conditions. 67
5.4 Control Forces, nominal conditions. 68
5.5 Chaser-target relative position. 70
5.6 Chaser-target relative velocity. 70
5.7 Chaser-target relative position. 71
5.8 Chaser-target relative position. 73
5.9 Chaser-target relative velocity. 73
5.10 Chaser-target relative position. 74
5.11 Chaser-target relative position. 75
5.12 Chaser-target relative velocity. 76
5.13 Chaser-target relative position. 76
5.14 5 hidden-layers Neural Network. 78
5.15 Chaser Attitude angles, nominal conditions. 79
5.16 Chaser Attitude rate, nominal conditions. 80
5.17 Control torques, nominal conditions. 81
5.18 Chaser attitude angles. 82
5.19 Chaser attitude rate. 83
5.20 Control Torques. 83
5.21 Chaser attitude angles. 84
5.22 Chaser attitude rate. 85
5.23 Control Torques. 85
5.24 Chaser attitude angles. 86
5.25 Chaser attitude rate. 87
5.26 Control Torques. 87
5.27 Chaser attitude angles. 89
5.28 Chaser attitude rate. 90
5.29 Control Torques. 90

Chapter 1

Introduction

The purpose of this thesis is to implement a control strategy of rendez-vous and docking operations between two spacecrafts, belonging to the class of small satellites. Proximity operations between two vehicles are extremely delicate from the point of view of relative motion control, which implies the development and design of extremely accurate control systems and sensors. The control systems are fundamental, since an object in orbit is subjected to various disturbing forces and torques, which tend to modify its orbital path and induce undesired accelerations. Attitude and orbit control systems are complex tools, consisting of many hardware and software, designed to fulfil the specific mission requirements. The main objective of this elaborate is the development of a control algorithm, which drives the system control unit, through its implementation and simulation in Matlab®/Simulink® . The general theory of automatic controls of dynamic systems was analysed, leading to a better understanding of the control strategies taken into account; the chosen algorithm belongs to the family of algorithms based on machine learning, a vast field of methods and applications, developed trying to emulate, at a considerably more basic level, the biological nervous system. A control algorithm of translational and rotational motion for the proximity manoeuvre, based on neural networks, was then carried out, a tool capable of learning a task to be performed, based on input and output data. The set of input/output data to be supplied to the neural network for its training has been obtained through the implementation of a controller based on a Proportional-Integrative-Derivative algorithm.
The mathematical model that describes the motion of a satellite in orbit is based on the dynamics and kinematics of a rigid body. Therefore, the translational and rotational dynamics of a rigid body has been presented, first at theoretical level, then at simulation model level: in terms of Euler-Hill equations, to describe the relative translational motion between the two spacecraft, and of moment Euler

equation to describe the evolution over time of the rotational dynamics and kinematics of the Chaser vehicle with respect to the Target vehicle.
The simulation model includes the mathematical modelling of the disturbance forces and torques affecting the vehicles during the manoeuvre. Since the mission considered takes place in low-Earth orbit, the elements to be considered are: the atmospheric drag, the solar radiation pressure, the gravity gradient and the Earth magnetic fields.
By analysing and studying the results reported in the last chapter, we want to highlight the greater adaptability of neural networks to the change of external conditions and simulation time with respect to the PID control.

1.1 Mission description

The mission considered in this work consists in performing proximity operations between two 6U CubSats in low-Earth orbit. The manoeuvre sequence starts with the deployment, from the International Space Station (ISS), of the two CubSats docked to each other. Then, the CubSats separate: one of them, the Chaser, move away from the Target up to a distance of 10 km. From that distance, it starts the approach phase towards the Target and come to a relative distance of about 2 km on one of the main axis of LVLH frame (R-bar, V-bar or H-bar). The main focus of this dissertation, although, will be the orbit and attitude control of the last phases: the rendez-vous and docking phases, as shown in figure 1.1, that takes place from a relative distance of approximatively 50-100 m. For all the mission phases, the Target will be considered a cooperative target, that is position and attitude will be considered fixed. The reference orbit, where the CubeSats will be deployed and the operations will be carried out, is close to the one of the ISS. The orbit will be considered circular to simplify the equations of motion, with a radius od 400 km and an inclination of 51.64 degress. Nevertheless, the mathematical and simulation model that will be further developed should work for any low-Earth orbits, and this kind of mission should be performed without involving the ISS, but the CubSats could be launched from the ground with any suitable launch vehicle.

1.1.1 CubSat

The CubeSat standard was created by California Polytechnic State University, San Luis Obispo and Stanford University's Space Systems Development Lab in 1999 to create an affordable spacecraft for educational purposes. It facilitates frequent and affordable access to space with launch opportunities available on most launch vehicles. CubeSats are research spacecrafts, called nanosatellites, designed in terms of standard dimensions called units (U), where a 1U CubeSat is a 10 cm cube with a mass of up to approximately 1.5 kg, as depicted in figure 1.2. The present

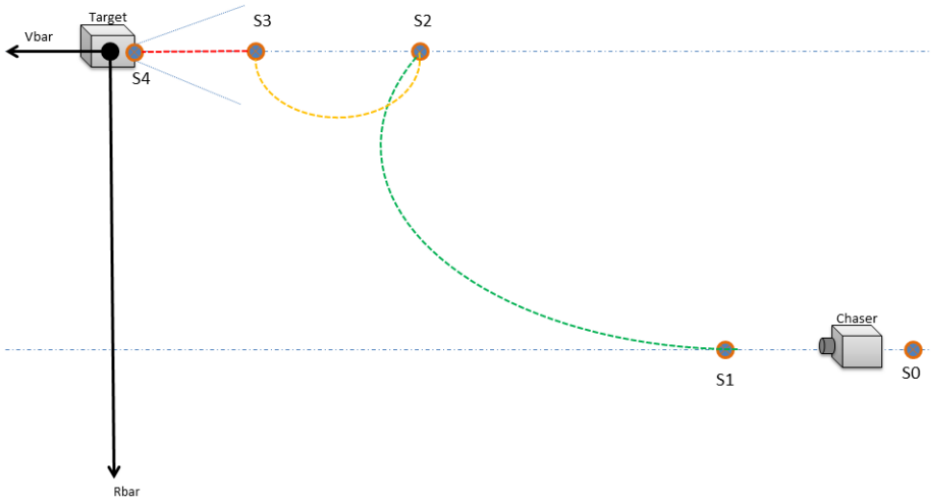


Figure 1.1: Rendez-vous and docking phases. s_0, s_1, s_2, s_3 and s_4 are key points of the last approaching phases.

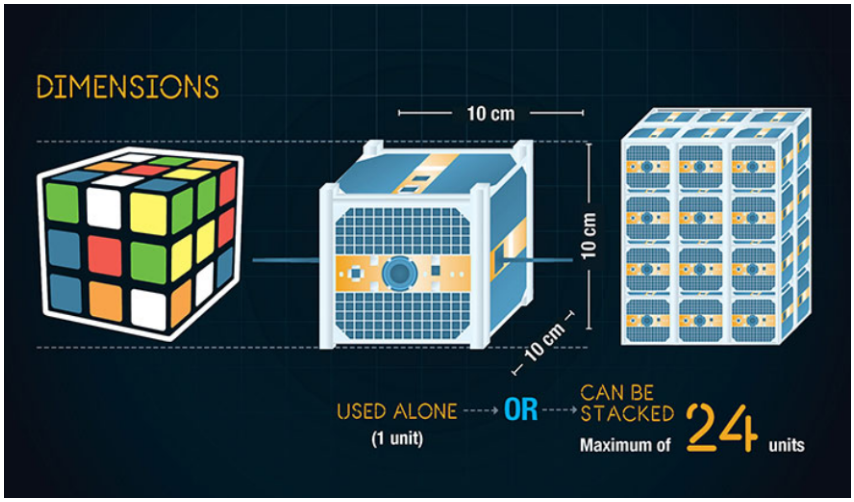


Figure 1.2: CubeSat.

work deals with 6U CubeSat, whose graphic representations are shown in figure 1.3; it is a cuboid with size of 30x20x10 cm. CubSats electronic components and

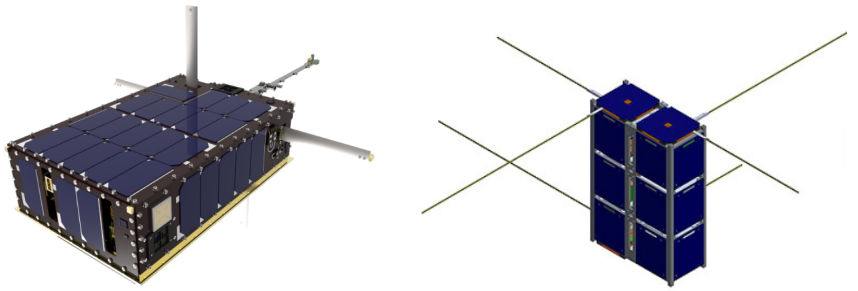


Figure 1.3: 6U CubSat.

structures are often realized by use of Commercially available Off-The-Shelf products (COTS[1]), reducing developing times and costs, being already available on the market and not needing an ad-hoc design. They are intended to be deployed from the International Space Station or launched as secondary payload from launchers in Low Earth Orbit (LEO), to perform many tasks, as communication, scientific, exploration and observation missions. Due to their affordability and versatility, CubSats are largely exploited also for commercial purposes. [1]

1.2 Space Environment

The space environment of interest in this thesis is the near-Earth environment, since the spacecrafts are meant to be launched in LEO, which altitude range varies from 200 to 1000 Km, as shown in figure 1.4. The spacecraft on-orbit life is interested by the interaction with many factors that tend to influence its trajectories and operating life of its components. Plasmas, high-energy charged particles, neutral gases, x-rays, ultraviolet (UV) radiation, meteoroids, and orbital debris cause degradation of materials, thermal changes, contamination, excitation, spacecraft glow, charging, radiation damage, and induced background interference. For control purposes, although, we take in consideration only the environment elements that influence the orbital trajectory and spacecraft attitude, such as disturbance forces and torques. In low-Earth orbits, the most important disturbance sources

[1]Commercially available off-the-shelf products are packaged solutions which are then adapted to satisfy the needs of the purchasing organization

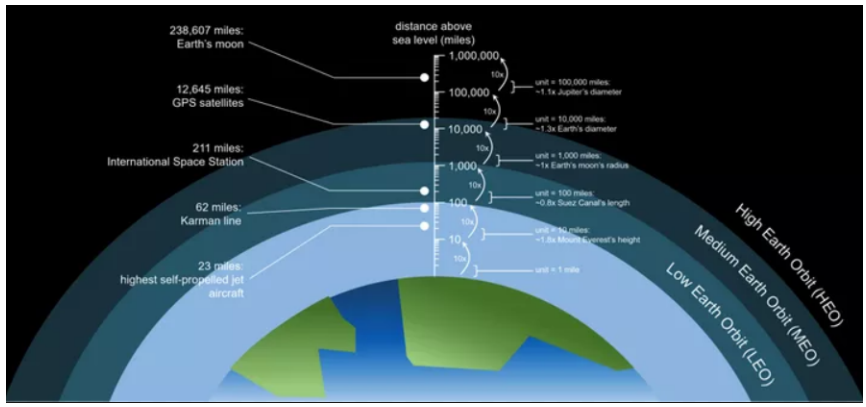


Figure 1.4: Low-Earth Orbit.

are: the atmospheric drag, the gravity-gradient, the solar radiation pressure and the Earth's magnetic field.
The atmospheric drag is due to the fact that some atmospheric particles are still present at LEO altitudes and, even though their small amount, they generate a small drag on spacecraft, which can lead to orbit decay. The same drag force will produce a disturbance torque on the spacecraft due to any offset existing between the aerodynamic centre of pressure and the centre of mass.
Planetary gravitational fields decrease with distance r from the centre of the planet according to the Newtonian law. Thus, an object in orbit will experience a stronger attraction on its side facing the planet than on the side facing outer space. This imbalance, the gravity-gradient, generates a torque acting on the spacecraft.
The solar radiation pressure is the force exerted by solar radiation particles on the surface of a spacecraft when it's in sunlight; it may, as well, generate a torque.
The Earth magnetic field, coupled with the dipole moment of the spacecraft, generates a disturbance torque on it. Nevertheless, it is common to take advantage of the planetary magnetic field as a control torque to counter the effects of other disturbances. [6]

1.3 Attitude and orbit Control

Spacecraft attitude and orbit control are executed by two different subsystems: **the Attitude Determination and Control System** (ADCS) and **the Guidance, Navigation and Control System** (GN&C). The motion of a spacecraft is defined by its position and velocity, that describe the translational motion of the centre of mass of the spacecraft, and the attitude and attitude rate, defining the rotational motion of the body of the spacecraft about the centre of mass. Orbit and attitude systems are constituted by the hardware, software and processes used to measure and control the spacecraft trajectory and orientation during all phases of a mission.

GN&S maintains and changes a vehicle's position and velocity, using three functions: the **navigation** represents the subsystem sensors area, that determines the current position and velocity of the spacecraft; **guidance** is the controller function and **control** uses navigation and guidance informations to change spacecraft position and velocity, by means of actuators. The guidance and navigation function measures and maintains the position of the spacecraft's centre of mass. In some applications (for instance to rendezvous and dock with other vehicles or to deploy other vehicles) propulsive manoeuvres and navigation and guidance of the spacecraft require an on-board Guidance Navigation and Control Subsystem. Given the desired velocity, guidance acts as a controller. It compares this velocity to the inertial velocity from the inertial navigation system, which acts as a sensor, and sends command to the engines or actuators, which apply a force to the spacecraft. The principal actuators used in these circumstances are:

- cold gas systems;
- monopropellant hydrazine;
- bi-propellant nitrogen tetroxide/monomethylhydrazine;
- solid propellant;

ADCS stabilizes the vehicle and orients it in desired directions during the mission despite the external disturbance torques acting on it, therefore it is tightly coupled to the propulsion and navigation subsystems and requires the use of sensors to measure the current spacecraft attitude. Attitude dynamics analysis is complex due to three factors: the attitude information is inherently vectorial, requiring three coordinates for its complete specification; it deals with rotating, hence non-inertial, frames; rotations are inherently order dependent in their description. ADCS is typically a major vehicle subsystem, with requirements that quite often drive the overall spacecraft design. Its components tend to be relatively massive, power consuming, and demanding for specific orientation, alignment tolerance, field of view,

structural frequency response, and structural damping. A body in space is subject to small but persistent disturbance torques from a variety of sources. The ADCS is designed within the spacecraft in order to resist these torques.
Attitude control systems can be classified in **passive** and **active** control systems and in **closed-loop** and **open-loop** control systems. The discriminating concept between open-loop and closed-loop control architectures is the presence of a feedback line. In closed-loop control, unlike in open-loop control, the measurement, by means of sensors, of the spacecraft actual attitude is compared with the desired attitude. Taking into account the error value, control torques are generated by means of actuators, to restore/acquire the desired attitude. The correction process is therefore continuous. Passive stabilization techniques take advantage of the disturbance torques to control the spacecraft, choosing a design to emphasize one and mitigate the others, and/or uses the mass characteristics of the spacecraft itself, depending on the mission. In active attitude control, the spacecraft attitude is measured and compared with a desired value. The error signal is then used to determine a corrective action (control torque) to generate a manoeuvre by means of the onboard actuators. Attitude actuators can be based on:

- thrust: reaction control jets;
- controlling momentum: reaction wheels, momentum wheels, control moment gyros;
- magnetism: magnetic torquers or magnetic coils.

In proximity manoeuvres, the most important requirement of ADCS is the determination and control accuracy, i.e. how well the spacecraft orientation with respect to an absolute reference is known and how well the vehicle attitude can be controlled with respect to a commanded direction. [11]

1.4 Thesis Organization

The problem under examination is developed firstly mathematically, analysing the equation of motion of a rigid body, to describe the spacecrafts relative motion, in the **Mathematical Model Chapter**. In this Chapter, the Hill-equations and Euler moment equations, applied to the specific case, are reported. Then, the general Control Theory is briefly delineated before introducing the control algorithm taking in consideration. In the **Control Theory Chapter**, Artificial Neural Network and PID control are introduced. In the **Simulation Model Chapter** all the processes and equations that lead to the definition of the Simulink® model are described in detail. The last Chapter, **Simulation Results**, contains all simulation graphs and data collected.

Chapter 2

Mathematical model

2.1 Reference frames

To approach the problem of a satellite motion in orbit, several reference frame needs to be defined. [4]

1. ECEF (Earth Centered Earth Fixed) frame, shown in figure 2.1, is considered inertial for what concerns missions in Earth orbit. Its origin, O, is in the centre of the Earth, x_E in the equatorial plane, pointing toward the mean of the vernal equinox; z_E is normal to the equatorial plane and pointing north, y_E is in the equatorial plane, such that $z_E = x_E \times y_E$.

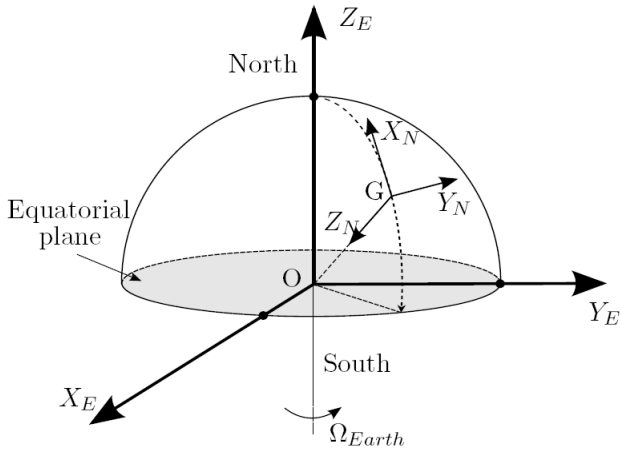


Figure 2.1: ECEF reference frame.

2. Local Orbital frame, in figure 2.2, has its origin in the Centre of Mass (CoM) of the spacecraft; x_{orb} is defined such that $x_{orb} = y_{orb} \times z_{orb}$ (x_{orb} is in the direction of the orbital velocity vector but not necessarily aligned with it), y_{orb} is in the opposite direction of the angular momentum vector of the orbit and z_{orb} is radial from the spacecraft CoM to the centre of the Earth.

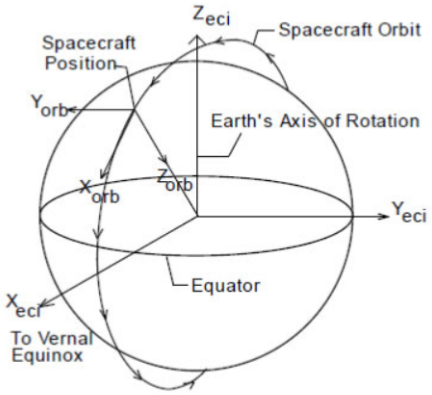


Figure 2.2: Spacecraft Local Orbital frame.

3. The Body Fixed Reference frame, in figure 2.3 is a moving coordinate frame, fixed on the spacecraft. The origin, O_S is located in the satellite centre of mass, the directions of the axes are along the main inertia axes of the spacecraft and $z_{Body} = x_{Body} \times y_{Body}$ forming a right handed system.

2.2 Satellite translational dynamics and kinematics

2.2.1 Equation of motion of a body in an inertial frame

The equation of motion of a satellite in space is derived by solving the two-body problem, which considers two rigid point masses acted upon only by the mutual force of gravity between them, as shown in figure 2.4. Their centre of mass positions is defined in relation to an inertial reference frame XYZ. Both the bodies are mutually subjected by the gravitational force of the other: F_{12} is the force exerted on m_1 by m_2 and F_{21} is the force exerted on m_2 by m_1. The position vector R_G of the centre of mass of the whole system, shown in figure 2.4, is defined by:

$$\mathbf{R}_G = \frac{m_1\mathbf{R}_1 + m_2\mathbf{R_2}}{m_1 + m_2}$$

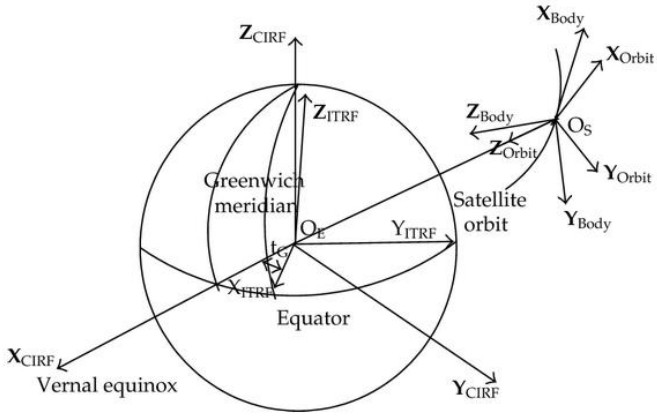


Figure 2.3: Spacecraft Body Fixed Reference frame.

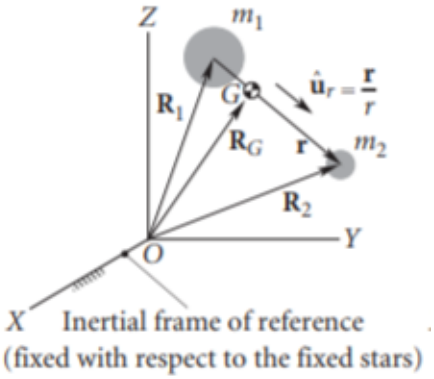


Figure 2.4: Two-body problem.

