POLITECNICO DI TORINO

Master degree course in Aerospace Engineering

Master Degree Thesis

Design and verification of an adaptive control system for small satellites involved in Rendez-vous and docking missions.

Supervisors
Prof. Sabrina CORPINO
Eng. Fabrizio STESINA

Candidate
Gabriele AQUINO
Matricola: 243881

a.a. 2018/2019

Abstract

The thesis aims at studying the rendez-vous and docking operations between two 6U Cubesats, Missions of RVD between cubesats are expected in the next future, enabling these small satellites for a large set of missions both in Earth Orbit and for interplanetary explorations. The challenge of these missions stays in the accurate knowledge of the attitude and relative position of the spacecraft and the accuracy of the manoeuvres control, taking into account safety margins.

In particular, the thesis deals with an innovative control system of the rotational and linear motion of the chaser spacecraft, based on the Artificial Neural Network (ANN).These networks act like the human neurons and learn how to reacts to specific inputs thanks to a process of learning. In this case, this training session has been conducted with a Linear Quadratic Regulator (LQR) controller. This solution confers versatility to the control and the capability to adapt the control, strategy to sudden disturbances, uncertainties and misbehaviours.

The first part of dissertation describes the objective and the high level requirements of the proximity operations mission and the constrains towards the subsystems. Then, the spacecraft GNC and ADCS architecture are presented with focus on their components. The second part of the first chapter deals with the design of the controllers, reporting the theoretical aspects related to the control techniques traditionally adopted, with great interest posed on Linear Quadratic Regulator (LQR) and the proposed innovative solution based on the Artificial Neural Network (ANN).

In the second part of this thesis, the controller is then designed thank to the development of a mathematical models in Matlab/Simulink environment that represents the models: in particular, the relative dynamics of the satellites and their attitude are described, by adding all the forces and torques applied to the satellites and then the controllers are added . The last chapter presents the simulations results, conducted before with LQR controller and then with the controlled based on neural network.

Finally, the results in both the configurations are compared highlighting the cases for which these two control techniques can be an efficient to manoeuvre a satellite involved in a Rendez-Vous and Docking (RVD) manoeuvre.

Summary

Index of figures

List of tables

1. Introduction

As the title of this dissertation may suggests, the objective is to design a control system whom is able to perform a successful operation of autonomous rendez-vous and docking between two small satellites. This type of mission is becoming increasingly used to complete various tasks like removal of space debris, on-orbit servicing and spatial repairing.

When it comes to rendez-vous and docking (RVD) missions, it means a procedure where one object named the chaser, usually a satellite, comes more closer to another one of its kind until, through some certain type of manoeuvre, they attached together and became a single body (fig.1). In this paper, it is studied a RVD mission between two 12U satellites, or rather a 120 x 120 x 120-cm cubes. Every moves that the chaser (or the target) makes comes from one specific system: the Attitude Determination and Control System (ADCS). The objective of this system is to determine the attitude through the use of on-board sensors (gyroscopes, magnetometers, sun/earth sensors), while the control of the attitude is performed using actuators (reaction wheels, thrusters and magnetorquers) combined with a non-linear control system in order to steady the satellite along the three axis.

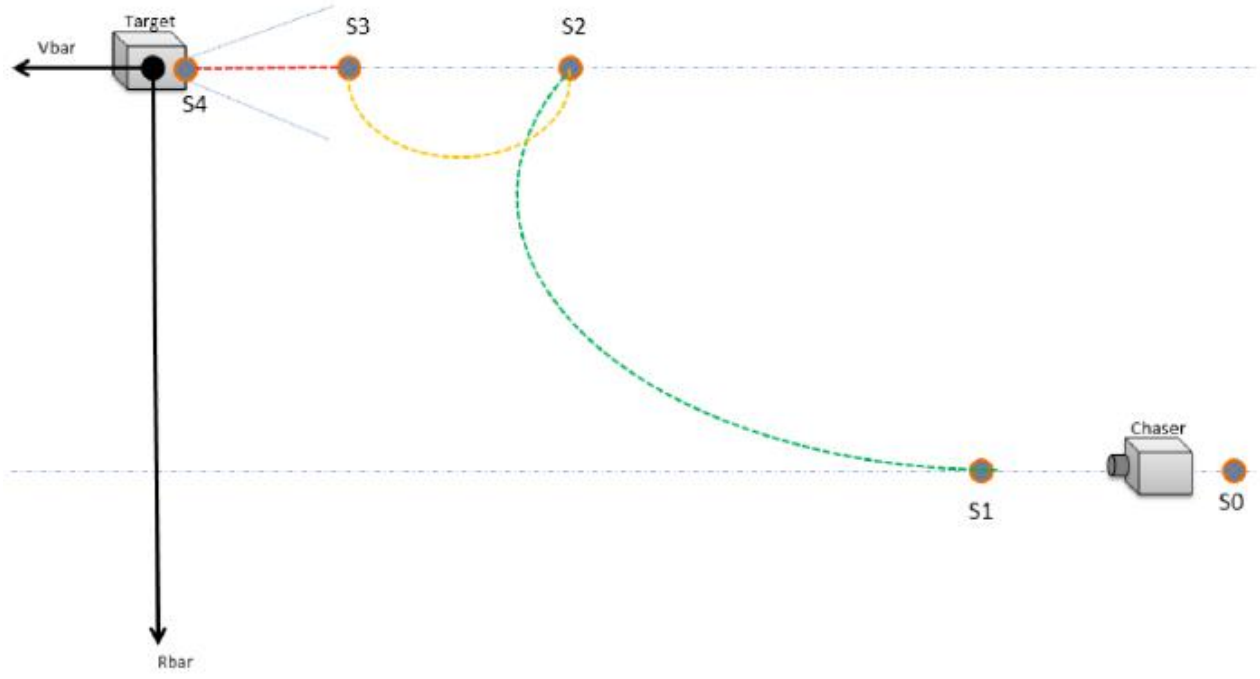

Figure 1: Rendez-vous and docking phases.

1.1 Mission description

The mission took in exam in this dissertation will consider the Target a cooperative target, i.e. that its attitude and position are known and it stands still in space. The starting manoeuvre is the

deployment one: the two CubeSats are attached to the ISS and with this operation can set them free. After that, there is the separation between the Chaser and the Target: the first one moves away until it reaches a distance of 10 km. The third and the last phase is the approach one: the Chaser moves 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 study starts during this last phase of the mission, the RVD one: this implies that the distance s_0 between the Chaser and the Target is approximately around 200 m or below. With the aim of simplify the mathematical equation, the orbit will be considered circular and the proximity operations will be performed on an orbit close to the one of the ISS. In this case, the orbit has a radius of 400 km and an inclination of 51.64 degrees. Nevertheless, the developed model can work both for any LEOs and without involving the ISS.

1.2 Attitude and orbit control system

In order to control the movement and the position of the spacecraft during the developing of the mission, there are two subsystems in charge: the Attitude Determination and Control System (ADCS) and the Guidance, Navigation and Control System (GN&C). Their duties, for example, can comprehend the alignment of the solar panels toward the Sun or pointing the antenna correctly for data transmission.

1.2.1 GN&C

This subsystem has to function to maintain and change both the position and the velocity of a spacecraft. This is allowed by exploiting three functions:

1. Navigation: this function regulates the position and the velocity of the vehicle, so it involves the on-board sensors;
2. Guidance: it is the controller function;
3. Control: using the actuators, it can change the position and the velocity thanks to information coming from navigation and guidance.

The three function works together: the guidance function compares the actual velocity, provided from the sensors of the navigation system, to the desired one. So, it sends command to the actuators, which apply a force to the spacecraft. It is necessary to have a brief discussion on the actual sensors and actuators used in todays CubeSats.

1.2.1.1 Attitude sensors

The sensors listed below are used both for target as chaser satellites in order to acquire attitude information.

- **Sun sensor**: Sun Sensor on a Chip (SSOC) is a two-axis and low-cost sun sensor for high accurate sun-tracking, pointing and attitude determination (Fig.2). The device measures the incident angle of a sun ray in two orthogonal axes, providing a high sensitivity based on the geometrical dimensions of the design.

Figure 2: nanoSSOC-D60 digital sun sensor.

Its performances are Tab.1: "Technical specifications of nanoSSOC-D60" (https://www.cubesatshop.com/product/nanossoc-d60-digital-sun-sensor/):

nanoSSOC-D60		
Parameter	**Value**	**Comments**
Sensor type	2 axes	Orthogonal
Field of view (FOV)	± 60°	Angular size of the view cone
Accuracy	< 0.5°	3σ error
Precision	< 0.1°	
Supply voltage	3.3 V	5V under request
Average consumption	< 21 mA < 23 mA	Dark Light
Temperature range	-30 to + 85 °C	
Weight	6.2 g	

Table 1: Technical specifications of nanoSSOC-D60.

Another useful component is the one fed by the New Space System company. The top of the gamma is the NSS CubeSat Sun Sensor (Fig.3). The device produces four analogue voltages that are dependant on the incident angle of sunlight in the horizontal and vertical directions. Each sensor is supplied with a calibration algorithm that calculates the sun vector from these four voltages to an accuracy of +/-0.5 degrees. Its performances are listed in Tab.2: "Technical performances of NSS CubeSat Sun Sensor" (https://www.cubesatshop.com/product/nss-cubesat-sun-sensor/).

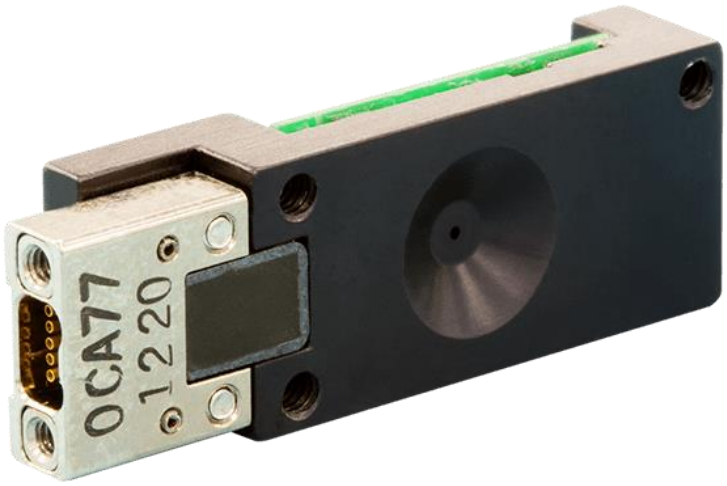

Figure 3: NSS CubeSat Sun Sensor.

NSS CubeSat Sun Sensor.		
Parameter	**Value**	**Comments**
Sensor type	2 axes	Orthogonal
Field of view (FOV)	114°	
Accuracy	< 0.5°	3σ error
Precision	< 0.1°	
Supply voltage	5 V	
Average consumption	< 10 mA	
Temperature range	-25 to + 50 °C	
Weight	< 5 g	

Table 2: Technical performances of NSS CubeSat Sun Sensor.

- **Magnetometer**: A magnetometer is a crucial part of a satellite attitude determination and control system, both for detumbling and attitude determination. It was picked the NSS Magnetometer (Fig.4)**.** The sensor provides x-, y- and z-axes magnetic field component measurements, as well as a sensor temperature measurement which is used for the temperature compensation of the magnetic field measurement. Ideally mounted outside the spacecraft at the end of a rigid boom the NewSpace Systems magnetometer includes low noise, precision processing and analogue-to-digital conversion circuitry; all of which improves the linearity and reduces the drift sensitivity of the sensor head. The integrated processing circuitry and sensor head provide the mission an accurate and stable magnetic field measurement at low power consumption.

Figure 4: NSS Magnetometer.

Technical specifications are listed in Tab.3: "Performances of NSS Magnetometer" (https://www.cubesatshop.com/product/nss-magnetometer/)

NSS Magnetometer	
Number of axis	Three, orthogonal
Axial alignment	< ±1°
Field measurement range	-60000 nT to +60000 nT
Noise density	< 8 nT rms/Hz at 1 Hz
Resolution	< 8 nT
Update rate	< 18 Hz
Power	< 550 mW
Power supply	5 V_{DC}
Thermal	-25 to +70 °C
Dimensions	96 mm x 45 mm x 20 mm
Mass	< 65 g

Table 3: Performances of NSS Magnetometer.

Alternatively, it can be used the NanoSense M315 Magnetometer, or simply M315, designed by GOMspace company (Fig.5). It is a reliable and fast magnetometer, perfectly suited for nano-satellites that require high performance from ADCS. Thanks to its compact design, it can be mounted everywhere, especially in parts far away from magnetic disturbance sources. The M315 is a highly accurate and low noise sensor with a very low temperature dependency. However, it is important to note that it can be

difficult to achieve the lowest noise due to magnetic noise in near field environment. Its performances are listed in Tab.4: "NanoSense M315 Magnetometer" (https://satsearch.co/products/gomspace-nano-sense-m315-magnetometer).

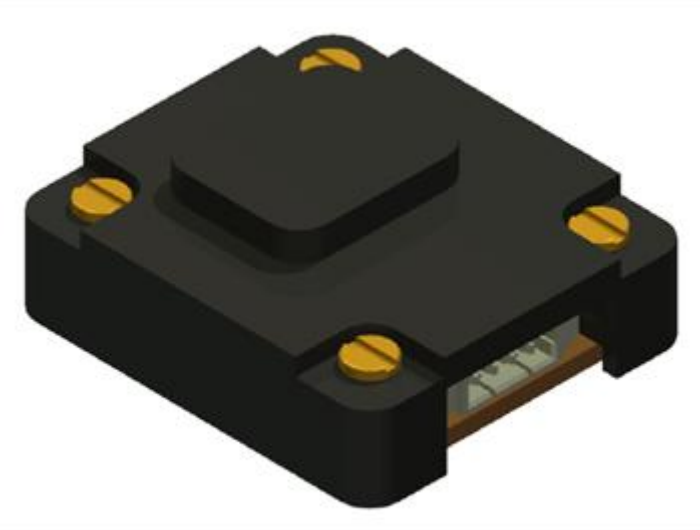

Figure 5: NanoSense M315 Magnetometer.

M315	
Number of axis	Three, orthogonal
Linearity (at ±200 µT)	0.5 %
Field measurement range	-800 µT to +800 µT
Noise (1-sigma)	15 nT
Frequency	100 Hz typical; 400 Hz max
Sample frequency	140 Hz
Thermal	-40 to +85 °C
Mass	< 8 g
Size	23 x 20 x 8 mm
Supply voltage	3.3 V typical; 5 V max

Table 4: NanoSense M315 Magnetometer.

- **Star tracker**: the first Star tracker took in consideration was the MAI-SS Space Sextant (Fig.6). The MAI-SS Star Tracker is low in cost and intended for CubeSat and NanoSat missions. The unit is completely self-contained and features lost in space star identification. Moreover, the system contains algorithms to provide sun avoidance for baffle-less operations.

Figure 6: MAI-SS Star Tracker.

Performances Tab.5: “MAI-SS Star Tracker specifications” (https://satsearch.co/products/gomspace-nano-sense-m315-magnetometer)

MAI-SS Star Tracker	
Accuracy (Cross Axis / Boresight)	4 arcsec / 27 arcsec
Acquisition time	130 ms Acq; 105 ms Track
Update rate	4 Hz
Sun exclusion w/wo buffle	45° / 90°
Operating temperature	-40 to 80 °C
Weight	170 g
DC Voltage	5 V
Average power consumption	1.5 W

Table 5: MAI-SS Star Tracker specifications.

Another strong candidate was the KU Leuven Star Tracker (Fig.7). The custom algorithms are optimized for accuracy, robustness and low computational cost. The star tracker has a built-in baffle to reduce the effect of stray light.

Figure 7: KU Leuven Star Tracker.

Technical specifications are reassumed in “Star Tracker Features” (Tab.6).

KU Leuven Star Tracker	
Accuracy (Cross Axis / Boresight)	2 arcsec / 10 arcsec
Acquisition time	130 ms Acq; 105 ms Track
Update rate	10 Hz
Sun exclusion w/wo buffle	40° / 90°
Operating temperature	-40 to 80 °C
Weight	250 g
DC Voltage	5 V
Average power consumption	< 1 W (nominal)

Table 6: Star Tracker Features (https://satsearch.co/products/new-space-systems-nsgy-001).

Figure 8: STIM300.

- **Gyroscope:** the gyroscopes, three for each satellite's axis, have the purpose to maintain the spacecraft aligned in a pre-determined position. In this case, it was chosen the STIM-300 from Sensonor (Fig.8). STIM300 is a small, tactical grade, low weight, high performance non-GPS aided Inertial Measurement Unit (IMU). It contains 3 highly accurate MEMS gyros, 3 high stability accelerometers and 3 inclinometers. The IMU is factory calibrated and compensated over its entire operating temperature range.

  Its performances are provided in the following table "STIM300 data" (Tab.7):

STIM-300 Gyroscope	
Angular rate	±400 °/s
Power supply	5 V
Operating temperature	-40 to +85 °C
Power consumption	1.5 W, max: 2 W
Bias Range	-250 to +250 °/h
Resolution	24 bits
Bandwidth (-3 dB)	262 Hz
Weight	< 55 g

Table 7: STIM300 data (https://www.cubesatshop.com/product/mai-400-reaction-wheel/).

In alternative, it can be used the *NSGY-001* by NewSpace System (Fig.9). It is a stella gyro that uses a COTS sensor and optics resulting in a very low cost attitude determination system that maintains accuracy during the eclipse phase. The NewSpace stellar gyroscope can be used to propagate a spacecraft's attitude from a known initial condition, without drift, while sufficient stars are common across frames. It can achieve this by using algorithms that tolerate noise and does not require a star database. It is thus far more robust against radiation damage than a standard star mapper solution would be if based on the same components.

Figure 9: NSGY-001.

Its performances are listed in Tab.8: "NSGY-001 characteristics" (https://www.cubesatshop.com/wp-content/uploads/2016/06/CubeWheel-large.png)

NSGY-001	
Power supply	5 V
Operating temperature	-40 to +85 °C
Power supply	5 V_{DC}
Power consumption	< 200 Mw (average)
Thermal (operational)	-25 to +50 °C
Rate estimation accuracy	≤ 0.20 degrees/s (boresight) ≤ 0.05 degrees/s (cross-boresight)
Maximum slew rate	≥ 1.00 degrees/s
Standard update rate	> 1 Hz

Table 8: NSGY-001 characteristics.

- **GPS Receiver:** the spacecraft needs the presence of a GPS receiver in order to give information about its position during its path toward the target. Because of that, the final choice is the *NSS GPS Receiver* (Fig.10). It is a 12-channel hardware-based receiver which utilizes a well-established GPS chipset. Targeted towards low-cost SmallSat constellations, it has been adapted for space altitude and velocity through the use of custom software modifications. The NSS GPS includes an unregulated, isolated 28 V power input and differential interfaces. It employs latch-up detection/ protection and a watchdog timer for increased reliability and robustness.
  Its characteristics are summarized in "NSS GPS Receiver performances" (Tab.9).

Figure 10: NSS GPS Receiver.

NSS GPS Receiver	
Position accuracy	< 10 m
Velocity accuracy	< 25 cm/s
Update rate	1 Hz
Operating frequency	L1 (1575.42 MHz)
Mass	110 g
Power	1 W
Thermal (operational)	-10 up to +50 °C
Power supply	3.3 V_{DC} nominal; 5 V_{DC} max

Table 9: NSS GPS Receiver performances (https://www.cubesatshop.com/product/nss-gps-receiver/).

Another choice can be the GPS receiver *piNAV-L1* (Fig.11) designed by NanoAvionics. It was designed with the aim to provide accurate position data with limited power and mass budget. In fact, it requires only 10% of power in comparison with conventional space-grade GPS ("piNAV-L1 features ", Tab.10).

Figure 11: GPS receiver "piNAV-L1".

NSS GPS Receiver	
Altitude	Up to 3600 km
Velocity accuracy	Up to 0.5 km/s
Update rate	1 Hz
Operating frequency	L1 (1575.42 MHz)

Mass	47 g
Power consumption	120 mW (typical)
Thermal (operational)	-10 to +50 °C
Power supply	2.7 V to 3.6 V

Table 10: piNAV-L1 features (https://n-avionics.com/wp-content/uploads/2018/07/piNAV-L1-new.jpg).

- **Cameras**: one of the possible choice is the *Chameleon Imager* by SAC (Fig.12). It is a compact CubeSat imager that takes advantage of the space-qualified electronics of the Gecko imager and combines this with high-performance optics to maximize imaging capability in small satellite. One of the biggest advantages is that the images are captured directly to the integrated mass stored; so, it means that there is no need of a additional payload storage capacity on the vehicle. Nevertheless it was thought for 3U CubeSat, it is available for larger satellite too.

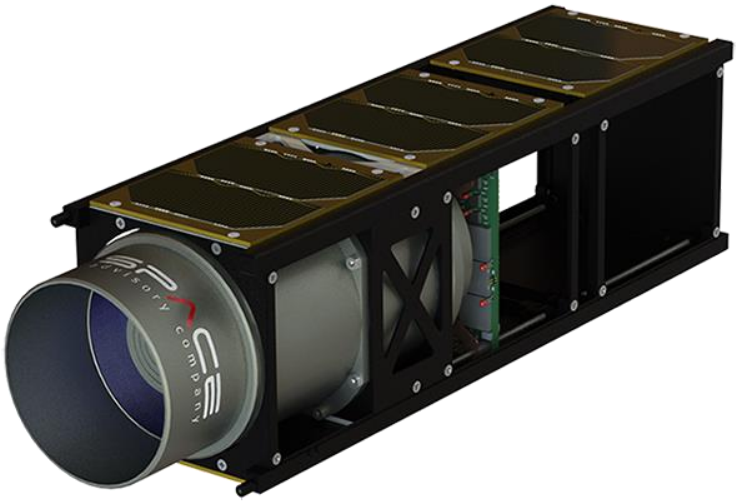

Figure 14: Chameleon imager.

Its performances are shortened in the table below ("Chameleon imager performances.", Tab.11):

Chameleon imager	
Spatial resolution @ 500 km	9.6 m PAN; 19 m MS; 29 m HS
Swath @ 500 km	Up to 32 km
Data format	10-bit or 20-bit
Integrated mass data storage	Up to 160 Gigabytes
Power usage	< 3.5 W (imaging mode) < 2.5 W (readout mode)
Mass (incl. electronics)	1.35 kg

Thermal	+10 to +30°C (operative) -20 to +70 °C (survival)
Spectral bands	Bayer RGB or PAN + 8 Multispectral bands or 150 band Hyperspectral

Table 11: Chameleon imager performances (https://www.cubesatshop.com/product/chameleon-imager/).

A strong competitor of the camera developed by SAC is the *C3D Cubesat Camera* designed by XCAM company (Fig.13). Originally developed in partnership with The Open University, C3D has successfully flown and transmitted images on CubeSat missions such as the UK Space Agency's UKube-1 and AlSat Nano. The C3D CubeSat camera system is an imaging payload which can be used for a variety of applications including Earth observation or, for the mission inquired in this thesis, RV&D missions.

Figure 15: C3D CubeSat Camera.

Its characteristics are shown in "C3D Specifications (Tab.12)" (http://www.xcam.co.uk/sites/default/files/styles/product_image/public/C3D%20board_3.jpg?itok=ih-jeKY3) :

C3D CubeSat Camera	
Spatial resolution @ 500 km	360 m GDS
Image Sensor	1.3 MP CMOS, 5/4 aspect ratio, RGB or B&W
Data format	8-bit raw and thumbnail (1:10)
Integrated mass data storage	16 MB SDRAM / 8 MB Flash
Peak power consumption	845 mW
Mass	85 g
Thermal	-25 to +65 °C (operating) -35 to +75 °C (survival)
Pixels	5.3 µm (1280 x 1024)

Wavelength range	400 – 650 nm (RGB, extended with B&W)

Table 12: C3D Specifications.

1.2.1.2 Attitude actuators

Being powerful enough to change the attitude and for other reasons like the lightness, it was chosen to take on-board only the reaction wheels, with the aim of actuators. However, in case the speed is very high, due to the conservation of the angular momentum, the amount of momentum they have to do with sometimes exceed the construction limit. So, they need to be desaturate through a process named "momentum damping" which includes the help of other actuators. These may be magnetorquers or thrusthers. In this case, it was picked the magnetorquer.

- **Reaction Wheels**; it was chosen to get on-board the MAI-400 Reaction Wheel (Fig.14). It is offered as a complete, standalone reaction wheel assembly unit for CubeSats and NanoSat missions.

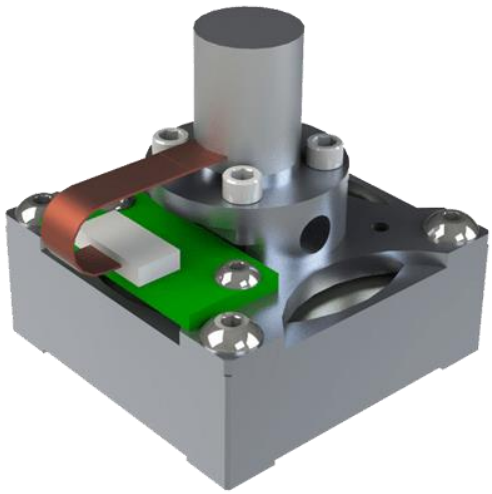

Figure 16: MAI-400 Reaction Wheel..

In the following table "MAI-400 specifications" (Tab.13), there are listed its performances (https://www.cubesatshop.com/product/mai-400-reaction-wheel/):

MAI-400 Reaction Wheel	
Maximum Torque	0.635 mNm
Momentum Storage @ 10000 rpm	11076 mNms
Mass	110 g
Operating temperature	-40 to 85 °C

Rotor dynamic balance	< 40 mg-mm
Maximum current	0.44 A
Steady state (500 RPM)	0.17 A
Idle	0.09 A
DC Voltage	5 V

Table 13: MAI-400 specifications .

Another choice can be using the CubeWheel reaction/momentum wheel from the CubaSpace company (Fig.15). Used to control the attitude of nanosatellites, the compact module includes a brushless DC motor with vacuum-rated bearings, as well as the require drive electronics and speed control algorithms.

Figure 17: CubeWheel (Large).

In "CubeWheel specifications"(Tab.14) are scheduled the specifications of this type of reaction wheel ("https://www.cubesatshop.com/wpcontent/uploads/2016/06/CubeWheel)

CubeWheel	**Small**	**Medium**	**Large**
Maximum Torque	0.23 mNm	1.0 mNm	2.3 mNm
Momentum Storage @ max rpm	1.7 mNms	10 mNms	30 mNms
Mass	60 g	140 g	220 g
Operating temperature	-10 to 70 °C	-10 to 70 °C	-10 to 70 °C
Peak power	0.72 W	< 1.5 W	< 2.2 W
Average power	0.12 W	< 0.24 W	< 0.27 W
DC Voltage	3.3 V	3.3 V	3.3 V

Table 14: CubeWheel specifications.

- **Magnetorquer**: this type of actuators, besides the duty of desaturate the reaction wheels, magnetorquers are largely used in CubeSats mission because they offer a method of controlling the attitude of a spacecraft. This can be achieved either directly, by interacting with the local Earth's magnetic field can provide coarse attitude pointing. One of the top choices is the NCTR-M012 Magnetorquer Rod by NewSpace (Fig.16). The use of a magnetic alloy rod produces an amplification effect over an air cored magnetorquer. This allows a system that uses less power, which is critical for CubeSat missions. The rods can enable a mission with increased manoeuvrability and reduced detumble rates.

Figure 18: NCTR-M012 Magnetorquer Rod.

Its performances are reassumed in "NCTR-M012 performances" (Tab.15) ("https://www.cubesatshop.com/wp-content/uploads/2018/05/NCTR-M012-Magnetorquer-Rod.png"):

NCTR-M012	
Magnetic moment	1.19 Am^2
Linearity	< ±5%
Residual moment	< 0.005 Am^2
Mass	< 50 g
Thermal (operational)	-20 to +60 °C
Power supply	5 V (DC)
Power	< 800 mW (nominal)

Table 15: NCTR-M012 performances.

Figure 21: NCTR-M002 Magnetorquer Rod.

NewSpace company provides another type of magnetorquer: the NCTR-M002 (Fig.17). Differences between the M002 magnetorquer and the precedent one can be seen in Tab.16: "NCTR-M002 specifications":

NCTR-M002	
Magnetic moment	0.2 Am^2
Linearity	< ±5%
Residual moment	< 0.005 Am^2
Mass	< 30 g
Thermal (operational)	-20 to +60 °C
Power supply	5 V (DC)
Power	< 200 mW (nominal)

Table 16: NCTR-M002 specifications ("https://www.cubesatshop.com/product/nctr-m002-magnetorquer-rod/").

- **Thruster(s)**: in order to allow the vehicle to move through the space, it needs to be endowed with a propulsion system, in particular with some thrusters, besides the solar panels. The first thruster selected is the *PM400* designed by Hyperion Technologies B.V. (Fig. 18). It is an high trust propulsion capability to 6-12U CubeSats and similar platforms. Low system complexity and zero propellant toxicity allow for simple and robust operations, both on the ground and when in orbit. The medium tank pressure and high storage density of liquid propellants enable high safety factor tanks to be used with little mass penalty. Its specification are reunited in "PM400 Specifications" (Tab.17)

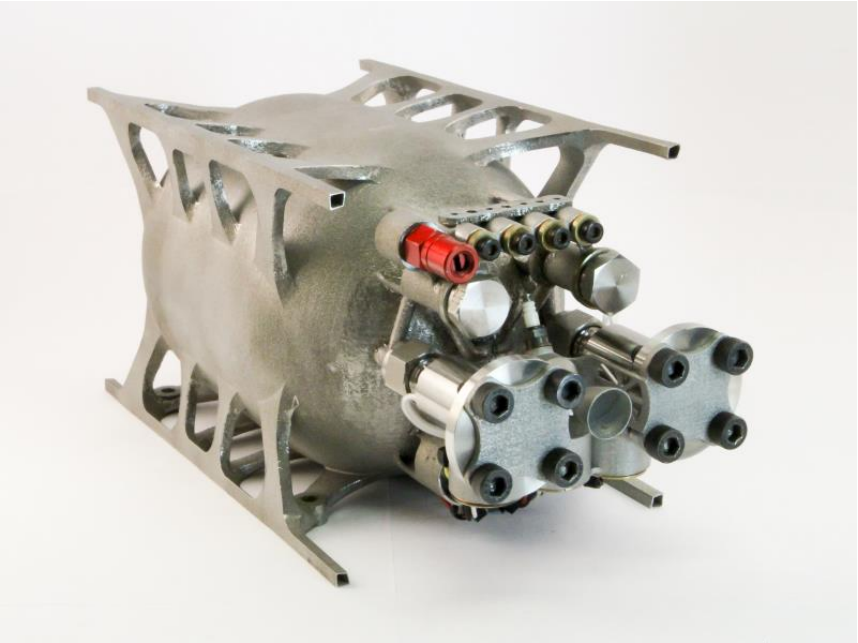

Figure 22: PM400.

PM400	
Total impulse	> 1750 Ns
Thrust	1 N
I_{sp}	> 285 s
ΔV	> 230 m/s
Operating temperature	-5 to +53 °C
Supply voltage	5 V
Power requiring	< 6 W (during firing) < 0.1 W (sleep)
Outer dimensions	200 x 100 x 100 mm
Nom. Propellant storage pressures	45 (Ox) / 7.5 (fuel) bar
Dry mass	< 1400 g
Propellant mass	625 g

Table 17: PM400 Specifications (https://satsearch.co/products/hyperion-technologies-pm400).

If thrust is expendable compared to weight, a great solution is to utilize the relative of PM400 from the same company: *PM200* (Fig.19). Its performances are reunited in Tab.18: "PM200 specifications" (https://satsearch.co/products/hyperion-technologies-pm200).

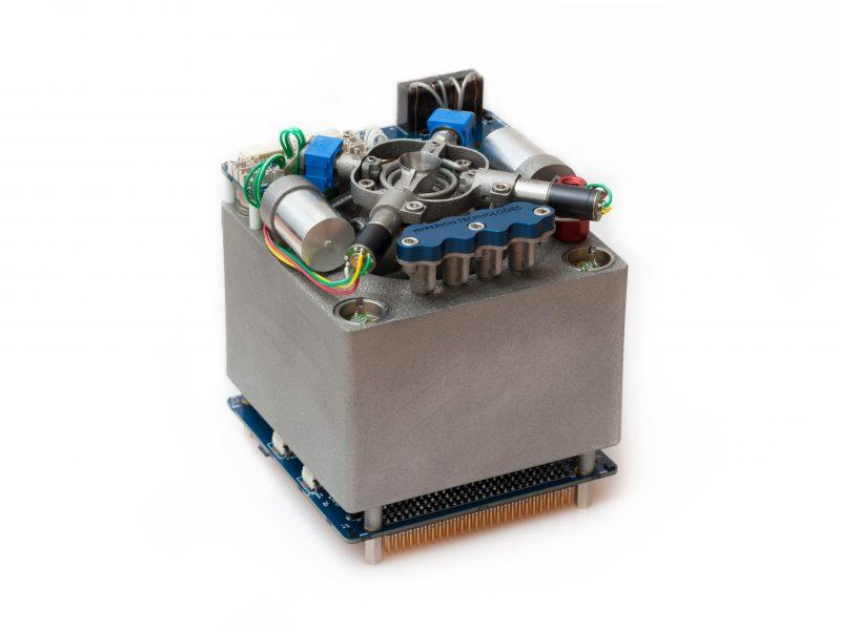

Figure 23: PM200.

PM200	
Total impulse	> 850 Ns
Thrust	0.5 N
I_{sp}	> 285 s
ΔV	> 230 m/s
Operating temperature	-5 to +35 °C
Supply voltage	5 V
Power requiring	< 6 W
Outer dimensions	< 0.1 W
Nom. Propellant storage pressures	45 (Ox) / 9 (Fuel) bar
Dry mass	1100 g
Propellant mass	310 g

Table 18: PM200 specifications.

Obviously, this last type of thruster can not be utilized in case of precision manoeuvre. For fulfil this type of task, the vehicle needs to rely on other kind of propeller. One of this genre is *IFM Nano Thruster* designed by Enpulsion (Fig.20). Used in many precedent missions like Rosetta, it is based on the Field Emission Electric Propulsion (FEEP) principle. It means that there is a field ionization from the tips of a porous tungsten crown at positive potential and consequent electrostatic acceleration of the Indium ions. One of the advantages is the much smaller size respect to similar thruster thanks to the use of solid propellant, which involves absence of a propellant management system. Its performances are listed in Tab.19: “IFM Nano Thruster specifications”

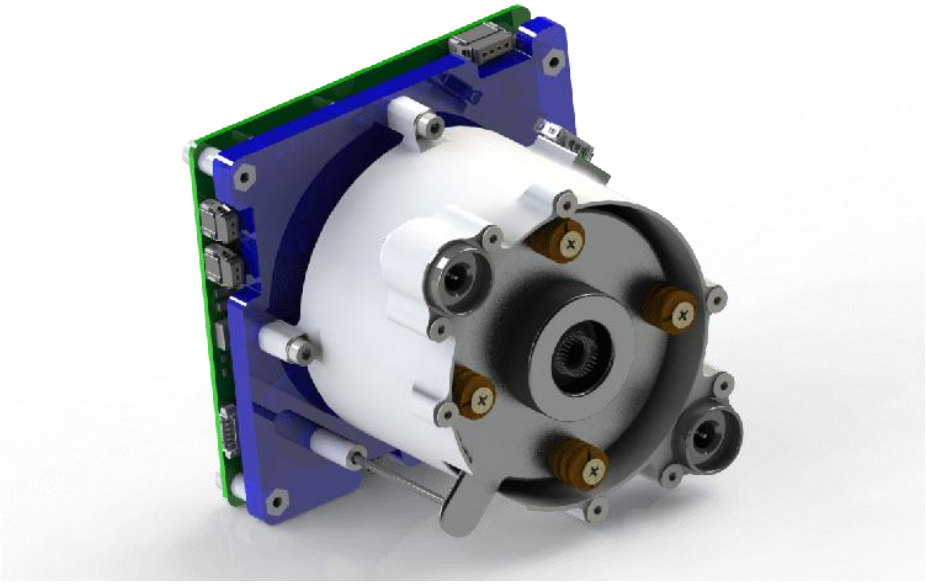

Figure 24: IFM Nano Thruster.

IFM Nano Thruster	
Total impulse	More than 5000 Ns
Thrust	350 µN (nominal)
I_{sp}	2000 to 5000 s
Dynamic thrust range	10 µN to 0.5 mN
ΔV	1038 m/s
Operating temperature	-20 to 50 °C
Supply voltage	12 V or 28 V
Power requiring	35 W
Outer dimensions	94 x 90 x 78 mm
Dry mass	640 g
Wet mass	870 g

Table 19: IFM Nano Thruster specifications (https://www.enpulsion.com/order/ifm-nano-thruster/).

A compromise between this type of propeller is *CubeSat Propulsion "EPSS"* developed by NanoAvionics (Fig.21). Suitable for satellites below 150 kg, it is based on "green" ADN mono-propellant. This type of thruster ensures a 6% higher specific impulse and 24% higher energy density compared to hydrazine employed system. Choosing this propeller can be the best fit in order to joining ESA's and NASA's new clean space initiatives. The performances of EPSS are specified in "EPSS performances " (Tab.20).

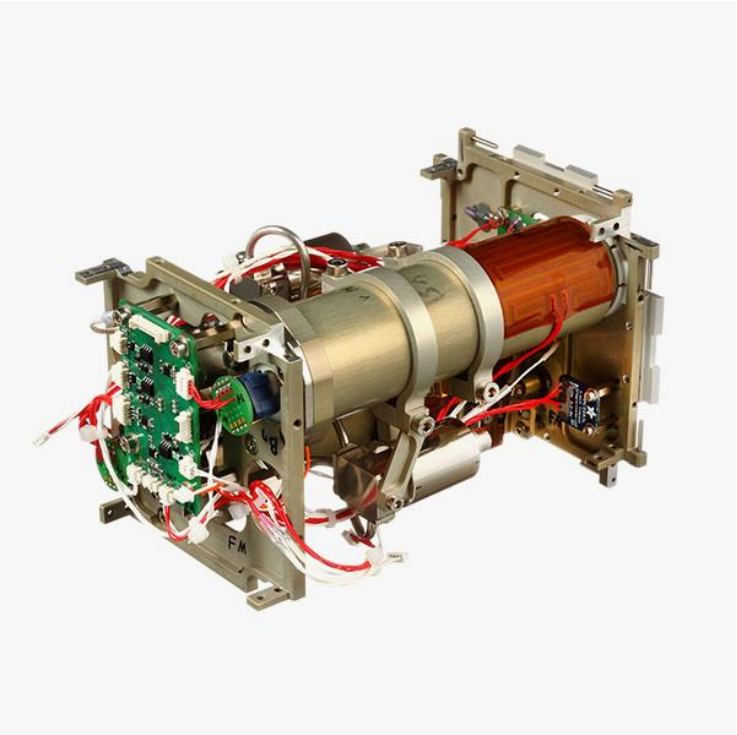

Figure 25: EPSS.

EPSS	
Propellant	AND Blend
Thrust	100 mN (nominal)
I_{sp} Vacuum	225 s
Min. Impulse Bit	0.002 Ns
Propellant throughput	1 kg
System pressure	10 bar
Chamber temperature	< 1600 °C
Power Consumption	0.05 W (Idle) 5-7.5 W (Peak) 4.5 W (Operational)

Table 20: EPSS performances (https://n-avionics.com/wp-content/uploads/2018/07/EPSS).

2. Control Theory

2.1 Dynamic system control

2.1.1. Dynamical system

In order to describe a dynamical system, it results necessary to find some functions which, changing in time, are useful to describe the evolution of the system. These functions are called signals and there two principal of them: the input signal *u(t)* and the output signal *y(t)*. With the aim of describe the model of the system, it is necessary to create a mathematical model, formed by an indefinite number of differential equations, which are useful to perform the control the system. Because the mathematical model is an approximation of the real one, there will be always two types of uncertainties: a dynamic and a parametric ones.

Like said before, in control theory the dynamic system is governed by a series of differential equations. If this system has definite dimension and it is continuous in time, it is necessary only a first order differential equations system in order to define the behave of the system. This description is named state equation and it is constituted by three fundamental variables:

1. The input signal $u(t) \in \mathbb{R}^n$;
2. The output signal $y(t) \in \mathbb{R}^n$;
3. The system state $x(t) \in \mathbb{R}^n$.

These three functions formed the late called first order differential equations system

$$\begin{cases} \dot{x}(t) = f[x(t), u(t); t] \\ y(t) = h[x(t), u(t); t] \end{cases} \tag{39}$$

Analysing the system, the first differential equation is the dynamic one because the development of the system is described through the presence both of the input time *t* and its evolution *t+dt*:

$$\dot{x}(t) = f[x(t), u(t); t] \Longrightarrow x(t + dt) = x(t) + f[x(t), u(t); t]dt \tag{40}$$

The second equation is obviously the static one, because the output signal is not derived in time.

With the aim of have an easier and better analyse of the system, it results crucial for practical reason to manipulate the state equation through the Laplace transform, in order to obtain the transfer functions. The first order differential equations system can be written in the following way:

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) + Du(t) \end{cases} \tag{41}$$

The first operation was to operate the Laplace transform to the first equation:

$$\begin{gathered} sX(s) - x(0) = AX(s) + BU(s) \\ (sI - A)X(s) - x(0) = BU(s) \\ X(s) = (sI - A)^{-1}x(0) + (sI - A)^{-1}BU(s) \end{gathered} \tag{42}$$

Differently, the second equation was transformed in the following way:

$$Y(s) = CX(s) + DU(s) \tag{43}$$

Combining (42) with (43), the second equation became:

$$Y(s) = C(sI - A)^{-1}x(0) + [C(sI - A)^{-1}B + D]U(s) \tag{44}$$

It was assumed that *x(0) = 0* in order to find a fundamental function: the transfer one *G(s)*.

$$\begin{gathered} Y(s) = G(s)U(s) \\ G(s) = C(sI - A)^{-1}B + D \end{gathered} \tag{45}$$

The transfer function can be written in a rational representation (?), defining:

1. K_G as the gain;
2. $z_1 \ldots z_m$ as G(s) zeros;
3. $p_1 \ldots p_m$ as G(s) poles.

$$G(s) = \frac{b_m s^m + b_{m-1}s^{m-1} + .. + b_1 s + b_0}{s^n + a_{n-1}s^{n-1} + .. + a_1 s + a_0} = K_G \frac{\prod_{i=1}^{m}(s - z_i)}{\prod_{i=1}^{m}(s - p_i)} \tag{46}$$

2.1.1.1 Dynamical system control

The control of a dynamic system expect to make the output signal *y(t)* to be similar to a reference signal, named desired signal *r(t)*. In order to do that, it is necessary to impose a command signal *u(t)* to the system. Like said in the previous chapter, it is possible to distinguish two type of control system: the open-loop control and the closed-loop one. The first one, named "non-feedback controller" too, has the particularity that the control action is independent from the output and does not modifies its values using a disturbance signal (fig.52).

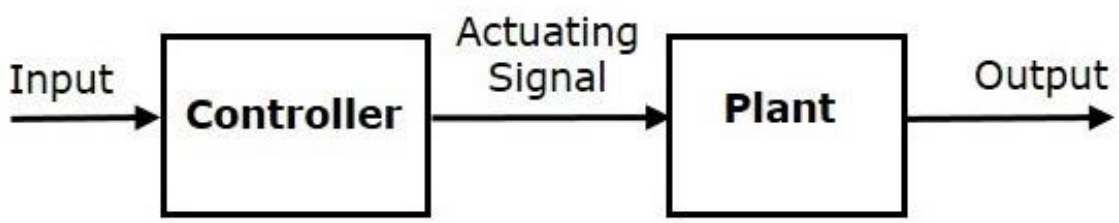

Figure 26: Open-loop controller.

The closed-loop control introduce, instead, a feedback signal that comes from the output and adjusts directly the input signal (fig.53) and, consequently, the performance of the system. Obviously, the closed-loop control has a better impact in terms of precision and attenuation of the disturbances, if present.

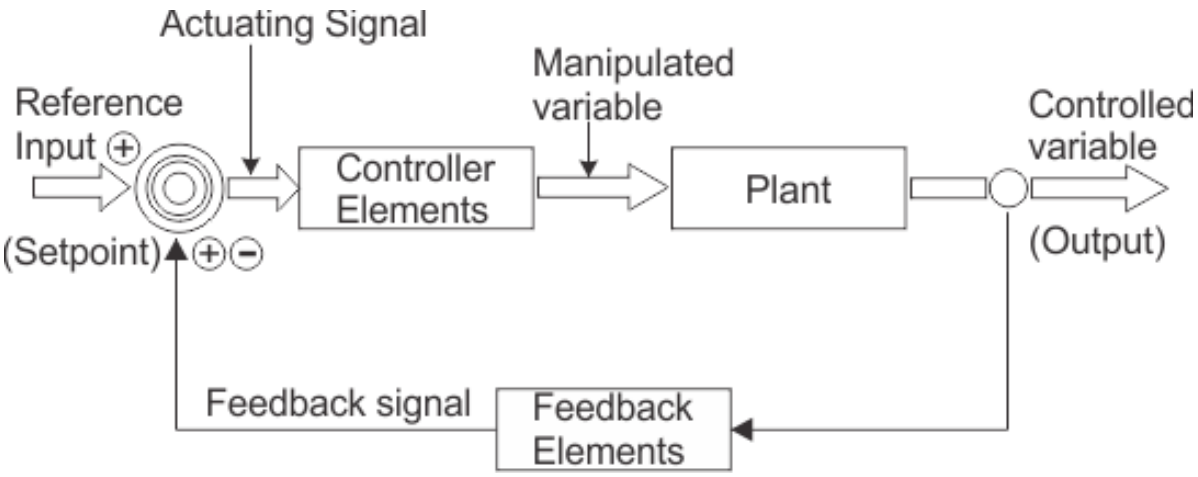

Figure 27: Closed-loop control.

2.2. ADCS

The Attitude Determination and Control System has the duty to orients toward a certain direction the vehicle during the developing of the mission. Due to external interferences, this system need to be linked to the sensors in order to measure the actual attitude of the spacecraft. Dynamics analysis of the spacecraft is very complex for many motives, in particular because it involves non-inertial frames. Consequently, this system is very important to develop during the design phase both for its importance as for other reasons like its power consuming or the demand for specific orientation.

In space, a vehicle, or generally a body, is subject to many, although small, and permanently disturbance torques. The aim of the ADCS is to struggle versus this torques and allow the spacecraft to fulfil its mission. This system can be classified in passive and active control system and in open-loop and closed loop control system. The principal difference between this last two type of control resides in the presence of a feedback line. In fact, in the closed loop case, the actual attitude data is compared with the desired one and they generate an error value. This one, in a continuous process, is used by actuators in order to generate control torques with the aim of restore or acquire the desired attitude. Regarding passive/active control, they differ from each other in that the former uses, depending on the singular value, one or other disturbance torques to control the vehicle. Differently, the active stabilization technique benefits of the comparison with a desired value. Like said before, the generated error value is utilized for stabilize a corrective action and generate an appropriate manoeuvre by the onboard actuators.

2.2.1 Control Algorithms

Over the years, there have been developed many control algorithms in order to command the orientation of a satellite body, obviously each of them presents advantages or shortcomings. The most utilized are (Yadava, Hosangadi, Krishna, Paliwal, & Jain, 2018):

1. **H_∞ controller**: it consists in a feedback control law able to stabilize the system achieving guaranteed performance. Given an execution time, the controller operates on minimizing the H_∞ norm of the closed loop, which represents the maximum energy gain, coming from an external disturbance to the error signal (Stoorvogel, 2000) (Fig.22) . In case of the

worst-case disturbance, the system operates in order to regulate the state going to minimize the output energy. The shortcomings of using H_∞ controller are the very low velocity response and the restriction of apply this method only to linear control problems.

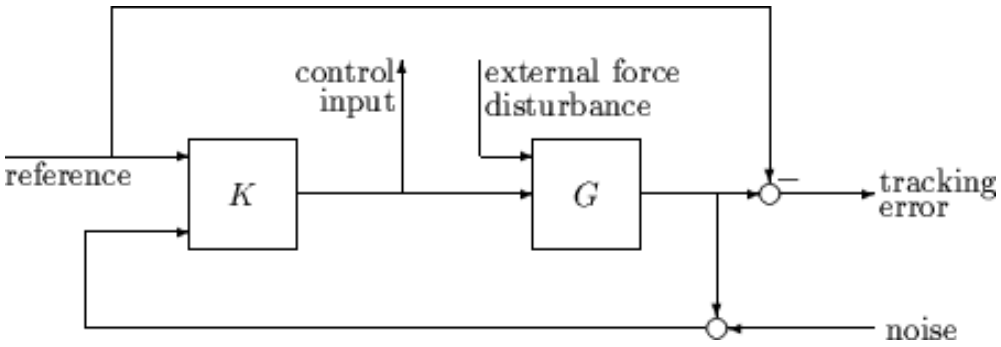

Figure 28: Typical H_∞ controller Closed-Loop Performance.

2. **PID controller**: despite being very sensitive to noisy, which imply the using of filters, this type of controller is one of the most popular in aerospace field. It consists in three principal actors (Astrom, 2002): a proportional component, an integral term and a derivative term (Fig.23). The first one is necessary in order to have a right amount of gain to the system; the integral term provide stability to the system; in the end, the third element is useful in order to reduce the overshoot in the system.

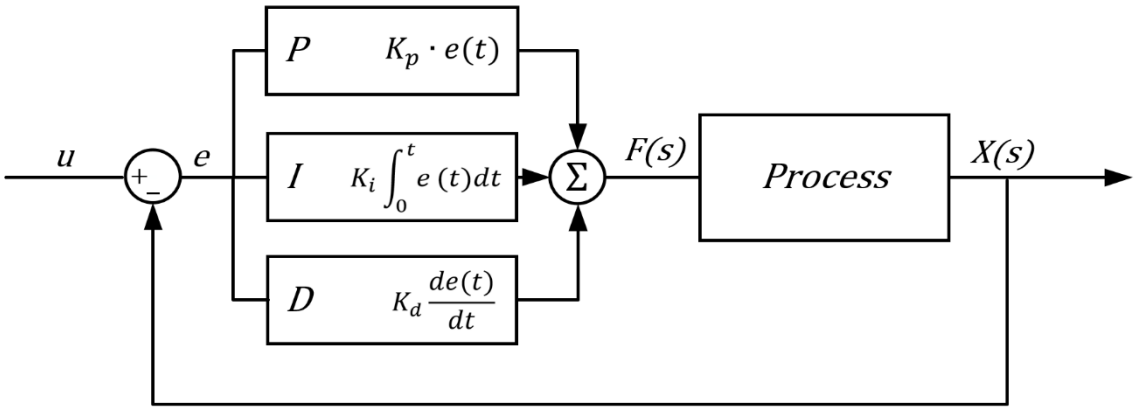

Figure 31: Structure of parallel PID control system.

3. **Fuzzy logic controller**: based on fuzzy logic, it differs in that the other control strategy its logical variables assumes any values between 0 and 1 (Singhala, Shah, & Bhavikkumar, 2014). It allows to convert a strategy based on experience into automatic control. Many studies have shown how this type of control have high efficiency than PID controller, but it is very complex to create and implement and, subsequently, requires high costs (Fig.24).

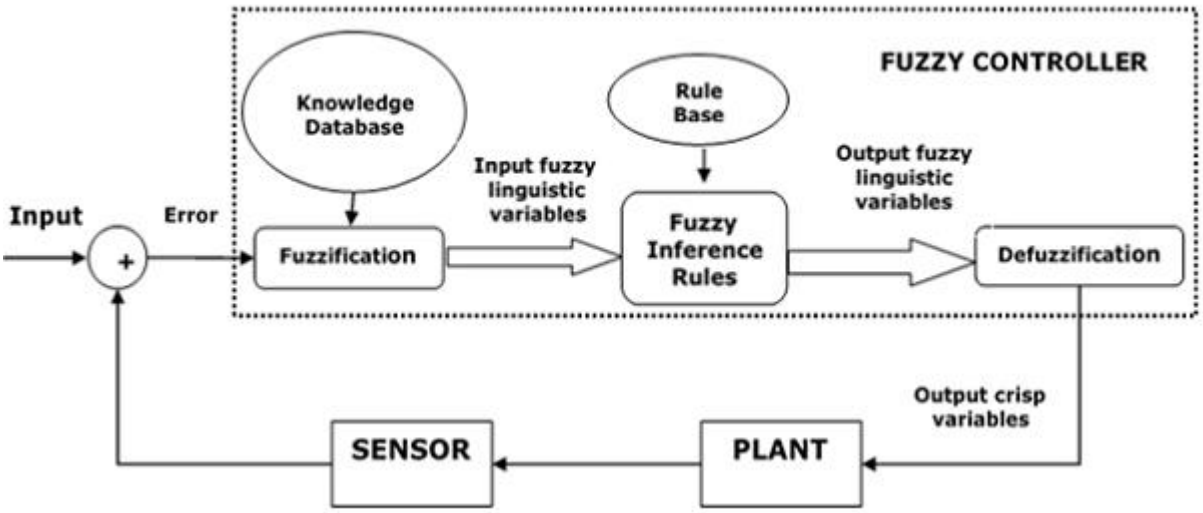

Figure 34: Fuzzy logic controller architecture.

4. **MP controller**: the model predictive control (MPC) instead of positional current values, it is based on the incremental values both of manipulated variables over the control horizon and the controlled variables at the end of the prediction horizon (Wojsznis, Mehta, Wojsznis, Thiele, & Blevins, 2007). In particular, using the prediction horizon allows a self-regulating process where the future steady state is guaranteed (Fig.25).

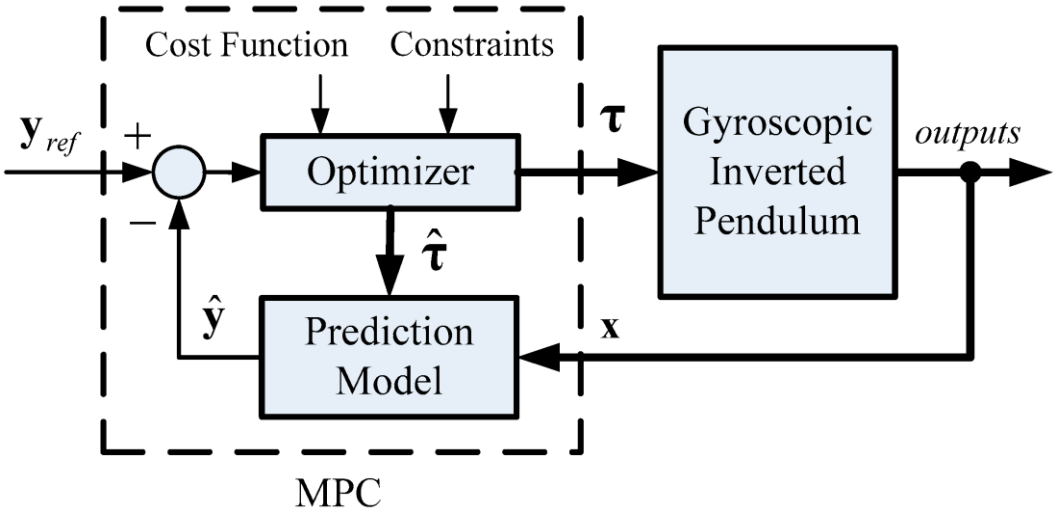

Figure 35: Block diagram for model predictive control..

5. Linear Quadratic Regulator (LQR): these type of controller is the one effectively used during the simulation. Its characteristics are widely explained in the next paragraph.

2.2.2.1 Linear Quadratic Regulator (LQR)

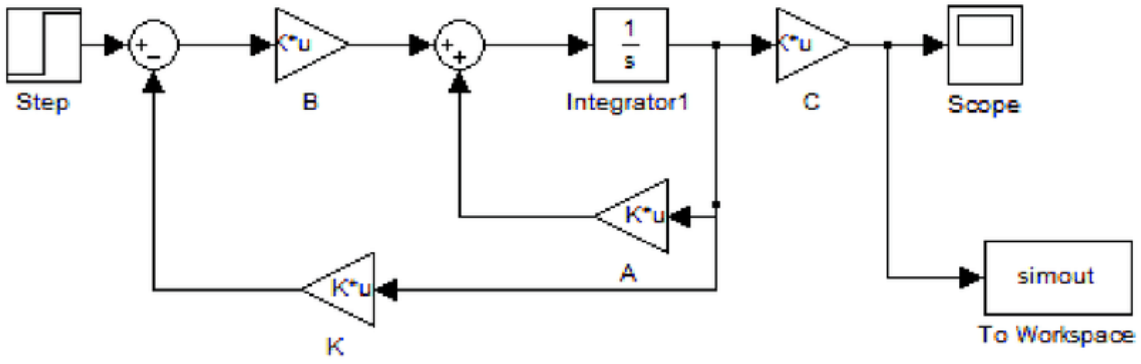

Figure 36: Example of an LQR controller.