The velocity and acceleration of the system centre of mass, relative to the inertial reference frame, are the derivative and the second derivative of R_G, respectively:

$$\mathbf{v}_G = \dot{\mathbf{R}}_G = \frac{m_1\dot{\mathbf{R}}_1 + m_2\dot{\mathbf{R}}_2}{m_1 + m_2}$$

$$\mathbf{a}_G = \ddot{\mathbf{R}}_G = \frac{m_1\ddot{\mathbf{R}}_1 + m_2\ddot{\mathbf{R}}_2}{m_1 + m_2}$$

As shown in figure 2.4, $\mathbf{r}$ is the position vector of m_2 relative to m_1:

$$\mathbf{r} = \mathbf{R_2} - \mathbf{R_1}$$

$\hat{\mathbf{u}}_r$ is the unit vector pointing from m_1 towards m_2:

$$\hat{\mathbf{u}}_r = \frac{\mathbf{r}}{r}$$

where $r = ||\mathbf{r}||$ is the module of the position vector. The force of gravitational attraction, which acts along the line joining the centres of mass of two bodies, is:

$$\mathbf{F}_{21} = \frac{Gm_1m_2}{r^2}(-\hat{\mathbf{u}}_r) = -\frac{Gm_1m_2}{r^2}\hat{\mathbf{u}}_r$$

the minus sign of the unit vector is due to the fact that the force vector is directed from m_2 to m_1; G is the universal gravitational constant. Taking into consideration Newton's second law of motion, $\mathbf{F}_{21} = m_2\ddot{\mathbf{R}}_2$, we obtain:

$$-\frac{Gm_1m_2}{r^2}\hat{\mathbf{u}}_r = m_2\ddot{\mathbf{R}}_2 \tag{2.1}$$

For m_1, we have, as well:

$$\frac{Gm_1m_1}{r^2}\hat{\mathbf{u}}_r = m_1\ddot{\mathbf{R}}_1 \tag{2.2}$$

To obtain the equation of relative motion between the two bodies, we multiply the equation 2.1 by m_1 and the equation 2.2 by m_2:

$$-\frac{Gm_1^2m_2}{r^2}\hat{\mathbf{u}}_r = m_1m_2\ddot{\mathbf{R}}_2$$

$$\frac{Gm_1m_2^2}{r^2}\hat{\mathbf{u}}_r = m_1m_2\ddot{\mathbf{R}}_1$$

Subtracting the second equation from the first one:

$$m_1m_2(\ddot{\mathbf{R}}_2 - \ddot{\mathbf{R}}_1) = -\frac{Gm_1m_2}{r^2}(m_1 + m_2)\hat{\mathbf{u}}_r$$

$$\ddot{\mathbf{r}} = -\frac{G(m_1 + m_2)}{r^2}\hat{\mathbf{u}}_r$$

We define the gravitational parameter μ as:

$$\mu = G(m_1 + m_2)$$

So, the equation of motion becomes:

$$\ddot{\mathbf{r}} = -\frac{\mu}{r^3}\mathbf{r} \tag{2.3}$$

This represents the second order differential equation that governs the motion of the body m_2 relative to m_1. [5]

2.2.2 Keplerian orbit and orbital parameters

According to Kepler's laws of planetary motion, the only possible paths for an orbiting object on the two-body problem are a family of curves called *conic sections*, which includes circle, ellipse, parabola and hyperbola, shown in figure 2.5. To obtain a form of the two-body problem equation of motion that can be integrated, we have to cross multiply it with $\mathbf{h}$, the specific angular momentum:

$$\mathbf{h} = \mathbf{r} \times \mathbf{v}$$

where $\mathbf{r}$ is the position vector of the satellite and $\mathbf{v}$ is the velocity vector. Their cross product remains constant along the orbit. Thus, multiplying equation 2.3 with $\mathbf{h}$, we have:

$$\ddot{\mathbf{r}} \times \mathbf{h} = \frac{\mu}{r^3}(\mathbf{h} \times \mathbf{r}) \tag{2.4}$$

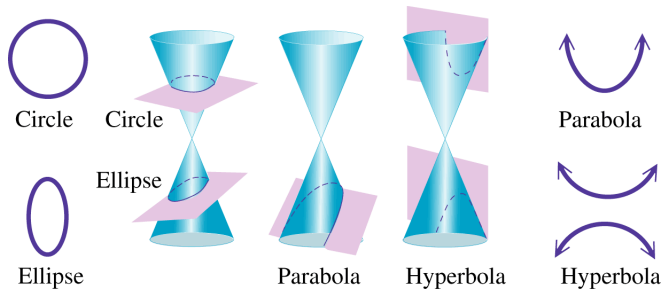


Figure 2.5: Conic sections.

$$\frac{\mu}{r^3}(\mathbf{h} \times \mathbf{r}) = \frac{\mu}{r^3}(\mathbf{r} \times \mathbf{v}) \times \mathbf{r} = \frac{\mu}{r^3}[\mathbf{v}(\mathbf{r} \cdot \mathbf{r}) - \mathbf{r}(\mathbf{r} \cdot \mathbf{v})] = \frac{\mu}{r}\mathbf{v} - \frac{\mu \dot{\mathbf{r}}}{r^2}\mathbf{r}$$

Note that μ times the unit vector is also:

$$\mu \frac{d}{dt}\left(\frac{\mathbf{r}}{r}\right) = \frac{\mu}{r}\mathbf{v} - \frac{\mu \dot{r}}{r^2}\mathbf{r}$$

Thus, we can rewrite the equation 2.4 as:

$$\frac{d}{dt}(\dot{\mathbf{r}} \times \mathbf{h}) = \mu \frac{d}{dt}\left(\frac{\mathbf{r}}{r}\right)$$

Integrating both sides:

$$\dot{\mathbf{r}} \times \mathbf{h} = \mu \frac{\mathbf{r}}{r} + \mathbf{B}$$

where $\mathbf{B}$ is the vector constant of integration. Now, we multiply both sides of the equation by $\mathbf{r}$:

$$\mathbf{r} \cdot \dot{\mathbf{r}} \times \mathbf{h} = \mathbf{r} \cdot \mu \frac{\mathbf{r}}{r} + \mathbf{r} \cdot \mathbf{B}$$

$$h^2 = \mu r + rB \cos \nu$$

where ν is the angle between the constant vector $\mathbf{B}$ and the radius vector $\mathbf{r}$. Solving for r, we obtain the trajectory equation expressed in polar coordinates:

$$r = \frac{h^2/\mu}{1 + (B/\mu) \cos \nu} \tag{2.5}$$

Equation 2.5 is mathematically identical, in form, to the general equation of a conic section written in polar coordinates, with the origin located at a focus and the polar angle, ν, the angle between $\mathbf{r}$ and the point on the conic nearest the focus:

$$r = \frac{p}{1 + e \cos \nu}$$

p is a geometrical constant of the conic called *semi-latus rectum* and the constant e is called the *eccentricity* and it determines which type of conic section the equation represents:

- $e = 0$, Circular orbit;
- $0 < e < 1$, Elliptical orbit;
- $e = 1$, Parabolic orbit;
- $e > 1$, Hyperbolic orbit.

The other parameters that define a keplerian orit are: *semi-major axis* (a) , the sum of the periapsis and apoapsis distances divided by two; the orbit *inclination* (i), vertical tilt of the ellipse with respect to the reference plane, measured at the ascending node; *longitude of the ascending node* (Ω), horizontally orients the ascending node of the ellipse, with respect to the reference frame's vernal point; *argument of periapsis* (ω), defines the orientation of the ellipse in the orbital plane, as an angle measured from the ascending node to the periapsis. From this description of orbital motion, it derives that the focus of the conic orbit must be located at the centre of the central body; the mechanical energy of a satellite does not change as the satellite moves along its orbit, so it must slow down where r increases and accelerate as r decreases; the orbital motion takes place ina plane which is fixed in inertial space; the specific angular momentum of a satellite, $\mathbf{h}$, remains constant. [2]

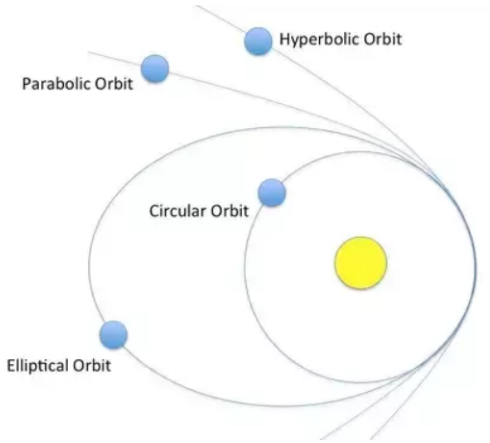


Figure 2.6: Types of orbits.

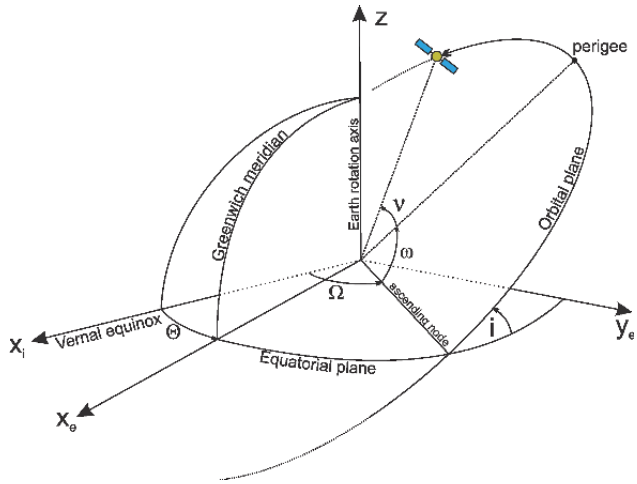


Figure 2.7: Orbital parameters.

2.2.3 Euler-Hill equations

In proximity manoeuvres as rendez-vous and docking operations between two satellites, the relative distance and velocity between them can be calculated defining a non-inertial coordinate system. The equations of motion of the Chaser satellite are developed with respect to this moving frame. The system origin is fixed in the centre of mass of the Target satellite and it moves with it. These equations,

the **Euler-Hill equations**, describe the relative motion of satellites with respect to the mating point. The derivation of these equations can be carried out by the linearisation of the equation of motion in orbit. In figure 2.8, the Target satellite A is positioned at a distance equal to $\mathbf{r_0}$ from the Earth and the position vector of the Chaser B is $\mathbf{r}$. The position vector of the Chaser vehicle relative to the target is $\delta\mathbf{r}$:

$$\delta\mathbf{r} = \mathbf{r_0} + \mathbf{r}$$

The relative position vector $\delta\mathbf{r}$ is very small with respect to the satellites distance

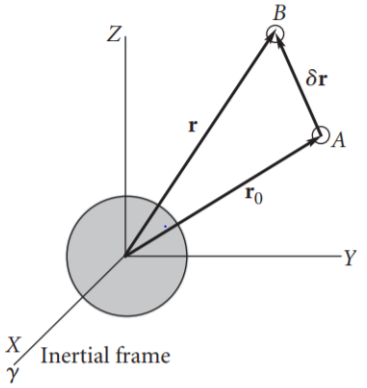


Figure 2.8: Relative position of Chaser and Target satellites.

from the origin of the inertial reference frame.
The equation of motion of the Chaser vehicle B is:

$$\ddot{\mathbf{r}} = -\mu\frac{\mathbf{r}}{r^3}$$

where $r = ||\mathbf{r}||$ and μ is the gravitational constant of the planet. The equation of motion of the chaser relative to the target is:

$$\delta\ddot{\mathbf{r}} = -\ddot{\mathbf{r}}_0 - \mu\frac{\mathbf{r_0} + \delta\mathbf{r}}{r^3}$$

Manipulating this equation, considering that $\delta\mathbf{r}$ is very small and neglecting all terms of higher order than one, the Chaser-Target relative equation of motion can be written as:

$$\delta\ddot{\mathbf{r}} = -\frac{\mu}{r_0^3}\left[\delta\mathbf{r} - \frac{\mathbf{3}}{\mathbf{r_0^2}}(\mathbf{r_0}\cdot\delta\mathbf{r})\mathbf{r_0}\right]$$

As said before, the reference frame will now be centred in the Target body centre of mass; the x axis lies along $\mathbf{r_0}$:

$$\hat{i} = \frac{\mathbf{r_0}}{r_0}$$

The y axis is in the direction of the local horizon, and the z axis is normal to the orbital plane of the Target, such that $\hat{\mathbf{k}} = \hat{\mathbf{i}} \times \hat{\mathbf{j}}$. The inertial angular velocity of the moving frame of reference is ω, and the inertial angular acceleration is $\dot{\omega}$. For the relative acceleration formula:

$$\ddot{\mathbf{r}} = \ddot{\mathbf{r}}_0 + \dot{\omega} \times \delta\mathbf{r} + \omega \times (\omega \times \delta\mathbf{r}) + \mathbf{2}\omega \times \delta\mathbf{v_{rel}} + \delta\mathbf{a_{rel}}$$

The relative position, velocity and acceleration in the moving reference frame are

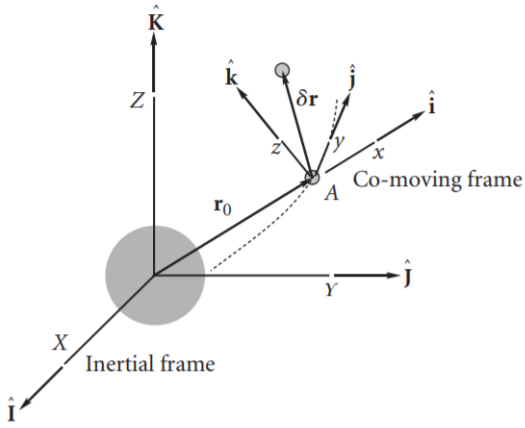


Figure 2.9: Non-inertial coordinate reference frame.

given by:

$$\delta\mathbf{r} = \delta x\hat{\mathbf{i}} + \delta y\hat{\mathbf{j}} + \delta z\hat{\mathbf{k}}$$

$$\delta\mathbf{v_{rel}} = \delta\dot{x}\hat{\mathbf{i}} + \delta\dot{y}\hat{\mathbf{j}} + \delta\dot{z}\hat{\mathbf{k}}$$

$$\delta\mathbf{a_{rel}} = \delta\ddot{x}\hat{\mathbf{i}} + \delta\ddot{y}\hat{\mathbf{j}} + \delta\ddot{z}\hat{\mathbf{k}}$$

Assuming that the Target orbit is circular, $\dot{\omega} = 0$ and the angular velocity can be written as:

$$\omega = n\hat{\mathbf{k}}$$

where n is the mean motion and it's constant. Thus we obtain:

$$\delta\ddot{\mathbf{r}} = (-n^2\delta x - 2n\delta\dot{y} + \delta\ddot{x})\hat{\mathbf{i}} + (-n^2\delta y + 2n\delta\dot{x} + \delta\ddot{y})\hat{\mathbf{j}} + \delta\ddot{z}\hat{\mathbf{k}}$$

The mean motion n is:

$$n = \frac{v}{r_0} = \frac{1}{r_0}\sqrt{\frac{\mu}{r_0}} = \sqrt{\frac{\mu}{r_0^3}}$$

Thus,

$$\frac{\mu}{r_0^3} = n^2$$

Substituting n in the equation, it becomes:

$$(\delta\ddot{x} - 3n^2\delta x - 2n\delta\dot{y})\hat{\mathbf{i}} + (\delta\ddot{y} + 2n\delta\dot{x})\hat{\mathbf{j}} + (\delta\ddot{z} + n^2\delta z)\hat{\mathbf{k}} = 0$$

So, the homogeneous Euler-Hill equations are:

$$\begin{aligned} \delta\ddot{x} - 3n^2\delta x - 2n\delta\dot{y} &= 0 \\ \delta\ddot{y} + 2n\delta\dot{x} &= 0 \\ \delta\ddot{z} + n^2\delta z &= 0 \end{aligned}$$

If disturbance and control accelerations are taken into account, changing notation, the previous equations become:

$$\begin{aligned} \ddot{x} - 2n\dot{y} - 3n^2 &= a_x \\ \ddot{y} + 2n\dot{x} &= a_y \\ \ddot{z} + n^2 z &= a_z \end{aligned}$$

where x, y, z are the relative distances between the two spacecrafts in the three directions, respectively; the a_i terms contain the disturbance and control accelerations. [5]

2.3 Satellite rotational dynamics and kinematics

2.3.1 Rotational kinetics of a rigid body

One way to define a spacecraft attitude is through the Euler angles, (ϕ, θ, ψ). They provide the orientation information by the angular difference between the axis of two reference frames, in this case, between the body frame (xyz) and the inertial frame (XYZ), like shown in figure 2.10. Thanks to the Euler angles, it is possible to define a transformation matrix between the body frame (xyx) and the LVLH frame (XYZ), $R_o^b(\phi, \theta, \psi)$:

$$\begin{bmatrix} \cos\phi\cos\psi - \sin\phi\sin\psi\cos\theta & -\cos\phi\sin\psi - \sin\phi\cos\theta\cos\psi & \sin\phi\sin\theta \\ \sin\phi\cos\psi + \cos\phi\cos\theta\sin\psi & -\sin\phi\sin\psi + \cos\phi\cos\theta\cos\psi & -\cos\phi\sin\theta \\ \sin\theta\sin\psi & \sin\theta\cos\psi & \cos\theta \end{bmatrix}$$

2.3.2 Quaternions

In attitude control problems, it is very common to prefer the use of unit quaternions to express rigid-body attitude, due to the fact that quaternions present no inherent

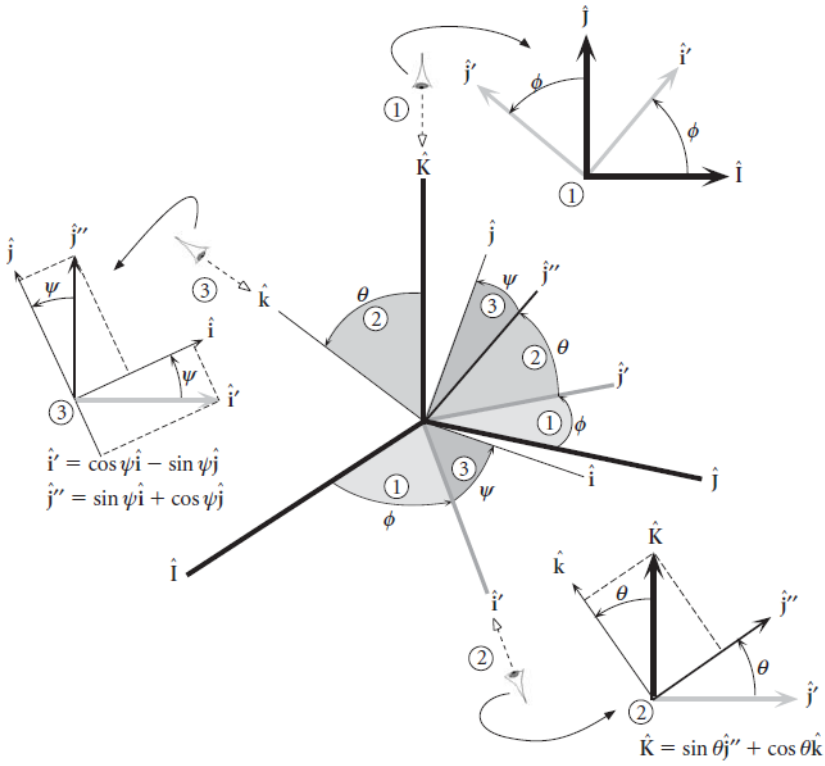


Figure 2.10: Euler angles.

geometric singularity and the linear equation to be integrated in time in order to determine their evolution as a function of angular velocity components is less computationally expensive than that derived for the Euler's angles. A quaternion is a 4×1 matrix which elements consists of a real scalar part s, which is the first element of the matrix, and a complex vector part $\vec{v}$.

$$q = \begin{bmatrix} s \\ \vec{v} \end{bmatrix} = \begin{bmatrix} s \\ v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}$$

The unit quaternion satisfies: $\mathbf{q}\mathbf{q}^T = 1$, which also means that:

$$q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1$$

Every unit quaternion express a rotation in $\mathbb{R}^3$. A spacecraft's kinematics equation gives the dependency of the time derivative of its relative orientation in space from the angular rate:

$$\dot{q} = \frac{1}{2}\omega \cdot \mathbf{q} \tag{2.6}$$

$$\begin{bmatrix} \dot{q_0} \\ \dot{q_1} \\ \dot{q_2} \\ \dot{q_3} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & -\omega_1 & -\omega_2 & -\omega_3 \\ \omega_1 & 0 & \omega_3 & -\omega_2 \\ \omega_2 & -\omega_3 & 0 & \omega_1 \\ \omega_3 & \omega_2 & -\omega_1 & 0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}$$

The transformation matrix between body frame and LVLH frame, written with quaternions, is:

$$R_o^b = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2 * (q_1\dot{q}_2 - q_3 \cdot q_0) & 2 \cdot (q_1 \cdot q_3 + q_2 \cdot q_0) \\ 2 \cdot (q_1 \cdot q_2 + q_3 \cdot q_0) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2 \cdot (q_2 \cdot q_3 - q_1 \cdot q_0) \\ 2 \cdot (q_1 \cdot q_3 - q_2 \cdot q_0) & 2 \cdot (q_2 \cdot q_3 + q_1 \cdot q_0) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}$$

The satellite's attitude can then be determined by integrating equation 2.6. To express the spacecraft attitude with Euler angles, the following equations must be used for the transformation:

$$\theta = asin[-2(q_1 q_3 - q_0 q_2)]$$

$$\phi = atan\left[\frac{2(q_0 q_1 + q_2 q_3)}{q_0^1 + q_1^2 - q_2^2 - q_3^2}\right]$$

$$\psi = atan\left[\frac{2(q_1 q_2 + q_0 q_3)}{q_0^2 - q_1^2 - q_2^2 + q_3^2}\right]$$

2.3.3 Rotational dynamics of a rigid body: Euler's moment equation

The Euler moment equation describes the rotation of a rigid body, using a rotating reference frame with its axes fixed to the body and parallel to the body's principal axes of inertia. This equation derives from the Newton second law, according to which, in an inertial reference frame, the time derivative of the angular momentum h equals the applied torque:

$$\frac{d\mathbf{h}}{dt} = \mathbf{M}$$

The angular momentum is equal to:

$$\mathbf{h} = \mathbf{I}\omega$$

where ω is the angular velocity about the principle axes, $\mathbf{I} \in \mathbb{R}^{\mathbf{3}}$ is the inertia matrix:

$$\mathbf{I} = \begin{bmatrix} I_x & -I_{xy} & -I_{xz} \\ -I_{yx} & I_y & -I_{yz} \\ -I_{zx} & -I_{zy} & I_z \end{bmatrix}$$

where I_x, I_y and I_z are the moments of inertia about the x, y and $z - axes$, respectively and $I_{xy} = I_{yx}$, $I_{xz} = I_{zx}$ and $I_{yz} = I_{zy}$ are the products of inertia defined as:

$$I_x = \int_V (y^2 + z^2)\rho dV$$

$$I_y = \int_V (x^2 + z^2)\rho dV$$

$$I_z = \int_V (x^2 + y^2)\rho dV$$

$$I_{xy} = I_{yx} = \int_V xy\rho dV$$

$$I_{xz} = I_{zx} = \int_V xz\rho dV$$

$$I_{yz} = I_{zy} = \int_V yz\rho dV$$

where ρ is the mass density of the body.
If the axes of the body frame coincides with the principal axes of inertia, as in the case considered in this work, the inertia matrix reduces to:

$$\mathbf{I} = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix}$$

In a rotating reference frame, the time derivative must be replaced with:

$$\frac{d\mathbf{h}}{dt} + \omega \times \mathbf{h} = \mathbf{M}$$

Since the inertial matrix doesn't depend on time, the equation becomes:

$$\mathbf{I}\dot{\omega} + \omega \times (\mathbf{I}\omega) = \mathbf{M}$$

2.4 Perturbing Forces and Torques

Spacecraft dynamics is influenced by several environmental forces and torques which the body is subjected to, during his life. Perturbing forces tend to modify orbit parameters and can lead to orbit decay; perturbing torques influence the spacecraft rotational motion and induce rotational accelerations. In LEO, the main disturbance elements to take into account to design the control system are: the aerodynamic drag, the gravity gradient, the solar radiation pressure and the Earth magnetic field.

2.4.1 Aerodynamic Drag

A spacecraft orbiting the Earth at low altitude is affected by a small drag, due to residual atmospheric particles in the upper layers of the atmosphere. The drag will create an acceleration component in the opposite direction of the velocity vector. This small force acting on the surface of the vehicle can lead to orbit decay, if not controlled. The atmospheric drag is:

$$\vec{F_a} = \frac{1}{2}\rho V^2 A C_D \frac{\vec{V}}{V}$$

where

- ρ is the atmospheric density, which at an altitude of 400 km is equal to $2.803 \cdot 10^{-12} kg \cdot m^{-3}$;
- V is the spacecraft velocity;
- A is the projected area of the spacecraft surface normal to $\vec{V}$;
- C_D is the drag coefficient, which is 2.2.

This drag force will produce a disturbance torque on the spacecraft, if any offset between the aerodynamic centre of pressure and the centre of mass exists. The aerodynamic torque will be:

$$\vec{T_a} = \vec{r}_{cp} \times \vec{F_a}$$

where $\vec{r}_{cp}$ is the centre-of-pressure (cp) vector , measured from the centre of mass, in body coordinates.

2.4.2 Solar radiation pressure

Solar radiation particles (photons) hitting the spacecraft surface can generate, in addition to many other effects, also a mechanical pressure, which leads to periodic variations of the orbital elements. It is a fluctuating perturbation and has more

effect at high altitudes, (e.g. geocentric orbits) and on light objects with a large surface. This perturbation is periodic as it applies only when the spacecraft faces the sun (if the Earth's albedo is neglected) and depends on the attitude of the satellite. The radiation pressure is expressed as follows:

$$\vec{F}_{sun} = -\lambda C_R P_0 A \vec{u}$$

where

- λ is the shadow function, that has a value of 0 if the satellite is in the Earth's shadow;
- C_R is the radiation pressure coefficient, which lies between 1 and 2. Assuming a value of 1.5 means that half of the photons are absorbed and half are reflected;
- $P_0 = 4.644 \cdot 10^{-6} Nm^{-2}$ is the solar pressure assumed constant;
- A is the spacecraft surface projected area normal to sun vector;
- $\vec{u}$ is the vector pointing from the Earth to the Sun.

Solar radiation pressure can produce, also, a torque on the spacecraft:

$$\vec{T}_{sun} = \vec{r}_{sp} \times \vec{F}_{sun}$$

where $\vec{r}_{sp}$ is the vector from body centre of mass to spacecraft optical centre of pressure.
Solar radiation torque is independent of spacecraft position or velocity, as long as the vehicle is in sunlight, and is always perpendicular to the sun line.

2.4.3 Gravity gradient

An orbiting object around a planet is subjected to a gravitational gradient, due to the newtonian law, which states that the planetary gravitational fields intensity follow the inverse-square law. The gravitational attraction on the side of the spacecraft closer to the planet is bigger than the one acting on the side more distant from the planet. This differential attraction, if applied to a body having unequal principal moments of inertia, results in a torque tending to rotate the object to align its minimum inertia axis with the local vertical, causing a periodic oscillatory motion. The gravity-gradient torque for a satellite in a near-circular orbit is:

$$\vec{T}_{grav} = \frac{3\mu}{R^3} \mathbf{u_e} \times [\mathbf{I}] \cdot \mathbf{u_e}$$

where

- $\mu = 3.986 \cdot 10^{14} m^3/s^2$ is Earth's gravitational coefficient;
- R is the distance of the spacecraft from the Earth's centre;
- $\mathbf{I}$ is the spacecraft inertia matrix;
- $\mathbf{u_e}$ is the unit vector from planet to spacecraft.

2.4.4 Earth magnetic field

Earth has a strong magnetic field, which field lines enclose around the planet, forming the *magnetosphere*, as shown in figure 2.11. The interaction of the magnetic

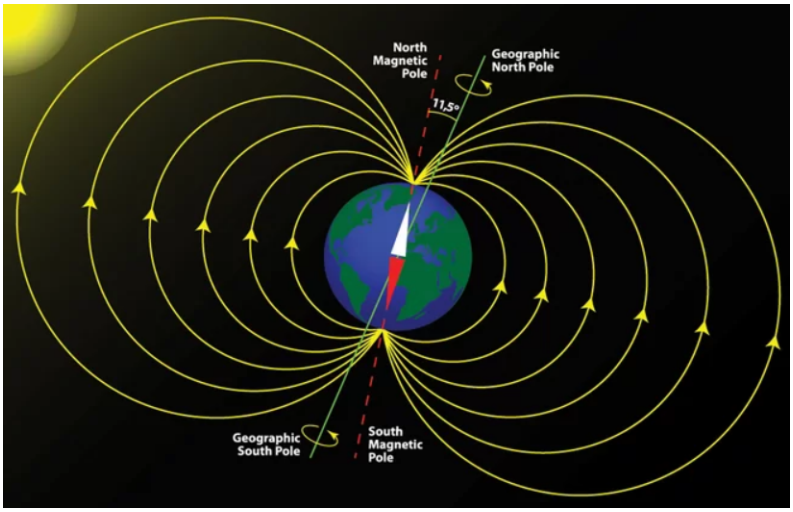


Figure 2.11: Earth magnetic field.

field and the spacecraft dipole moment, generates a torque on the spacecraft, equal to:

$$\vec{T}_{mag} = \vec{m} \times \vec{B}$$

where $\vec{m}$ is the spacecraft dipole moment, due to current loops and residual magnetization, measured in Am^2 per turns and $\vec{B}$ is the Earth magnetic field vector, expressed in body frame, measured in Tesla, T. Its magnitude is proportional to the magnetic moment of the Earth, $\mu = 7.96 \cdot 10^{-15} Tm^3$ and to $1/r^3$, where r is the radius vector to the spacecraft.

Chapter 3

Control Theory

3.1 Dynamic system control

3.1.1 Dynamical system

A dynamical system is described by variables evolving in time. The functions that describe the variables evolution in time are called **signals**. The fundamental signals of a dynamical system are the input signal $u(t)$ and and the output signal $y(t)$; depending on the controllability of the input signal, it can be a command or a disturbance signal. The dynamical systems are governed by differential equations, that constitute a model of the system. In control theory, a mathematical model is used to describe the system upon which to perform the control. This leads to two types of uncertainties: a parametric and a dynamic uncertainties, due to the fact that a model is always an approximation of the system that it represents.
Every dynamical system, which is a continuous time and finite dimension system, can be described by a first order differential equations system:

$$\dot{x}(t) = f[x(t), u(t); t]$$

$$y(t) = h[x(t), u(t); t]$$

dove $x(t) \in \mathbb{R}^n$ is the system state, $u(t) \in \mathbb{R}^n$ is the input, $y(t) \in \mathbb{R}^n$ is the output and n is the system order. This description of a dynamical system is called **state equation**. The first equation is dynamic:

$$\dot{x}(t) = f[x(t), u(t); t] \Longrightarrow x(t + dt) = x(t) + f[x(t), u(t); t]dt$$

The state at $t + dt$ depends on the state and on the input at time t. So, the time evolution is described. The second equation, $y(t) = h[x(t), u(t); t]$ instead, is static,

because it expresses a relation between quantities at the same time.

For practical reason, it is better to analyse the system in terms of **transfer functions**. The transfer function is obtained by applying **the Laplace transform** to the state equation. If we consider the state equations of a Linear-Time Invariant (LTI) system:

$$\dot{x}(t) = Ax(t) + Bu(t)$$

$$y(t) = Cx(t) + Du(t)$$

applying the Laplace transform to the first equation, we have:

$$sX(s) - x(0) = AX(s) + BU(s)$$

$$(sI - A)X(s) - x(0) = BU(s)$$

$$X(s) = (sI - A)^{-1}x(0) + (sI - A)^{-1}BU(s)$$

Applying the Laplace transform to the second equation, we obtain:

$$Y(s) = CX(s) + DU(s)$$

Considering that $X(s) = (sI - A)^{-1}x(0) + (sI - A)^{-1}BU(s)$,

$$Y(s) = C(sI - A)^{-1}x(0) + [C(sI - A)^{-1}B + D]U(s)$$

Let $x(0) = 0$:

$$Y(s) = G(s)U(s)$$

$$G(s) = C(sI - A)^{-1}B + D$$

where $G(s)$ is the transfer function of the system. It can, also, be seen in a rational representation as:

$$G(s) = \frac{b_m s^m + b_{m-1}s^{m-1} + ... + b_1 s + b_0}{s^n + a_n - 1s^{n-1} + ... + a_1 s + a_0} = K_G \frac{\prod_{i=1}^{m}(s - z_i)}{\prod_{i=1}^{n}(s - p_i)}$$

Where $p_1, ..., p_n$ are $G(s)$ poles, $z_1, ..., z_m$ are $G(s)$ zeros and K_G is the gain.

3.2 Dynamical system control

Dynamical system control is based on the imposition of a command signal $u(t)$ to the system, in order to make the output $y(t)$ to follow a desired signal $r(t)$, set as a reference. We can point out a first discrimination in control approaches: **open-loop control** and **closed-loop control**. In open-loop control 3.1, the input does

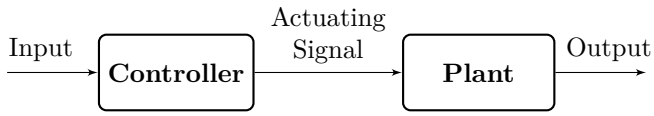


Figure 3.1: Open-loop control.

not depend on the output and, also, the controller doesn't take into account any possible disturbance signals.

In closed-loop control, represented in figure 3.2, instead, the input signal depends on the output and on the disturbing signals acting on the plant, if present. So, thanks to the feedback signal, the controller corrects the behaviour of the system on the basis of the output, step by step. This leads to more precision while controlling the system and to a more effective attenuation of disturbances.

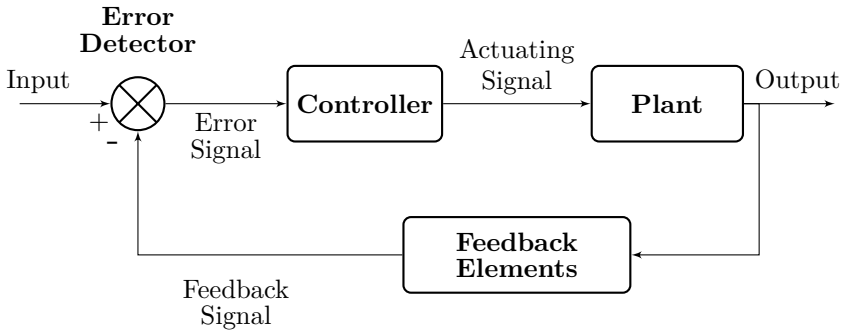


Figure 3.2: Closed-loop control.

In this work, the Neural network control and the PID control have been chosen amongst all existing control techniques and they have been described in next sections.

3.3 Neural control

3.3.1 Neural Network

Artificial Neural Network models are mathematical and computational emulations of the biological nervous system. These networks aim to retrace biological organism brain behaviour, in order to reproduce the capability to generalize from a large number of inputs and learn from external stimuli; therefore they are used, in many

fields, for the resolution of complex problems involving numerical data. [10]
These algorithms can learn to perform a task by being trained with example data, as they are able to establish a relationship between input and output data, without further informations. The mathematical model of a neural network is an approximation of what in neuroscience is known as "Neural Circuit". Hence, as a neural circuit is a population of neurons interconnected by synapses, more simplistically, an artificial neural network is a group of artificial neurons linked by connection links. The artificial neurons are the elementary units of the network, connection links interrelate them and transmit the informations in form of signals among the neurons, that process it. The connection links are associated to a specific weight, which properly multiplies the signal transmitted. [10]
There are many types of neural networks in use in machine learning algorithms: in this work, the one taken in consideration is the **feed-forward neural network**, which is a neural network in its simplest form. This network is characterized by the linearity of the information path: the signal propagates from the input layer to the output layer, without any cycle.

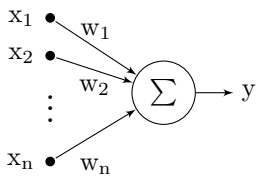


Figure 3.3: Scheme of the artificial neuron.

The first computational model for artificial neurons was proposed by McCulloch and Pitts. The single neuron produce an output signal based on an *activation function* f:

$$y = f(\sum_{i=1}^{n} w_i x_i - b)$$

The activation function depends on the resulting input, given by the sum of the input signals $x_1, x_2, ..., x_n$ multiplied for the associated weights of the connection links $w_1, w_2, ..., w_n$; while b is the firing threshold 3.3.
The f function is the step function, so the outputs are binary :

$$f(x) = \begin{cases} 1 & if\ x \geq 0 \\ 0 & if\ x \leq 0 \end{cases}$$

Therefore, the neuron is activated if the weighted sum of the inputs is bigger than

the threshold:

$$f(\sum_{i=1}^{n} w_i x_i - b) = \begin{cases} 1 & if \ \sum_{i=1}^{n} w_i x_i - b \geq 0 \\ 0 & if \ \sum_{i=1}^{n} w_i x_i - b \leq 0 \end{cases}$$

The neural network is formed by these elementary units, grouped in layers. Depending on the number of layers, there are **single-layer neural networks**, also called **perceptron**, and **multi-layer neural networks**. The single-layer neural network consists in one input layer of input nodes and an output layer of neurons.
The multi-layer neural network has more than one layer of neurons: it consists of one input layer, one or more middle layers and one output layer 3.4. The middle layer is called "hidden" because its outputs can't be directly observed.

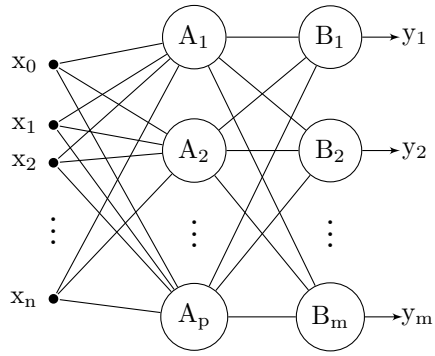


Figure 3.4: Two-layer neural network.

To be utilized, neural networks have to be trained to perform a specific task. The training is made by feeding the network with a specific example set of inputs and outputs. Only when the neural network is trained, it can be used.

3.3.2 Training the neural network

The key problem for neural network training and learning is to define which kind of functions can be represented by a neural network. Using the Stone-Weierstrass theorem in the theory of approximation of functions, the **Universal approximation theorem** has been proven:

Theorem: *A feed-forward network with a single hidden layer containing a finite number of neurons can approximate continuous functions on compact subsets of* $\mathbf{R}^n$*, under mild assumptions on the activation function.*

This means that all continuous functions whose domains are closed and bounded in $\mathbb{R}^n$ can be approximated by neural networks with smooth activation functions. The learning process takes the name of "**supervised learning**", when it occurs through known input-output pairs. The network shall find a function that optimizes the action that produces such outputs from the corresponding inputs. If (x_i, y_i), i=1,...,m is the set of example pairs, $y_i = \phi(x_i)$ where ϕ is the control law, the algorithm shall find a function ϕ_N that produces $\phi_N(x)$ outputs similar enough to $\phi(x)$. The approximation function ϕ_N is provided by the neural network and it is contained in the set of functions F from the input space X to the output space Y. The fact that the approximation functions belonging to F are suitable for ϕ is guaranteed by the **universal approximation property** of neural networks, written above.
An information about the amount of training data is also needed to identify the class of functions that obtain a good approximation. At this purpose, the Vapnik-Chervonenkis dimension of class of functions computable by the neural network has been defined. The concept of VC dimension of a class of sets can be extended to the case of a class of functions by using subgraphs. The subgraph of a function f is the subset $\mathbb{R}^n \times \mathbb{R}$ defined by:

$$S(f) = (x, t) : x \in \mathbb{R}^n, t \in \mathbb{R}, t \leq f(x)$$

The dimension of F is defined by the VC dimension of the class of its subgraphs[1]:

$$S(f) : f \in F$$

In conclusion, neural networks with finite VC dimensions are trainable. [10]

3.3.3 The backpropagation algorithm

Multi-layer neural networks are trained by means of the **backpropagation algorithm**, which is a generalization of the **Delta rule**, a learning algorithm for *single-layer neural networks*. The Delta rule is based on the optimization of the performance of the neural network: it aims to reduce the error between the neural network outputs, o^q, and the target outputs y^q (for all q=1,..,m), by changing the weights. The measure of the error taken into account is:

$$E = \sum_{q=1}^{N} E^q$$

[1] A subgraph of a function f: $\mathbb{R}^n \to \mathbb{R}$ is the set of points lying on or below its graph.

where E^q is:

$$E^q = \frac{1}{2}\sum_{i=1}^{m}(y_i^q - o_i^q)^2$$

In order to use the *gradient descent method*[2] for our purpose, the weight w_{ij} has to be differentiable, being w_{ij} the weight from node j to output neuron i. So, the following expression has to be differentiable:

$$o_i^q = f_i\left(\sum_{j=0}^{n} w_{ij}x_j^q\right)$$

where x_j^q is the input given to the network. Hence, the activation function f_i on the i^{th} neuron has to be a differentiable function.
Applying the gradient descent method to E, to find its minimum, we move proportionally to the negative of the gradient of E, -∇E, updating each weight w_{ij} as:

$$w_{jk} \rightarrow w_{jk} + \Delta w_{jk}$$
$$\Delta w_{jk} = -\eta\frac{\partial E}{\partial w_{ij}}$$

where $\frac{\partial E}{\partial w_{ij}}$ is the vector of partial derivatives representing the ∇E at a point w with components w_{ij} and η is a positive number, called **learning rate**.

$$\frac{\partial E}{\partial w_{ij}} = \sum_{q=1}^{N}\frac{\partial E^q}{\partial w_{ij}}$$

$$\frac{\partial E^q}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}}\left(\frac{1}{2}\sum_{i=1}^{m}(y_i^q - o_i^q)^2\right) = (o_i^q - y_i^q)\frac{\partial}{\partial w_{ij}}f_i\left(\sum_{i=0}^{n} w_{ji}x_i^q\right)$$

Since $\frac{\partial}{\partial w_{ij}}(o_i^q - y_i^q) = 0$ for $i \neq j$, and

$$o_j^q = f_j\left(\sum_{i=0}^{n} w_{ji}x_i^q\right)$$

Therefore,

$$\frac{\partial E^q}{\partial w_{ij}} = x_k^q(o_j^q - y_j^q)f_j'\left(\sum_{i=0}^{n} w_{ji}x_i^q\right) = \delta_j^q \cdot x_k^q$$

[2]The gradient descent method is a first-order iterative optimization algorithm for finding the minimum of a function. To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient of the function at the current point.

where

$$\delta_j^q = (o_j^q - y_j^q) f_j' \left(\sum_{i=0}^{n} w_{ji} x_i^q \right)$$

for the j^{th} output neuron. [10]

$$\Delta w_{jk} = \sum_{q=1}^{N} \Delta^q w_{jk} = -\eta \sum_{q=1}^{N} \frac{\partial E^q}{\partial w_{jk}} = \sum_{q=1}^{N} -\eta \delta_j^q x_k^q$$

This leads to find a certain weight vector, w^*, such that the ∇E at w^* is zero. To train more complex neural networks, as the multi-layer neural networks 3.5, we need to generalize the Delta rule, obtaining the backpropagation algorithm.

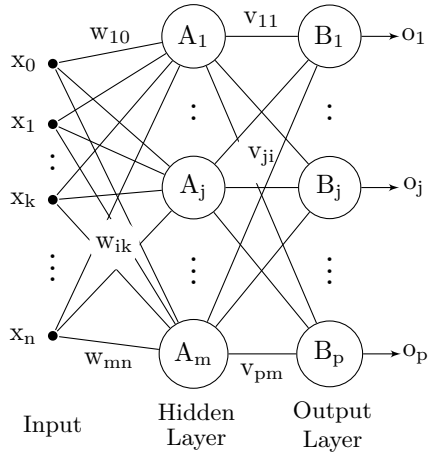


Figure 3.5: Two-layer neural network.

The main problem encountered in applying the delta rule to a multi-layer neural network is the fact that when the errors are analysed, it's not possible to detect which hidden neurons are responsible for the error, because y_j^q is the target patterns of the output layer only. However, to change the weight as needed, it's sufficient analysing the partial derivative of E with respect to the output o_i, i=1,...,p. If the activation function is differentiable, then it's possible to apply the gradient descent method to find the w^* that minimizes the error E, also in this case, doing some changes in the formulas. The weights of the hidden layer must be introduced: v_{ji} is the weight of the link connecting the hidden neuron i to the output neuron j, and w_{ik} the weight of the link connecting the input node k to the hidden neuron i. The delta rule can be applied to v_{ji}, by considering the hidden layer as an input

layer:

$$\Delta v_{ji} = \sum_{j=1}^{N} \Delta^q v_{ji} = -\eta \sum_{j=1}^{N} \frac{\partial E^q}{\partial v_{ji}} = \sum_{j=1}^{N} (-\eta \delta_j^q z_i^q)$$

where z_i^q is the net input to the hidden neuron i,

$$z_i^q = f_i \left(\sum_{k=0}^{m} x_k^q w_{ik} \right)$$

and

$$\delta_j^q = (o_j^q - y_j^q) f_i' \left(\sum_{k=1}^{m} v_{jk} z_k^q \right)$$

The update of the weight w_{ik} for a hidden neuron i and input node k is:

$$\Delta^q w_{ik} = -\eta \frac{\partial E^q}{\partial w_{ik}}$$

with

$$\frac{\partial E^q}{\partial w_{ik}} = \frac{\partial E^q}{\partial o_i^q} \frac{\partial o_i^q}{\partial w_{ik}}$$

where o_i^q is the output of the hidden neuron i,

$$o_i^q = f_i \left(\sum_{l=0}^{n} w_{il} x_l^q \right) = z_i^q$$

$$\frac{\partial o_i^q}{\partial w_{ik}} = f_i' \left(\sum_{l=0}^{n} w_{il} x_l^q \right) x_k^q$$

$$\delta_i^q = \frac{\partial E^q}{\partial o_i^q} = \sum_{j=1}^{p} \frac{\partial E^q}{\partial o_j^q} \frac{\partial o_j^q}{\partial o_i^q}$$

where j is in the output layer. The $\frac{\partial E^q}{\partial o_j^q}$ are known from previous calculations.

$$o_i^q = f_i' \left(\sum_{l=1}^{m} v_{il} z_l^q \right) v_{ji}$$

The δ_i^q, for hidden neurons i, are computed from the already-known values of δ_j^q for all j in the output layer. So, first the network is fed the input patterns x^q forward to the output layer and then the δ_j^q are calculated for all output neurons j. Next, these δ_j^q are propagated backwards to the layer below (in this case, the

hidden layer) in order to calculate the δ_i^q for all neurons i of that layer. At this point there are two approaches for training the network: **the batch approach** and **the incremental approach**. The batch approach consists on updating the weights w_{ik}, according to:

$$\Delta w_{ik} = -\eta \sum_{q=1}^{N} \frac{\partial E^q}{\partial w_{ik}}$$

after all N training patterns are presented to the neural network.
The incremental approach, instead, changes the weights w_{ik} after every training pattern is fed to the neural network, using:

$$\Delta^q w_{ik} = -\eta \frac{\partial E^q}{\partial w_{ik}}$$

The stopping criterion can be a threshold or the error function E.