The first step in order to train the artificial neural network is to create an LQR controller (Fig.26). The principal goal of this type of controller is to reach the desired position (and velocity) going to create a gain matrix K_{LQR}. This matrix is used in order to calculate the control acceleration a_{LQR} with the following formula:

$$a_{LQR} = K_{LQR} x_e \tag{1}$$

Where x_e is the tracking error, which is defined as the difference between the desired vector and the actual state. Through the computation of a_{LQR}, it is possible to find the value that minimized the cost function J:

$$J = \frac{1}{2}\int_0^{\infty} (x_e^T Q x_e + u^T R u + 2u^T N x_e) dt \tag{2}$$

Q is the state gain matrix**, R** is the control effort gain and **N** is a gain matrix. In this case, the N matrix is set to zero, while the Q and the R matrices are build in this way (Bevilacqua, Lehmann, & Romano, 2011):

$$Q = \begin{bmatrix} \frac{\alpha_{Q1}}{x_{max}^2} & 0 & 0 & & & \\ 0 & \frac{\alpha_{Q2}}{y_{max}^2} & 0 & 0\ 0\ 0 & 0\ 0\ 0 & 0\ 0\ 0 \\ 0 & 0 & \frac{\alpha_{Q3}}{z_{max}^2} & & & \\ & & & \frac{\alpha_{Q4}}{\dot{x}_{max}^2} & 0 & 0 \\ 0\ 0\ 0 & 0\ 0\ 0 & 0\ 0\ 0 & 0 & \frac{\alpha_{Q5}}{\dot{y}_{max}^2} & 0 \\ & & & 0 & 0 & \frac{\alpha_{Q6}}{\dot{z}_{max}^2} \end{bmatrix} \tag{3}$$

$$R = \begin{bmatrix} \frac{\beta_{R1}}{u_{x,max}^2} & 0 & 0 \\ 0 & \frac{\beta_{R2}}{u_{y,max}^2} & 0 \\ 0 & 0 & \frac{\beta_{R3}}{u_{z,max}^2} \end{bmatrix}$$

Thanks to its structure, these two types of matrices are able to find an equilibrium between control effort and performance. In the first matrix, the relative positions of the Chaser respect to the desired position are equal to the Chaser's current distance to goal, i.e r_g. So:

$$x_g = y_g = z_g = r_g \tag{4}$$

The same thing has been done with the numerator of each diagonal component:

$$\alpha_{Q1} = \alpha_{Q2} = \alpha_{Q3} = \alpha_{Q4} = \alpha_{Q5} = \alpha_{Q6} = r_g \tag{5}$$

Differently, the weight on the relative error velocity it is calculated as:

$$\dot{x}_{max} = \dot{y}_{max} = \dot{z}_{max} = \frac{r_{init}}{r_m} r_g \tag{6}$$

Where r_{init} is the chaser initial distance to the target and r_m is the maximum allowed distance from the goal.

Switching to the R matrix, the denominators of the diagonal has been set equal to the maximum control u_{max} that can be reached by the thrusters mounted on the spacecraft. This parameter can be computed considering the thrust force F_t and the mass of the satellite m_s. So:

$$u_m = \frac{F_t}{m_s} \tag{7}$$

$$u_{xmax} = u_{ymax} = u_{zmax} = a_{zmax} = u_m$$

While the numerator terms are set equal to r_g too.

The principal advantage of using this type of controller is the using of relative dynamics which is each step computed and improved with little adjustment (McCamish, Romano, & Yun, 2009). On the other hand, the LQR technique can not be a fast method in order of collision avoidance. Infact, there are necessary numerous sequences of computations in order to avoid an impact with an object present in the operating environment. For these reasons, using an LQR for this purpose involves high cost in terms of implementations and time of calculations.

With the aim of calculating the control torques, it has been decided to choose a different path. In particular, it has been followed the example of Yaguang Yang to create a quaternion-based LQR spacecraft control design (Yang, 2004). The first thing to do is to stabilize a fast settling time. In this case, it has been taken a value of $T_s = 10\ s$. Its rate can be also calculated using the following formula:

$$T_s = \frac{4}{\beta_3 \omega_z} \tag{8}$$

Assuming a value of $\beta_3 = 0.8$, it is possible to obtain $\omega_z = 0.5$. In order to have a system globally stable, it has be calculated an another parameter:

$$\alpha = \frac{1}{J_{33}\omega_z^2} = \frac{1}{J_{11}\omega_x^2} = \frac{1}{J_{22}\omega_y^2} = 0.0840 \tag{9}$$

Consequently, it has been possible to define the other two components of the angular velocity vector:

$$\omega_y = \frac{1}{\sqrt{J_{22}\alpha}} = 0.4629 \tag{10}$$

$$\omega_x = \frac{1}{\sqrt{J_{11}\alpha}} = 0.4082$$

Setting $\beta_1 = \beta_2 = 1$, the components of the feedback matrices D and Krot are automatically given by the following formulas:

$$d_1 = \frac{2\beta_1\sqrt{J_{11}}}{\sqrt{\alpha}} = 51.4344 \tag{11}$$

$$d_2 = \frac{2\beta_2\sqrt{J_{22}}}{\sqrt{\alpha}} = 58.3212$$

$$d_3 = \frac{2\beta_3\sqrt{J_{33}}}{\sqrt{\alpha}} = 11.9048$$

Moreover:

$$k_1 = k_2 = k_3 = \frac{2}{\alpha} = 23.8095$$

So, the final forms of D and Krot are:

$$D = \begin{bmatrix} d_1 & 0 & 0 \\ 0 & d_2 & 0 \\ 0 & 0 & d_3 \end{bmatrix} \tag{13}$$

$$Krot = \begin{bmatrix} k_1 & 0 & 0 \\ 0 & k_2 & 0 \\ 0 & 0 & k_3 \end{bmatrix}$$

The gain utilize to obtain the target values during the rotational motion of the spacecraft is:

$$G = [D\ Krot] \tag{14}$$
$$u = -Gx$$

2.3 Artificial Neural Networks (ANNs).

How many times it has been heard that the brain is capable to perform computations many times more powerful than the fastest existing computer. This particular ability is due to the fact that it is organized into structural components, named neurons, which are highly specialized for the processing and transmission of cellular signals. An explanatory example is the human vision:

besides representing the surrounding space, it gives to the brain the necessary information that it needs to interact with the environment in very short period of time (about 100-200 nms) (Sabatini & Regehr, 1996).

Inspired by neural circuit, artificial neural networks (ANN), also known as "neural networks", are an useful mathematic/computational tool for the resolution of problems regarding control, data analysis and pattern recognition. In aerospace field, they are a strong candidate for attitude control thanks to its inherent nonlinear behaviour, which makes them a natural choice in order to control nonlinear system. In order to achieve a good performance, ANN are formed by individual processing units named neurons, or processing units, grouped in layers. Accordingly, it can be written the following definition of neural network as an adaptive machine (Haykin, 2011):

"*A neural network is a massively parallel distributed processor made up of simple processing units that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:*

1. *Knowledge is acquired by the network from its environment through a learning process;*
2. *Interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge.*"

To achieve a desired design objective, the first step is to train neurons to respond to certain stimuli in the desired way. Thus, it results necessary to perform a learning process, called learning algorithm, in order to modify the synaptic weights of network.

The aim of this paper is to show the benefits and disadvantages of neural networks involved in control of small satellites that have to perform a rendez-vous and docking missions, compared to other type of control methods, like the LQR controller.

2.3.1 Artificial Neural Networks

Like said before, ANNs are computational models which, providing certain input data, are able to process information and learning from them. The neural network is formed by a circuit of information-processing unit, the neurons, represented in Fig.27. In input, there are m input signals, whom is made a weighed summation on the values of the connections that lead the inputs to neuron, resulting:

$$x_1 w_1 + x_2 w_2 + \ldots + x_m w_m \tag{15}$$

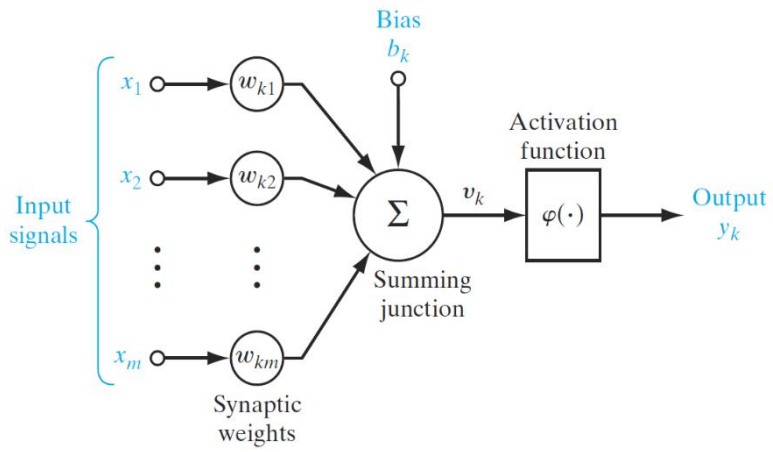

Figure 37: Nonlinear model of a neuron.

The results of this equation (15) is passed to an activation function ϕ which generates the real input of the neuron. Obviously, there can be more than one layer combining one or more neurons. Normally, the basic level of a neural network are three (Fig.28).

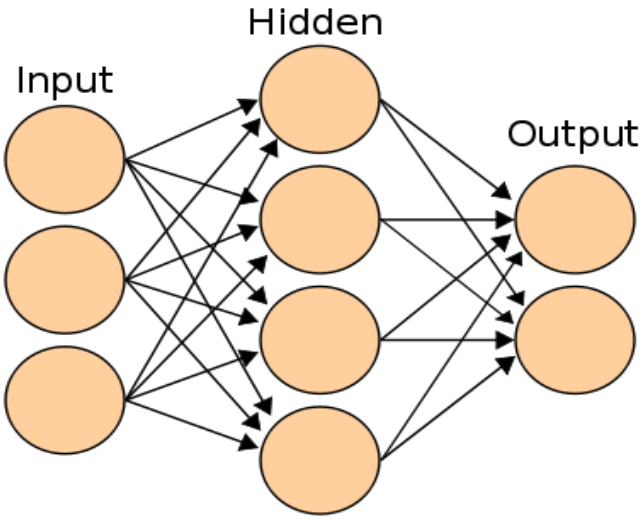

Figure 40: Structure of a single layer neural network.

To every layer, various basic elements can be identified:

1. The first layer, the *input* one, contains the connecting links, or a set of synapses, each of which is characterized by a weight or strength of its own. In particular, the synapse j that receives a signal x_j, amplified by the synaptic weight w_{kj}, and which is connected to neuron k. It is necessary to state that the first subscript of the synaptic weight indicates the neuron in question, while the second one refers to the input end of the synapse to which the weight refers;
2. The second layer, the hidden level, is formed by N unit, called linear combiner or adder, that operate a sum of the different input signals, multiplied by the various synaptic weights;
3. The output of a neuron, which its amplitude is limited to some finite value (typically [0,1] or [-1,1]) by the activation function ϕ, is the third level.

It can be written a pair of equations that describes in mathematical terms the neuron k:

$$u_k = \sum_{j=1}^{n} w_{kj} x_j \tag{16}$$

And

$$y_k = \varphi(u_k + b_k) \tag{17}$$

where u_k is the linear combiner output, not showed in Fig.29, and b_k is the bias. The effect of b_k is to apply an affine transformation to the output u_k through the formula:

$$v_k = u_k + b_k \tag{18}$$

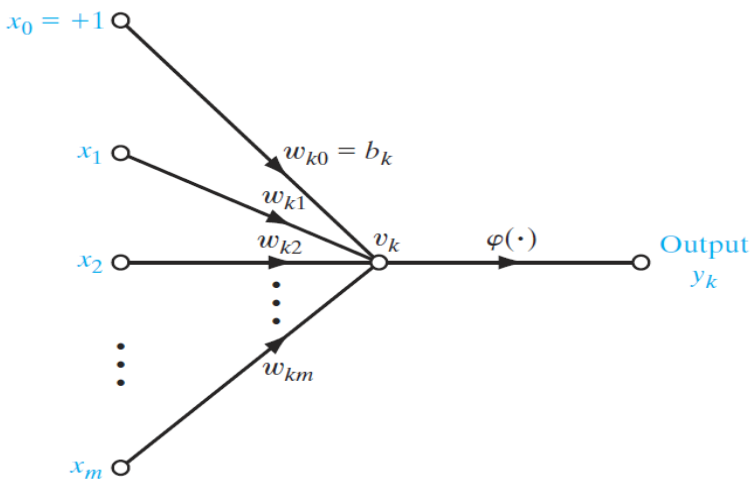

Figure 44: Nonlinear model of NN where w_{k0} replace the effect of the bias.

Based on the value of the bias, the relation between u_k and the induced local field v_k is changed. If b_k is positive or negative, the formula is modified like the Fig.3 shows:

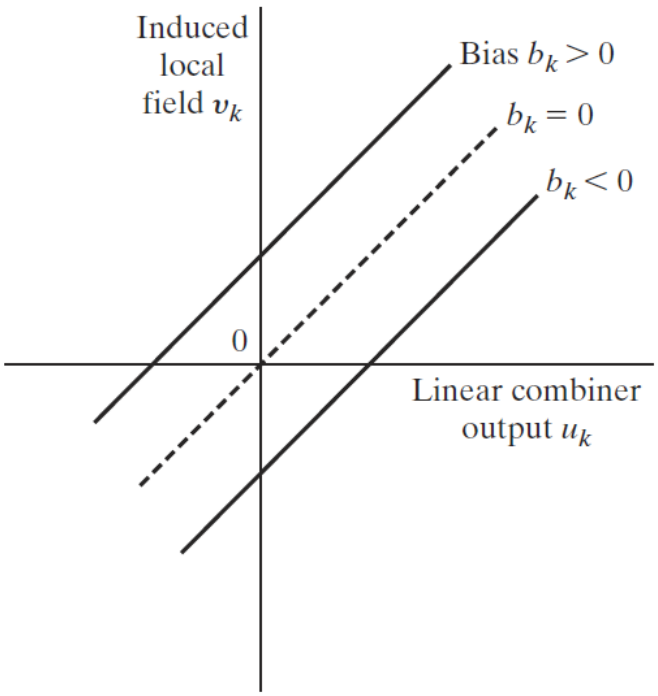

Figure 43: Effect of the bias b_k on the affine transformation.

Specifying and adding the effect of the bias sign, it is possible to modify the structure of the network (Fig.30). Indeed, it needs to be added a new synapse, where its input is

$$x_0 = +1 \tag{19}$$

and which corresponds a weight of

$$w_{k0} = b_k \tag{20}$$

2.3.1.1 Activation function ϕ

There are many types of activation function that can be identified, but the most important one are:

1 *Threshold function*: better known as unit step function (Fig.31), it is a function described by the following relationship

$$\varphi(v) = \begin{cases} 1 & if\ v \geq 0 \\ 0 & if\ v < 0 \end{cases} \tag{21}$$

The output of neuron k will be modified too

$$y_k = \begin{cases} 1 & if\ v \geq 0 \\ 0 & if\ v < 0 \end{cases} \tag{22}$$

In the end, the induced local field can be calculated as

$$v_k = \sum_{j=1}^{n} w_{kj} x_j + b_k \tag{23}$$

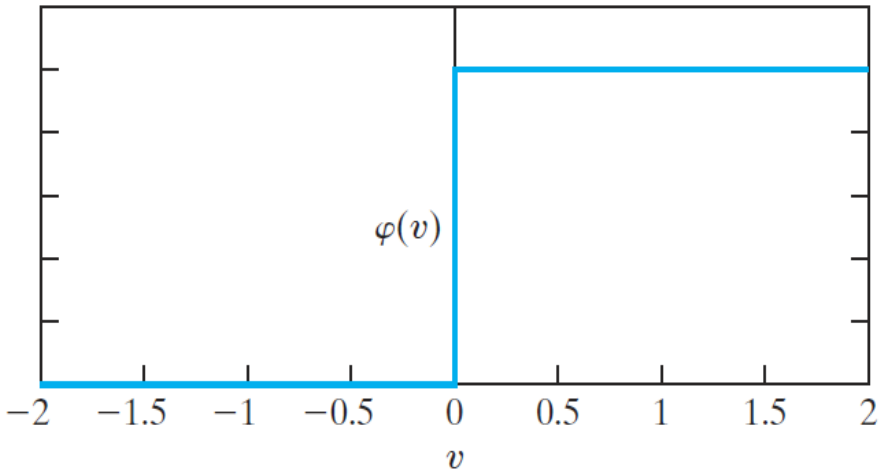

Figure 47: Threshold function.

Sigmoid function: this is the most used activation function during the creation of ANN, having the shape of an "S" (Fig.32). A classic example of this type of function is the hyperbolic tangent function, defined by

$$\varphi(v) = \tanh(v) \tag{24}$$

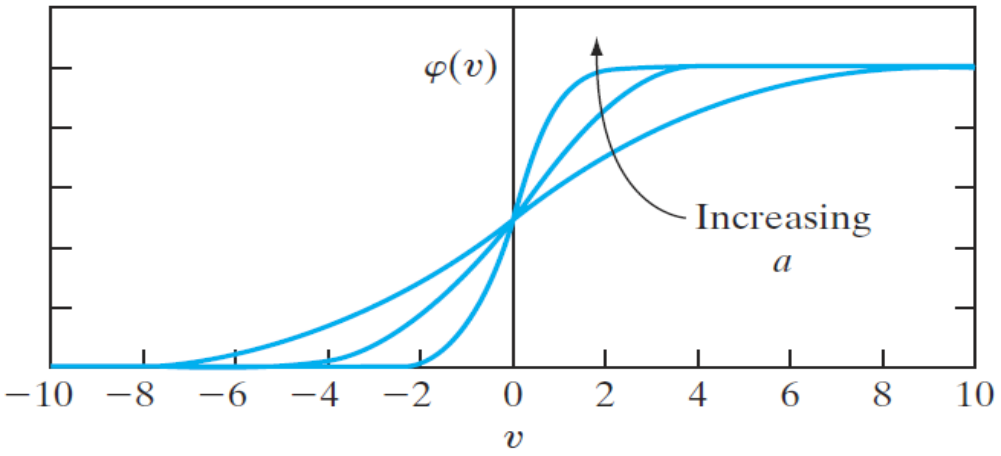

Figure 50: Sigmoid function with a variable slope parameter a.

2.3.2 Architectures of ANNs.

There are many ways in order to organize a neural network. For example, it can be decided to add more hidden layers of neurons, having in input the output of previous one level. The choice of the architecture it is fundamental because it directly influences the learning algorithm utilised to train the network.

It can be identified three different classes of network:

1. **Single-layer Feedforward Networks**

This is the simplest structure of layered network (Fig.33). It is organized in three areas: on the left side, there is the input layer of source nodes, augmented of respective weights, which goes directly into the activation function, which is the second area; at least, there is the output layer that can be formed by one or more neurons. The defect of this network is that, differently from more complex architectures, there are no feedback from the output area, so the input it is

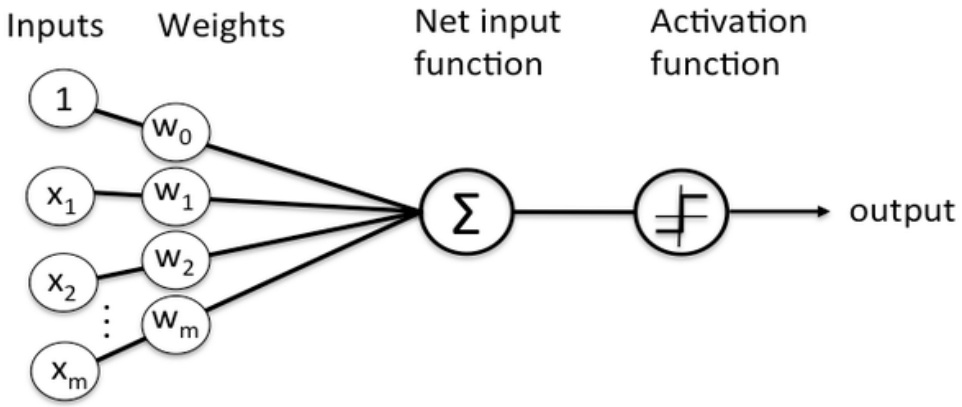

Figure 53: Single-layer Feedforward Networks.

influenced by it.

2. **Multilayer Feedforward Networks**

As it can be seen from the Fig.33, this type of network differences itself by the presence of more than one layer of neurons, called hidden layers or hidden units. The purpose of those levels is to create a separation from the input and output layer, in order to effect more

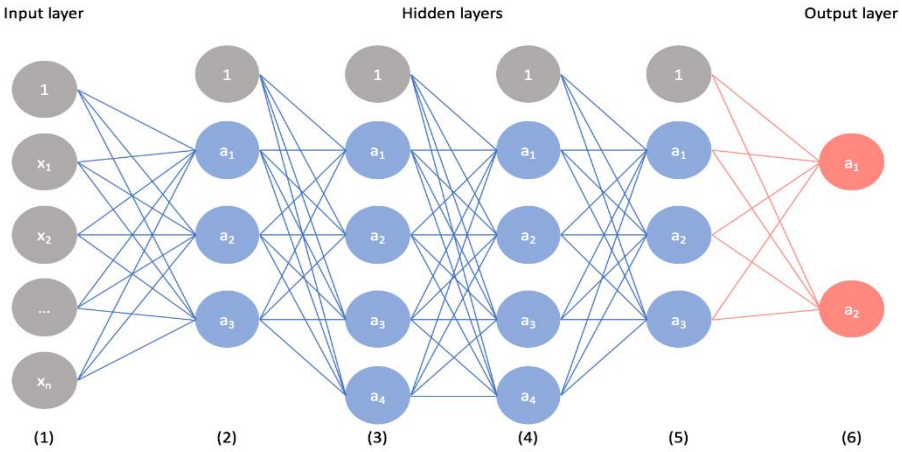

Figure 56: Multilayer Feedforward Networks.

calculations and extract statistics that are more specific.

The principal characteristic of this type of architecture is that the inputs of the one layer will be the output from the previous one level. So, the overall response of the network will be composed from the sum, or in generally by a combination, of the inputs of every layer. A network as shown in the previous image is called "fully connected", because every nodes/neurons of every layer is called with the node of the next layer; vice versa, it will be said to be "partially connected".

3. **Recurrent Networks**

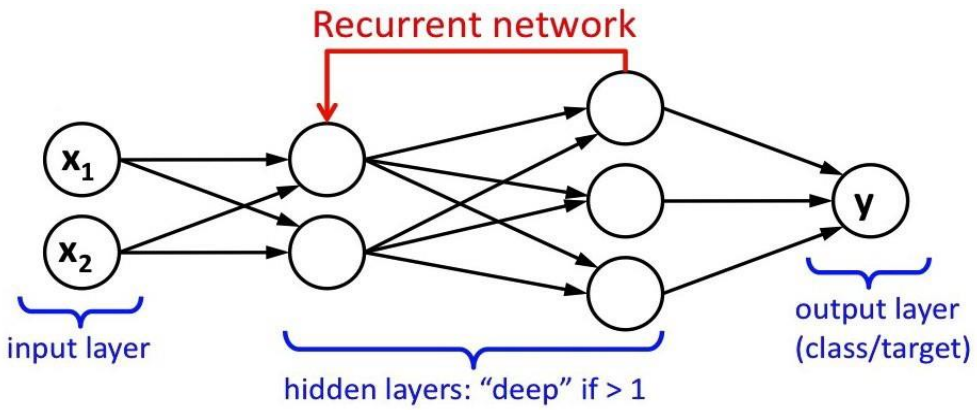

Figure 59: Recurrent network.

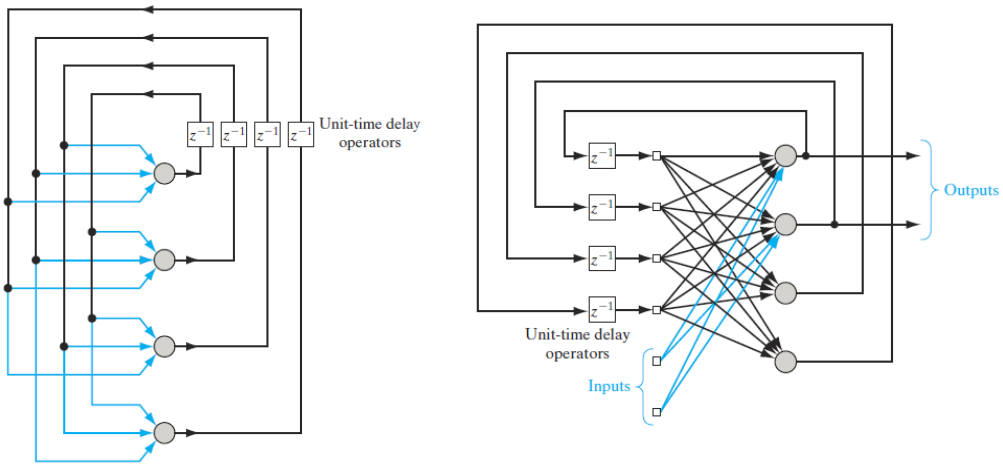

Figure 62: Recurrent network with no self-feedback and no hidden layers (on the left); recurrent network with hidden neurons (on the right).

Utilizing this architecture, there is a new function that can be used to improve the performance of the network: the feedback loop. As illustrated in Fig.9, a recurrent network can be formed by a number of hidden layers, where the output of the last hidden layer is used as input of first level too. This type of network can be divided in two subgroups: if the output of a neuron is fed back into its own input, this is called a self-feedback loop; vice versa, it is referred as recurrent network (Fig.35 & 36).

2.3.3 Training and learning process

Artificial neural networks are such a good and useful tool if they are training well. Indeed, such as an infant who starts to speak hearing his parents, so the NNs understand and are influenced by the environments where they are immersed. Before speaking of the training process, it is necessary to expand the concept of knowledge.

2.3.3.1 Knowledge

For a neural network, knowledge refers to the way the circuit interpret, predict and how it responds to the impulse coming from the extern environment. In order to achieve some specified goals, the network has to learn a model of the world around itself. The interpretation of the world has to contain two major information:

1. Prior information: information about what the environment is and what has been known;
2. Measurements: through sensors and other instruments, the neural network has to be able to observe and understands how it is supposed to operate. Collecting this measurements, it can be formed a pool of information, called training data, useful to train the neural network.

In order to achieve a good knowledge representation, is a general common sense to respect four rules:

I. **Similar inputs have to be classified into the same class, because they produce similar representations inside the circuit;**
II. **Inputs of separate classes have to produce diverse representations (this rule is dual of the precedent one);**
III. **If a characteristic is particular important, the network should use a big amount of nodes for portraying that detail**.
It is useful to introduce two parameters: the probability of detection and the probability of false alarm. The first indicates to the network the probability that, the object that it is considering, is a target; vice versa, the probability of false alarm defines the chances of misunderstanding a mark. Applying Rule 3, it ensures the high degree of accuracy in decision making process;
IV. **The network must to be structured in such a way that invariances and prior information are not to be learned by the system; this allow a simpler design of the circuit.**
As results of this last rule, there are born neural networks with specialized structure. Differently from a fully connected network, their rate of information transmission is very high, deriving from peculiarity of having a smaller number of free parameters. In this way, they learn faster because they necessitate a smaller data set for training. Consequently, the use of specialized network allows to lowering the costs because they are smaller.

2.3.3.2 Prior information

In order to put prior information into the structure, there are no defined rules. Instead, there are utilized two techniques: using local connections, in order to restrict the ANN architecture, and limiting the pick of synaptic weights.

To satisfy the first constraint, every receptive field of the various hidden neurons are constituted by an equal number of input source. In this way, the network have a smaller size. The receptive field is defined as "the region of the input field over which where the incoming stimuli can influence the output signal produced by a neuron" (Haykin, 2011) (Fig.37).

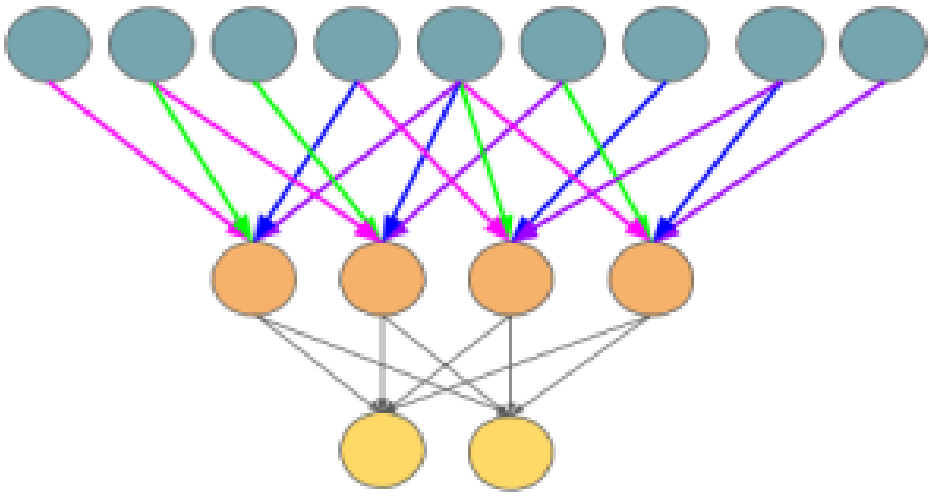

Figure 65: Combined use of receptive field and weight sharing.

2.3.3.2 Invariances

There are some types of signals that have not to be influenced by transformations of the observed signals. Some techniques can be used in order to achieve this goal:

1. Invariance by structure: designing the structure in a certain way, it is possible to create the same output although the input is transformed. For example, a way it is to imposing the weight of two different neuros, having the same input, equal;
2. Invariance by training: during a period of training, the network understands how to discriminate between diverse aspects of the same problem/example. There are two problems with this method. This first one is the computational demand imposed to the network, which is very high; the second one is that is not obvious that this training will also enable the network to recognize other objects of different classes invariantly;
3. Invariant feature space*:* this technique consists of create an invariant classifier-type neural network. It rests on the premise that it may be possible to extract features that characterize the essential information content of an input data set and that are invariant to transformations of the input. If such features are used, then the network as a classifier is relieved of the burden of having to delineate the range of transformations of an object with complicated decision boundaries.. The use of an invariant feature space offers three distinct advantages. First, the number of features applied to the network may be reduced to realistic levels. Second, the requirements imposed on network design are relaxed. Third, invariance for all objects with respect to known transformations is assured.

2.3.3.4 Learning process

When an human being wants to learn something from the surrounding environment, he can chose between two ways of learning process: with or without the help of someone, normally named as "teacher". In particular, the latter form can be divided into two subcategories: unsupervised learning and reinforcement learning.

2.3.3.4.1 Learning with a teacher

Known as supervised learning too (Chaturvedi, 2008), this type of method is shown is Fig.38. This closed-loop feedback system is characterized by three principal blocks: the environment, the teacher and the learner.

- The environment, as already said before, is where the neural network operates and which receives input data;
- The teacher is the one who possess the necessary knowledge, in form of input-output examples, of the stimuli which came from the environment. Indeed, it is able to produce a desired response, that represents the "perfect" action that the network may perform;
- The learner, or rather the ANN, does not know the operative context (i.e. the environment). The goal is to emulate the response of the teacher. In order to do that, the actual response is

correct with an error signal, formed by the difference between the response from the teacher and the one coming from the network.

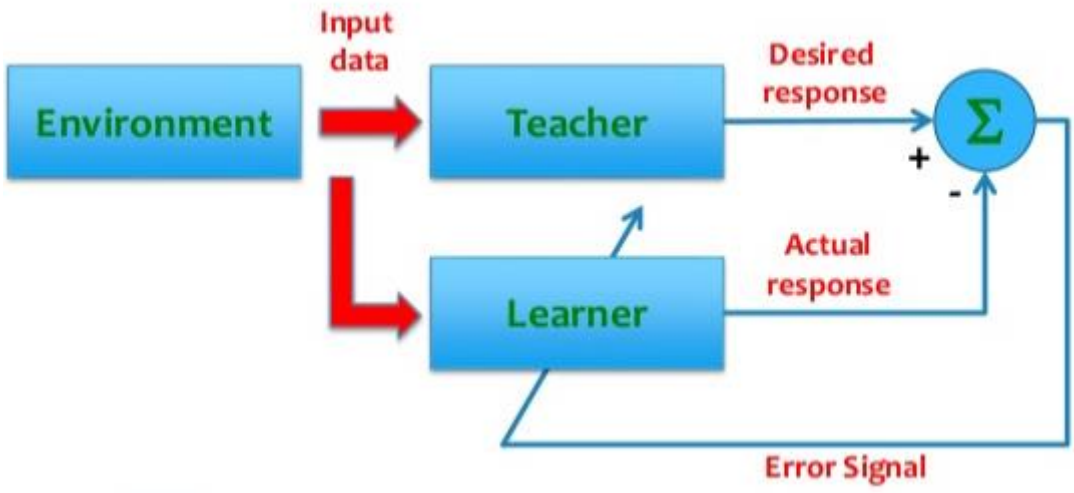

Figure 66: Supervised learning.

In this way, thanks to the training, the teacher transfers his knowledge of the environment to the learner, who stores these information in form of synaptic weights. These parameters are useful to understand how well the system perform. In order to that, it can be utilized the mean-square error function, defined by some free parameters (i.e. the weights). It will be create a multidimensional error surface, where the coordinates are the free parameters and every operation is visualized as a point on the surface. The performance of the system can be improved moving down the point in discuss toward a specific minimum point: it may be a global or a local minimum. Through the gradient of the error surface, defined as a vector which points in the direction of steepest descent, the learner can understand the actual behaviour of the system. The algorithm starts from a generic point x_0 and calculates the gradient $\nabla f(x_0)$, giving the possibility to define the moving direction. After a specific step, a new point x_1 with the relative gradient can be estimated. This iterative process continues until the gradient goes down to zero.

2.3.3.4.2 Learning without a teacher

As the name suggest, this technique plan to not use a teacher during the learning process. Accordingly, there are no examples that the network can use to understand the environment around itself. As said in the precedent paragraph, there are two paths that can be followed:

1) Reinforcement learning

The block diagram pictured in Fig.39 shows one type of reinforcement-learning system. It is based on two principal actors: the environment and the learning system. The latter is projected to learn through a method called delayed reinforcement. It consists of a generation of a reinforcement signal which, added to state signal coming from the environment, will generate the signal of action. Thanks to the continued interaction with the environment, the learning is operated going to minimize a scalar index of performance (Modi & Jethva, 2016).

In this case, the performance are represented by a cost-to-go function, defined as the expectation of the cumulative cost of actions taken over a sequence of steps instead of simply the immediate cost. The goal of the learning system is to find the actions that best approximate the general behaviour of the system and sends a feedback signal to the environment.

The advantage of utilizing this type of learning process is represented by the fact that it is independent from a teacher, so the network learn to approach with the surrounding environment on the basis of its experience only. Nevertheless, there are two major drawbacks:

1. Due to the absence of the teacher, there is no supervising system that is charge to control the learning process of the system and his response to extern stimuli;
2. A generation of reinforcement signal implies that the ANN has to individually understand the sequence of actions that lead to that final outcome, assigning a credit for each steps; the outcome is only evaluated by the reinforcement signal instead.

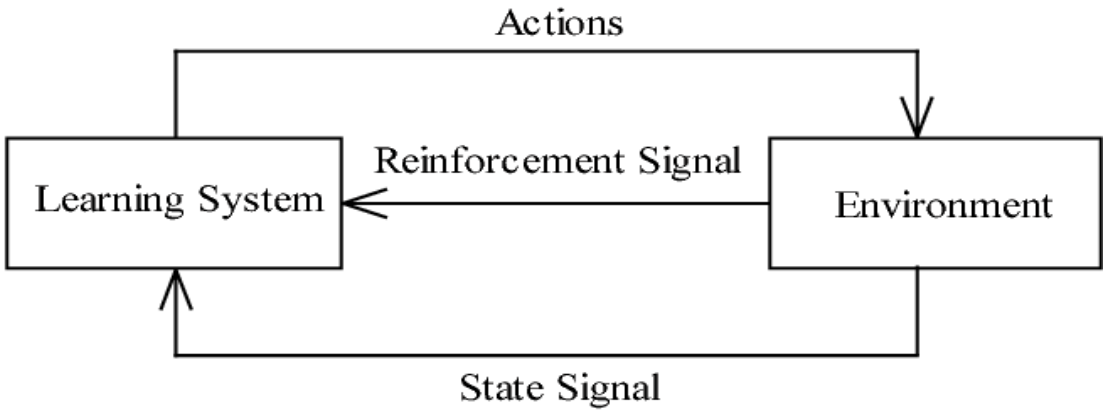

Figure 67:Reinforcement learning.

2) Unsupervised learning

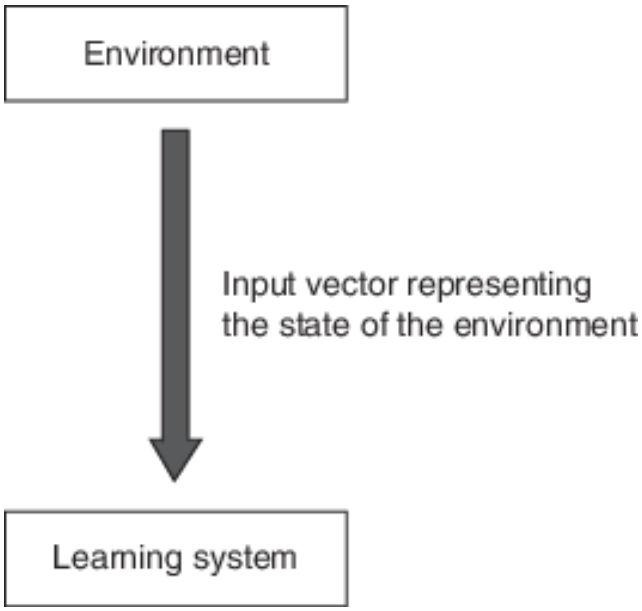

Figure 70: Block diagram of unsupervised learning.

In this case, besides the absence of a teacher, there is no reinforcement signal neither. The structure of the system is shown in Fig.40 and it is very simple. There is a vector describing the state of the environment which is the input data of the learning system; the network, thanks to regularity of the statistical stimuli from the extern, stars to develop the ability to create new classes able to approximate the features of the input (Atiya, 1990).

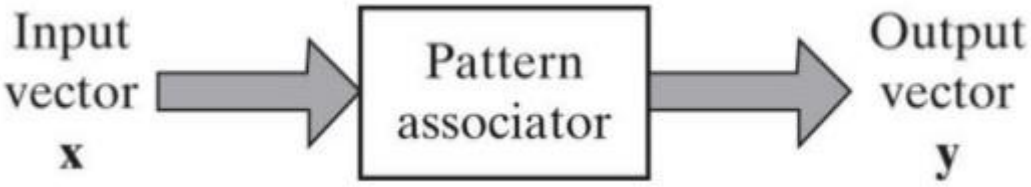

Figure 71: Pattern relation between the input vector x and the output vector j.

This type of learning is performed using a strategy of challenge between the neurons. The network can be divided, for example, in two or more layers. In the first one, the available data are received from the input layer. The feature contained in the initial stimuli are provided to the second layer, the one with the neurons. Here, it is applied a competitive strategy: the "winner-takes-all" one. The neuron which offer the greatest total input wins the game and will be the one to be turned on; the others will be switched off.

2.3.3.5 Learning tasks

Utilizing a certain type of learning task, which represent the universality of the NN, influences the way the network learn to interact with the environment.

2.3.3.5.1 Associative memory

Like the human brain acquire information by association and stored them in a distributed memory, the same is done for neural network. Association can be of two types: autoassociation e heteroassociation (Borders, et al., 2017). The first one, which use the unsupervised learning, consists of memorizing a set of input patterns by presenting them cyclically to the network; after, a distorted version of the original vector is presented to the network, which must be able to recall the right pattern. Heteroassociation is based on coupling a set of input pattern with a specific set of output vectors. Hence, it involves the supervised learning process.

For example, let define the key pattern x_j and the memorized pattern y_j (Fig.41). Defining the number of patterns stored in the network n, the associative memory permits to combine these two vectors by the relation

$$x_j \rightarrow y_j, \quad k = 1,2, \dots, n \tag{24}$$

Consequently, the parameter n gives a measure of the storage capacity of the network; hence, it must be as large as possible. A big difference between the two types of association relies on the dimensions of the last two vectors. Infact, if the network storing utilizing the autoassociation memory, x_j will be the equal to y_j; whereas, in heteroassociation memory, the dimension of the

output vector can be different from the input one. This can be a problem when the network has to recall a specific pattern, because x_j contains the key for its retrieval too.

In associative memory, there are two phases includes in the operation:

1. Storage phase: following the path marked by the Eq.24, this phase consists in training the network;
2. Recall phase: giving in input to the network a distorted version of x_j, this phase involves the retrieval of the right y_j.

Especially in the second phase, the training results fundamental in order to recall the right memorized pattern. If this does not happen, it is said that an "error in recall" has occurred.

2.3.3.5.2 Pattern recognition

This second type of learning tasks is described as "*the process whereby a received pattern/signal is assigned to one of a prescribed number of classes*" (Haykin, 2011). The pattern recognition is a process formed by various phases. The first step is recurrently present to the network a set of input patterns of diverse category or classes. After that, a never seen before pattern, which relies to the same population of the previous ones, will be presented to the network. His duty is to identify the new input and classifying it and its information correctly. In order to do that, the patterns are divided in some representative points and placed in a multidimensional decision space, which is created internally to the network. This space is divided into regions, each one is associated with a class.

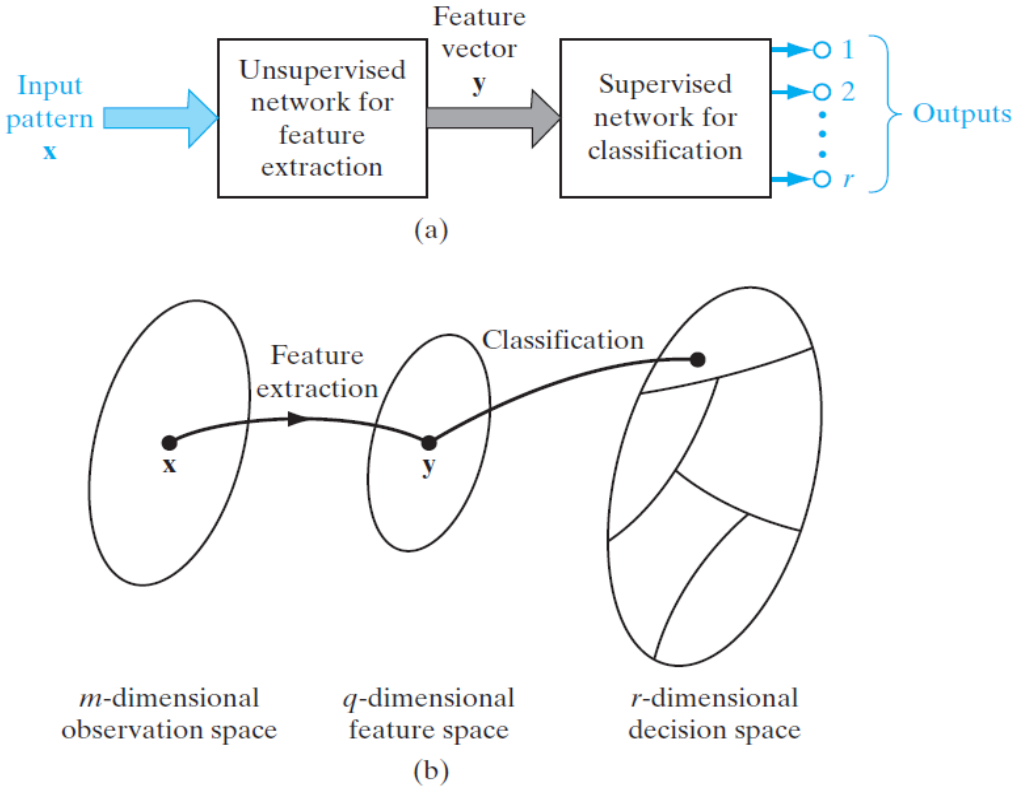

Figure 72: First approach of pattern classification.

Pattern recognition can be exploited using one of this two approach:

- The first one consists in combining an unsupervised network with a supervised one respectively for feature extraction and for classification (Fig. 42a). As shown in the picture, results necessary to define some parameter. The first one is m, the observables: it consists in a representation of a pattern. So, it may be viewed as a point x in the m-dimensional observation (data) space. The extraction of the feature and its classification going through a middle phase, where the point x became a new point y in a q-dimensional feature space. This operations needs to be done in order to minimize the dimension of the feature. Subsequently, it is classified into one of the classes of the r-dimensional decision space (Fig.43b);
- The second method involves the use of a feedforward network which exploits a supervised learning process. The extraction and the classification of the process is explicated thanks to some hidden layers in the network formed by various computational units.

2.3.3.5.3 Function approximation

Function approximation is another learning task required to the neural network. It is based on a relationship between an input s and its related output z, which is

$$z = f(s) \tag{25}$$

The problem is the vector-valued f which is unknown. In order to solve this problem, it is common use to utilize a set of labeled examples:

$$\Gamma = \{(s_i, z_i)\}_{i=1}^{N} \tag{26}$$

It is mandatory for the network to produce a function F that describes a certain input-output mapping that respect the disequation

$$\|F(s) - f(s)\| < \varepsilon \quad for\ all\ s \tag{27}$$

where ε must be a positive number as small as possible. This can be done having a right number of free parameters and a big size of initial examples Γ.

This type of approach is perfect for a supervised learning. Hence, it results necessary to explore in which ways this kind of network can approximate a not known input-output mapping. There are basically two possible passable roads:

1. **System identification**: assuming to provide to the entire system an input vector s, the network will be guided in the learning process by an unknown system, which will act as a teacher. This one can be, for example, a system formed by labeled examples where the time is invariant. The scheme of the block diagram is shown in Fig.42. Both the network and the system will produce an output, y_i and z_i respectively. The difference between these two signals will create the error signal vector e_i. This one will be feedback to the neural network in order to adjust its free parameters, in order to stay below the tolerance ε;
2. **Inverse modelling:** the goal of this structure is to calculate the input signal s through the equation

$$s = f^{-1}(z) \tag{28}$$

The difficult of this system is to calculate the inverse function of f, because there can be more than one solution. Anyway, the scheme of this type of system is pictured in Fig.43. Differently from the previous case, this time the vector z_i is used as input vector, s_i is the desired response

and e_i reports the difference between s_i and the output of the network y_i. Like before, the error signal vector is useful in order to adjust the free parameters of the system and be within the required tolerance range

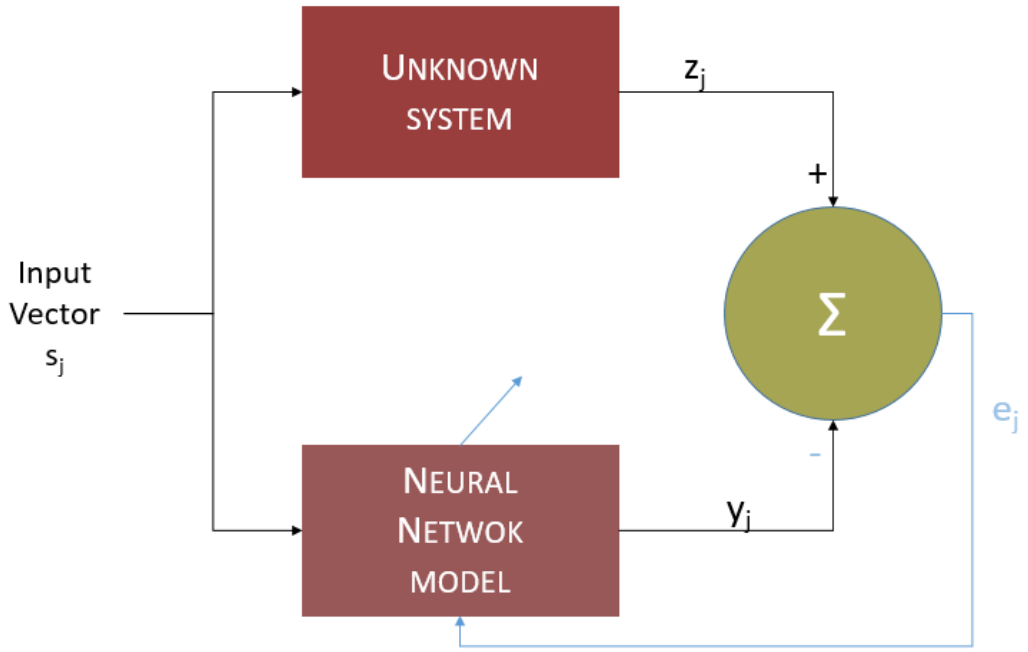

Figure 78: System identification of a supervised system.

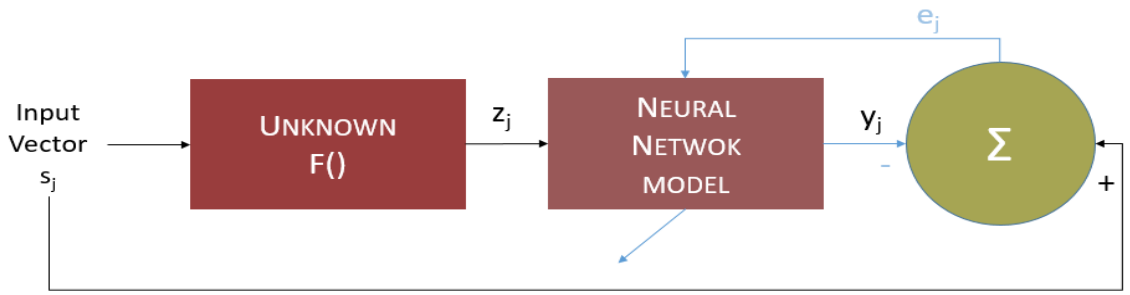

Figure 75: Block diagram of inverse system modelling.

2.4 Final notes

In this introductive part, it was presented a general summary about the neural networks and its properties. The most important one is for sure the process of learning, that can be obtained in three ways:

1. Supervised learning: with the goal of minimizing a cost function, this procedure requires to define a preferred response in order to accomplish a particular mapping of input-output;
2. Unsupervised learning: the network has to learn by itself how to respond to the input coming from the external environment;
3. Reinforcement learning: the network and its environment are continuously in contact, with the intention of create an input-output mapping minimizing the scalar index of performance.

Choosing the supervised learning implies using a set of labeled examples, where each example contains an input signal and its relative desired response. The problem of this type of learning is linked to the massive cost, in terms of time too, of collecting a set of labeled examples. The second learning process, the unsupervised one, relies only on the capacity of the network to respond to a set of stimuli. In this way, there's no more the necessity of a training output sample, but, on the other hand, to an input data can corresponds more than one response. In order to try to solve this problem, it is evolving a new type of learning process: the semisupervised learning (Kingma, Rezend, Mohamed, & Welling, 2015). It consists of using both labeled and unbranded examples, with the aim of creating a network that approximates efficiently a large-scale pattern-classification problem.

Finally, the reinforcement learning is the compromise between supervised and unsupervised learning, with the learning system and the environment that constantly cooperates between them. The leaning system reacts to a certain stimuli and learns from the response of the environment to that action

3. Model & Simulation

3.1 Reference frames

When a study is conducted on a moving satellite, it is necessary to establish several reference frames. They are:

1. **Earth Centered Earth Fixed** (ECEF): shown in Fig.45, this frame has the following characteristics:
    - Origin O in the centre of the Earth;
    - The x-axis x_E is located on the equatorial plane, defined at the vernal equinox and with the positive direction in the direction of the constellation of Aries;
    - z_E is perpendicular to the last axis and points toward the Polar star, i.e. the North;
    - y_E is automatically defined because it completes the right-handed set.

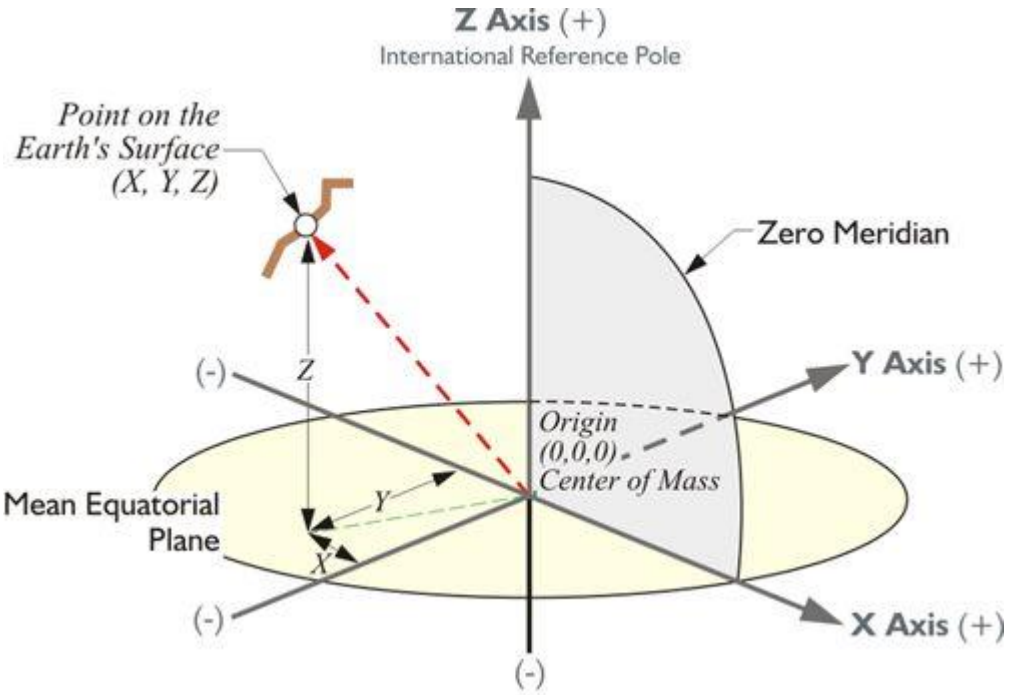

Figure 81: ECEF reference frame.

2. **Local Orbit frame**: this time, the origin is in the Centre of Mass (CoM) of the vehicle. The z-axis, z_{orb}, is radial from the CoM of the spacecraft to the centre of the Earth; y_{orb} is in the opposite direction of the angular momentum vector of the orbit; finally, the x-axis x_{orb} is in the direction of the orbital velocity, but it is automatically defined because $x_{orb} = y_{orb} \times z_{orb}$. This frame is displayed in Fig.46.

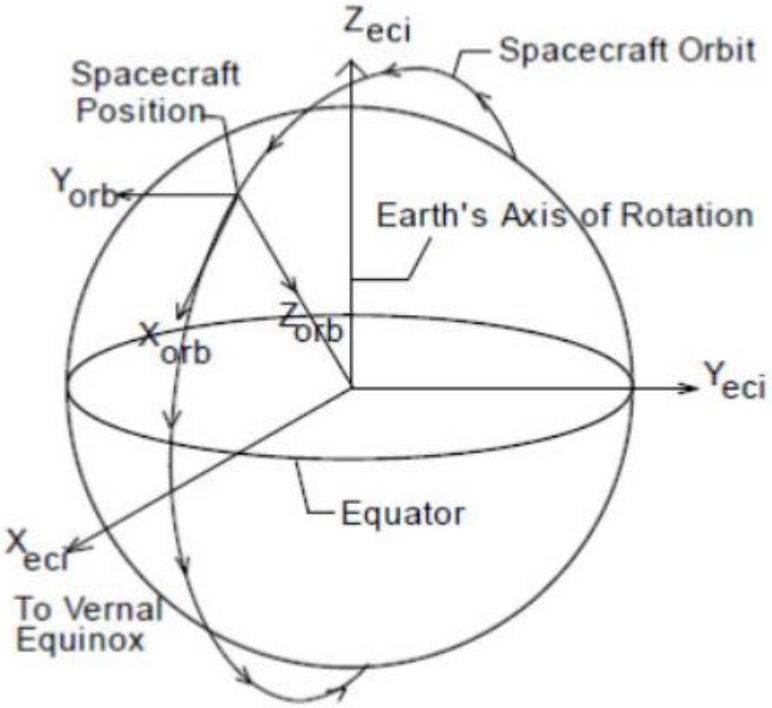

Figure 82: Spacecraft Local Orbital frame.

3. **Body Fixed Reference frame**: liked shown in Fig.47, this is a non-inertial coordinate system, because it is fixed on the spacecraft. It has for origin the CoM of the vehicle, the main inertia axis of the spacecraft represents the direction axis, with $z_b = x_b$ x y_b thanks to the right-hand rule.

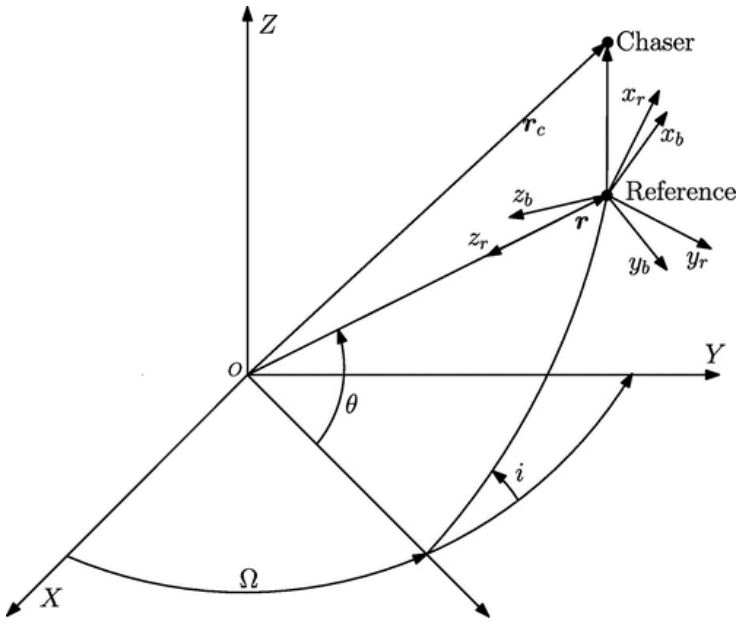

Figure 83: Spacecraft Body Fixed Reference frame.

3.2 Operative environment

In order to make precise calculations while the spacecraft is in orbit, it is mandatory to consider the disturbances forces and torques due to space environment. The main disturbance elements which can modify orbit parameters are summarized in this subchapter.

3.2.1 Aerodynamic Drag

If a satellite moves towards a low orbit, it has affected by the Earth's atmosphere. It makes its presence felt by a small drag, which decrease the velocity of the vehicle e modifies its path too. The atmosphere can be identified as a force in opposite direction of the velocity vector and can be calculated thanks to the following formula:

$$\overrightarrow{F_{drag}} = -\frac{1}{2} C_D \rho V^2 \cos(\alpha)\, A \tag{28}$$

Where:

- ρ is the atmospheric density. At an attitude of 400 km, its value is equal to $2.803 \cdot 10^{-12} kg \cdot m^{-3}$;
- V is the spacecraft velocity;
- Acos(α) is the projected area perpendicular to V;
- C_D stands for the commonly known drag coefficient. It is normally taken as 2.2.