3.3.4 Direct neural control

The direct neural control is a way of exploiting neural networks in control problems. We talk of direct neural control when the controller is a neural network, as in figure 3.6; instead, when the controllers are built based on a neural network model of the plant, we talk about indirect neural control. Direct neural controllers are designed by modelling inverse dynamics of the plant. The network must be trained as the controller using numerical input-output data or a mathematical model of the system. For developing a neural controller, several types of neural network architectures can be used. As been said in 3.3.1, the feedforward networks has been employed: feedforward neural configurations are multi-layer networks composed by various hidden layers of neurons, which number depends on the choice of the user, between the input and output layer. The learning algorithm used for training the neural network is the Levenberg-Marquardt algorithm.

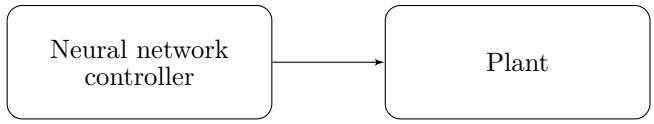


Figure 3.6: Direct neural control scheme.

3.3.5 Feed-forward neural network and Levenberg-Marquardt algorithm

Neural networks are distinguished into two main categories, based on their architectures: **feed-forward neural networks** and **recurrent neural networks**. The feed-forward neural network is characterized by the absence of a feedback, a synaptic connection from the outputs towards the inputs; otherwise, if there exists such a feedback, then the network is called recurrent neural network. Feed-forward neural networks can be single layer or multi layer, depending on the number of hidden layers the user decide to employ. In the input layer, there is no computation performed; input signals are passed on to the output layer via the weights and the neurons in the output layer compute the output signals. The hidden neurons intervene between the external input and the network output, in order to enable the network to extract higher-order statistics. In Figure 3.7 the network is **fully connected** because every neuron in each layer is connected to every other neuron in the next forward layer.

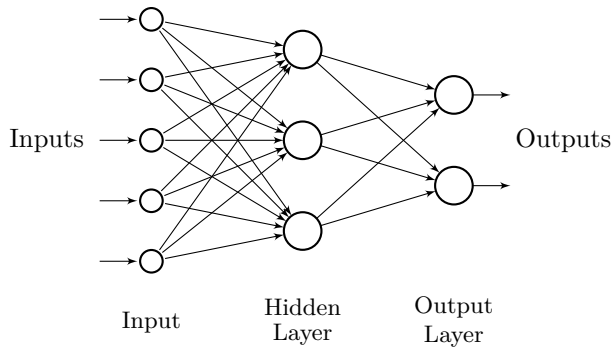


Figure 3.7: Multi-layer feed-forward neural network.

If some of the synaptic connections were missing, the network would be called **partially connected**. [12]

To train the network, the Levenberg-Marquardt (LM) algorithm has been used. The LM algorithm arise from the back-propagation algorithm, but combining the stability of the steepest descent method with the speed of the Newton algorithm. In the BP algorithm, the performance index E(w) to be minimized is defined as the sum of squared errors between the target outputs and the network's simulated outputs:

$$E(w) = \delta^T \delta$$

Where $w = [w_1, w_2, ..., w_N]$ consists of all weights of the network, δ is the error vector comprising the error for all the training examples. When training with the LM method, the increment of weights Δw can be obtained as follows:

$$\Delta w = [J^T J + \eta I]^{-1} J^T \delta$$

Where J is the Jacobian matrix, η is the learning rate which is to be updated using the β depending on the outcome. In particular, η is multiplied by decay rate $\beta(0 < \beta < 1)$whenever $E(w)$ decreases, whereas η divided by β whenever $E(w)$ increases in a new step. The standard LM training process can be illustrated in the following pseudo-codes:

1. Initialize the weights and parameter η (η=.01 is appropriate).

2. Compute the sum of the squared errors over all inputs $E_i(w)$.

3. Solve step 2. to obtain the increment of weights Δw.

4. Recompute the sum of squared errors $E_{i+1}(w)$.

5. Using $w + \Delta w$ as the trial w:

   if $E_{i+1}(w) < E_i(w)$ ***then***

   $$w = w + \Delta w$$

   $$\eta = \eta \cdot \beta(\beta = .1)$$

   and go back to step 2.

   else

   $$\eta = \frac{\eta}{\beta}$$

   and go back to step 4.

   end [13]

3.4 PID control

Proportional-Integral-Derivative (PID) control is the most common control algorithm employed in industrial control systems and in a wide range of applications. The reason why it is so largely diffused can be attributed to the fact that it is a

robust control and it is simple to implement.
It is a closed loop feedback, continuously modulated control, 3.8: the controller continuously calculates the error $e(t)$ between a desired value and the output of the plant. On the basis of this error, it applies a correction governed by a mathematical control law:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt}$$

where u is the control signal, e is the calculated error between the setpoint value and the plant output, and K_p, K_i and K_d are the proportional, integral and derivative gains, respectively. The control signal is thus a sum of three terms: the P-term, which is proportional to the error, the I-term, which is proportional to the integral of the error, and the D-term, which is proportional to the derivative of the error. The proportional, integral and derivative terms can be interpreted as actions on the present error, on the average of the past errors and on a prediction of future errors, respectively. With the proportional action the error decreases if the gain increases, but the system output becomes more oscillatory. When integral gain is increased, the steady state error[3] is removed, but the system behaviour is more oscillatory; for small values of K_i the system response tends slowly towards the reference. When no derivative action is performed, the behaviour is oscillatory and it becomes more damped as derivative gain is increased.

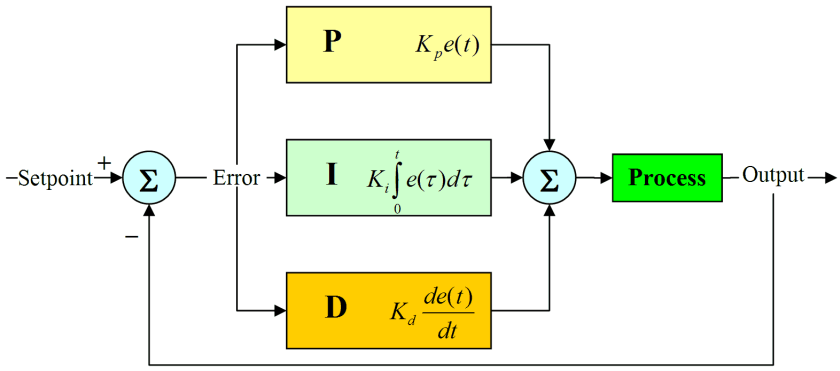


Figure 3.8: PID control scheme.

An important step in PID control developing is gains tuning. Optimizing gains

[3]The steady state error is defined as the difference between the input (command) and the output of a system in the limit as time goes to infinity.

values is fundamental for the convergence of the solution. Traditional control techniques based on modelling and design can be used, but there are also special methods for direct tuning based on simple process experiments.

Chapter 4

Simulation model

In this chapter the Matlab/Simulink model of the system is described. In figure 4.1, the overall Simulink architecture is depicted: the control blocks, one for translational motion and one for rotational motion, the dynamics blocks, that describe the spacecraft translational and rotational dynamics, and the block containing the disturbance forces. In figure 4.2, the functioning of the simulator is explained: the system consists of the translational and rotational dynamics blocks, which represent the chaser spacecraft motion with respect to the target. In the *Chaser Dynamics* block are contained the Hill equations, the kinematic equations and the Euler equations. The outputs of the dynamic block are sent to the two controllers, which compare the outputs with a reference set of values. If the two controllers detect a discrepancy with the desired values, they emit a signal to correct the spacecraft trajectory or rotational motion.
The input for the dynamics block are the disturbance torques and forces, which main sources are represented by the gravity gradient, the Earth magnetic field, the solar radiation pressure and the aerodynamic drag. The *Guidance, Navigation and Control* block gives the control input to the dynamics block, mediated by the *Thrusters and Reaction Wheels* block. The forces and torques are transformed to the *body-coordinate* system. The output of the system are the relative positions and velocities, the satellite attitude and attitude rate and the control forces and torques.

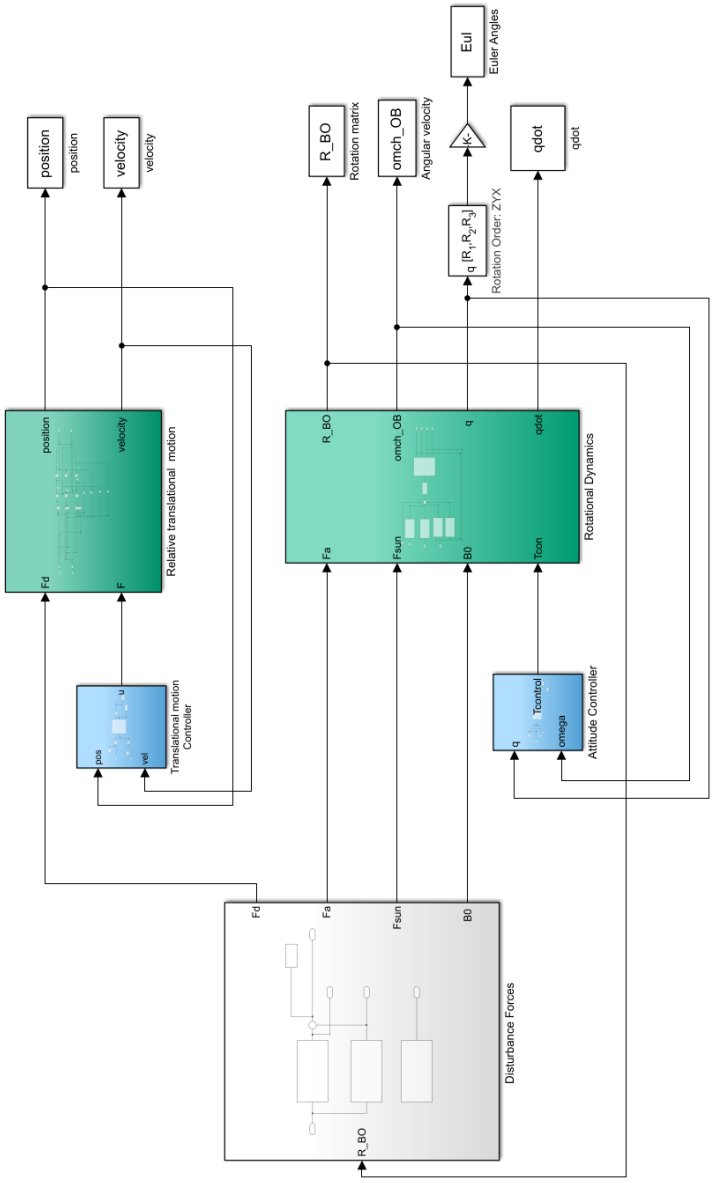


Figure 4.1: First level Simulink scheme.

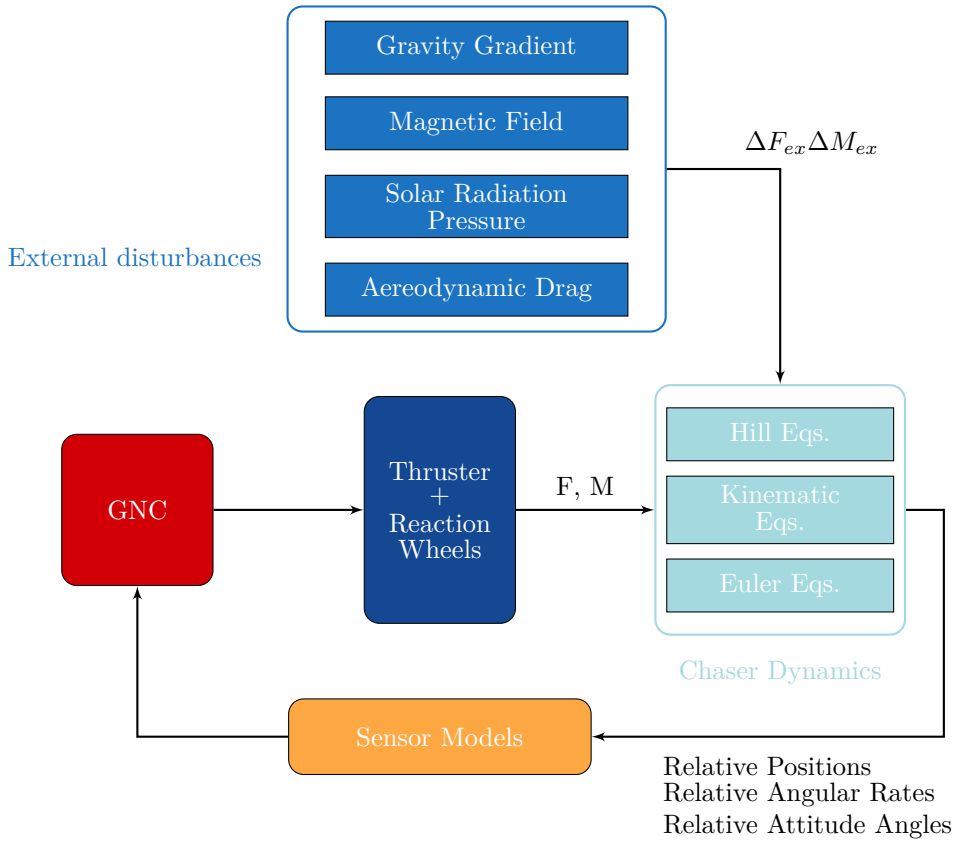


Figure 4.2: Spacecraft dynamics and control scheme.

4.1 Satellite Dynamics Model

4.1.1 Translational Dynamics model

The dynamics of the satellite is modelled taking into account both the translational and rotational dynamics of the satellite motion towards the Target. The translational motion is described in figure 4.3: the model represents the integration of the Hill equations of relative motions in the local orbital frame, LVLH:

$$\ddot{x} = \frac{1}{m_c} F_x + 2\omega\dot{z}$$

$$\ddot{y} = \frac{1}{m_c} F_y - \omega^2 y$$

$$\ddot{z} = \frac{1}{m_c} F_z - 2\omega^2\dot{x} + 3\omega^2 z$$

where F_x, F_y and F_z are the sum of the disturbance and control forces, expressed in N, in x, y and z direction, respectively; m_c is the mass of the Chaser satellite, in kg; and ω is the angular velocity of the spacecraft around Earth, in rad/s. By integrating these equation one time, we obtain the relative translational velocities in the three directions, in m/s; the second integration gives us the relative positions, in m. The linearisation of the Hill equations is represented in the state space form as:

$$\dot{\mathbf{x}}(k) = A(k)\mathbf{x} + B(k)\mathbf{u}$$

where $\mathbf{x}(k) = [\text{x y z } \dot{x}\ \dot{y}\ \dot{z}]$ is the state vector constituted by the three components of Chaser position and velocity with respect to the Target centre of mass, $u(k) = [F_x, F_y, F_z]$ is the vector of the control forces given by the controller, and A and B are defined as follows:

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 2\omega^2 \\ 0 & -\omega & 0 & 0 & 0 & 0 \\ 0 & 3\omega^2 & 0 - 2\omega^2 & 0 & 0 & \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1/m_c & 0 & 0 \\ 0 & 1/m_c & 0 \\ 0 & & 1/m_c \end{bmatrix}$$

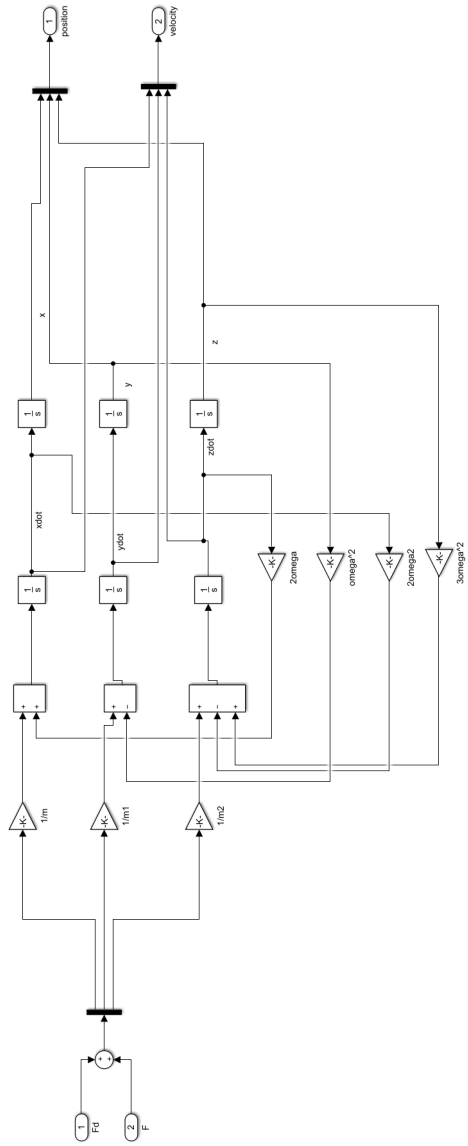


Figure 4.3: Relative Translational dynamics block.

The control output $\mathbf{u}$ is given by:

$$\mathbf{u} = K_p \mathbf{e}(k)$$

where K_p is the proportional gain and $\mathbf{e}(k)$ is the error vector, given by the difference of the state vector and the desired reference vector. So, the state-space equation for translational dynamics becomes [4]:

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 2\omega^2 \\ 0 & -\omega & 0 & 0 & 0 & 0 \\ 0 & 3\omega^2 & 0 - 2\omega^2 & 0 & 0 & \end{bmatrix} \mathbf{x} + \frac{1}{m_c} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \mathbf{u}$$

4.1.2 Rotational dynamics model

The rotational dynamic block is depicted in figure 4.4. It contains the disturbance and control torques, the kinetic and kinematic blocks. In figure 4.5, the kinetic block is shown, where the Euler equation is implemented:

$$\mathbf{I}\dot{\omega} + \omega \times (\mathbf{I}\omega) = \mathbf{T}$$

where, $\mathbf{I}$ is the satellite inertia matrix, ω and $\dot{\omega}$ are its angular velocity and its angular acceleration, respectively, with respect to the inertial reference frame; $\mathbf{T}$ is the sum of the control and disturbance torques.

$$\dot{\omega} = I^{-1}[\mathbf{T} - \omega \times I\omega]$$

By the integration of this equation, we obtain the angular velocity of the spacecraft with respect to the inertial reference frame.The Chaser angular velocity with respect to the orbital local frame, ω_{OB}, is:

$$\omega_{BO} = \omega_{BI} - R_{OB}^T \omega_{OI}$$

where ω_{OI} is the angular velocity of the orbital frame with respect to the inertial frame and R_{BO} is the rotation matrix. In the kinematics block, in figure 4.6, the quaternions and the angular velocity with respect to the body frame are obtained. For control problems, quaternions are more suitable than Euler angles, then the attitude kinematics equations are expressed in terms of quaternions:

$$\dot{\mathbf{q}}_{OB} = [Q_{OB}]\omega_{\mathbf{OB}}$$

where $\mathbf{q}_{OB} = [q_{OB0}, q_{OB1}, q_{OB2}, q_{OB3}]$ is the quaternion vector, expressing the Chaser attitude, with respect to the orbital frame and Q_{OB} is obtained through a

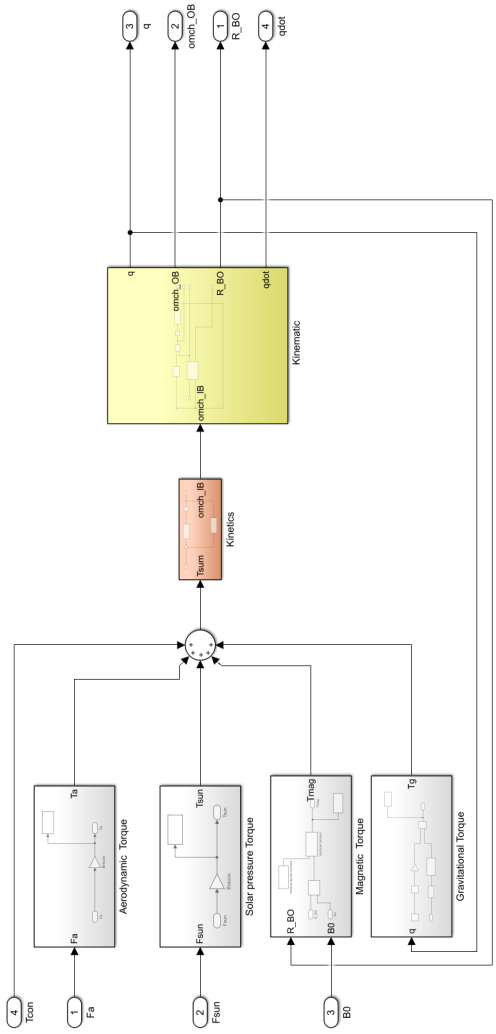


Figure 4.4: Chaser Rotational dynamics block.