In case the aerodynamic centre of pressure of the spacecraft and its centre of mass are not in the same position, this drag force will generate a disturbance torque T_a:

$$\overrightarrow{T_a} = \overrightarrow{r_{cp}} \times \overrightarrow{F_a} \tag{29}$$

r_{cp} is the centre-of-pressure vector in a body coordinates frame and it is measured starting from the centre of mass.

3.2.2 Solar radiation pressure

Another element that can create a change of the orbital elements can be the flux of photons emitted by the Sun. This particles generate a pressure on the surface of the spacecraft which they hit. Obviously, this is a periodical effect due to both the period the vehicle faces the Sun and moreover the attitude. This force can be expressed as:

$$\overrightarrow{F_{sun}} = -\lambda C_R P_0 A \vec{u} \tag{30}$$

In this case, these coefficients represent:

- The shadow function λ. When the satellite is in the Earth's window, its value is of 0;
- C_R is the radiation pressure coefficient. It relies between an interval of 1 and 2. For example, if it is made an assumption of 1.5, it means that the photons whom meet the spacecraft's surface are half absorbed and half rejected;
- P_0 is the solar pressure. It is considered constant at a value of $4.644 \cdot 10^{-6} Nm^{-2}$;
- A symbolizes the projected vehicles' area which is normal to the Sun vector;
- $\vec{u}$ is the vector pointing from the Earth to the Sun.

Like the precedent one, this disturb can produce a torque:

$$\overrightarrow{T_{sun}} = \overrightarrow{r_{sp}} \times \overrightarrow{F_{sun}} \tag{31}$$

r_{sp} is the vector from the spacecraft optical centre of pressure to body centre of mass.

3.2.3 Gravity gradient

Being an object which rotates around a planet, every spacecraft is subject to a gravitational gradient. Obviously, this attraction, varying trough an inverse-square law, is greater to the side more close to the planet. Consequently, this differential attraction will generate a torque which makes the spacecraft rotates in order to align the local vertical with its minimum inertia axis. This results in an periodic oscillation of the vehicle's path. For a satellite in a near-circular orbit, this particular torque can be expressed as:

$$\overrightarrow{T_{grav}} = \frac{3\mu}{R^3} u_e \times [I] \cdot u_e \tag{32}$$

Where:

- R stands for the distance between the Earth's centre and the spacecraft;
- u_e is the unit vector from vehicle to planet;
- I is the spacecraft inertia matrix;
- μ represents the Earth's gravitational coefficient. It is equal to $3.986 \cdot 10^{12} m^3/s^2$.

3.2.4 Earth magnetic field

As well known, the Earth is surrounded by a strong magnetic field: it is known as the magnetosphere (fig.48).

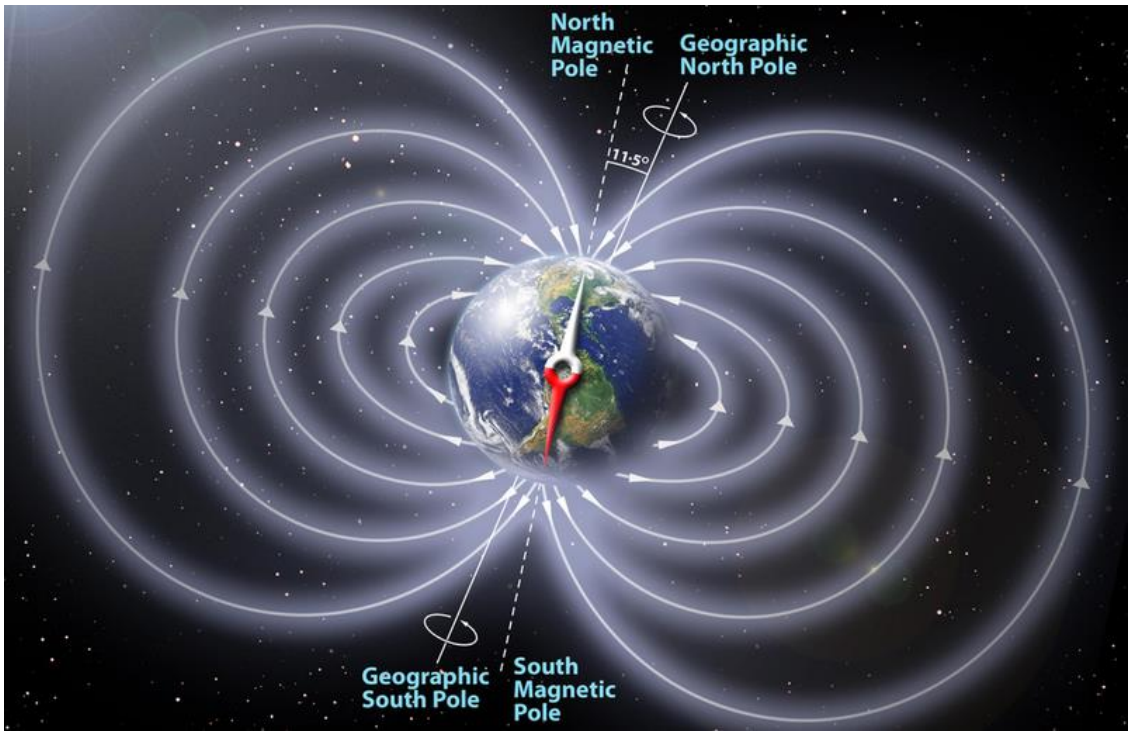

Figure 84: Earth's magnetosphere.

When the spacecraft is immersed in this field, it is subject to a torque calculated with the following formula:

$$\overrightarrow{T_{mag}} = \vec{m} \times \vec{B} \tag{33}$$

$\vec{B}$ stands for the Earth magnetic field vector and it is expressed is Tesla; $\vec{m}$ is the spacecraft dipole movement, so it is measured in Am^2.

3.3 Rotational and kinematics dynamics

3.3.1 Orbital parameters

An object orbiting around a planet has to respect the Kepler's law of planetary motion: this means that the satellite can follows only curves paths denominated conic sections. This family of conics comprehend circular, elliptical, parabolic and hyperbolic paths (Fig.49).

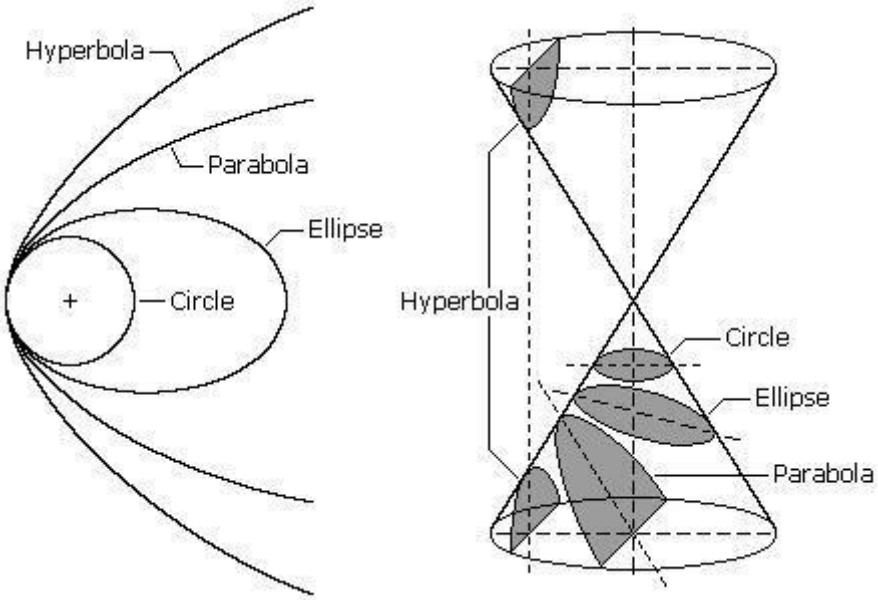

Figure 87: Conic sections.

In order to identify a specific orbit, it results necessary to define six orbital parameters. First, it is needed to introduce some coefficients. The first one is periapsis: it represents the distance between the origin of the orbit and its closer point on the satellite's path. Its opposite is the apopasis. The orbital elements are (Fig.50):

- **Semi-major axis a**: it is the half sum of apoapsis and periapsis distances. It allows to comprehend the dimension of the orbit;
- **Eccentricity e**: on the basis of its value, it is possible to understand the form of the path**.**
  - e = 0: circular orbit;
  - 0 < e < 1: elliptical orbit;
  - e = 1: parabolic orbit;
  - e > 1: hyperbolic orbit;
- **Longitude of ascending node Ω:** the position in the orbit where the path of the spacecraft passes through the ecliptic plane, i.e. the ascending nodes. It is measured from the vernal equinox;

- **Inclination i**: inclination between the orbit of the spacecraft and the reference plane. It is measured starting from the ascending node;
- **Argument of periapsis ω**: it defines the angle between the periapsis and the ascending node**;**
- **True anomaly ν**: it represents the position of the spacecraft at a specific time.

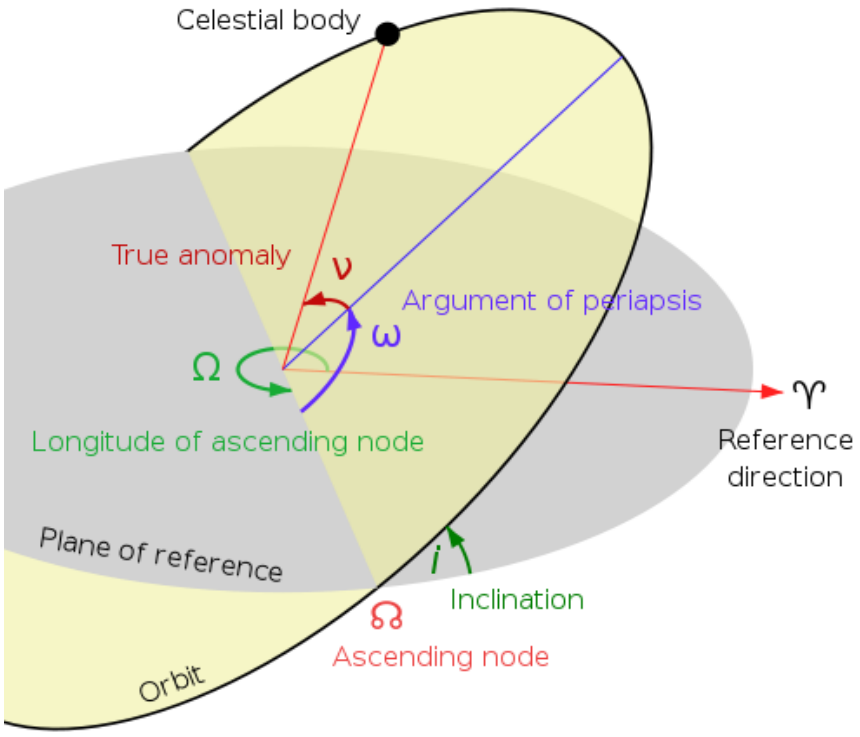

Figure 88: Orbital elements.

3.3.2 Euler-Hill equations

When the problem under consideration is the proximity manoeuvres of rendez-vous and docking, it results necessary to define a non-inertial coordinate system in order to measure the relative distance and velocity between the Target and the Chaser. This moving system is fixed in the Centre of Mass of the Target satellite and the Chaser moves with it. The motion of this two spacecraft are described, respect to the mating point, thanks to the Euler-Hill equations, which derives from the linearization of the equation of motion in orbit. The homogenous Euler-Hill equations are:

$$\begin{cases} \delta\ddot{x} - 3n^2\delta x - 2\delta n\dot{y} = 0 \\ \delta\ddot{y} + 2\delta n\dot{x} = 0 \\ \delta\ddot{z} + n^2\delta z = 0 \end{cases} \tag{32}$$

The previous equations can change if are considered both the disturbances and the control accelerations:

$$\begin{cases} \delta\ddot{x} - 3n^2 - 2n\dot{y} = a_x \\ \ddot{y} + 2n\dot{x} = a_y \\ \ddot{z} + n^2 z = a_z \end{cases} \tag{33}$$

It needs to define:

1. x,y,z: distance beetwen the two satellites along the three directions;
2. n: mean motion. It is defined as:

$$n = \sqrt{\frac{\mu}{r_0^3}} \tag{34}$$

3. The a_i terms takes into the account the disturbance and the control acceleration.

3.3.4 Rotational kinetics

Approximating the spacecraft as a rigid body, it is possible to define a method that allows to tie up the attitude of the satellite with its angular velocity ω. This is possible thanks to the Direction Cosine Matrix (or quaternions), which uses the Euler angles (φ, θ, ψ) (Fig.51), useful to represent the attitude in the model too.

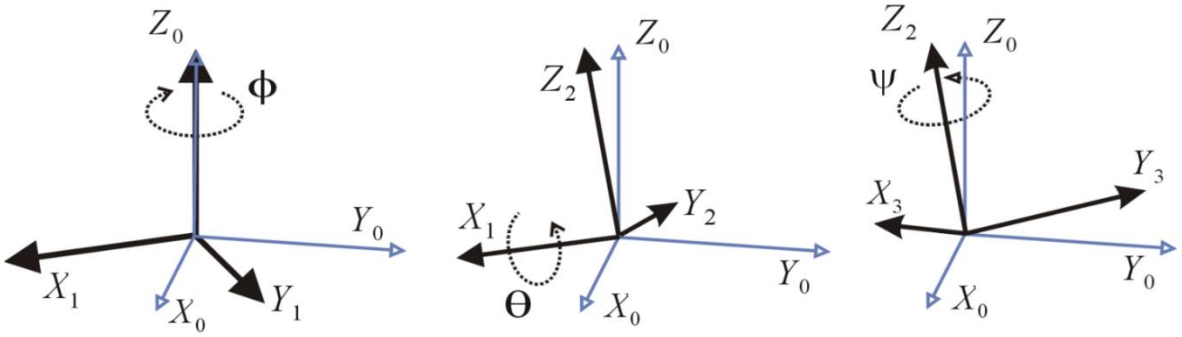

Figure 91: Euler angles.

In order to obtain the quaternion, it is necessary to integrate the following equation:

$$\dot{q} = \frac{1}{2} Q\omega \tag{35}$$

Where:

- Q represents the following matrix:

$$Q = \begin{bmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & q_3 & q_2 \\ q_3 & q_0 & -q_1 \\ -q_2 & q_1 & q_0 \end{bmatrix} \tag{36}$$

- ω is the rotation vector expressed in the body frame.

Using the second law of dynamics for rotating bodies, it is possible to relate the torque applied on the spacecraft to the angular momentum:

$$\dot{H} = T \tag{37}$$

From this formula, is easy to obtain the Euler moment equation:

$$J\dot{\omega} + \omega \times J\omega = T \tag{38}$$

where J is the inertia matrix.

3.4 Control design

In this chapter it is described the simulation model, in Matlab/Simulink, that allows to represent the motion of the Chaser and respect to the Target. The architecture of the model in Simulink environment is showed Fig.55: there are the dynamics blocks, which comprehend the rotational and translational dynamics of the spacecraft; the control blocks, both for translational and rotational motion and, finally, the block that contains the disturbance forces.

The model's process starts with an input signal for the dynamics block by Guidance, Navigation and Control (GNC) system, which is exploited thanks to the presence of thrusters and reaction wheels. The output signal from this last block are the forces and the torques needed to change the trajectory of the satellite. This commands are taken by the chaser dynamic blocks, which includes the Hill equations, the Euler equations and the kinematic equations too. This signal is modified by the disturbance torques and forces generated from solar radiation pressure, aerodynamic drag, ecc. Finally, it will be generated the output signals, which consists in satellite attitude, its relative velocity and position and the control forces and torques (Fig.54).

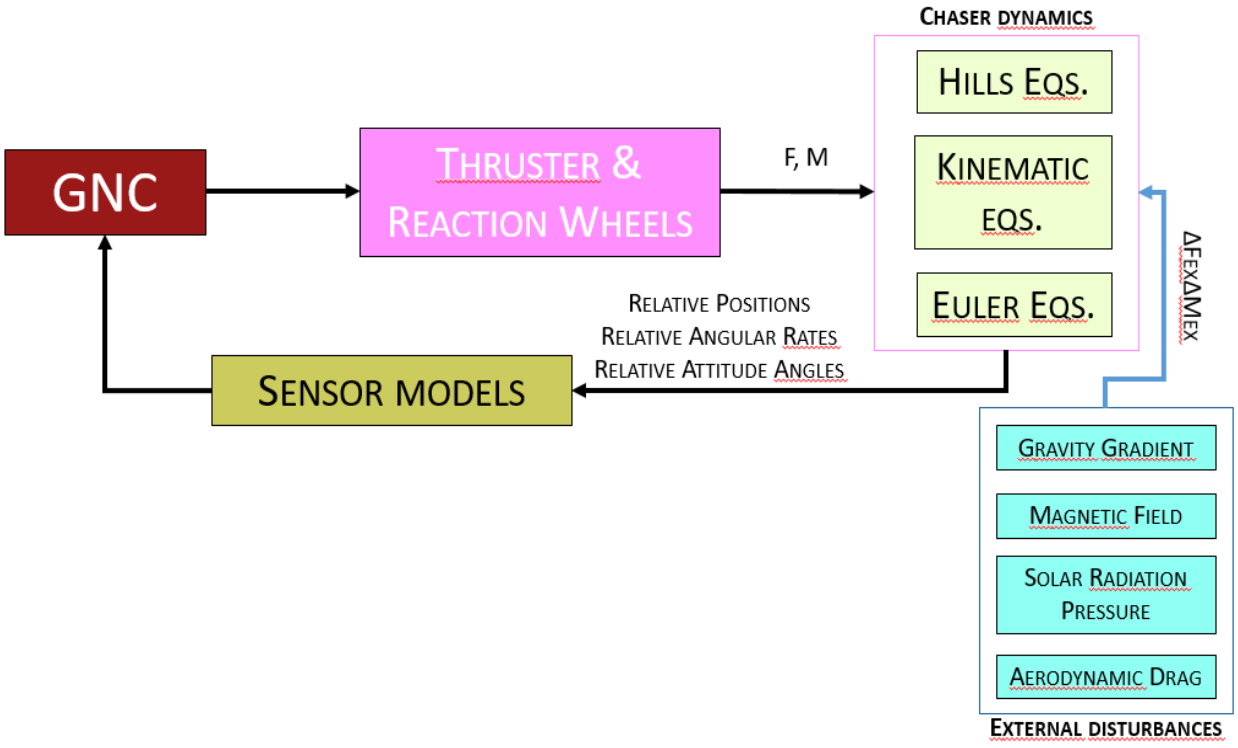

Figure 94: Dynamics block scheme.

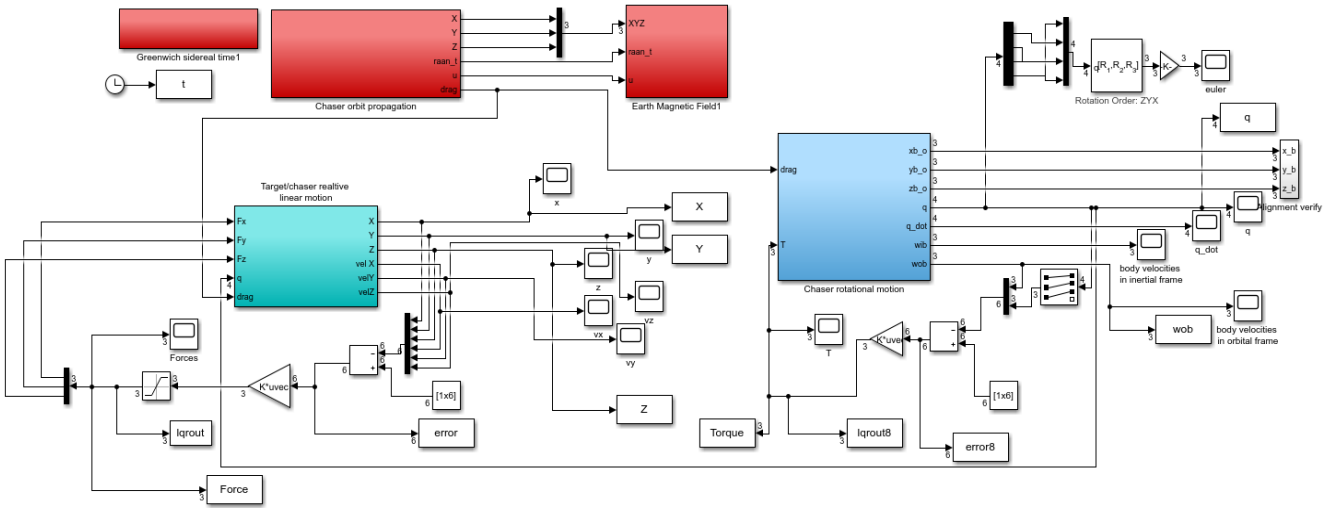

Figure 97: Simulink model.

3.4.1 Translational dynamics model

Both the translational and rotational dynamics are taking into account in order to model as best as possible the motion of the Chaser approaching the Target. The first motion, described in the local orbit frame LVLH, used the later called Euler-Hill equations and its Simulink representation is pictured in Fig.56:

$$\begin{cases} \ddot{x} = \frac{1}{m_c} F_x + 2\omega\dot{z} \\ \ddot{y} = \frac{1}{m_c} F_y - \omega^2 y \\ \ddot{z} = \frac{1}{m_c} F_z + 3\omega^2 z - 2\omega^2 \dot{x} \end{cases} \tag{47}$$

Where:

- m_c is the mass of the Chaser satellite;
- ω is the satellite's angular velocity, expressed in rad/s, while it is orbiting around the Earth;
- F_x, F_y, F_z are the amount of control and disturbance forces towards three direction, measured in N.

The translational velocity in x, y and z are acquired integrating one time the equation in (47). Repeating this operation, it was possible to find the relative positions. As done in the precedent chapter, it is possible to linearize the Euler-Hill equations. The first one was:

$$\dot{x}(k) = A(k)x + B(k)u \tag{48}$$

Where:

- A and B are two matrix empirically defined as:

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 2\omega^2 \\ 0 & -\omega & 0 & 0 & 0 & 0 \\ 0 & 3\omega^2 & 0 & -2\omega^2 & 0 & 0 \end{bmatrix} \tag{49}$$

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

- The state vector *x(k)* which ties up the position and the velocity of the Chaser to the centre of mass of the Target:

$$x(k) = [x\ y\ z\ \dot{x}\ \dot{y}\ \dot{z}] \tag{50}$$

- The vector which contains the value of three forces along each direction *u(k)*:

$$u(k) = [F_x, F_y, F_z] \tag{51}$$

In order to find the control output u it is necessary to utilize both the error vector *e(k)*, generated by the difference between the desired reference r and the state vector, and the proportional gain K_p. The control output was found thanks to the following formula:

$$u = K_p e(k) \tag{52}$$

In the end, it was possible to substitute these equations into the (48) formula and obtain the following structure of the state-space equation for translational dynamics:

$$\dot{x} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 2\omega^2 \\ 0 & -\omega & 0 & 0 & 0 & 0 \\ 0 & 3\omega^2 & 0 & -2\omega^2 & 0 & 0 \end{bmatrix} x + \frac{1}{m_c} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{53}$$

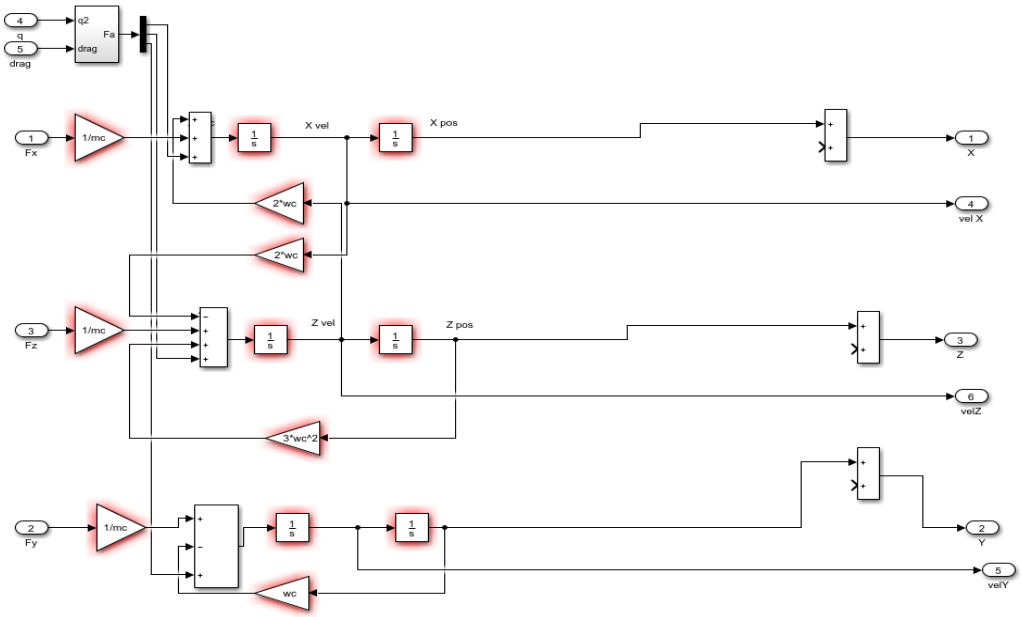

Figure 103: Translational model.

3.4.2 Rotational dynamics model

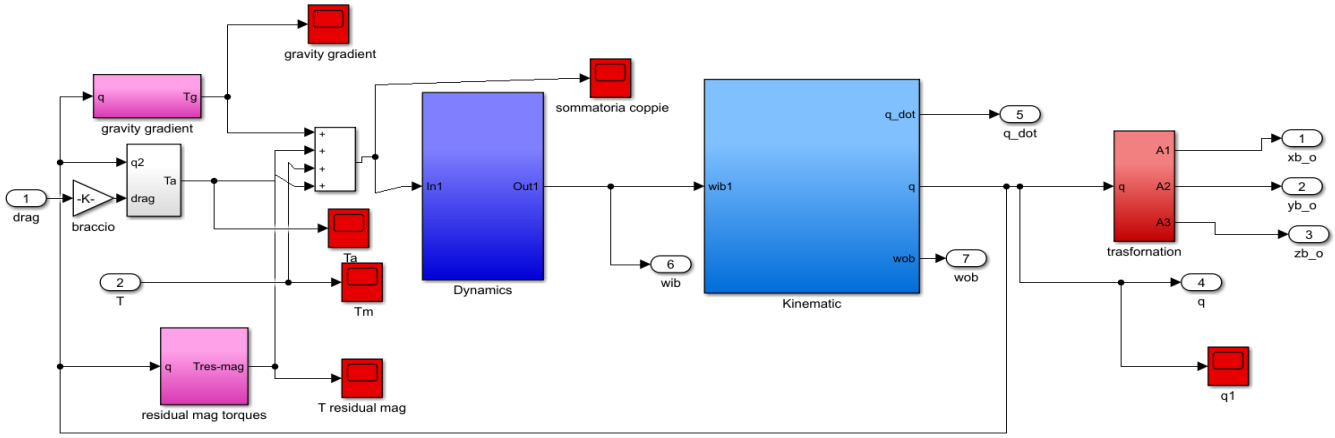

Figure 100: Rotational dynamics Simulink model.

For take into account the rotational dynamics it is mandatory to consider the contribution from the disturbance forces and the control torques. For that reasons, it was used the Euler equation into the block of the Simulink model (fig.57):

The equation utilised to describe the rotational dynamics contains both angular velocity and acceleration, respectively ω and $\dot{\omega}$, the inertia matrix I and the totality of disturbance and control torques T:

$$\mathrm{I}\dot{\omega} + \omega \times (I\omega) = T \tag{54}$$

Isolating the angular acceleration respect to the inertial reference frame and integrating it one time, it was possible to obtain the angular velocity of the Chaser:

$$\dot{\omega} + \omega \times (I\omega) = T \rightarrow \dot{\omega} = I^{-1}[T - \omega \times I\omega] \\ \rightarrow \omega_{BO} = \omega_{BI} - R_{OB}^{T}\omega_{OI} \tag{55}$$

Both the angular velocities are referred to the inertial frame, as well as the orbit velocity ω_{OI}; in the end, there is the rotation matrix R_{OB}. For practical reasons, it is easier to state the attitude kinematics equations using quaternions instead of Euler angles:

$$\dot{q_{OB}} = [Q_{OB}]\omega_{BO} \tag{56}$$

In these formulation compare:

- Q_{OB} which is deriving from the quaternions and it interpreted as a Matlab functions:

$$Q = \begin{bmatrix} -q_{OB1} & -q_{OB2} & -q_{OB3} \\ q_{OB0} & q_{OB3} & -q_{OB2} \\ -q_{OB3} & q_{OB0} & -q_{OB1} \\ q_{OB2} & -q_{OB1} & q_{OB0} \end{bmatrix} \tag{57}$$

- The quaternion vector q_{OB} which is written the orbital frame

3.4.3 LQR-controller

Like anticipated in the previous chapter, this controller is used in order to control both the velocity and the position of the Chaser with respect to the Target spacecraft. As reference for the controller, it has been taken the vector [0 0 0] both for the reference position and velocity, considering the Target like a cooperative spacecraft, i.e. it does not move during the manoeuvre.

As it can be seen in the figure pictured below, the LQR controller calculates at each iteration the actual trajectory of the Chaser comparing it with the reference. The error vector generated from the difference between this two vectors is multiplied with the gain K of the LQR system, calculated thanks to a specified MatLab function. As output it is obtained the control signal values, which generates the control forces, that has been specifically limited between a minimum and a maximum. This functionality field is stabilized from the thrusters chosen for the CubeSat in exam:

$$-F_{th,min} \leq u(k) \leq F_{th,max} \tag{58}$$

Speaking of the rotational dynamics instead (Fig. ?), it has been followed the same path. The quaternions vector combined with the angular velocity vector, they were compared to the reference vector, which was again [0 0 0], because the Target is not moving or spinning. Consequently, the error was tuned with the LQR gain and it generates the control torques, which it is again limited in a certain range gave this time by the Reaction wheels:

$$-T_{RW,min} \leq u(k) \leq T_{RW,max} \tag{59}$$

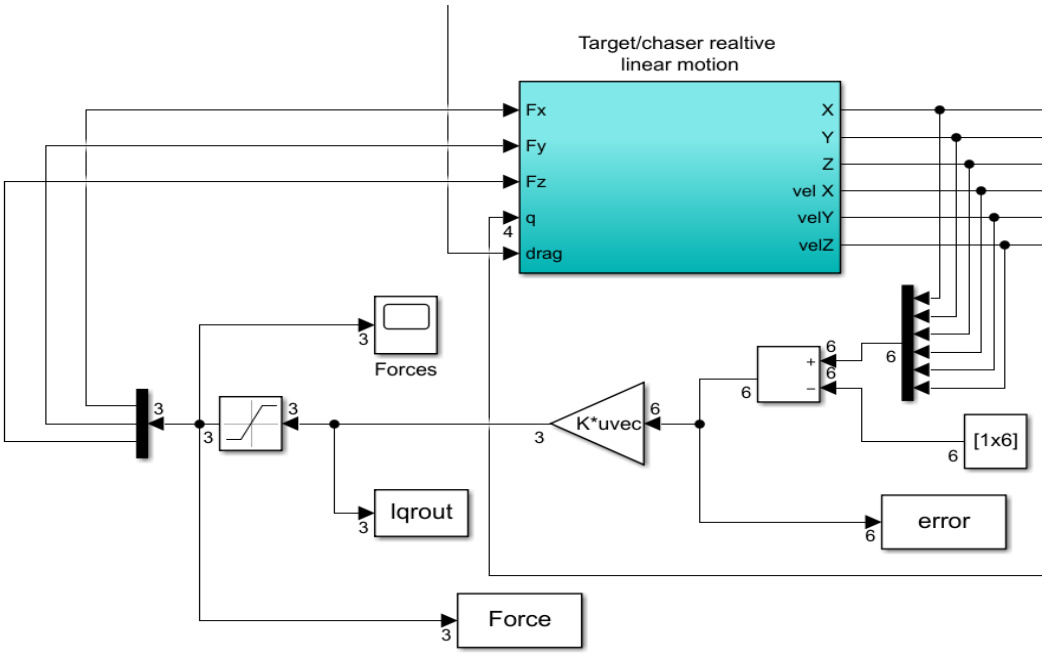

Figure 106: LQR controller for traslational motion.

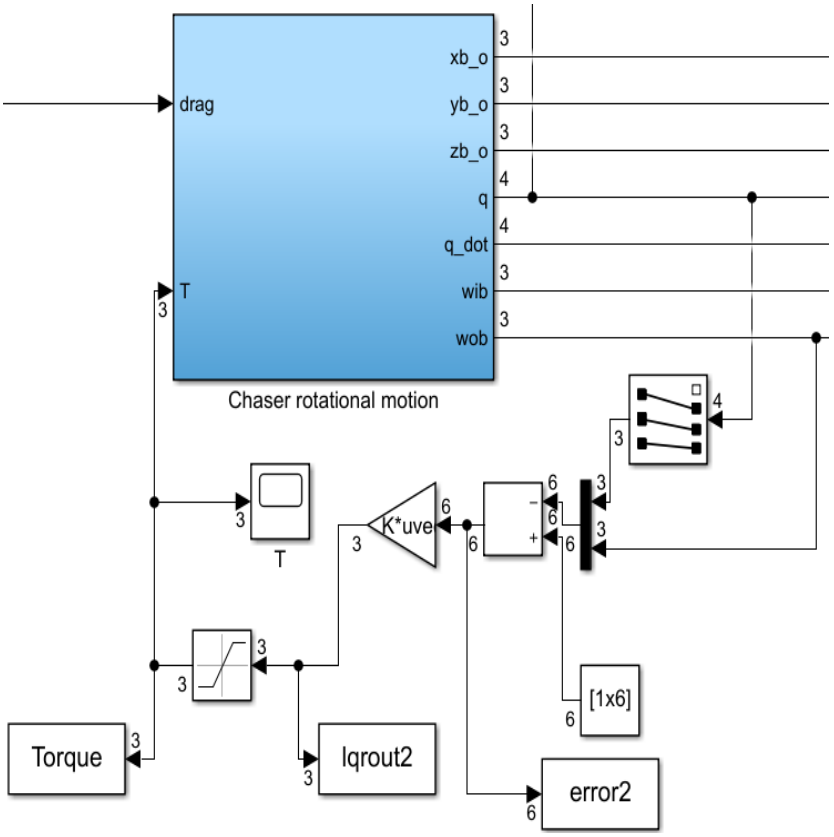

Figure 109: LQR controller for rotational motion.

3.4.4 Neural Network Controller

In order to generate a neural network that can be implemented in the Simulink model, it has been used the Mathworks Neural Network Toolbox, which allowed to create the Neural Network controller both for the rotational and translational motion. The two blocks are shown in Fig.60 and Fig. 61.

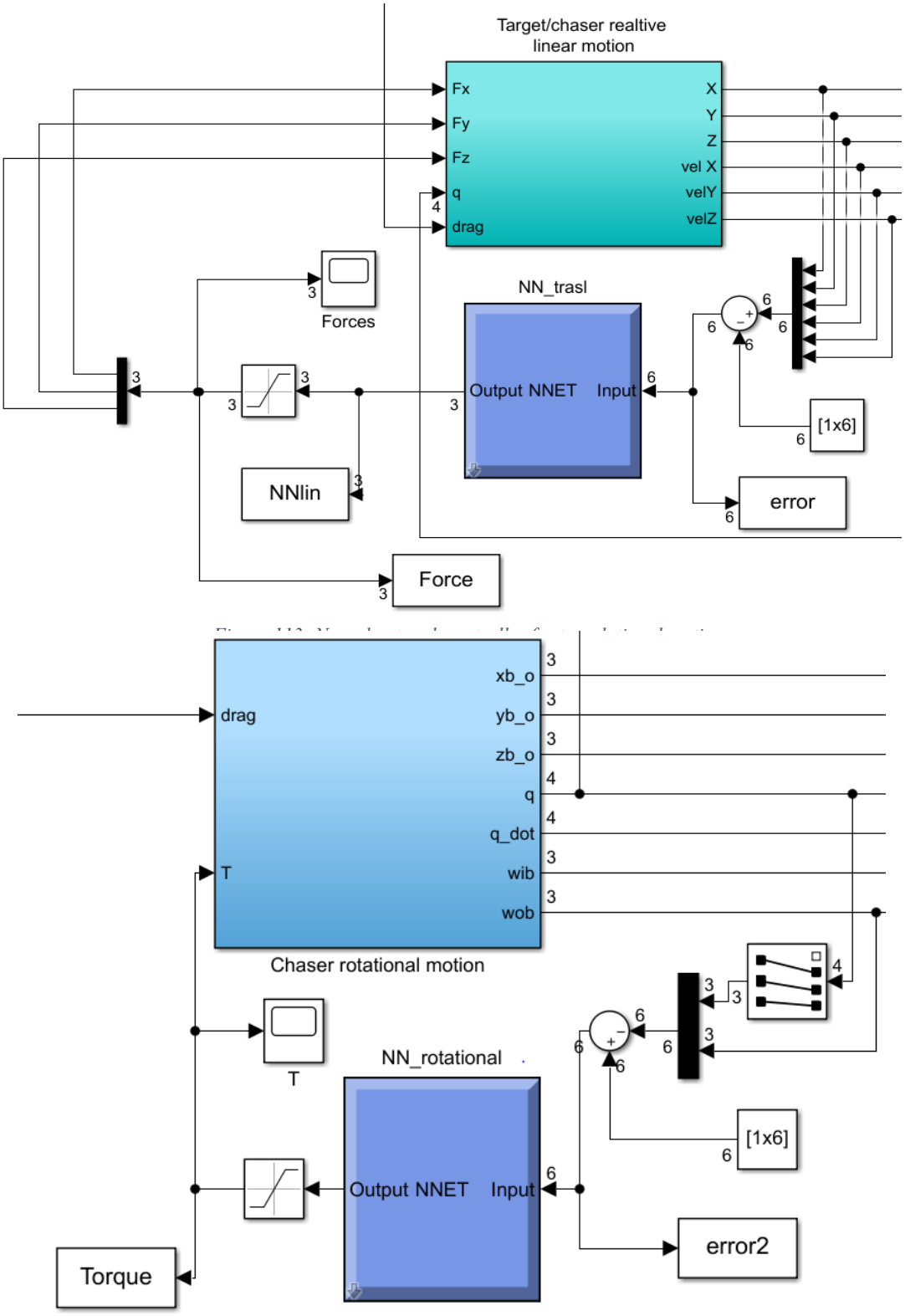

Figure 112: Neural network rotational motion.

When the command “nnstart” is launched in Matlab’s command window, the toolbox offers four types of the design methods:

1. Fitting app: using the supervised learning technique, it adapts the network to unknown data sets. Starting from the sample inputs, it trains the networks in order to generate the desired outputs. This type of technique is consequently particularly suited for modelling and controlling dynamic system;
2. Pattern recognition: starting from the supervised or unsupervised classification, it is based on finding some equal characteristics between a class of objects. This learning technique is useful, for example, in text classification or radar processing;
3. Clustering: it creates a neural network finding similarity hidden patterns or groupings in data, using the unsupervised methodology. For this reasons, it is used in object recognition;
4. Regression: the input and output are linked in a relationship recognized by the algorithms.

In this paper, it has been used the first design method. This technique foresees of solving a certain kind of problem using a two-layer feed-forward neural network. In particular, it draws from the Matlab workspace the necessary data (the input and output vector) in order to train efficiently the net. For defining the network, it is necessary to indicate the number of hidden layers, which are constituted by a not specified numbers of sigmoid hidden neurons, followed by linear output neurons. The hidden neurons are activated by a sigmoid function, which has the following formula:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{60}$$

The performance of the NN is evaluated looking at the mean squared error (MSE) and regression analysis. If the MSE has a near-zero value, it means that there is no error. Regarding the regression R instead, a value of 1 means a close relationship between outputs and targets. The obtained results can be recalculated if they are not quite right, going to retraining the net using a modified data set.

3.4.4.1 Training of the NN

3.4.4.1.1 Translational motion

The first step in order to obtain the NN block in Simulink environment is to obtain both the trajectory and the commands vectors from the LQR control. Generating the error vector and the LQR output vector, it has been possible to utilize this twos respectively for constituting the input data to present to the network and the target data that defines the desiring network output. Initially, it has been chosen a number of six hidden layers. Training the network, it has been possible to generate the regression plots. If the fit is perfect, the data are placed along a 45 degree line: this means that the output are equal to the targets. The figure pictured below (Fig. 62) shows how, after every training, the R value is changing. With the aim of approaching the unit value, it has been chosen to decrease the number of hidden layers: it this way, the network has not the necessary power to overfit the data. Retraining the net with this new number of hidden layers, it has been found very quickly the desired R rate and it has been possible to generate the Simulink control block.

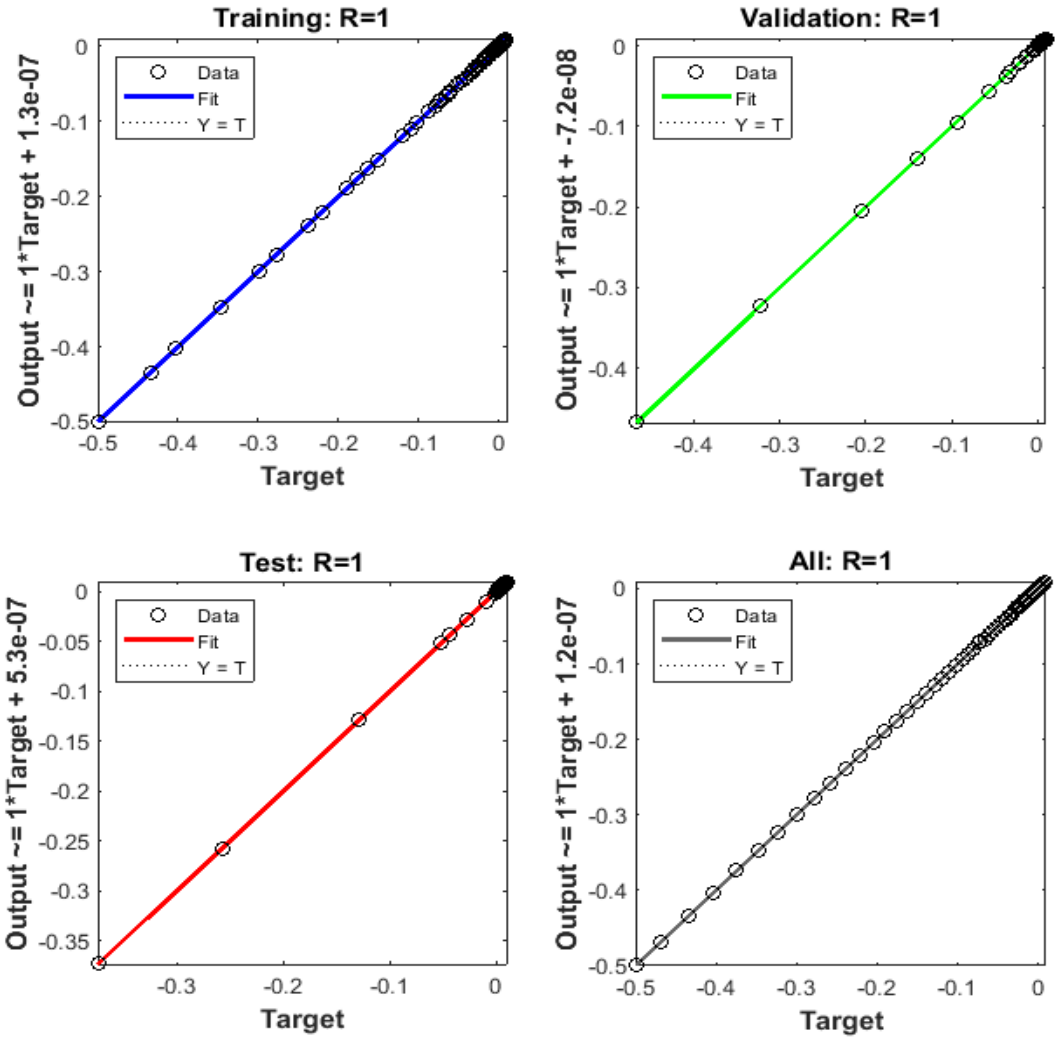

Figure 116: Final training session of translational motion.

3.4.4.1.2 Rotational motion

It is the same process of the translational motion. As input data it has been again used the error vector and as target the LQR output vector. The net has followed the same step of translational motion network, but, in this case, the number of hidden layers has been lowered to 5. After a certain amount of training session, the regression plot has been the following:

4. Results

As reported in the initial paragraph of this dissertation, the aim was to confronting the control of the spacecraft with one standard controller and with the neural network. In order to do so, through a Matlab code, there were stabilized the nominal properties of the Chaser and its initial condition. The physical characteristics were:

- **Mass**: 10 kg;
- $\mathbf{L_x}$: 0.2 m;
- $\mathbf{L_y}$. 0.1 m;
- $\mathbf{L_z}$: 0.3 m;
- $\mathbf{I_x}$: 0.0833 kgm^2;
- $\mathbf{I_y}$: 0.10833 kgm^2;
- $\mathbf{I_z}$: 0.04166 kgm^2.

The orbit parameters have to respect the boundary region gave by the LEO. The orbital elements chosen to be set are the orbit altitude h and the orbit inclination i. In particular:

- **h**: 400 km;
- **i**: 98°.

With the aim of obtain a desired response for the trajectory and attitude, it has been also calculated the gain of the LQR-controller, which is a 3x6 matrix. For having a full picture of the problem, it was necessary to take into account the disturbance forces. The type of force and its relative values are listed below:

- **Aerodynamic Drag**: $F_a \sim -10^{-6}$ N;
- **Solar radiation pressure**: $F_{sun} \sim 10^{-5}$ N;
- **Aerodynamic Torque**: $T_a \sim 0.2124e^{-7}$ 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.

Like anticipated in the previous subchapter, both the thrusters and the reaction wheel gave a boundary region which the control forces and torques must have retuned. In particular:

$$-0.4N \leq F_{th} \leq 0.4N \tag{61}$$

$$-0.01Nm \leq T_{RW} \leq 0.01Nm \tag{62}$$

4.1 Nominal condition: translation motion

The nominal condition of the Chaser were:

- **x**: 50 m;
- $\mathbf{v_x}$**:** 0.15 m/s;
- **y:** 2 m;
- $\mathbf{v_y}$**:** 0 m/s;
- **z:** 0 m;
- $\mathbf{v_z}$**:** 0 m/s.

These inputs were taking into account by the LQR-controller which utilized them in order to generate the control vectors. After that, it has been possible to train the NN controller, shows in in Fig.63, more than one time. In this way, it has been done an initial comparison between the behaviour of the spacecraft under the control of the LQR and of the Neural Network. Like pictured

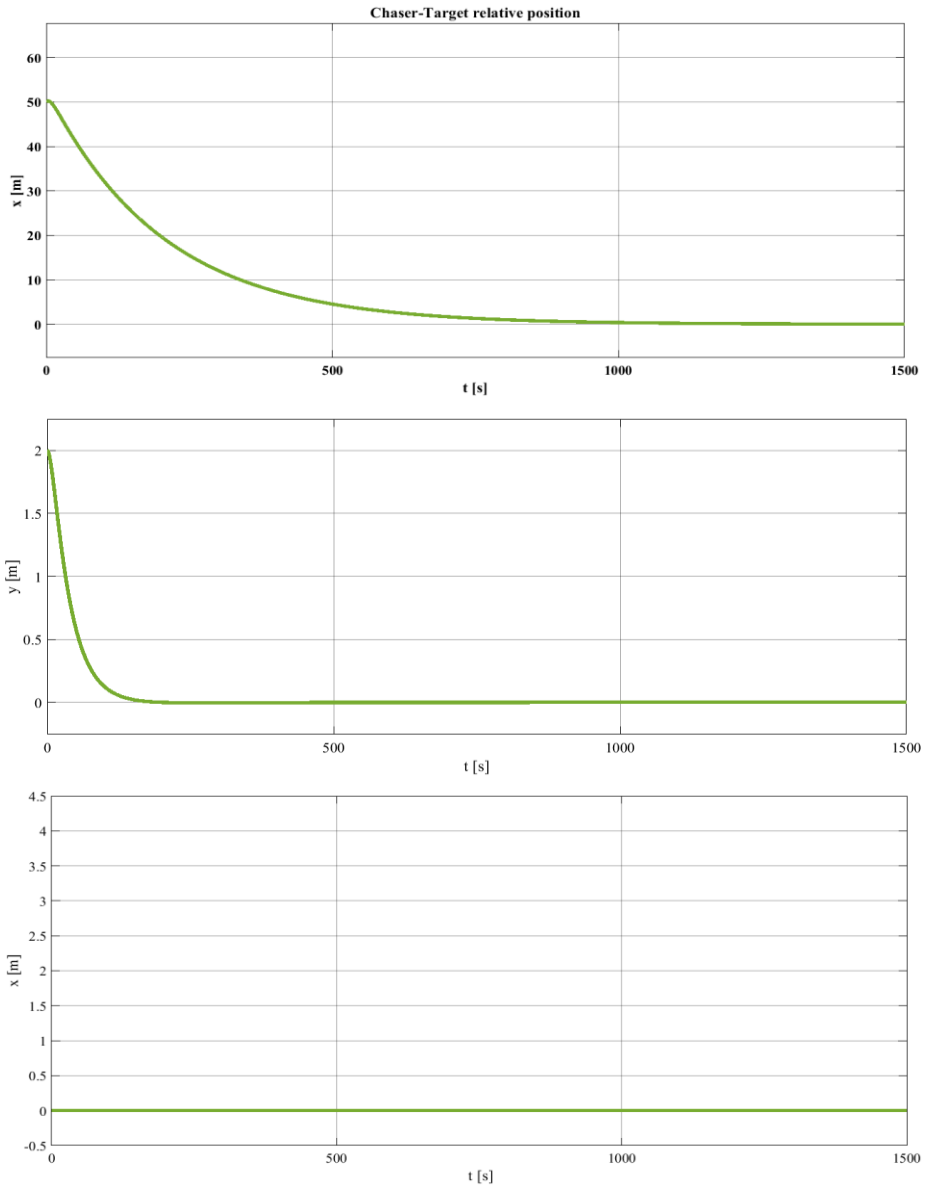

Figure 119: Chaser-Target relative position, LQR control.

in Fig.64, with the LQR, the Chaser reaches the Target after almost 1000 seconds in x, while the y target is accomplished after only 200 s. Of course, the z position stays equal to 0.

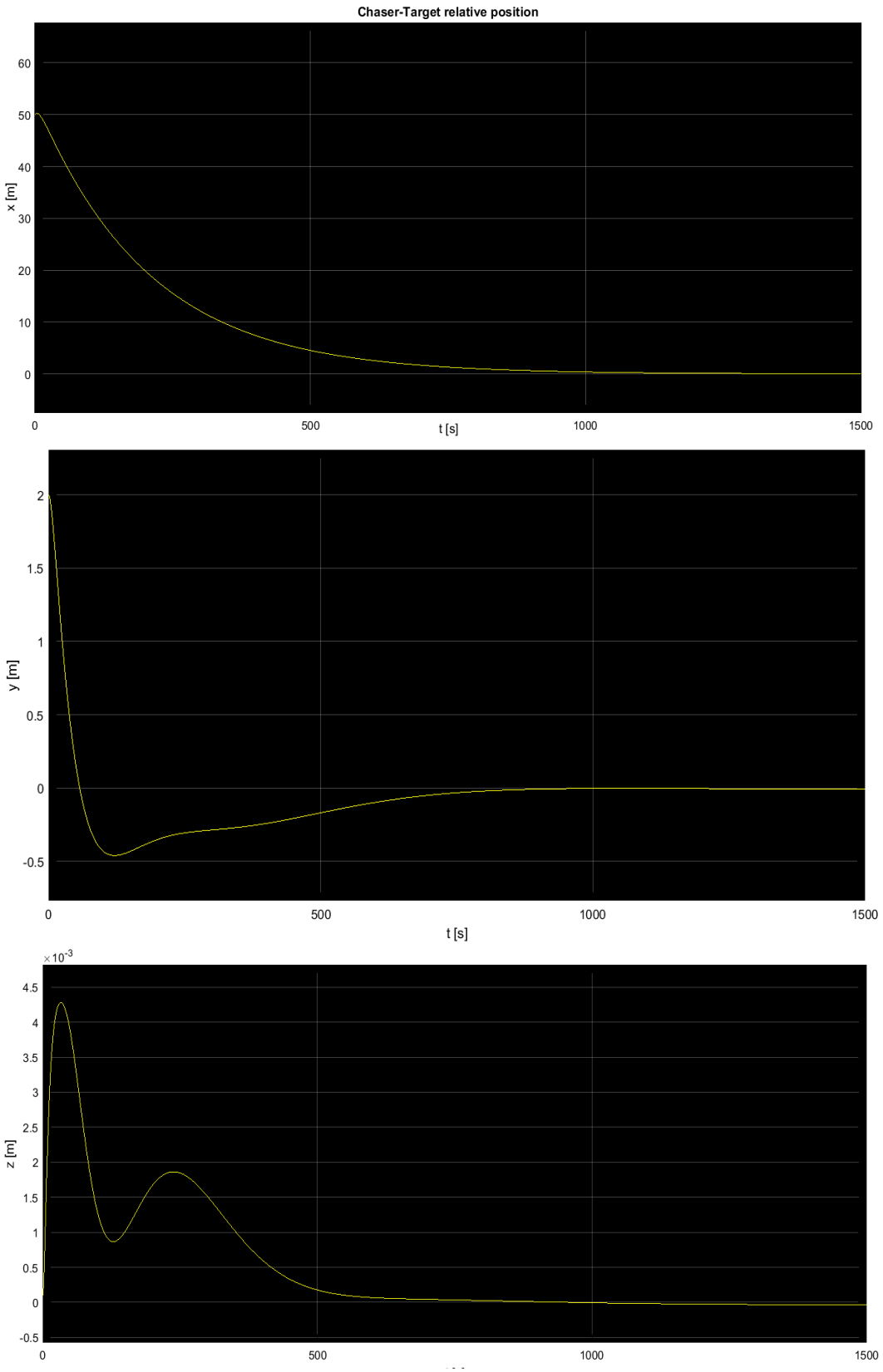

Figure 122: Chaser-Target relative position, NN control.

From the graphics presented below, it is possible to make a first comparison between the control made with the LQR and the control with the ANN. In this second case, it is possible to observe:

1. The desired x position is reached in an equal time both using LQR or the NN controller;
2. The y position computed with the neural network is different respect to the first case: now, the target it is reached after about 800 s;

3. Finally, it can be observed a fluctuation in the graphics of the z position. It is important to underline that this oscillation stays in a range of 10^{-3} m. So it is almost the same both for LQR and NN.

   Plotting the relative velocities, it has been possible to create the following graphics (Fig.65 & 66):

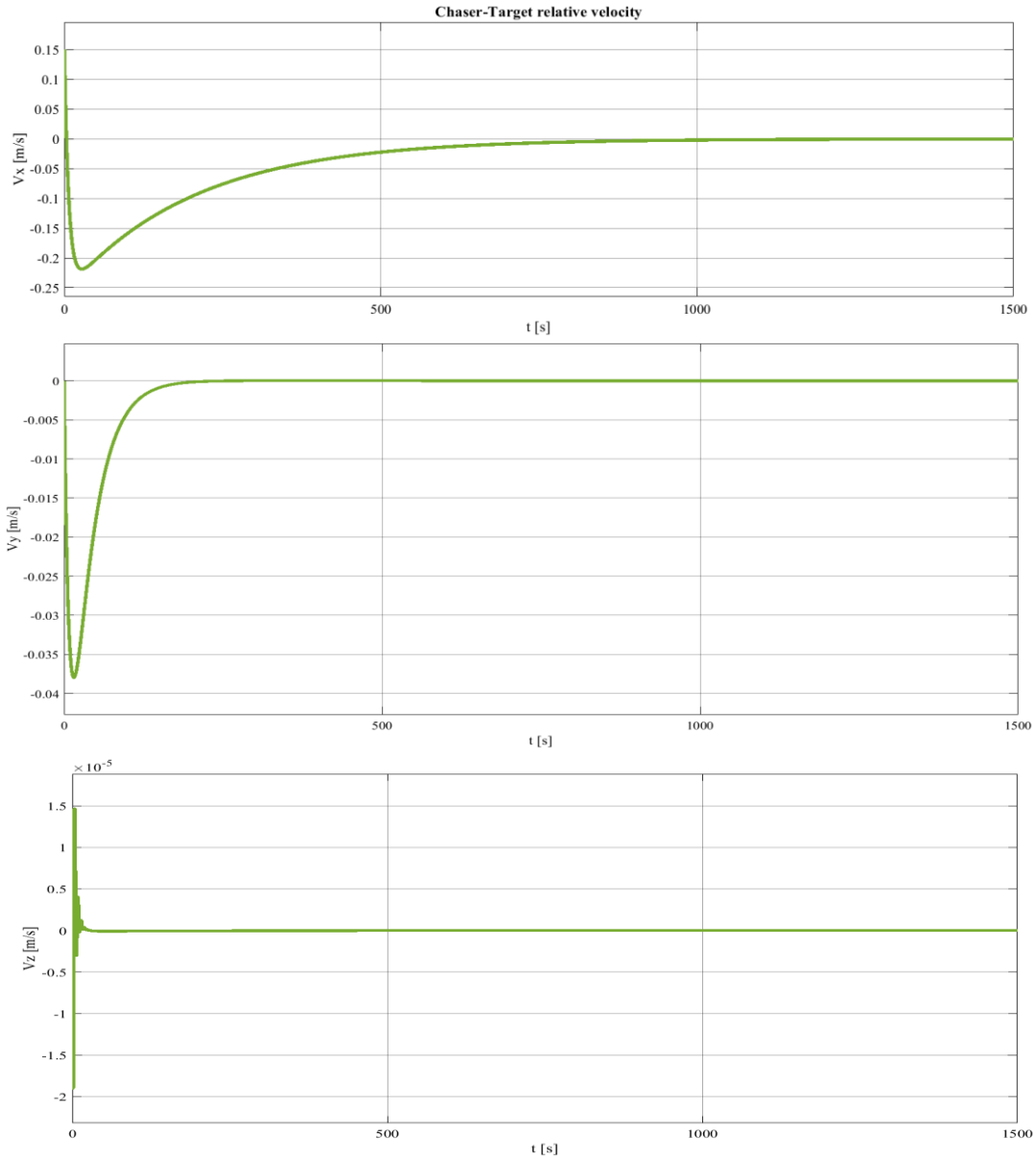

Figure 125: Chaser-Target relative velocity, LQR control.