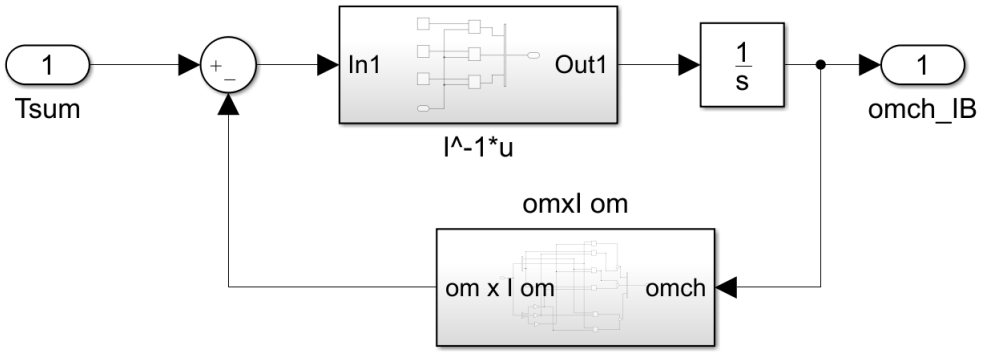


Figure 4.5: Kinetics.

Matlab interpreted function, from the quaternions:

$$Q_{OB} = \begin{bmatrix} -q_{OB1} & -q_{OB2} & -q_{OB3} \\ q_{OB0} & q_{OB3} & -q_{OB2} \\ -q_{OB3} & q_{OB0} & q_{OB1} \\ q_{OB2} & -q_{OB1} & q_{OB0} \end{bmatrix}$$

In this case, the state vector is $\mathbf{x}(k) = [q_{OB0}, q_{OB1}, q_{OB2}, q_{OB3}, \omega_{IBx}, \omega_{IBy}, \omega_{IBz}]$ and $\mathbf{u}(k)$ is the vector of the control forces given by the controller, $[T_x, T_y, T_z]$. The A and B matrices are:

$$A = \begin{bmatrix} 0_{4\times3} & [Q_{OB}] \\ 0_{3\times3} & \begin{bmatrix} 0 & \frac{\omega_{IBz}(I_y-I_z)}{I_x} & \frac{\omega_{IBy}(I_y-I_z)}{I_x} \\ -\frac{\omega_{IBz}(I_x-I_z)}{I_y} & 0 & -\frac{\omega_{IBx}(I_x-I_z)}{I_y} \\ \frac{\omega_{IBy}(I_x-I_y)}{I_z} & \frac{\omega_{IBx}(I_x-I_y)}{I_z} & 0 \end{bmatrix} \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{1}{2I_x} & 0 & 0 \\ 0 & \frac{1}{2I_y} & 0 \\ 0 & 0 & \frac{1}{2I_z} \end{bmatrix}$$

The control output $\mathbf{u}$ is given by:

$$\mathbf{u} = K_p \mathbf{e}(k)$$

where K_p is the proportional gain and $\mathbf{e}(k)$ is the error vector, given by the difference of the state vector and the desired reference vector. So the state-space equation for attitude dynamics becomes [4]:

$$\dot{\mathbf{x}} = \begin{bmatrix} 0_{4\times3} & [Q_{OB}] \\ 0_{3\times3} & \begin{bmatrix} 0 & \frac{\omega_{IBz}(I_y-I_z)}{I_x} & \frac{\omega_{IBy}(I_y-I_z)}{I_x} \\ -\frac{\omega_{IBz}(I_x-I_z)}{I_y} & 0 & -\frac{\omega_{IBx}(I_x-I_z)}{I_y} \\ \frac{\omega_{IBy}(I_x-I_y)}{I_z} & \frac{\omega_{IBx}(I_x-I_y)}{I_z} & 0 \end{bmatrix} \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{1}{2I_x} & 0 & 0 \\ 0 & \frac{1}{2I_y} & 0 \\ 0 & 0 & \frac{1}{2I_z} \end{bmatrix} \mathbf{u}$$

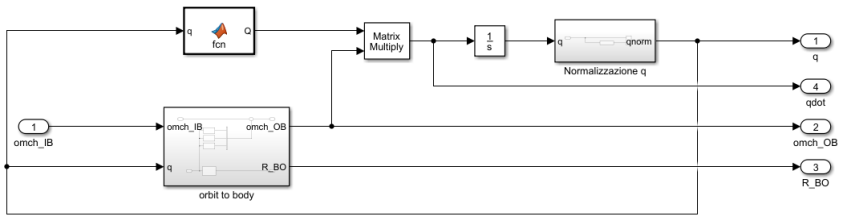


Figure 4.6: Kinematics.

4.2 PD-controller

PD-controller for translational motion is depicted in figure 4.7 . It controls both the translational relative position and velocity of the Chaser with respect to the Target spacecraft. The Target position and velocity are considered as reference by the controller: the LVLH reference frame is centred in the Target centre of mass, so the reference position vector is [0 0 0] and the reference velocity vector is [0 0 0], as the Target is considered a non moving object, i.e. a cooperative target. The Chaser actual trajectory, calculated in the translational dynamic block, is compared with the reference at each iteration; the error signals of position and velocity,$\mathbf{e}(k)$, are multiplied by proportional gains, properly tuned to make the system converge to the desired solution. The control signal values, generating the control forces to

correct the trajectory, are limited between a maximum and a minimum permissible value, given by the thrusters constrains:

$$-\mathbf{F}_{th,min} \leq \mathbf{u}(k) \leq \mathbf{F}_{th,max}$$

The same logic is followed by the attitude PID-controller, in figure 4.8. The atti-

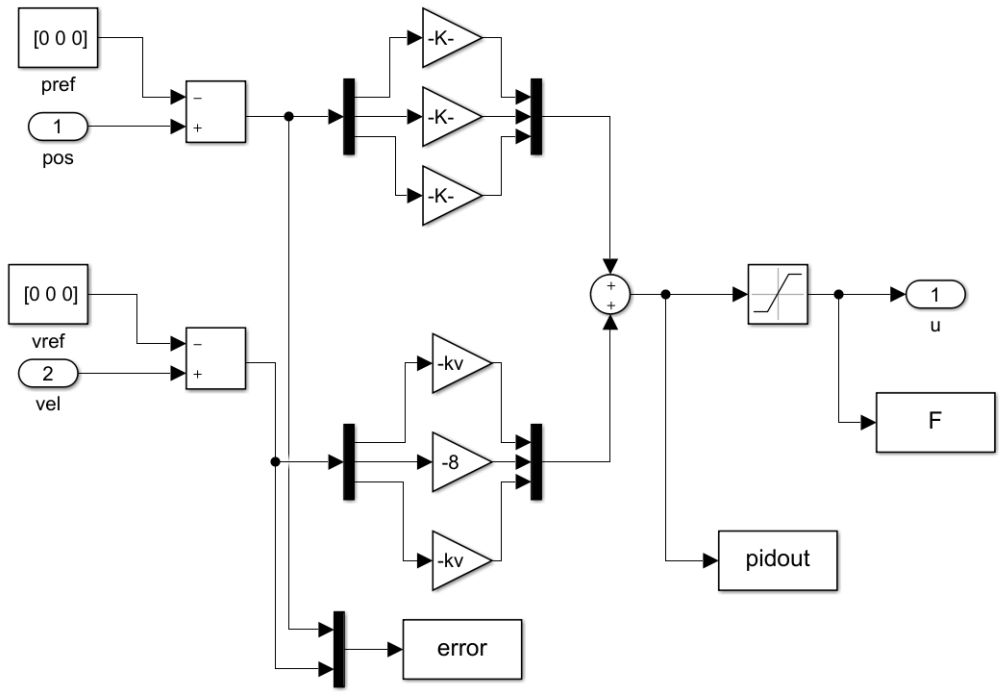


Figure 4.7: PID-controller for translational motion.

tude quaternions vector and the angular velocity vector calculated in the *Rotational dynamics block* are compared with the reference attitude and attitude rate: since the Target is considered non spinning, the reference angles and angular rate are both [0 0 0]. The error signals of attitude and attitude rate,$\mathbf{e}(k)$, are multiplied by proportional gains. The control signal values, generating the control torques, are limited between a maximum and a minimum permissible value, given by the actuators constrains; in this case we consider the Reaction wheels:

$$-\mathbf{T}_{RW,min} \leq \mathbf{u}(k) \leq \mathbf{T}_{RW,max}$$

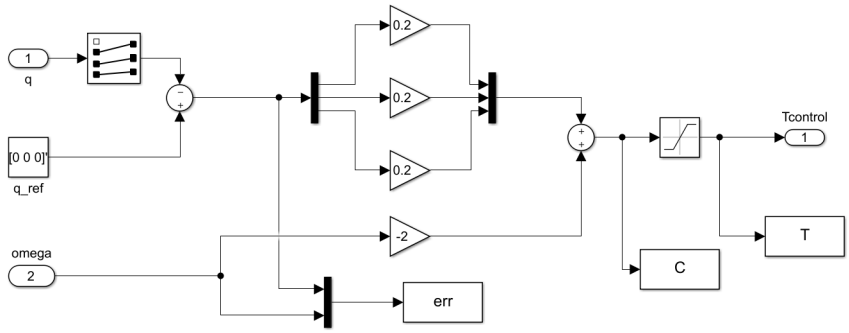
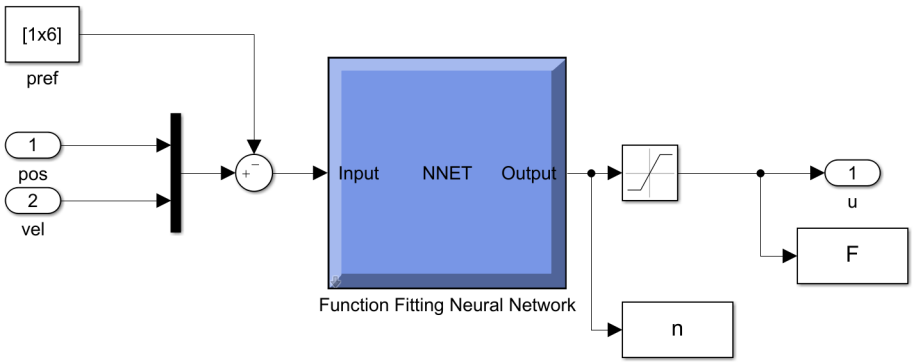


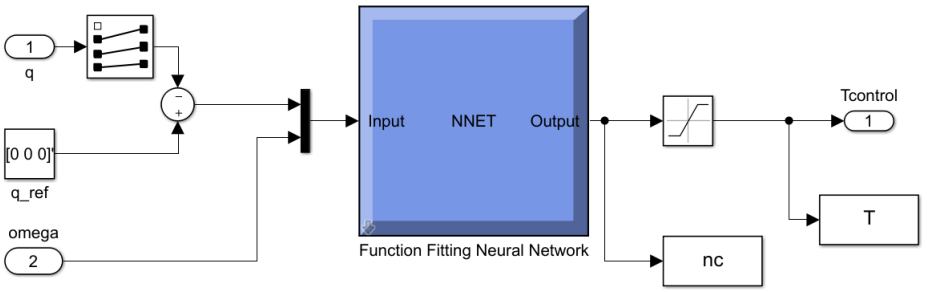
Figure 4.8: PID-controller for rotational motion.

4.3 Neural Network Controller

The Neural Network controller for translational and rotational motion, shown in figure 4.9, was realized thanks to the Mathworks' *Neural Network Toolbox*, which provides a tool to generate a neural network Matlab code and a Simulink block. The



(a) *Neural Network controller for translational motion.*



(b) *Neural Network controller for translational motion.*

Figure 4.9: Neural Network controller.

Neural Network Toolbox offers four types of the most common design techniques:

- Supervised Learing → Fitting app;
- Pattern Recognition → Pattern Recognition app;
- Clustering → Clustering app;
- Regression → Dynamic time series app;

Supervised Learning consists in training the neural network to produce the desired outputs in response to sample inputs, making them particularly suitable for modelling and controlling dynamic systems, classifying noisy data and predicting future events. The goal is to give the neural network the ability to generalize from the training data and to adapt to unseen data sets.
Pattern Recognition works by classifying the input data into objects or classes based on key characteristics, using the supervised or unsupervised classification. It is considerably useful in machine vision, radar processing, speech recognition and text classification.
Clustering learning technique is an unsupervised approach in which neural networks can be used for analysing exploratory data to find hidden patterns or groupings in data. This process involves grouping the data by similarity. Among the applications of the cluster analysis are the analysis of the genetic sequence, market research and object recognition.
Regression algorithms establish the relationship between a response variable (output) and one or more explanatory variables (inputs).

In this work, the Neural Networks used were generated by the Fitting app. The Neural Net Fitting app solves a data-fitting problem with a two-layer feed-forward neural network with sigmoid hidden neurons and linear output neurons. It allows the creation and training of a neural network, selecting input and output data from the MATLAB® workspace, dividing it into training, validation, and testing sets and defining the network architecture by choosing the number of hidden layers. The evaluation of the network performance is carried out, evaluating its mean squared error (MSE) and regression analysis. If the results are not satisfying, the app permits the retraining of the network with modified settings or on a larger data set. [7]
The activation function of the hidden neurons is a sigmoid function, which is differentiable, necessary requirement for a network to be trainable [10]. The sigmoid function is expressed by the formula,

$$f(x) = \frac{1}{1 + e^{-x}}$$

and has the form shown in figure 4.10.

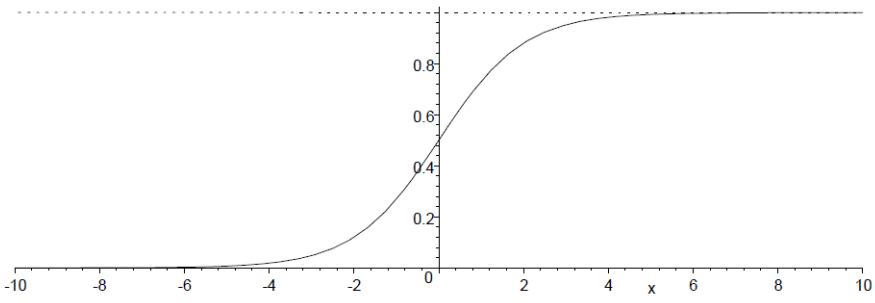


Figure 4.10: Sigmoid activation function.

4.3.1 Regression and Mean Squared Error

The network learning performance are expressed in terms of mean squared error (MSE) and of linear regression. Regression is a statistical technique to determine the linear relationship between two or more variables. Regression is primarily used for prediction and causal inference. The linear form of regression shows the relationship between one independent variable (X) and a dependent variable (Y), as shown in the formula:

$$Y = \beta_0 + \beta_1 X + u$$

The magnitude and direction of that relation are given by the slope parameter (β_1), and the status of the dependent variable when the independent variable is absent is given by the intercept parameter (β_0). An error term (u) captures the amount of variation not predicted by the slope and intercept terms. The linear regression model is designed to establish a relationship between a pair of variables in a data set. In this case, Y is the target data set, the dependent variables, and X is the input data set, the independent variables. The aim of the regression equation is to find the best fitting line relating the variables to one another [3].
MSE measures the average of the squares of the errors, that is the discrepancy between the estimated values and what is observed. The equation that defines the MSE is:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2$$

where n is the number of data points, $\tilde{y}_i$ is the estimated data and y_i is the observed data. It indicates how close a regression line is to a set of points. It does this by taking the distances from the points to the regression line and squaring them. The squaring is necessary to remove any negative signs. [9]

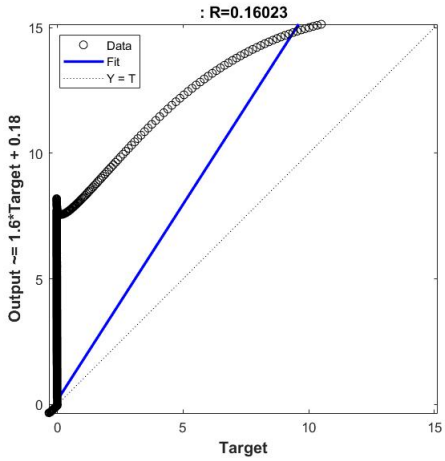
4.4 Training the Neural Network with PD-control

4.4.1 Translational motion NN control

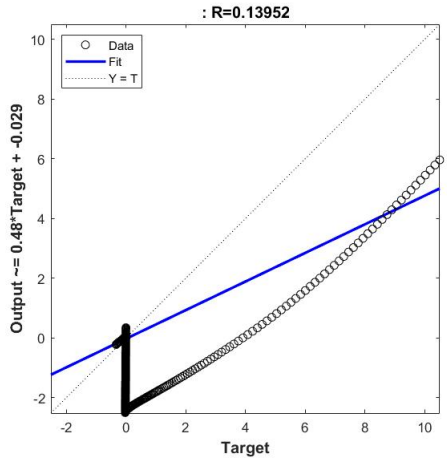
To generate the neural network Simulink block, first, the trajectory points and commands using PD control were obtained. The training data sets was constituted by the error vector, i.e. the PID controller input, and the PID output vector. So the neural network received as input the errors on the trajectory measured with respect to the desired position and velocity vectors, and as target data, it received the PD output commands.The data provided to the network for the training are divided, according to the user's wishes, into a percentage for training, one for the test and one for validation. The data used for validation are needed to validate that the network is generalizing and to stop training before overfitting. The test data will be used as a completely independent test of network generalization. The number

of hidden layers chosen to perform the training was 6. In figure 4.11, the regression plots relative to each training set are shown. For a perfect fit, the data should fall along a 45 degree line, where the network outputs are equal to the targets. In figure 4.11.a, the regression plot shows that the network was not successfully trained, as the targets do not fit the desired output. In this case, it is necessary to retrain the network, sightly changing the data training set. This will change the initial weights and biases of the network, and may produce an improved network after retraining. After the third training session, the regression plot shows that the target fits the data. Although, if we train the network once again, the performance drops, as depicted in figure 4.11.d. This is due to overfitting. The error on the training set is driven to a very small value, but when new data is presented to the network the error is large. The network has memorized the training examples, but it has not learned to generalize to new situations.The method used here for improving network generalization is decreasing the number of hidden layers, so the make the network just large enough to provide an adequate fit. The larger network, the more complex the functions the network can create; so if the network is small enough, it will not have enough power to overfit the data. [8]

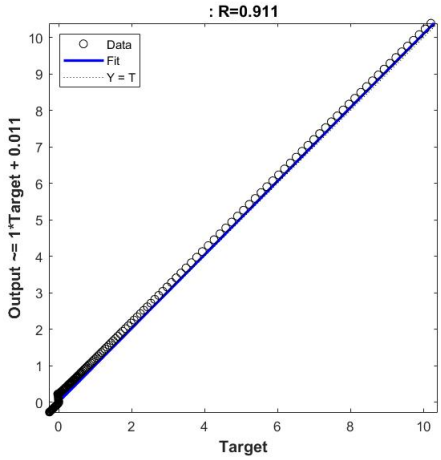
At this point, it is necessary to adjust the number of hidden neurons layers and train the network again. After other two training sessions, the regression plot shows satisfying results, 4.12. The following regression plots display the network outputs with respect to targets for training, validation, and test sets. The neural network control block thus generated is ready to be inserted and used in the rendez-vous manoeuvre Simulink model.



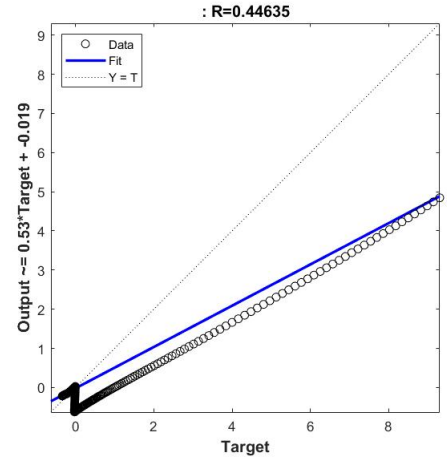
(a) *Regression plot, after first training session.*



(b) *Regression plot, after second training session.*

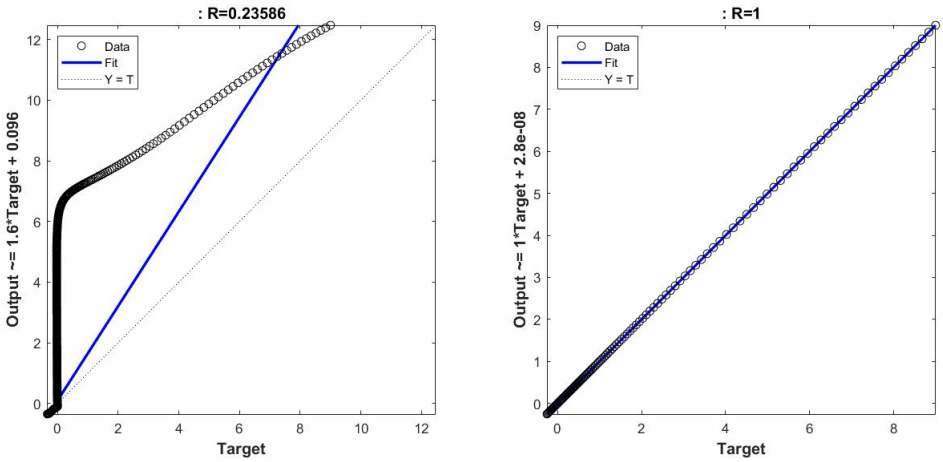


(c) *Regression plot, after third training session.*



(d) *Regression plot, after forth training session.*

Figure 4.11: Regression plot after every training session.



(a) *Regression plot, after fifth training session.* (b) *Regression plot, after sixth training session.*

Figure 4.12: Regression plot after every training session.