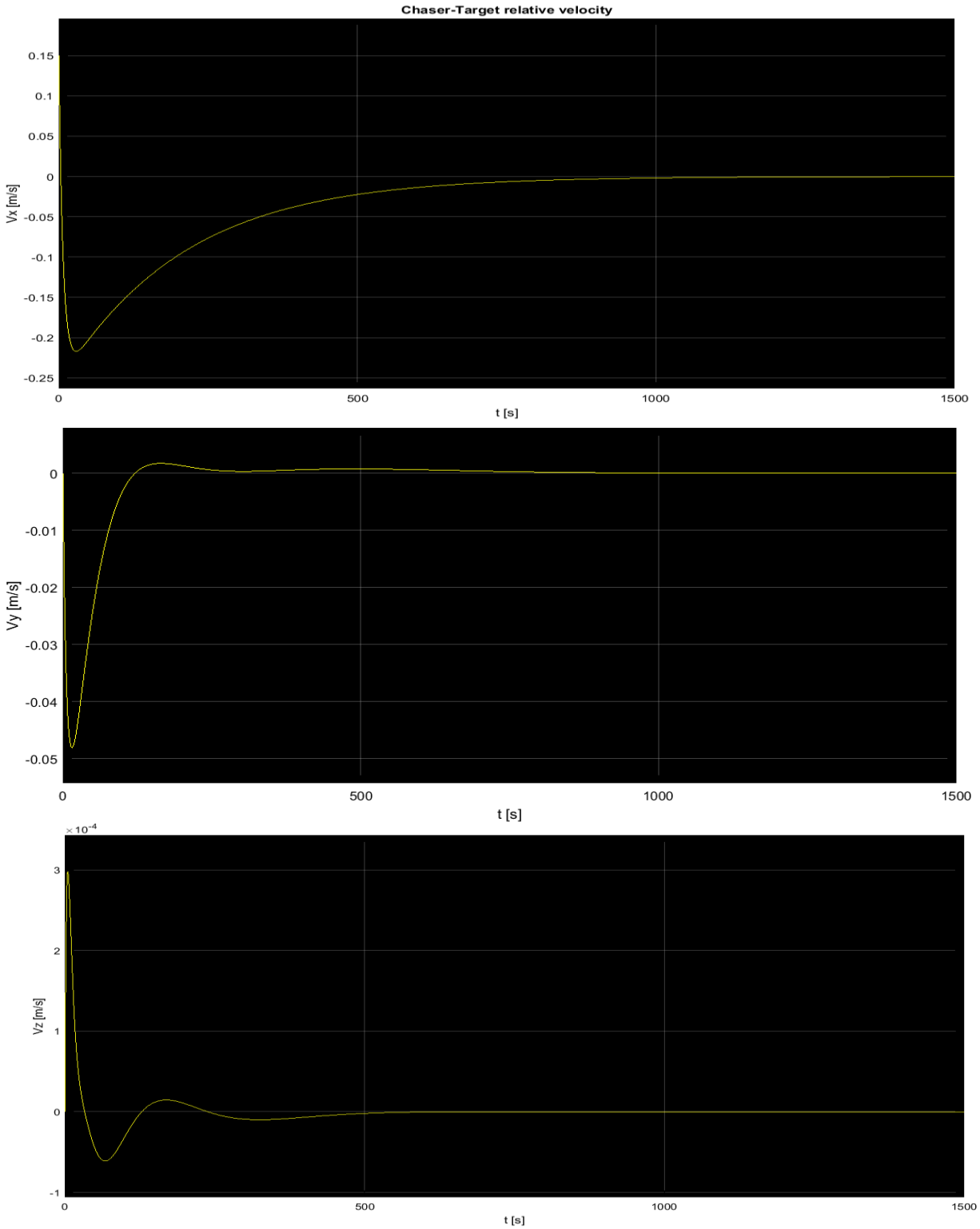

Figure 126:Chaser-Target relative velocity, NN control.

1. The curves of the two v_x velocities are practically the same;
2. The velocity in y direction reaches the desired goal below 200s in the first case, while it takes over 500 s to the ANN to give the same results;

3. Both the v_z present an initial overshoot, followed by a quickly achievement of the desired target

Finally, the last comparison has been made between the forces for each axis. The two figures (Fig.67 & 68) are pictured right below:

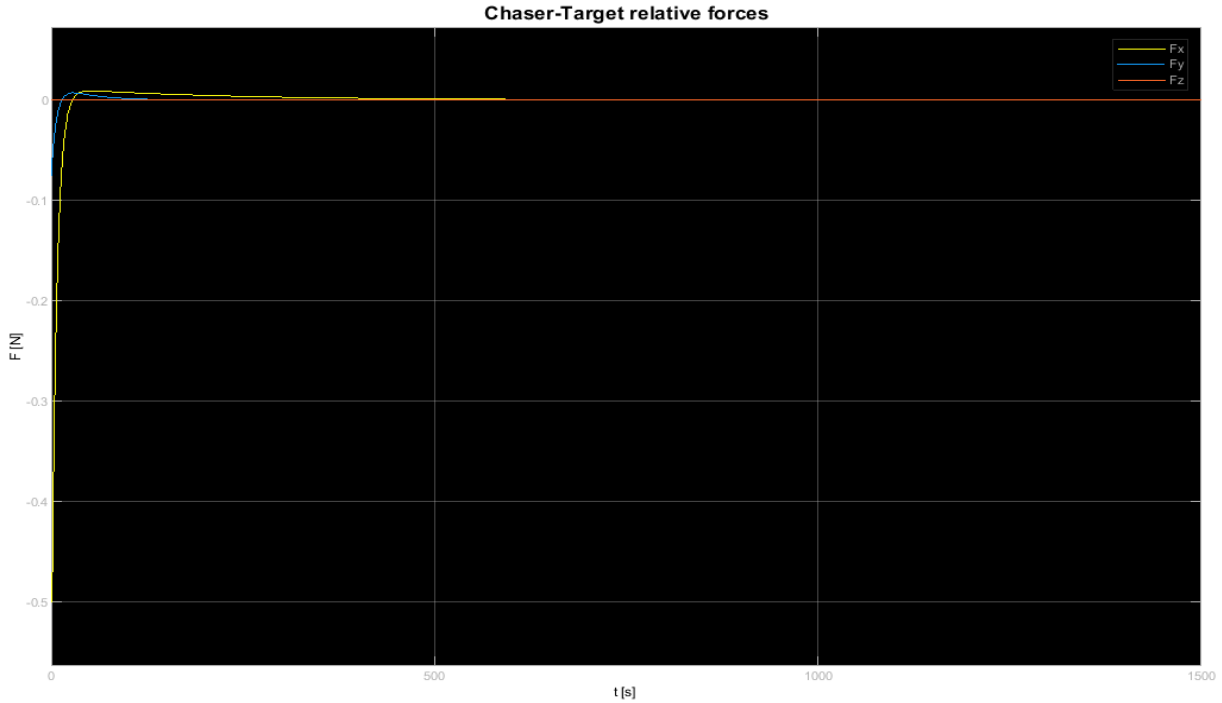

Figure 130: Chaser-Target relative forces, LQR control.

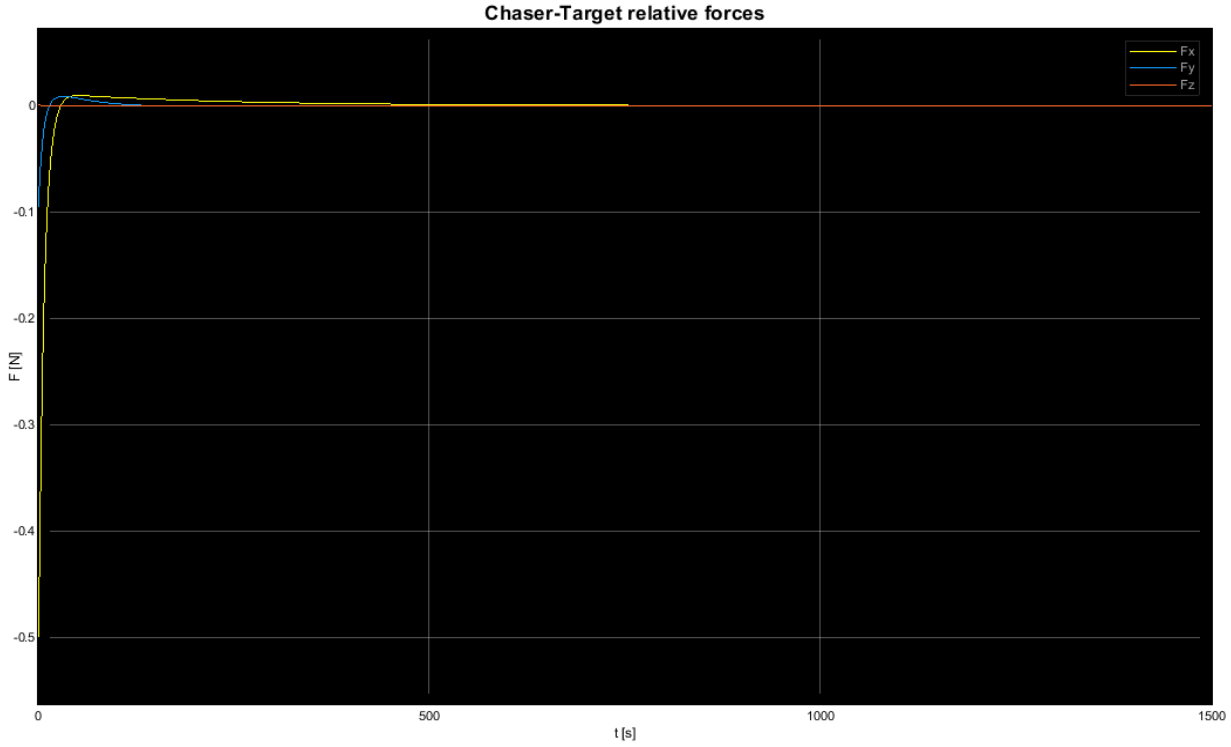

Figure 127: Chaser-Target relative forces, NN control.

It is possible to notices that, in the first case, the three forces reach the zero line after about 600 second. On the other hand, the forces controlled with the artificial network reaches the desired spot after 750 seconds.

After all this confrontations, it is already possible to note that, despite the fact that in most cases the neural network needs more time to reach the desired target, it responds generally very efficiently to the extern conditions. In the next chapter are therefore presented some simulations with different output for each of them.

4.2 Different conditions: translational motion

4.2.1 Simulation 1

After have studied the nominal conditions, the necessary next step has been to prove the Neural network changing the nominal conditions, setting new values to the position and velocity parameters, and adding a disturbance force too. In the first simulation, it has been chosen the following conditions:

- **x**: 60 m;
- $\mathbf{v_x}$**:** 0.15 m/s;
- **y:** 2 m;
- $\mathbf{v_y}$**:** 0.012 m/s;
- **z:** 0 m;
- $\mathbf{v_z}$**:** 0 m/s;
- **Disturbance force:** $-2e^{-6}$ N;

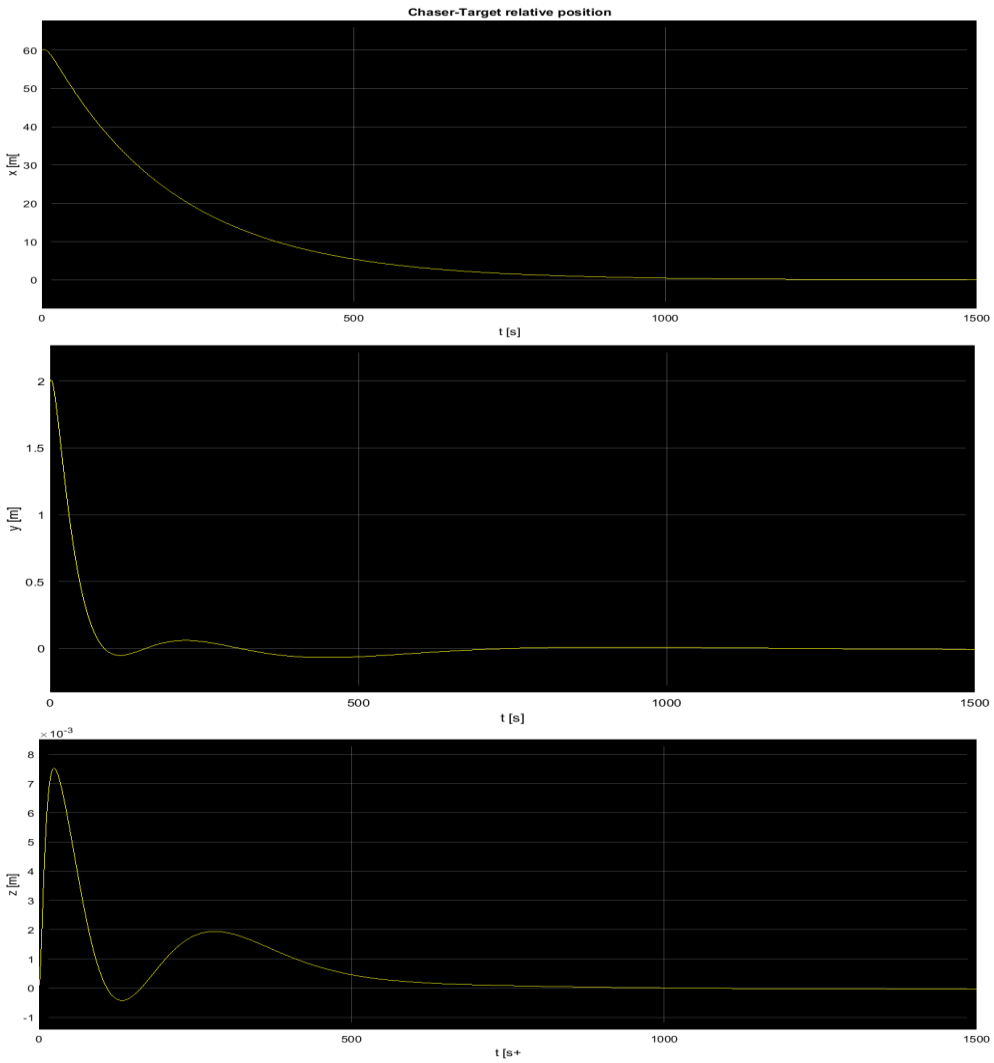

Figure 133: Chaser-Target relative position, NN control.

The simulation shows how the desired position has been reached, even though the conditions have been change. It is interesting to note that the disturbance force induce a noise which turns into an oscillation both in the velocity and force graphs. Despite of it and the change of initial conditions, the Chaser makes the manoeuvre in the right way. The results of this simulation are shown in Fig. 69, 70 and 71:

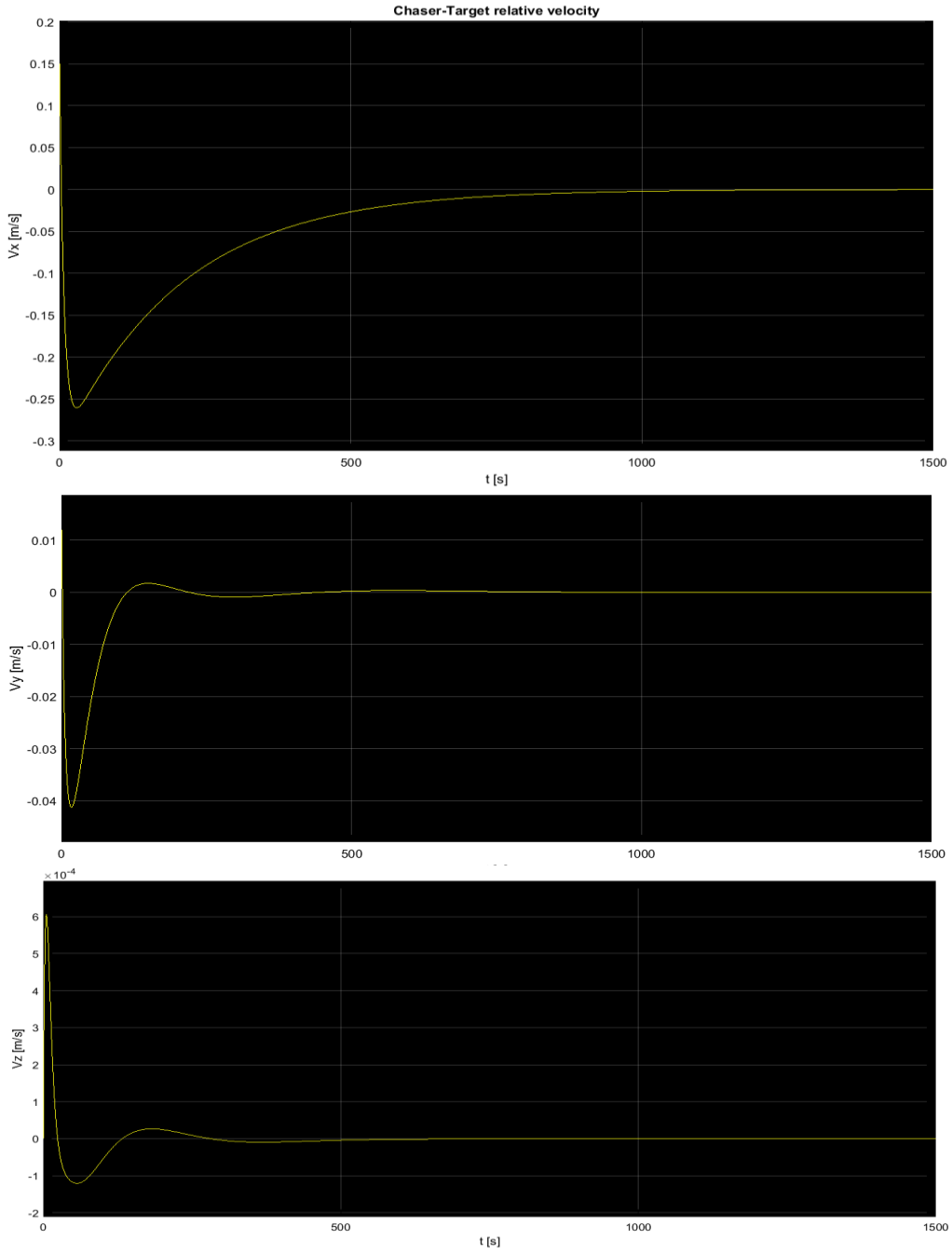

Figure 136: Chaser-Target relative velocity, NN control.

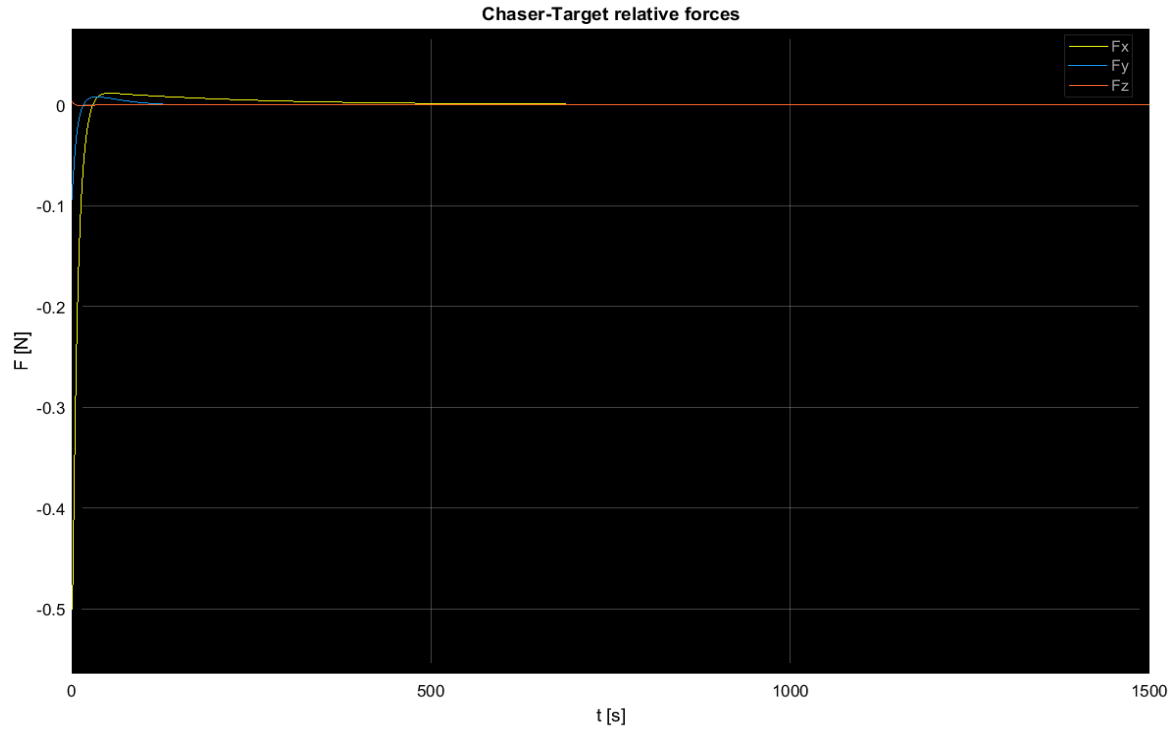

Figure 139: Chaser-Target relative forces, NN control.

The final values of position and the velocity along three axis are:

- $X_{f,nn}$ = 4.144e-02 m;
- $Y_{f,nn}$ = -7.926e-03 m;
- $Z_{f,nn}$ = -3.560e-5 m;
- $V_{xf,nn}$ = -1.968e-04 m/s;
- $V_{yf,nn}$ = -1.785e-05 m/s;
- $V_{yf,nn}$ = -1.758e-08 m/s;

4.2.2 Simulation 2

Differently from the first simulation, during the second one it has been decided to increase the y position and the disturbance force, at the expanse of the x position. The precise values are:

- **x**: 55 m;
- **v_x:** 0.15 m/s;
- **y:** 4 m;
- **v_y:** 0.012 m/s;
- **z:** 0 m;
- **v_z:** 0 m/s;
- **Disturbance force:** -2e^{-5} N;

Setting a simulation time of 1500 seconds, the obtained results are similar to the precedent simulation, the Chaser reaches again the Target, in spite of the changed parameters, in particular despite of the augmented disturbance force. The graphs of position, velocity and force are pictured below (Fig. 72,73 & 74):

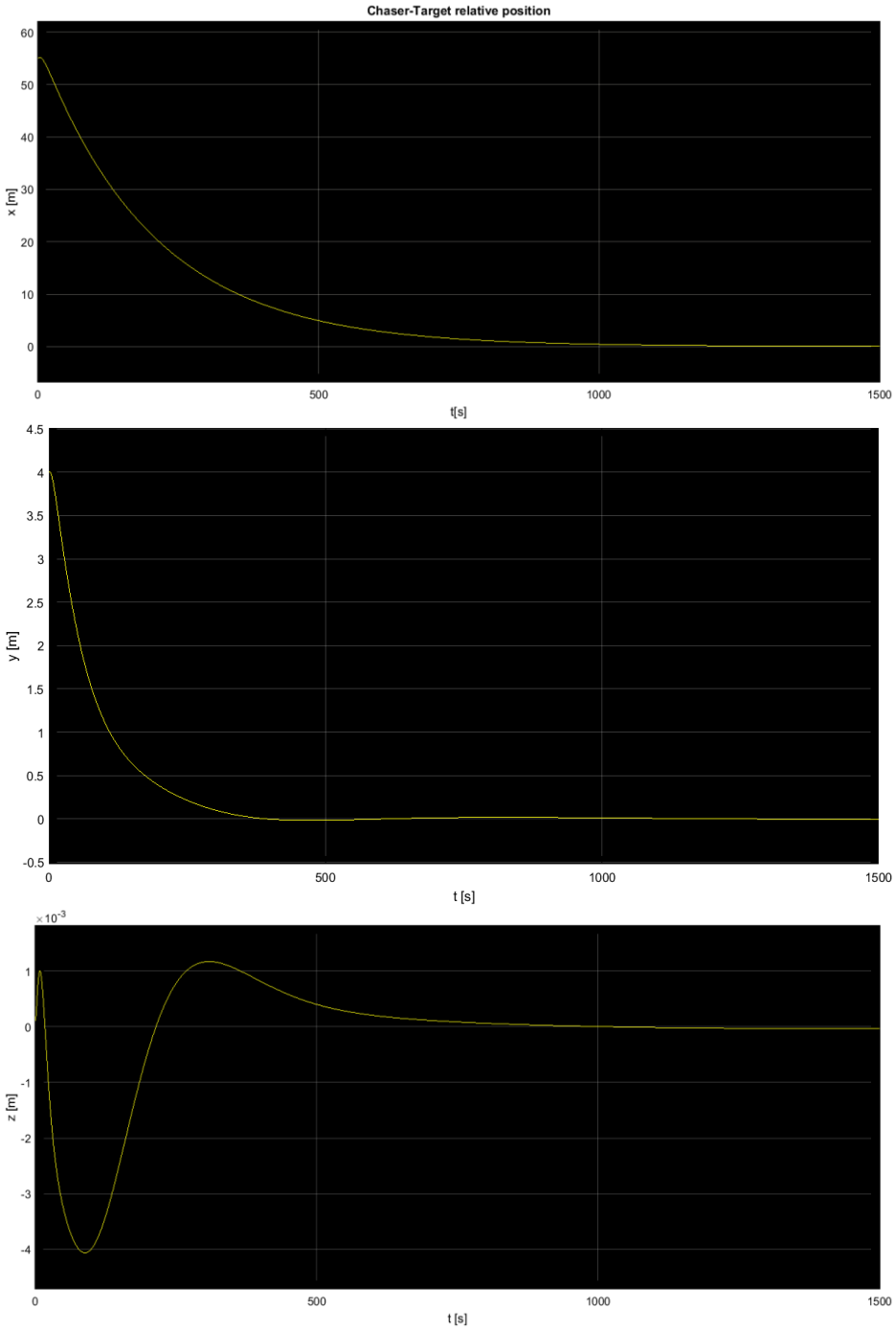

Figure 142: Chaser-Target relative position, NN control.

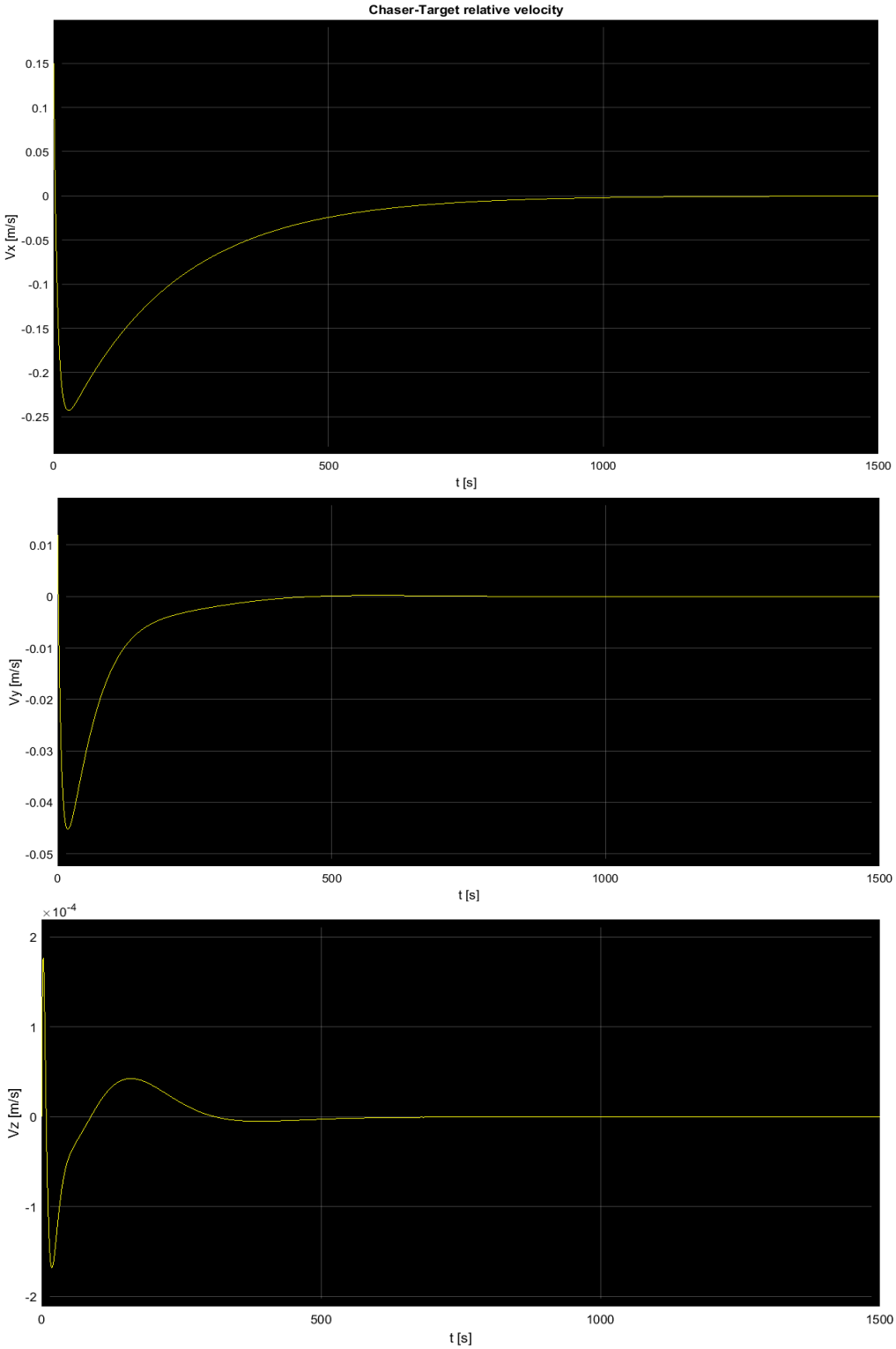

Figure 145: Chaser-Target relative velocity, NN control.