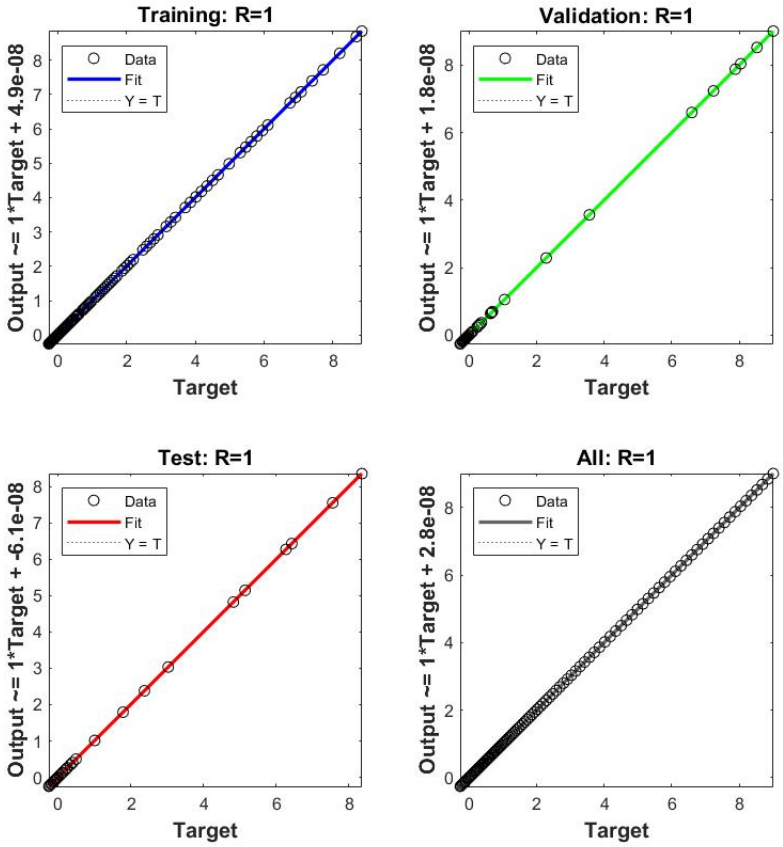
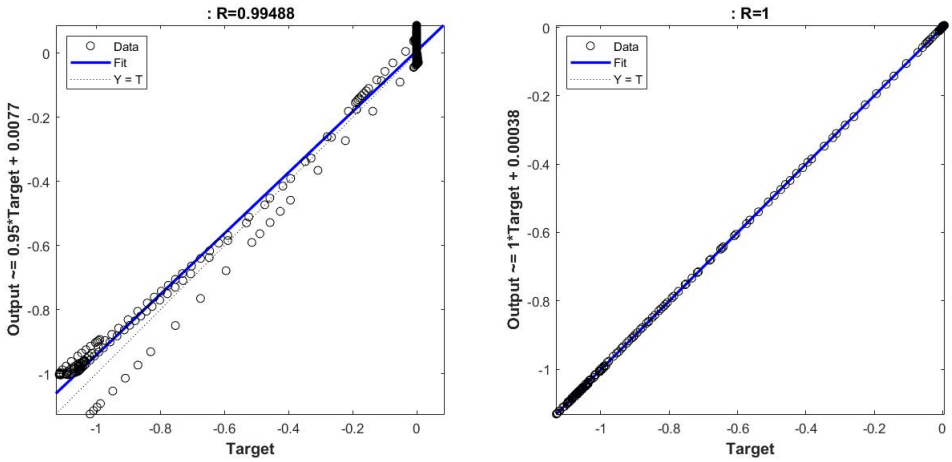


Figure 4.13: From the first up on the left, clockwise: Training regression plot, Validation regression plot, Test regression plot, All data regression plot.

4.4.2 Rotational motion NN control

To generate the neural network Simulink block, first, the attitude, attitude rate and control torque data points using PID control were obtained. The training data sets was constituted by the error vector, i.e. the PD controller input, and the PD output vector. So the neural network received as input the errors between the desired attitude and the actual attitude or attitude rate vectors, and as target data, it received the PID output commands. As said in previous paragraph for the trajectory controller, the data provided to the network for the training are divided, according to the user's wishes, into a percentage for training, one for the test and one for validation. The data used for validation are needed to validate that the network is generalizing and to stop training before overfitting. The test data will be used as a completely independent test of network generalization. The number of hidden layers chosen to perform the training was 5. In figure 4.14, the regression plots relative to each training set are shown. For a perfect fit, the data should fall along a 45 degree line, where the network outputs are equal to the targets. In figure 4.14.a, the regression plot shows that the the network was not successfully trained, as the targets do not fit the desired output. In this case, it is necessary to retrain the network, sightly changing the data training set. This will change the initial weights and biases of the network, and may produce an improved network after retraining [8]. After the second training session, the regression plot shows that the target fits the data, as shown in figure 4.14.b. The neural network control block thus generated is ready to be inserted and used in the rendez-vous manoeuvre Simulink model.



(a) *Regression plot, after first training session.* (b) *Regression plot, after second training session.*

Figure 4.14: Regression plot after every training session.

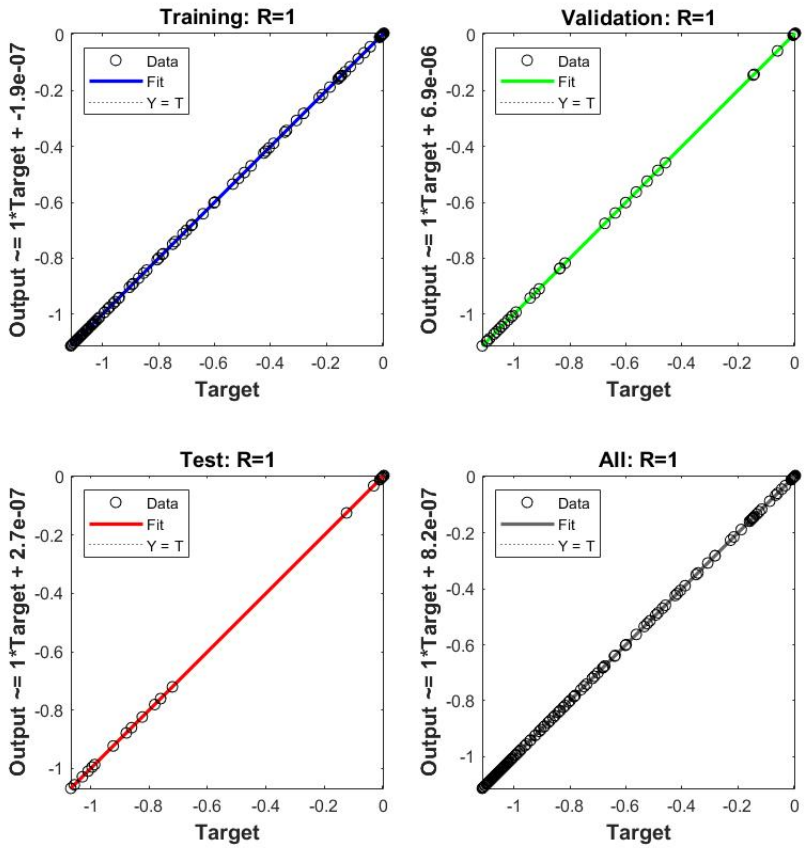


Figure 4.15: From the first up on the left, clockwise: Training regression plot, Validation regression plot, Test regression plot, All data regression plot.

Chapter 5

Simulation results

In this chapter, the simulation results are reported, in order to show the behaviour of the spacecraft in relation to the two different control strategies and its response to different initial conditions. The Chaser physical characteristics are shown in table 5.1: the mass, external dimensions and inertia elements respects the CubeSat standard, which require small masses and dimensions that has to be multiples of the 1U (1cm×1cm ×1cm). The orbit where the mission takes place is a low-

Table 5.1: Chaser nominal properties.

Mass (kg)	10
l_x (m)	0.2
l_y (m)	0.1
l_z (m)	0.3
I_x (kgm^2)	0.0833
I_y (kgm^2)	0.10833
I_z (kgm^2)	0.04166

Earth orbit, whose parameters are shown in table 5.2. The orbit is considered circular to avoid model complications. The orbital elements taken into account are the orbit altitude h, the orbit inclination i in table 5.2: The PD controller gains

Table 5.2: Orbital parameters.

Altitude, h (Km)	400
inclination (°)	98

obtained after properly tuning the controller, both for trajectory and attitude

control, are shown in tables 5.3 and 5.4: The disturbance forces and toques taken

Table 5.3: Trajectory PID-controller gains.

K_{px}	K_{py}	K_{pz}	$K_{p\dot{x}}$	$K_{p\dot{y}}$	$K_{p\dot{z}}$
-0.4	-0.08	-0.4	-40	-8	-40

Table 5.4: Attitude PID-controller gains.

K_{pq_1}	K_{pq_2}	K_{pq_3}	$K_{p\omega_x}$	$K_{p\omega_y}$	$K_{p\omega_z}$
0.2	0.2	0.2	-2	-2	-2

into account are due to aerodynamic drag, the Earth magnetic field, the gravity gradient and the solar radiation pressure. The translational motion is interested by the aerodynamic drag and the solar radiation pressure, since the magnetic field and the gravity gradient do not produce a torque on the spacecraft. The attitude dynamics, instead, in interested by all the said elements: aerodynamic, magnetic, gravitational and solar pressure torques. The maximum values of this forces and torques is approximately:

- **Aerodynamic Drag** : $F_a \sim -10^{-6}$ N;
- **Solar Radiation Pressure**: $F_{sun} \sim 10^{-5}$ N;
- **Aerodynamic Torque**: $T_a \sim 0.2124 \cdot 10^{-7}$ Nm ;
- **Solar Radiation Torque**: $T_{sun} \sim 10^{-8}$ Nm
- **Gravity Gradient Torque**: $T_g \sim 10^{-8}$ Nm
- **Magnetic Torque**: $T_m \sim 10^{-7}$ Nm

The limitations of the control forces and torques are dictated by the actuators constraints: CubeSats can be equipped with actuators suitable for their small dimensions and masses. So, the thrusters for trajectory control can have a minimum and maximum force:

$$-0.4N \leq F_{th} \leq 0.4N$$

The actuators for attitude control, the reaction wheels, can give a maximum and a minimum control torque:

$$-0.01Nm \leq T_{RW} \leq 0.01Nm$$

5.1 Translational motion control: nominal conditions

The following results represent the trajectory control of the Chaser with respect to the Target, in terms of relative positions and velocities along the three directions in the LVLH frame. These results were carried out considering the input data in table 5.1 for the Chaser physical characteristics and the orbital parameters in 5.2; all the disturbance forces acting on the spacecraft has been considered as indicated in the previous section; PID controller gains for trajectory control are indicated in table 5.3.
The Neural Network controller was trained several times, with different sets of input/output obtained with PD-control, and the final net architecture is depicted in figure 5.1; the neural Network here represented has 6 hidden neurons layers: The total simulation time is 1000 seconds, with a fixed step of 0.01; the initial

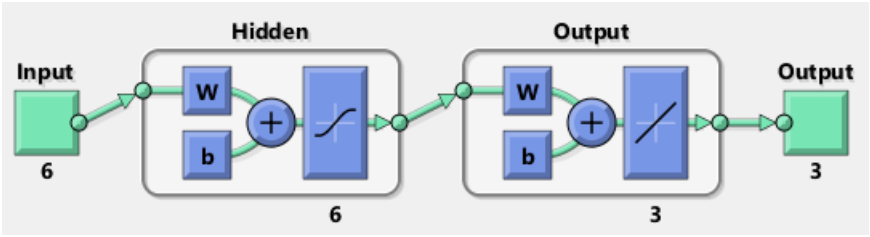


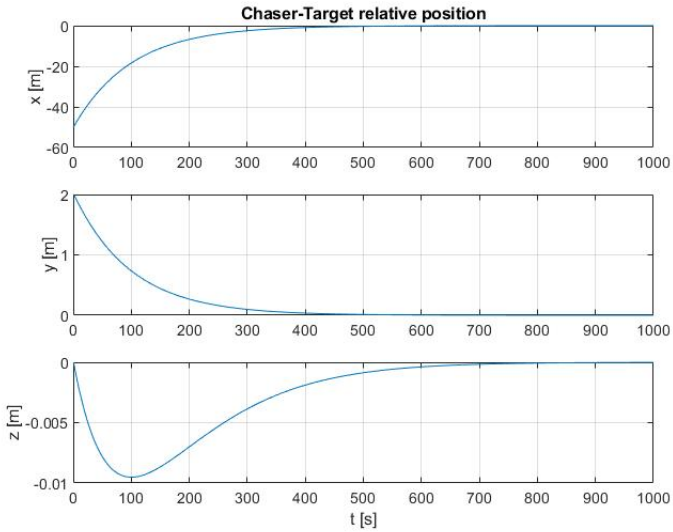
Figure 5.1: Neural Network architecture for trajectory control.

conditions for this set of simulations are:

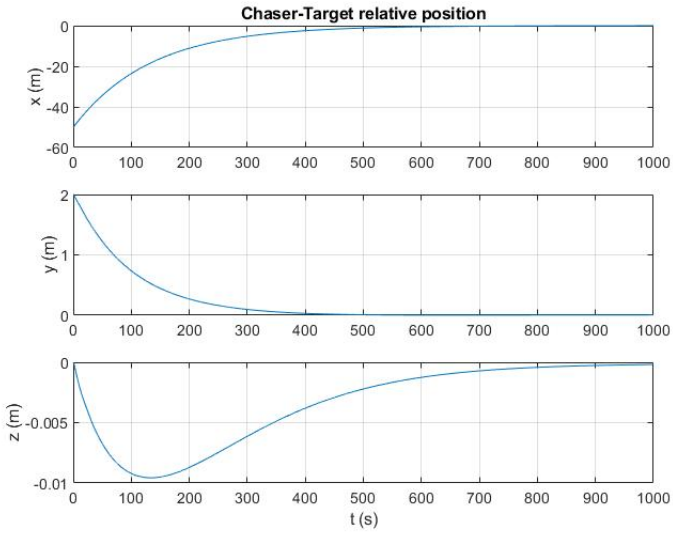
Table 5.5: Initial conditions.

x (m)	y (m)	z (m)	v_x (m/s)	v_y(m/s)	v_z(m/s)
-50	2	0	0.15	0	0

In figure 5.2, it is shown the comparison between the system controlled by PD and by Neural Network. It is evident that the Neural Network precisely emulates the PID behaviour, leading the system to convergence. The Chaser spacecraft reaches the Target position in approximately 500 seconds, both in x and y directions; it takes a little longer in the z direction, about 700 seconds. To have a better understanding of the NN controller behaviour, it is possible to estimate an error between the final position and velocity at the end of the simulation between the PD model and the NN model:

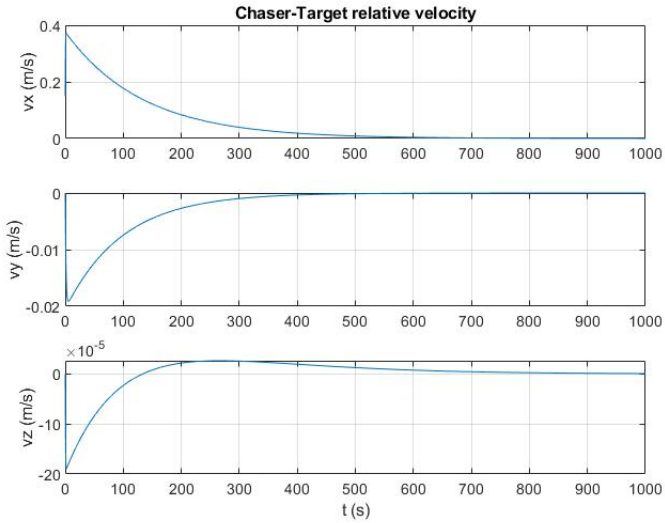


(a) *Chaser-Target relative position, PID control.*

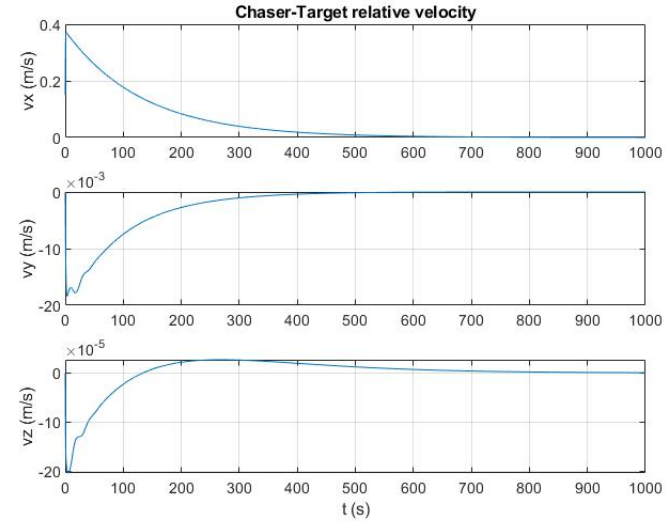


(b) *Chaser-Target relative position, NN control.*

Figure 5.2: Chaser-Target relative position, nominal conditions.

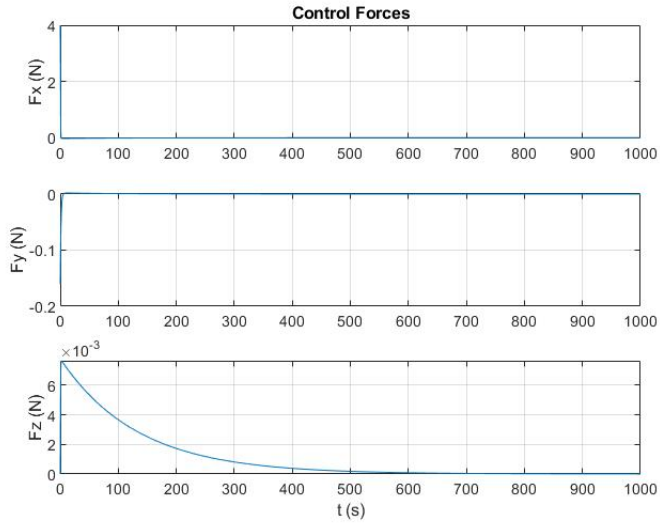


(a) *Chaser-Target relative velocity, PID control.*

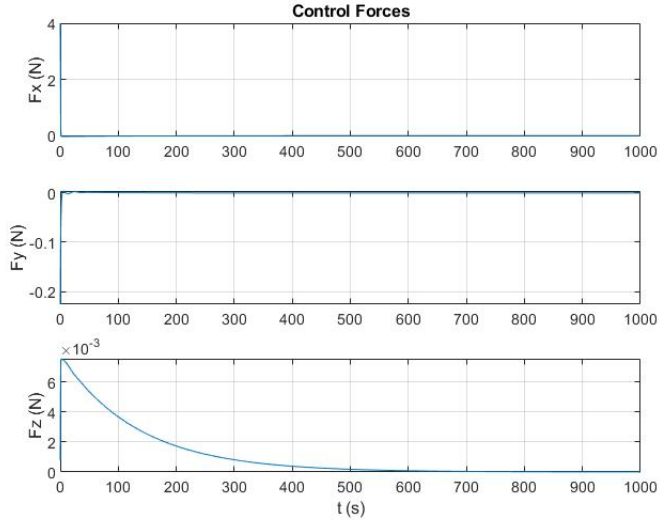


(b) *Chaser Attitude relative velocity, NN control*

Figure 5.3: Chaser-Target relative velocity, nominal conditions.



(a) *PID control Forces.*



(b) *NN control Forces*

Figure 5.4: Control Forces, nominal conditions.

- Final position, along x-axis: $x_{f,PD}$=-0.0273 and $x_{f,NN}$=-0.0274;
- Final position, along y-axis: $y_{f,PD} = 8.0800 \cdot 10^{-5}$ and $y_{f,NN}$=0.0055;
- Final position, along z-axis: $z_{f,PD} = -1.0628 \cdot 10^{-4}$ and $z_{f,NN} = -1.8642 \cdot 10^{-4}$;
- Final velocity, along x-axis: $vx_{f,PD} = 2.0520 \cdot 10^{-4}$ and $vx_{f,NN} = 2.0438 \cdot 10^{-4}$;
- Final velocity, along y-axis: $vy_{f,PD} = -8.1847\dot{1}0^{-7}$ $vy_{f,NN} = -7.0538 \cdot 10^{-6}$;
- Final velocity, along z-axis: $vz_{f,PD} = 6.9225e \cdot 10^{-7}$ and $vz_{f,NN} = 5.5802 \cdot 10^{-7}$;

The error on the final position along x-axis is approximately of e_x=0.36%; this is an indication of the good quality of the network approximation.

5.2 Translational motion control: changing initial conditions

5.2.1 Simulation 1

The Neural Network controller was tested by changing the initial conditions, as in table 5.6 and an additional disturbing force has been added. The simulation time is 1000 seconds, all physical Chaser characteristics and orbital parameters remain the same. In this graphs, it is shown the trajectory in terms of position and velocity

Table 5.6: Initial conditions. Simulation 1

x (m)	y (m)	z (m)	v_x (m/s)	v_y(m/s)	v_z(m/s)	Disturbance Force (N)
-60	2	0	0.15	0.012	0	$-2 \cdot 10^{-6}$

of the Chaser spacecraft with respect to the Target controlled by the same NN controller, and its control forces. The spacecraft arrives at the desired point, even if the initial starting position are changed and a new disturbance force is added, with respect to the previous simulation. This demonstrates the adaptability of the NN controller to unseen external conditions. In figure 5.5 and 5.6 are represented the Chaser-Target relative positions and velocities;in figure 5.7 are represented the control forces. We can observe that the changes in the initial conditions and the added disturbance force brought a little oscillation in the first instants of the simulation, but the system reach the solution, nevertheless.The final positions and velocities reached are:

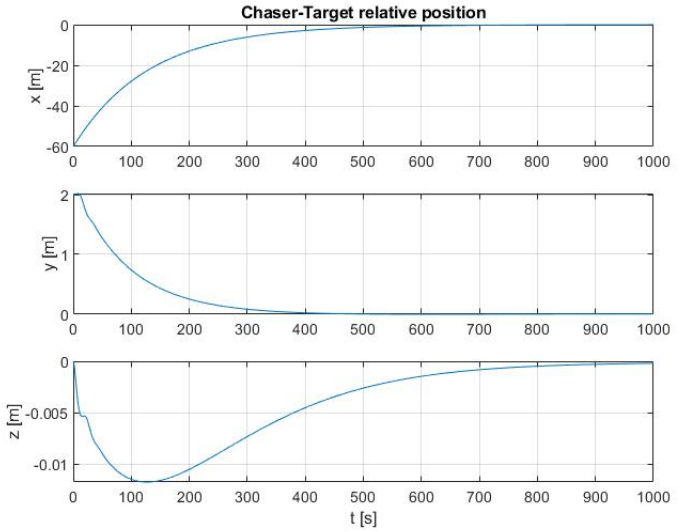


Figure 5.5: Chaser-target relative position.

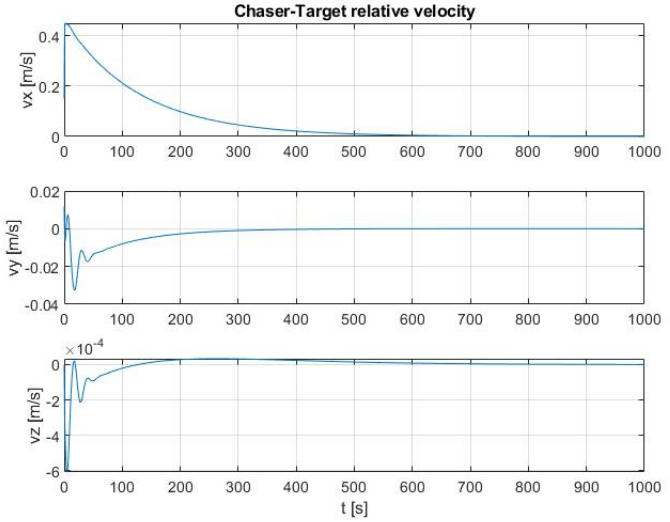


Figure 5.6: Chaser-target relative velocity.

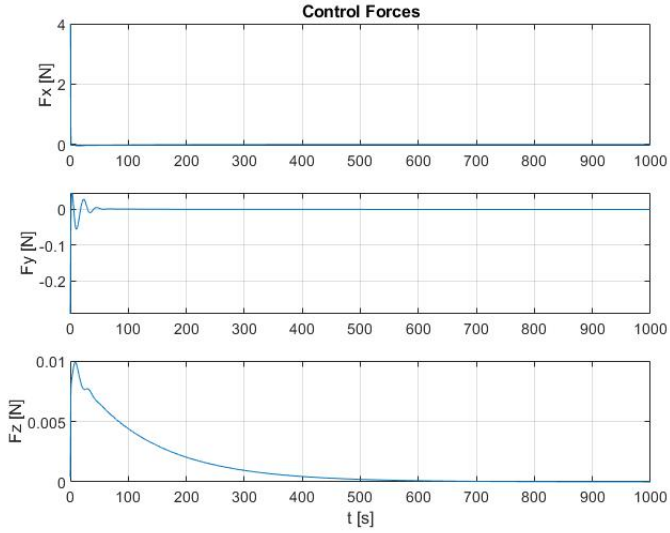


Figure 5.7: Chaser-target relative position.

- Final position, along x-axis: $x_{f,NN}$=-0.0316 m;
- Final position, along y-axis: $y_{f,NN}$=0.0053 m;
- Final position, along z-axis: $z_{f,NN} = -2.0060 \cdot 10^{-4}$ m;
- Final velocity, along x-axis: $vx_{f,NN} = 2.362 \cdot 10^{-4}$ m/s;
- Final velocity, along y-axis: $vy_{f,NN} = 8.7315 \cdot 10^{-6}$ m/s;
- Final velocity, along z-axis: $vz_{f,NN} = 6.5673 \cdot 10^{-7}$ m/s;

5.2.2 Simulation 2

In this second simulation, the initial conditions change again, as shown in table 5.7 and an additional disturbing force has been added, this time bigger than in simulation 1. The simulation time is 1000 seconds, all physical Chaser characteristics and orbital parameters remain the same.

Table 5.7: Initial conditions. Simulation 2

x (m)	y (m)	z (m)	v_x (m/s)	v_y(m/s)	v_z(m/s)	Disturbance Force (N)
-55	4	0	0.15	0.012	0	$-2 \cdot 10^{-5}$

- Final position, along x-axis: $x_{f,NN}$=-0.0323 m;
- Final position, along y-axis: $y_{f,NN}$=0.0056 m;
- Final position, along z-axis: $z_{f,NN} = -1.9503 \cdot 10^{-4}$ m;
- Final velocity, along x-axis: $vx_{f,NN} = 2.3929 \cdot 10^{-4}$ m/s;
- Final velocity, along y-axis: $vy_{f,NN} = 6.1970 \cdot 10^{-6}$ m/s;
- Final velocity, along z-axis: $vz_{f,NN} = 6.0907 \cdot 10^{-7}$ m/s;

In figure 5.8 and 5.9 are represented the Chaser-Target relative positions and velocities;in figure 5.10 are represented the control forces. As in the previous results, a little oscillation in the first instants of the simulation is present, but the system reaches the solution, in spite of the changed starting position/velocities and of the increased disturbance force.

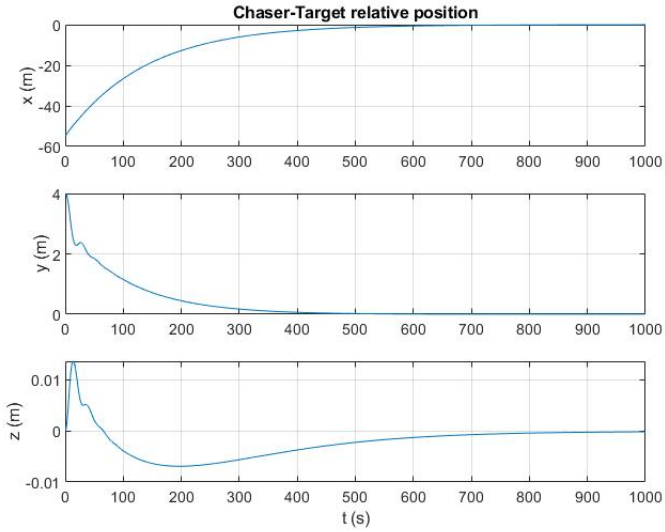


Figure 5.8: Chaser-target relative position.

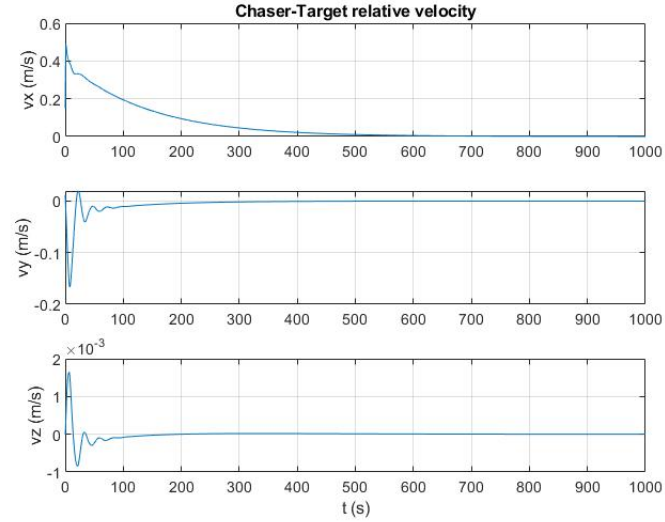


Figure 5.9: Chaser-target relative velocity.

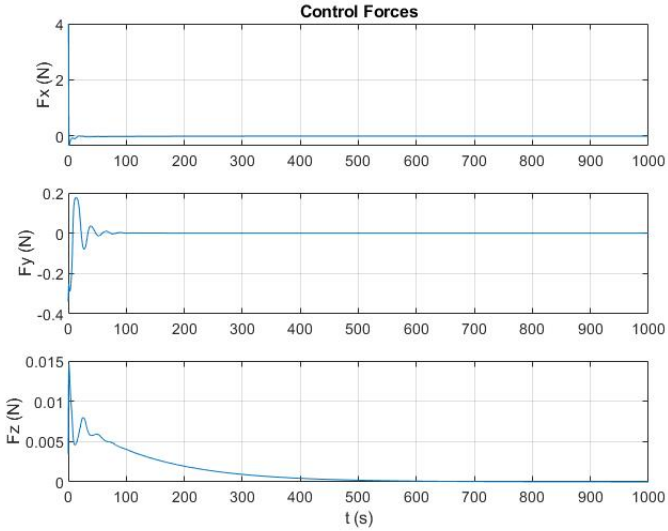


Figure 5.10: Chaser-target relative position.

5.2.3 Simulation 3

In this third simulation, the initial conditions change again, as shown in table 5.8 and the additional disturbance force is further increased. The simulation time is 1000 seconds, all physical Chaser characteristics and orbital parameters remain the same.

Table 5.8: Initial conditions. Simulation 3

x (m)	y (m)	z (m)	v_x (m/s)	v_y(m/s)	v_z(m/s)	Disturbance Force (N)
-30	4	0	0.15	0.015	0	$-2 \cdot 10^{-4}$

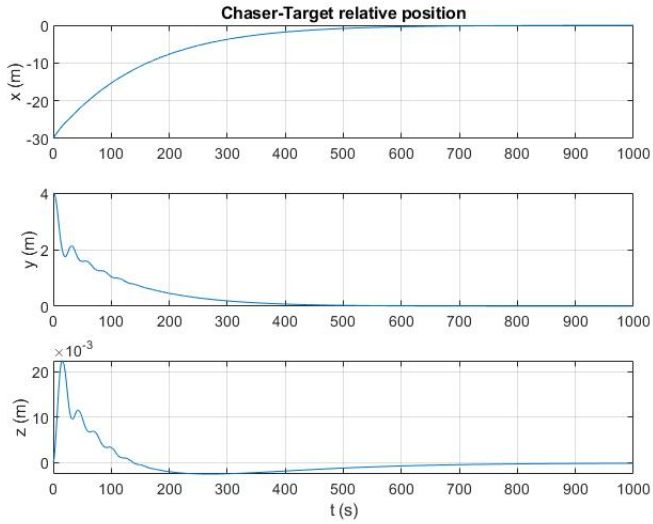


Figure 5.11: Chaser-target relative position.

- Final position, along x-axis: $x_{f,NN}$=-0.0233 m;
- Final position, along y-axis: $y_{f,NN}$=0.0064 m;
- Final position, along z-axis: $z_{f,NN} = -1.5862 \cdot 10^{-4}$ m;
- Final velocity, along x-axis: $vx_{f,NN} = 1.5121 \cdot 10^{-4}$ m/s;
- Final velocity, along y-axis: $vy_{f,NN} = 1.9565 \cdot 10^{-6}$ m/s;

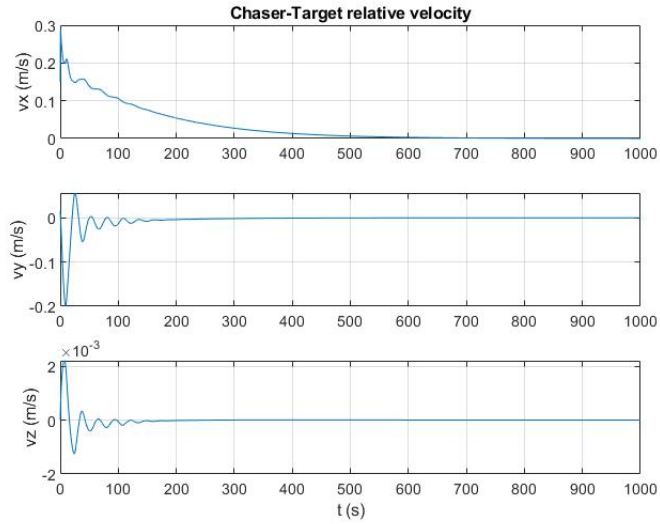


Figure 5.12: Chaser-target relative velocity.

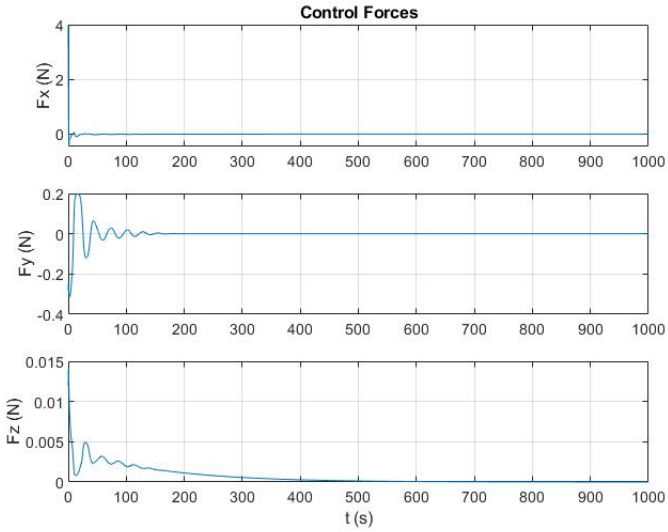


Figure 5.13: Chaser-target relative position.

- Final velocity, along z-axis: $vz_{f,NN} = 3.4401 \cdot 10^{-7}$ m/s;

In figure 5.8 and 5.12 are represented the Chaser-Target relative positions and velocities;in figure 5.13 are represented the control forces. In this case, the initial oscillation takes longer to be damped, but the system reaches acceptable solutions.

5.3 Attitude control: nominal conditions

In the following graphs, the attitude and attitude rate control is represented. It is shown the comparison between the satellite behaviour controlled by the PD-controller and by the Neural Network controller, in different conditions. In table 5.9, the initial conditions of the simulations are indicated, in terms of initial attitude and attitude rate.These results were carried out considering the input data in table 5.1 for the Chaser physical characteristics and the orbital parameters in 5.2; all the disturbance torques has been considered. The PD gains for attitude and attitude rate controller are all proportional, as indicated in 5.10. The same simulations, with the same initial conditions, is performed using, this time, the Neural Network control. The Neural Network controller was trained several times, with different sets of input/output obtained with PD-control, and the final net architecture has 5 hidden layers, as shown in figure 5.14. The simulation time is 1000 seconds, with a time step of 0.01.

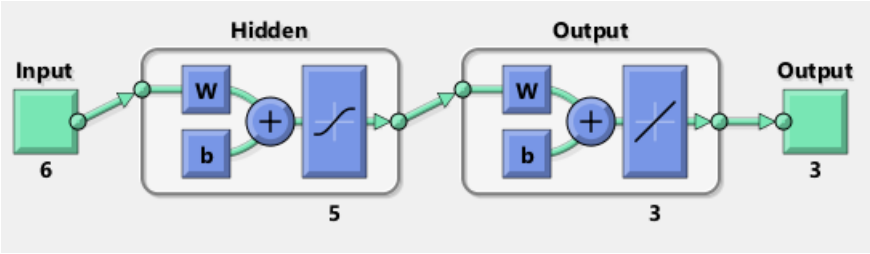


Figure 5.14: 5 hidden-layers Neural Network.

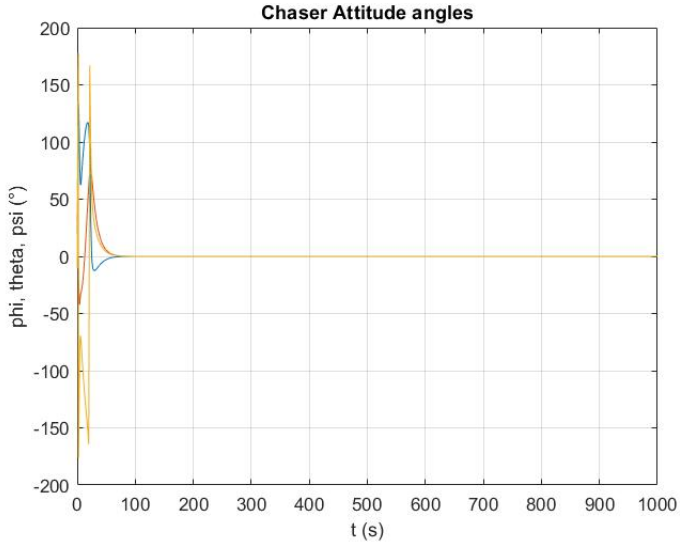
Table 5.9: Initial conditions.

ϕ (°)	θ (°)	ψ (°)	ω_x (rad/s)	ω_y(rad/s)	ω_z(rad/s)
20	30	-10	0.5	0.5	0.5

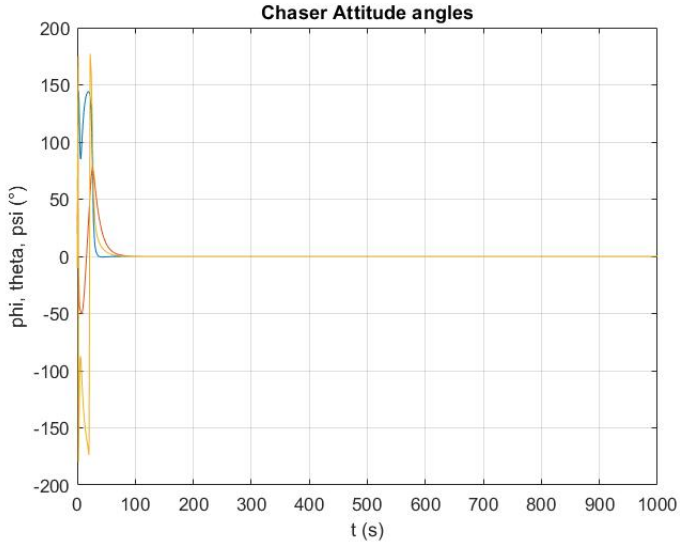
Table 5.10: Attitude PID-controller gains.

q	k_p	0.2
ω	k_p	-2

In figure 5.15 and 5.16 are shown the Chaser attitude and attitude rate controlled by PD-controller and by the NN controller. It is evident that the behaviour of the system under the two different controllers is basically identical. Also the control torques, in figure 5.17, follow the same trend. The dynamics is very fast

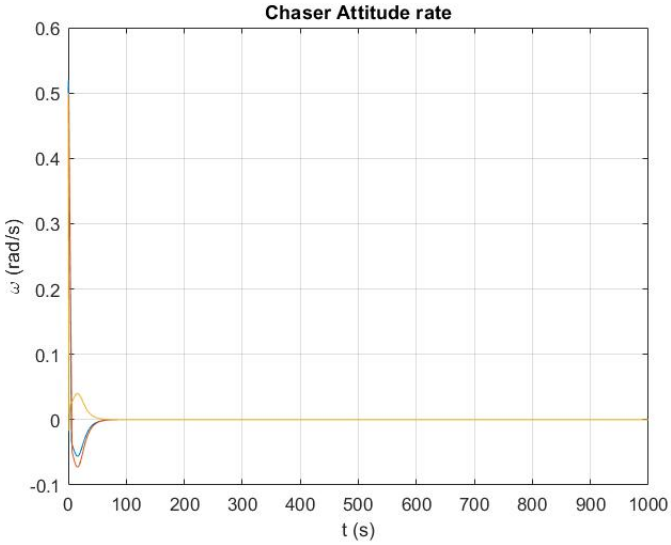


(a) *Chaser Attitude angles, PID-control.*

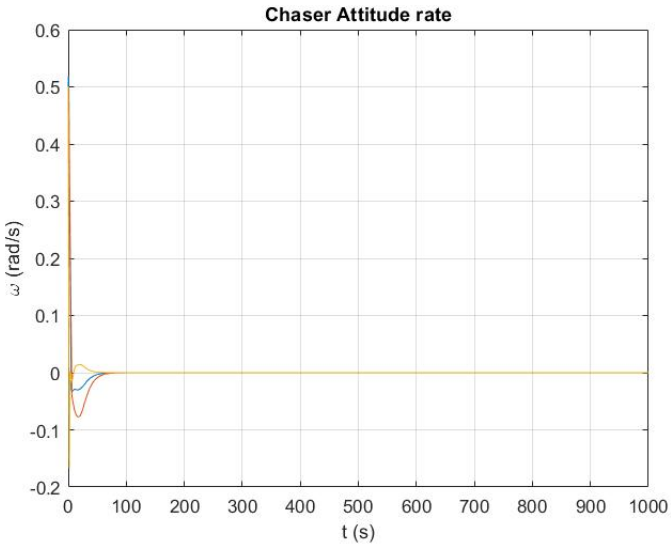


(b) *Chaser Attitude angles, NN-control.*

Figure 5.15: Chaser Attitude angles, nominal conditions.

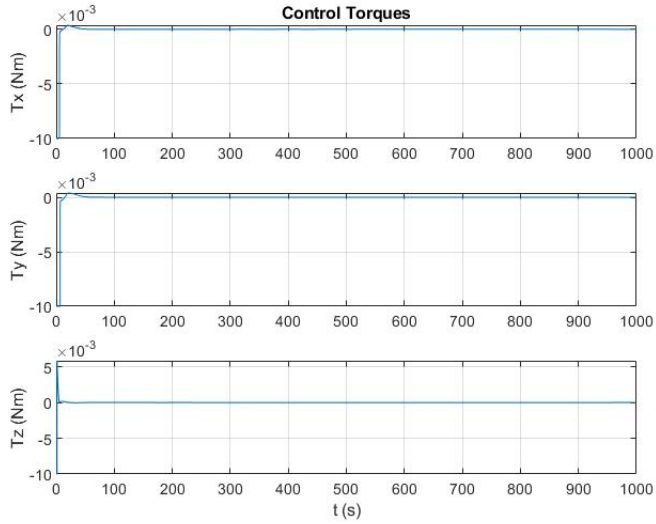


(a) *Chaser Attitude rate, PID-control.*

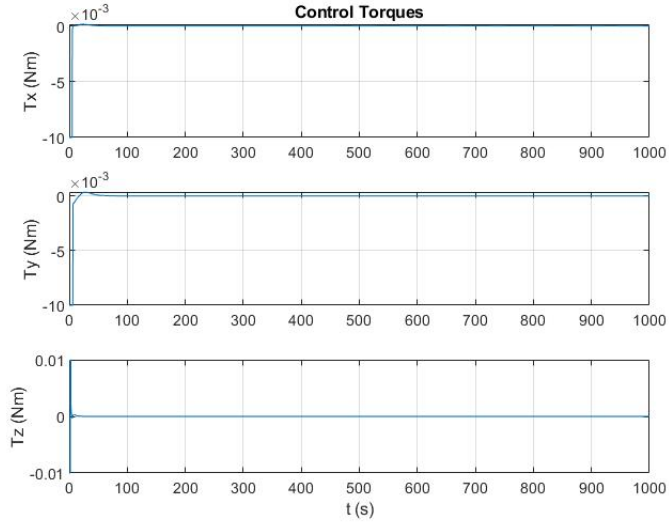


(b) *Chaser Attitude rate, NN-control.*

Figure 5.16: Chaser Attitude rate, nominal conditions.



(a) *Control torques, PID-control.*



(b) *Control torques, NN-control.*

Figure 5.17: Control torques, nominal conditions.

and the solution is reached in the first 100 seconds of the simulation, both on the attitude and attitude rate.

5.4 Attitude control: changing initial conditions

5.4.1 Simulation 1

The Neural Network controller was tested by changing the initial conditions, as in table 5.11 and an additional disturbing torque has been added. The simulation time is 1000 seconds, all physical Chaser characteristics and orbital parameters remain the same. In figure 5.18 and 5.19 the Chaser attitude and attitude rate

Table 5.11: Initial conditions.

ϕ (°)	θ (°)	ψ (°)	ω_x (rad/s)	ω_y(rad/s)	ω_z(rad/s)	Disturbance Torque (Nm)
15	21	-6	0.5	0.5	0.5	10^{-6}

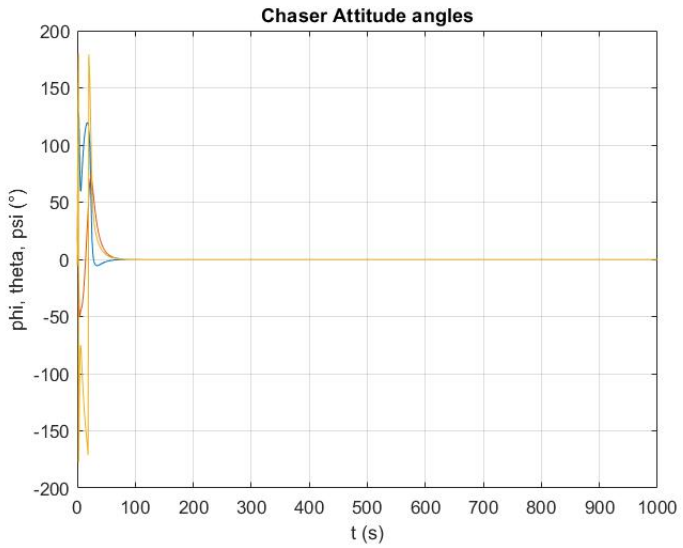


Figure 5.18: Chaser attitude angles.

are shown and in figure 5.20, the control torques are depicted. The NN controller is able to successfully control the Chaser attitude, despite the additional disturbance torque applied to the spacecraft and the changed initial conditions.

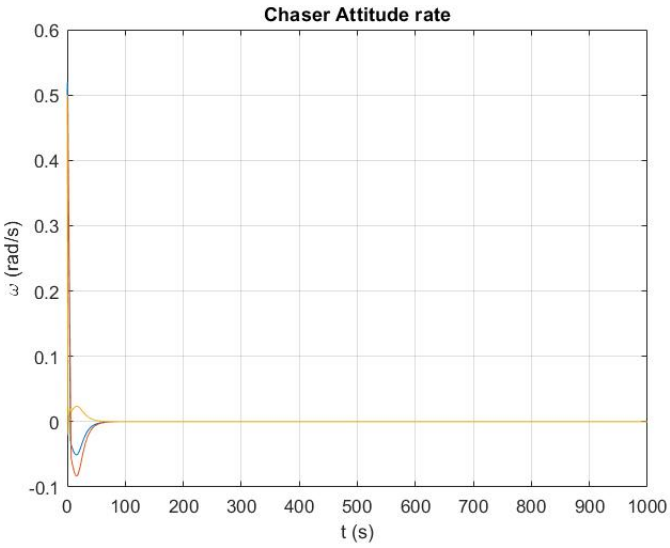


Figure 5.19: Chaser attitude rate.

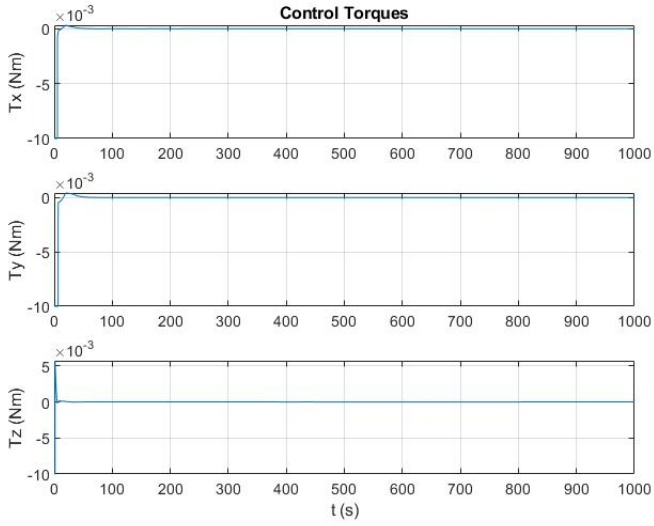


Figure 5.20: Control Torques.

5.4.2 Simulation 2

In this second simulation, the NN controller was tested by changing the initial conditions, as in table 5.12, and increasing the disturbing torque value. The simulation time is 1000 seconds, all physical Chaser characteristics and orbital parameters remain the same. In figure 5.21 and 5.22 the Chaser attitude and attitude rate are

Table 5.12: Initial conditions.

ϕ (°)	θ (°)	ψ (°)	ω_x (rad/s)	ω_y(rad/s)	ω_z(rad/s)	Disturbance Torque (Nm)
40	10	6	0.06	0.06	0.06	10^{-5}

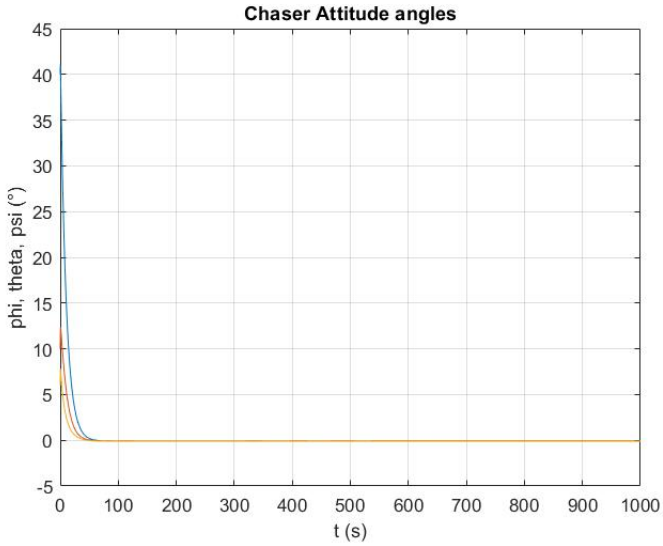


Figure 5.21: Chaser attitude angles.

shown and in figure 5.23, the control torques are depicted. The NN controller is able to successfully control the Chaser attitude, despite the additional disturbance torque applied to the spacecraft and the changed initial conditions. Since the initial angular velocity is much smaller than in the previous simulations, the solution converges faster.

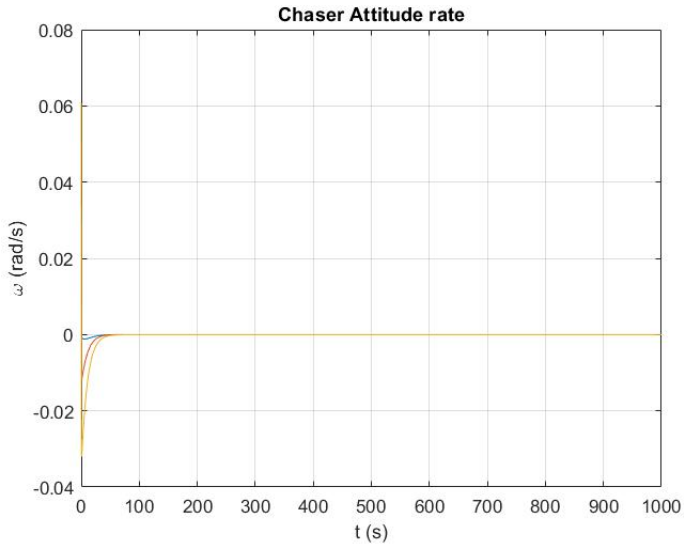


Figure 5.22: Chaser attitude rate.

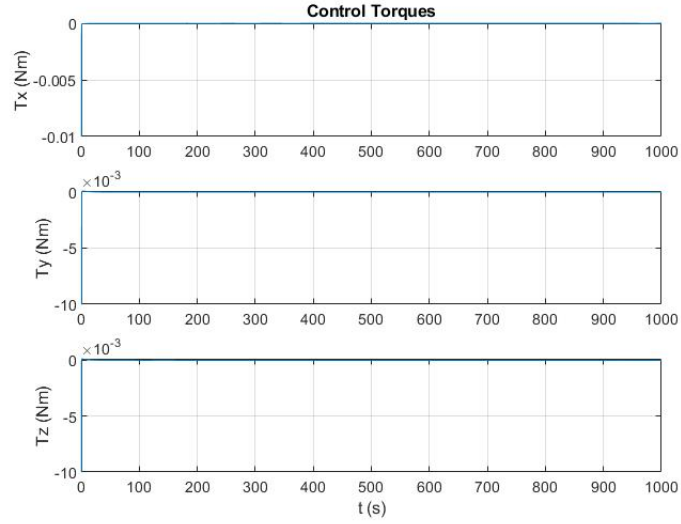


Figure 5.23: Control Torques.

5.5 Attitude control: adding sensor

In the same simulation model, a disturbing random signal was added to the quaternion vector, in order to simulate the presence of a sensor, to detect the current attitude.

5.5.1 Simulation 1

The initial conditions are depicted in table 5.13.

Table 5.13: Initial conditions.Simulation 1

ϕ (°)	θ (°)	ψ (°)	ω_x (rad/s)	ω_y(rad/s)	ω_z(rad/s)
40	10	6	0.06	0.06	0.06

- Disturbance Torque: 10^{-5} Nm;
- Sensor error:0.00001;

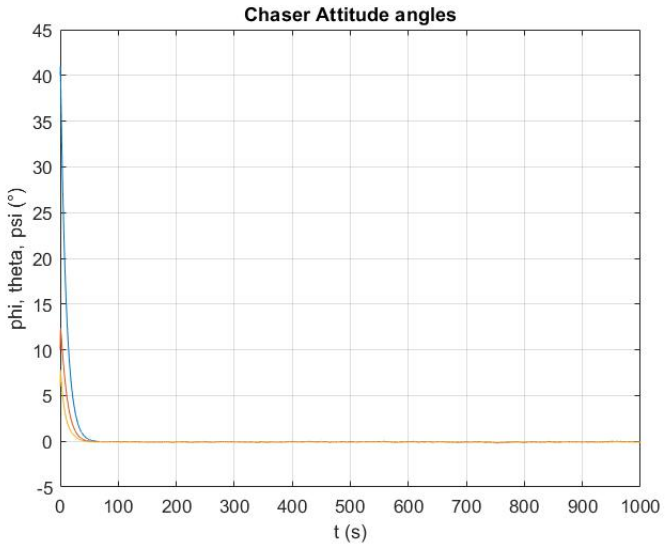


Figure 5.24: Chaser attitude angles.

In figure 5.24, 5.25 and 5.26 are represented the Chaser attitude, attitude rate and control torques, respectively. The error signal introduced by the sensor is

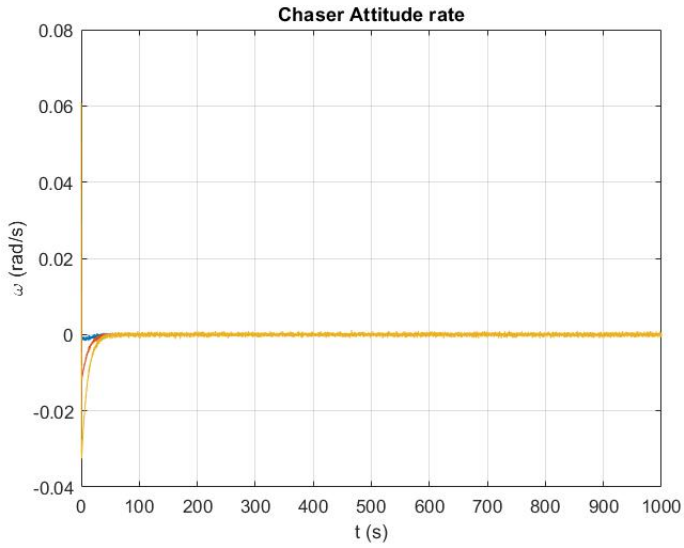


Figure 5.25: Chaser attitude rate.

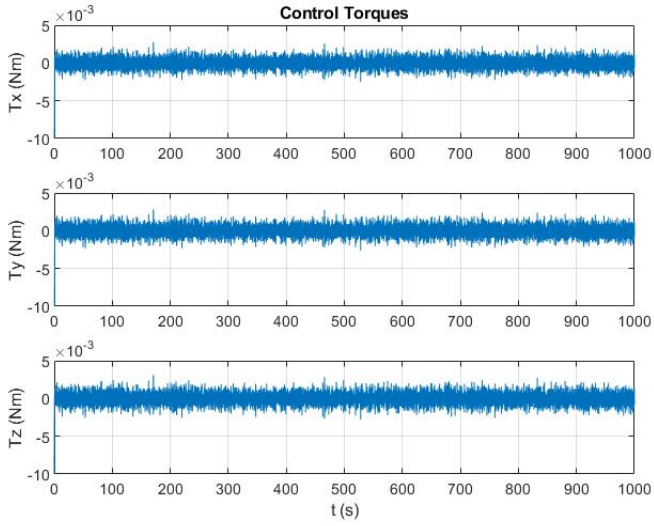


Figure 5.26: Control Torques.

evident in the control torques representation in figure 5.26. While the angular rate is subjected to persistent small oscillations around zero, in the attitude angles graphs, in figure 5.24, the angular oscillation is almost intangible. This results demonstrate that the NN control is able to control the system, even adding a noise signal on its inputs, which it was not trained for.

5.5.2 Simulation 2

In this second simulation, the error signal was augmented. The initial conditions are shown in table 5.14.

Table 5.14: Initial conditions. Simulation 2

ϕ (°)	θ (°)	ψ (°)	ω_x (rad/s)	ω_y(rad/s)	ω_z(rad/s)
50	15	8	0.06	0.06	0.06

- Disturbance Torque: 10^{-5} Nm
- Sensor error: 0.0001

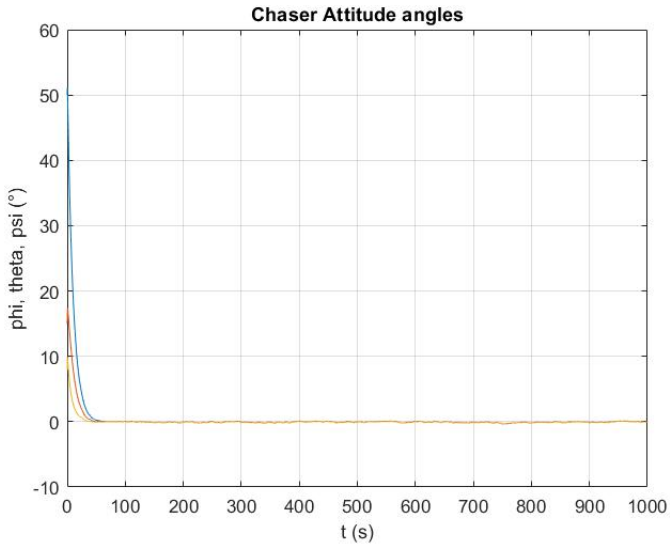


Figure 5.27: Chaser attitude angles.

In figure 5.27, 5.28 and 5.29 are represented the Chaser attitude, attitude rate and control torques, respectively. The error signal introduced, here, by the sensor is more evident in the control torques representation in figure 5.26. Now, the oscillations on the angular rate and attitude angles are more evident, although still small.

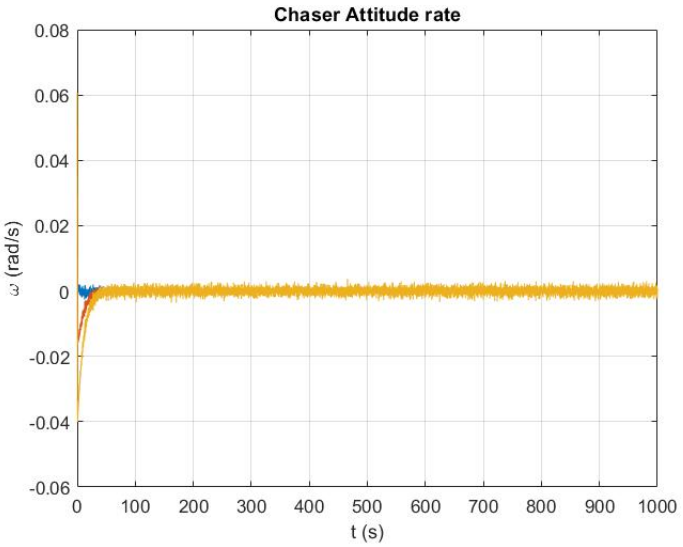


Figure 5.28: Chaser attitude rate.

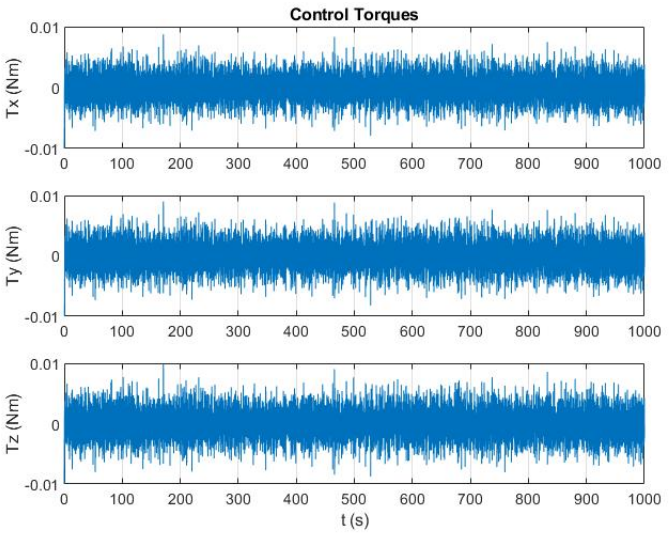


Figure 5.29: Control Torques.

5.6 Discussion

The results carried out in this chapter, thanks to Matlab/Simulink modelling of the system dynamics and controllers, shows that the neural network trained with PD-controller data sets of trajectory and rotational motion is able to control the same system almost with the same level of accuracy, with a reasonably small error. If the external conditions change, the NN controller is able to control the system. The translational motion simulations demonstrate that, if the initial positions and velocities change or if a disturbance force is added to the system, the NN control is able to control the system with the same level of accuracy, expect for a brief oscillations is the first instants of the simulations. Also increasing the force value, the controller is able to reach the desired solutions.
The rotational motion simulations show that the NN controller is able to imitate the PD behaviour and the attitude angles and angular velocities reach the expected solutions is less then 100 seconds. Also in this case, varying the initial conditions of angular position and velocity, the network is able to give the same results. Adding another disturbance torque, of increasing value, is not cause of controller malfunction.
For the rotational motion control, a sensor error was taken into consideration. To show its presence, a noise random signal was added to the quaternion vector. From the results, it is evident that, although with small and persistent oscillations, the system converges to the expected solution.
From all this results, it can be assumed that the Neural Network was reasonably well trained and that it has gained the ability to generalize.

Chapter 6

Conclusion

The present thesis was based on the development of an attitude and trajectory control strategy for a rendez-vous and docking manoeuvre between two small satellites. The proximity manoeuvre mission was described defining the mission profile, requirements and environment: the mission involves two 6U CubeSats, a Chaser which performs a rendez-vous towards a cooperative Target satellite. The mission requirements and constraints are guided by the design of the ADCS and GN&C subsystems, that demands very high accuracy. The complexity of the mission was addressed, taking into consideration the disturbance forces and torques acting on a spacecraft in LEO and considering the small dimensions and characteristics of the CubeSats satellites.
The main focus of this work was the implementation of a control algorithm able to control the trajectory and attitude of the Chaser spacecraft in its path towards the Target, from the last hold point to the mating point. A control algorithm based on deep learning neural network was made, based on PID-control of the same system. All the results were carried out by developing a Matlab®/Simulink® model of the system, based on the translational and rotational dynamics equations describing the spacecraft motion.
Neural network controllers for trajectory and attitude control were developed thanks to the Matlab® *Neural Network Toolbox*, a tool that allows the user to control and manage all the processes involved in the definition, training and testing of the neural network with a set of input/output data, present in the Matlab® workspace. In training the neural network was important to avoid overfitting, i.e. when the network learnt too well a single task and loses its capacity to generalize from inputs. But, it was also important to avoid undertraining, meaning that the network did not have enough training sets or amount of data to learn its task. Depending on the problem, it is necessary to choose the appropriate number of hidden layers constituting the network architecture and to establish the more suitable amount

of training sets and size of the data sets. For the translational dynamics control, it was necessary to train the network multiple times, with different data sets, to obtain satisfying performance. Instead, for the rotational dynamics, the number of training sets could be smaller.
The results obtained show that the NN control is able to obtain the desired solutions, even when the inputs change, indicating its capability to generalize.
From the results obtained, it was evident that the more the neural network is trained, or trained on a larger data set, the better it works and responds to different inputs; but, it is also important to avoid overtraining. Moreover, the number of hidden layers that constitute the network architecture, is determinant in optimizing the network performance. The more the network architecture is articulated, the better can fit complex functions, with increased accuracy. If the neural network is well trained, it can adapt to different external conditions and control the spacecraft towards the target with high precision.
For future works, the Neural Network should be generated on the basis of data obtained from a different control algorithm and should be trained with larger data sets, to increase its adaptive performances.

Bibliography

[1] THE CUBESAT PROGRAM. http://www.cubesat.org/about. [Online].

[2] Roger R. Bate, Donald D. Mueller, and Jerry E. White. *Fundamentals of Astrodynamics*. 1971.

[3] Dan Campbell and Sherlock Campbell. Introduction to regression and data analysis, statlab workshop. 2008.

[4] S Corpino, Stefano Mauro, Stefano Pastorelli, F Stesina, Gabriele Biondi, Loris Franchi, and Tharek Mohtar. Control of a noncooperative approach maneuver based on debris dynamics feedback. *Journal of Guidance, Control, and Dynamics*, 41:1–18, 10 2017.

[5] Howard D Curtis. Orbital mechanics for engineering students. *Amsterdam: Elsevier Butterworth Heinemann*, 2005.

[6] Bonnie F. James, O. W. Norton, and Margaret B. Alexander. The natural space environment: Effects on spacecraft. *NASA; United States*, Nov 1994.

[7] Mathworks. Deep Learning Toolbox. https://it.mathworks.com/products/deep-learning.html. [Online].

[8] Mathworks. Fit Data with a Shallow Neural Network. https://it.mathworks.com/help/deeplearning/gs/fit-data-with-a-neural-network.html. [Online].

[9] MIT. Linear Regression. http://www.mit.edu/~6.s085/notes/lecture3.pdf. [Online].

[10] Hung Nguyen, Ram Prasad, Carol Walker, and Elbert A. Walker. A first course in fuzzy and neural control. 11 2018.

[11] J R. Wertz. *Spacecraft Attitude Determination And Control*. 01 1978.

[12] Murat Sazli. A brief review of feed-forward neural networks. *Communications, Faculty Of Science, University of Ankara*, 50:11–17, 01 2006.

[13] Amir Suratgar, Mohamad Bagher Tavakoli, and Abbas Hoseinabadi. Modified levenberg-marquardt method for neural networks training. *Proceedings - Wec 05: Fourth World Enformatika Conference*, 6, 01 2005.