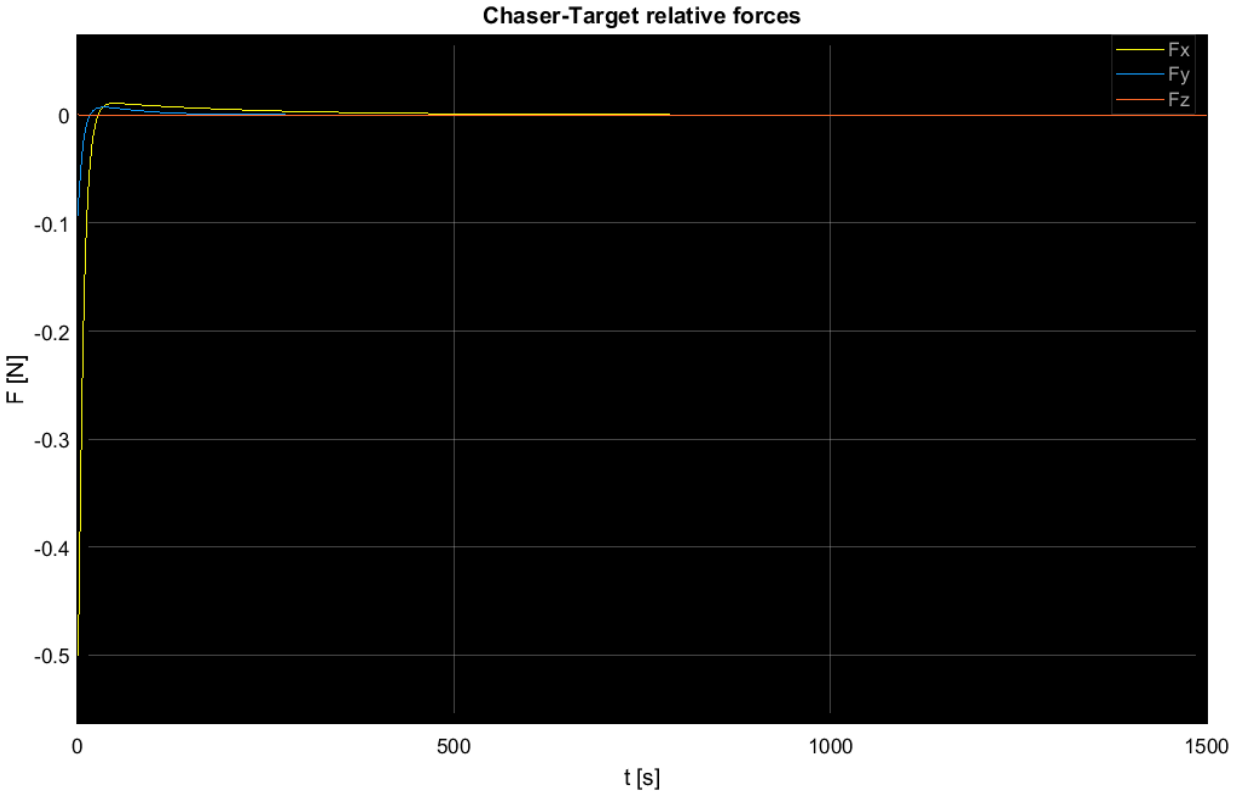

Figure 148: Chaser-Target relative forces, NN control.

In this case, the final values for each calculated parameters is:

- $X_{f,nn}$ = 3.793e-02 m;
- $Y_{f,nn}$ = -8.09e-03 m;
- $Z_{f,nn}$ = -3.593e-05 m;
- $V_{xf,nn}$ = -1.768e-04 m/s;
- $V_{yf,nn}$ = -1.783e-05 m/s;
- $V_{zf,nn}$ = -1.524e-08 m/s;

4.2.3 Simulation 3

In the final simulation, once again the position along the x axis is decreased, while the disturbance force value is further increased like the velocity in y. The orbital parameters, the Chaser physical characteristics and the simulation time remain all the same. For this third simulation, the input parameters are:

- **x**: 30 m;
- $\mathbf{v_x}$**:** 0.15 m/s;
- **y:** 4 m;
- $\mathbf{v_y}$**:** 0.015 m/s;
- **z:** 0 m;
- $\mathbf{v_z}$**:** 0 m/s;
- **Disturbance force:** -2e^{-4} N;

Running the simulation, the following graphs (Fig. 75,76 & 77) are obtained:

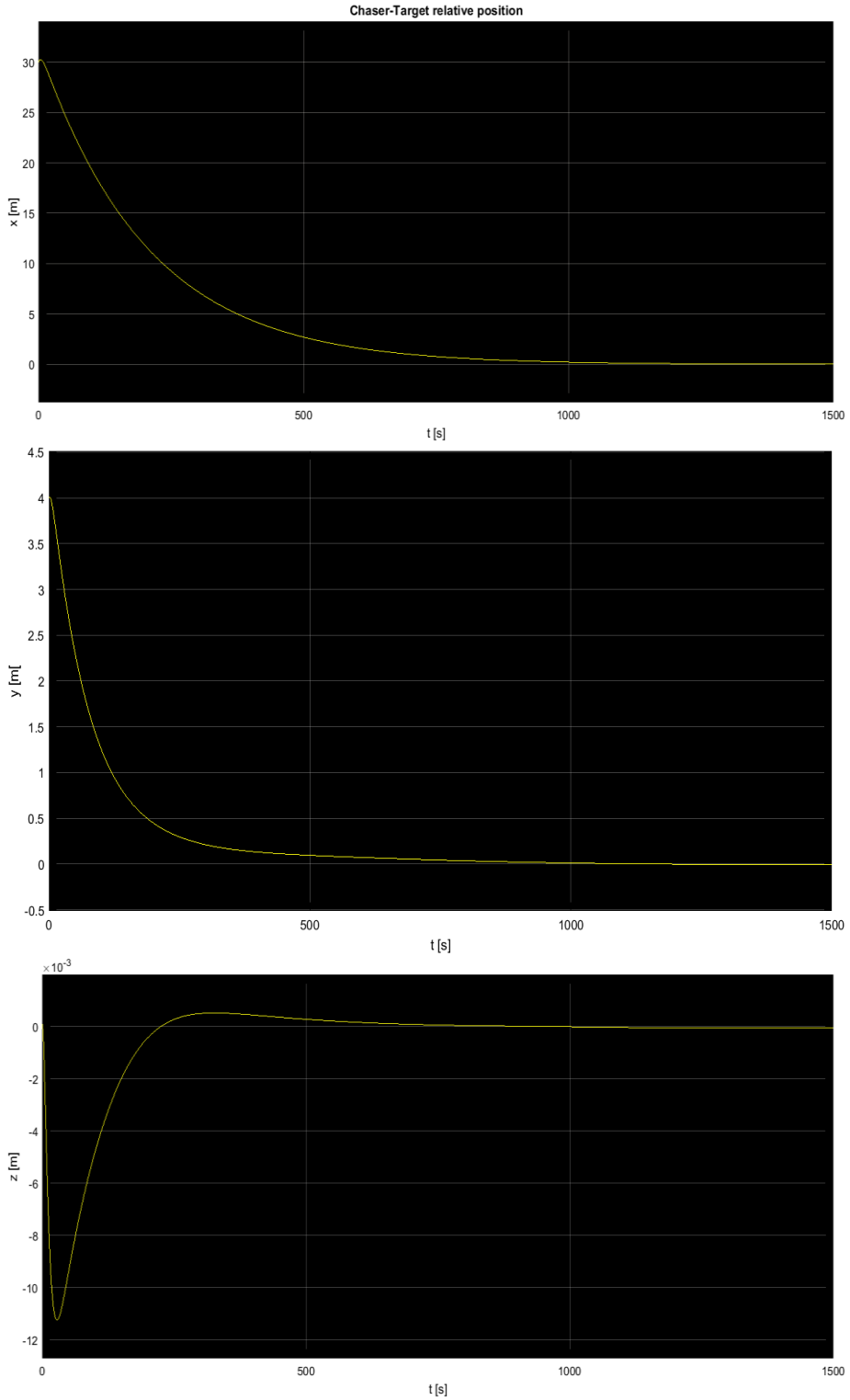

Figure 151: Chaser-Target relative position, NN control.

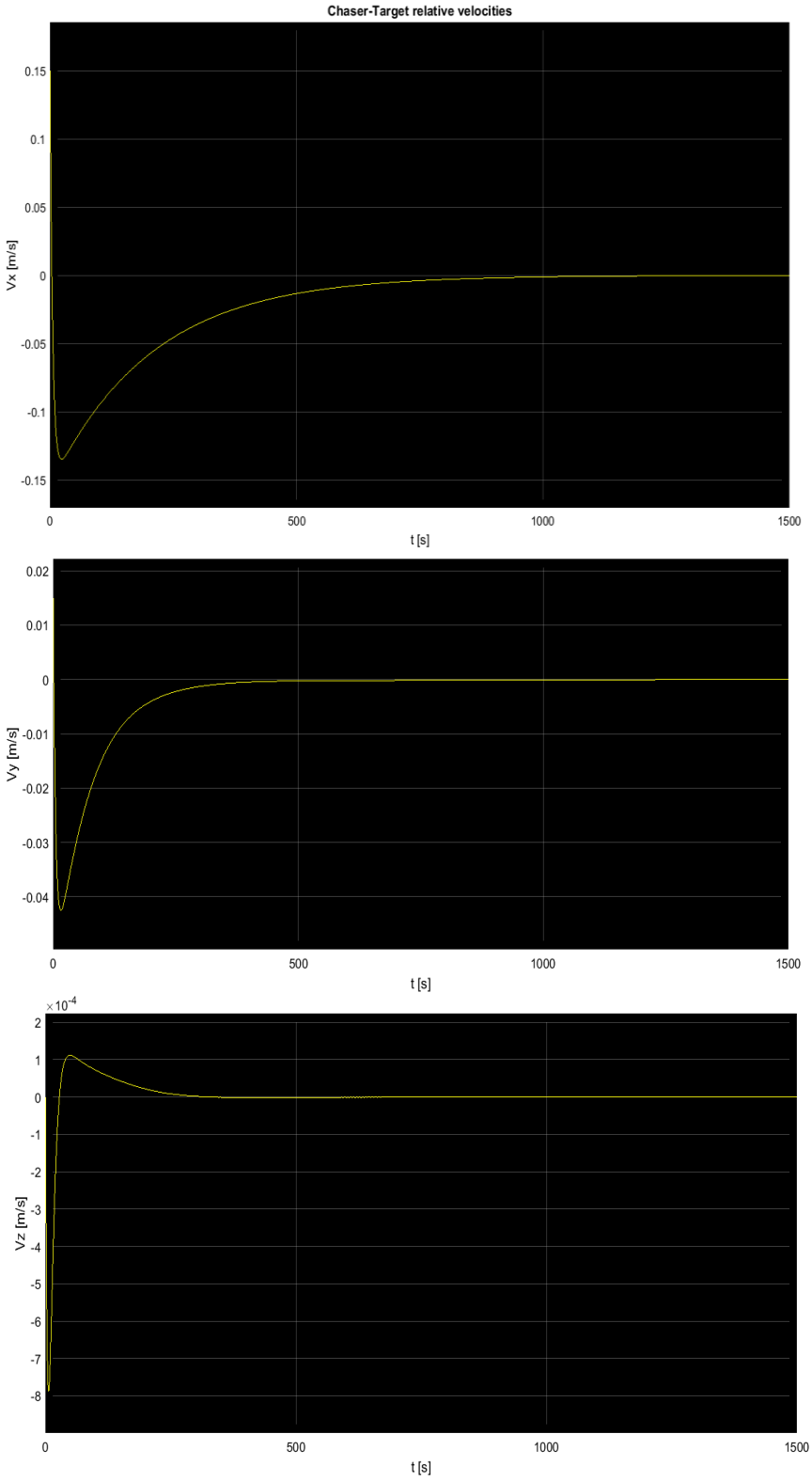

Figure 154: Chaser-Target relative velocity, NN control.

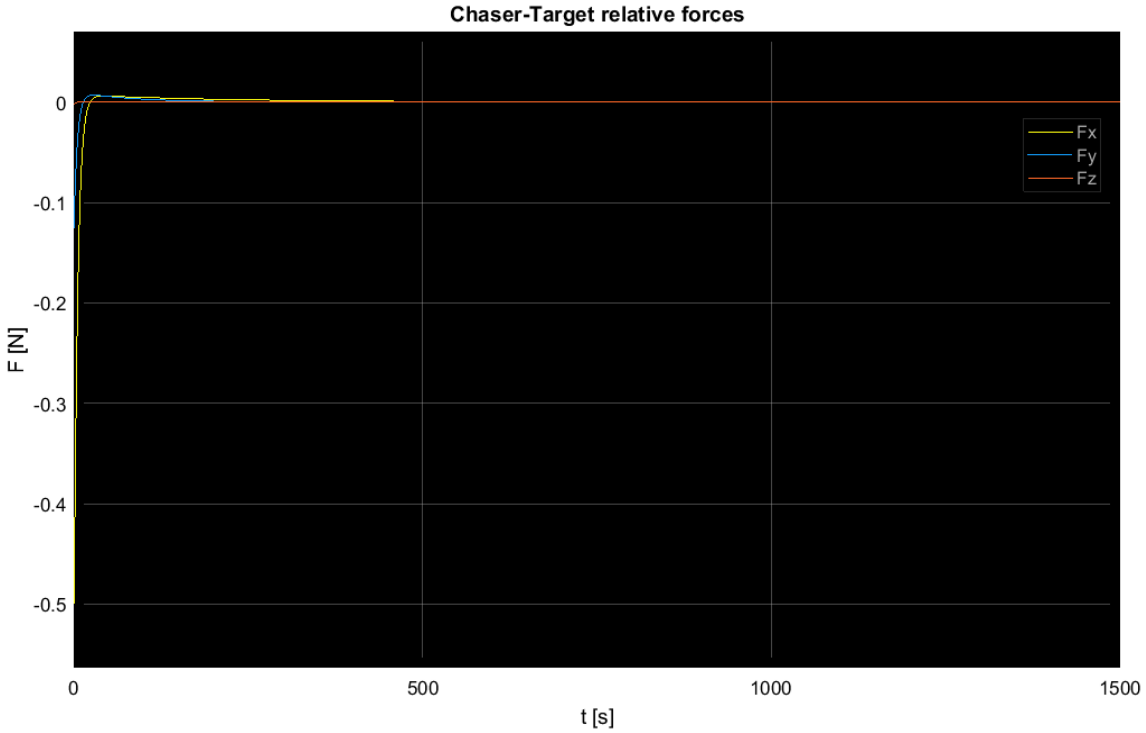

Figure 157: Chaser-Target relative forces, NN control.

It is possible to see that the desired response is reached once again, nevertheless the system takes longer time to "fight" with the disturbance force and shut down the noise. The final values of position and velocity are:

- $X_{f,nn}$ = 2.172-02 m;
- $Y_{f,nn}$ = -9.368e-03 m;
- $Z_{f,nn}$ = -3.750e-05 m;
- $V_{xf,nn}$ = -9.686e-05 m/s;
- $V_{yf,nn}$ = -1.395e-05 m/s;
- $V_{zf,nn}$ = -7.612e-09 m/s;

4.3 Rotational motion: nominal conditions

After studied the translational motion, it is the moment to relate with the rotational motion of the spacecraft. Like done before, it has been made a confront between the control obtained with the LQR technique and the one generated by the Artificial Neural Network outputs. In this case, the network has been trained like did for the translational motion, but it is now formed by 5 hidden layers. The simulation time remain the same, while the new nominal conditions are:

- $\varphi = 10°$;
- $\theta = 10°$;
- $\psi = 10°$;
- ω_x = 0.5 rad/s;
- ω_y = 0.5 rad/s;
- ω_z = 0.5 rad/s;

In the figure pictured below, there are shown the Chaser attitude angles and the Chaser angular velocities when it has been controlled with the LQR technique (fig.78,79 & 81). Right after that, there are pictured the same parameters, but for the control with the Artificial Neural Network (fig.80,82 and 83).

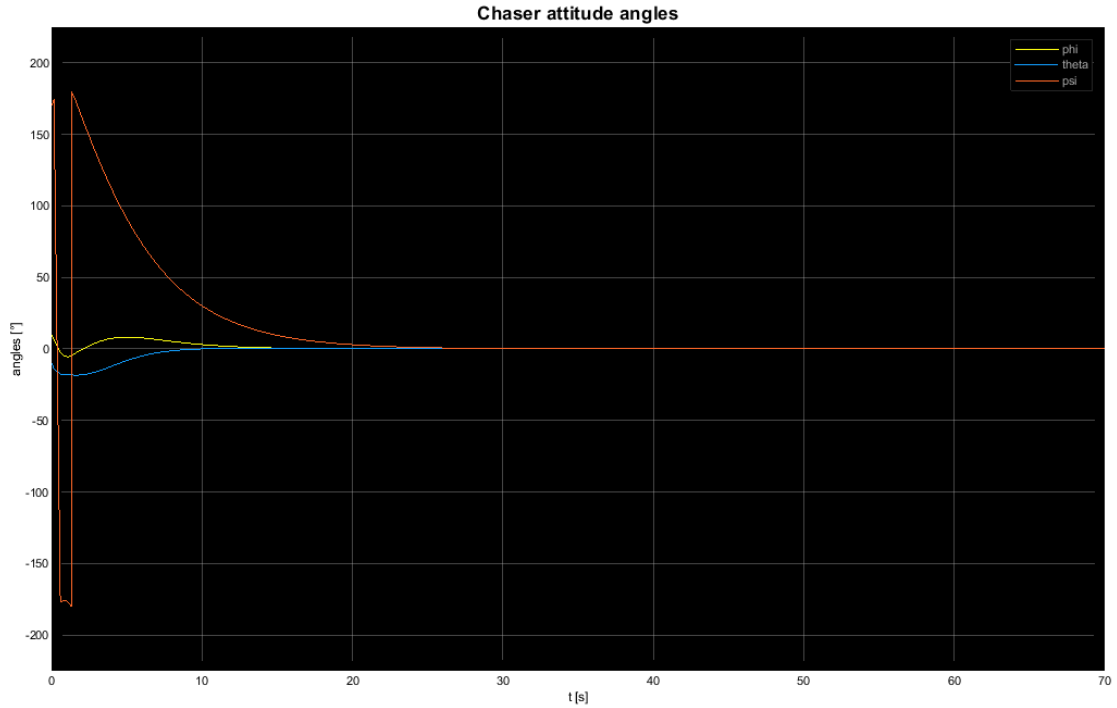

Figure 160: Chaser attitude angles, LQR control.

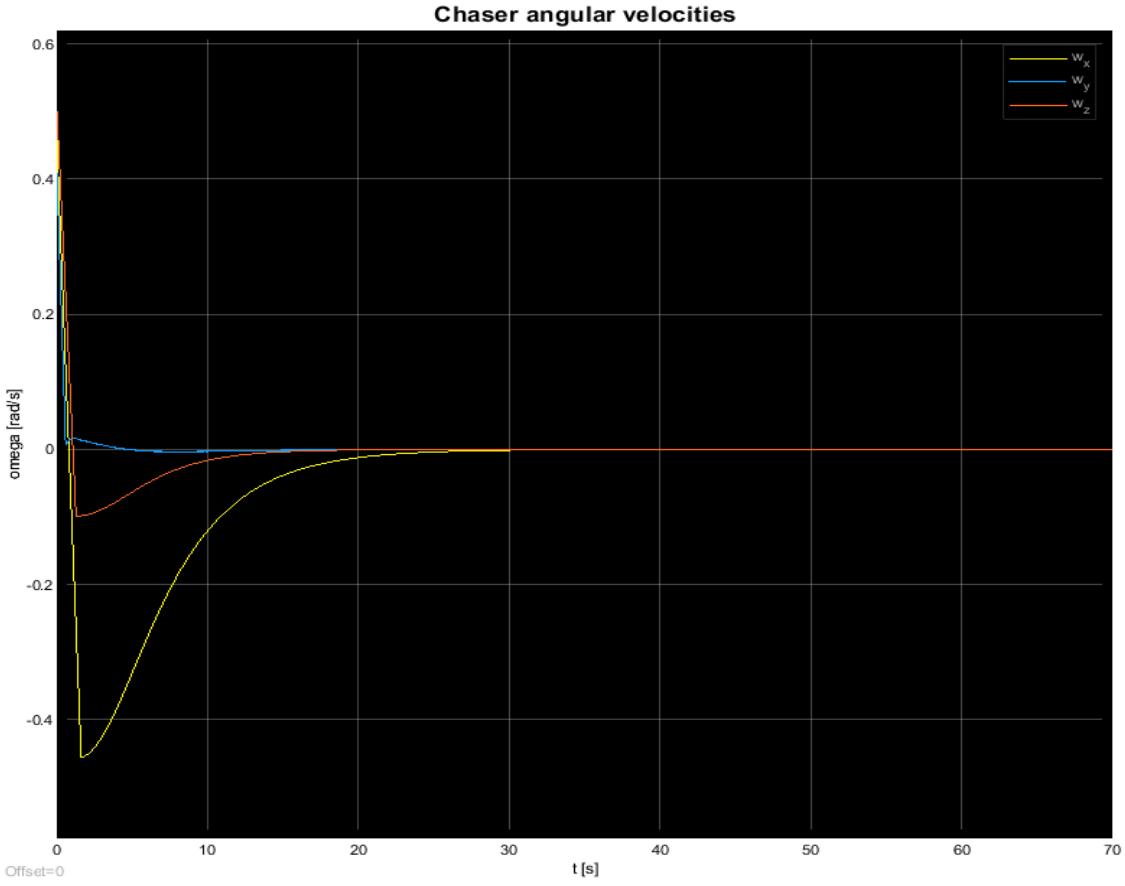

Figure 163: Chaser angular velocities, LQR control.

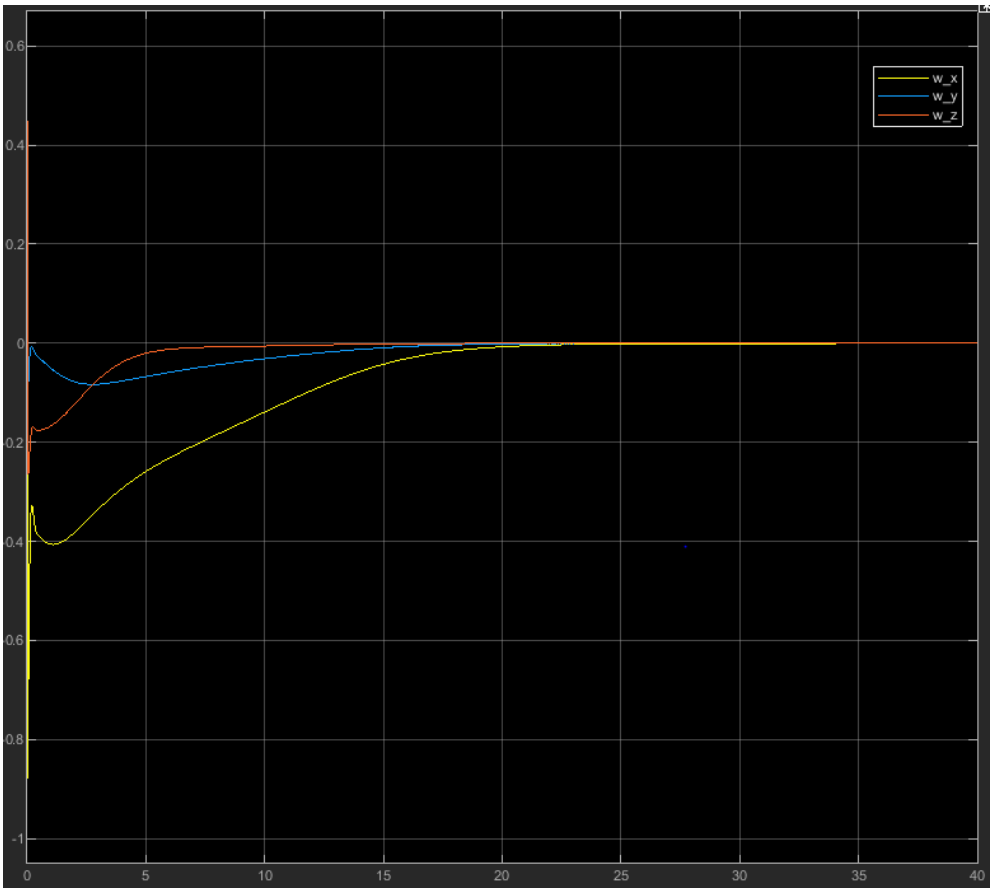

Figure 164: Chaser attitude rates, NN control.

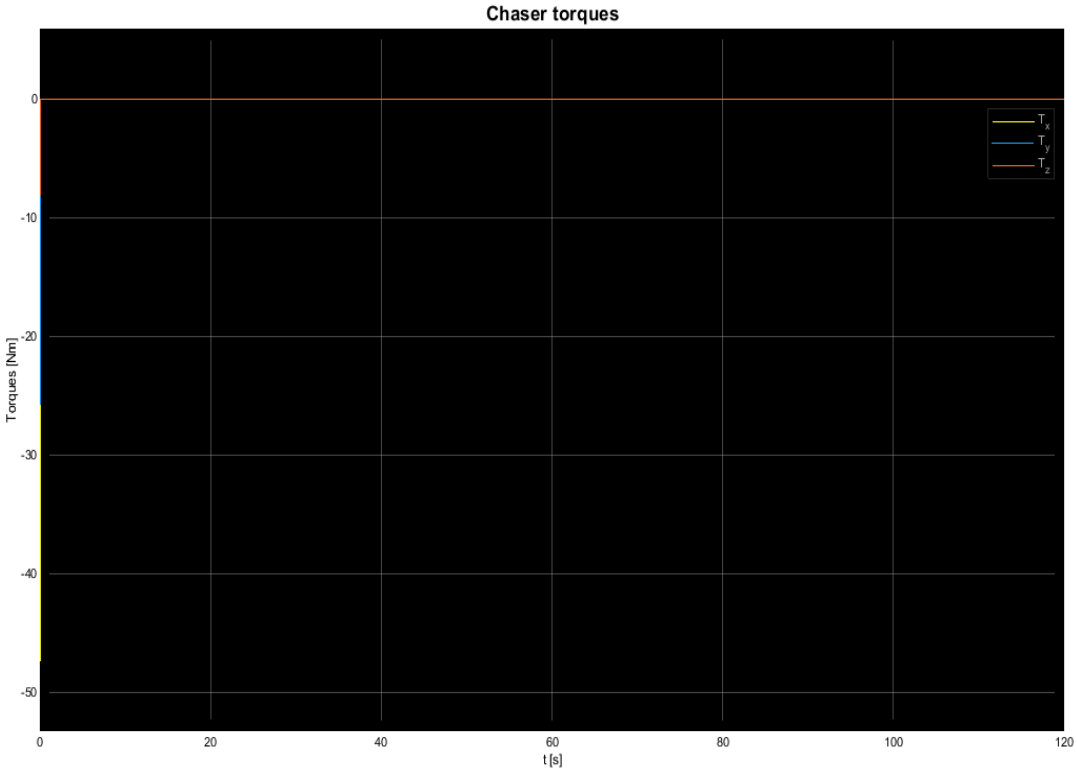

Figure 165: Chaser torques, LQR control.

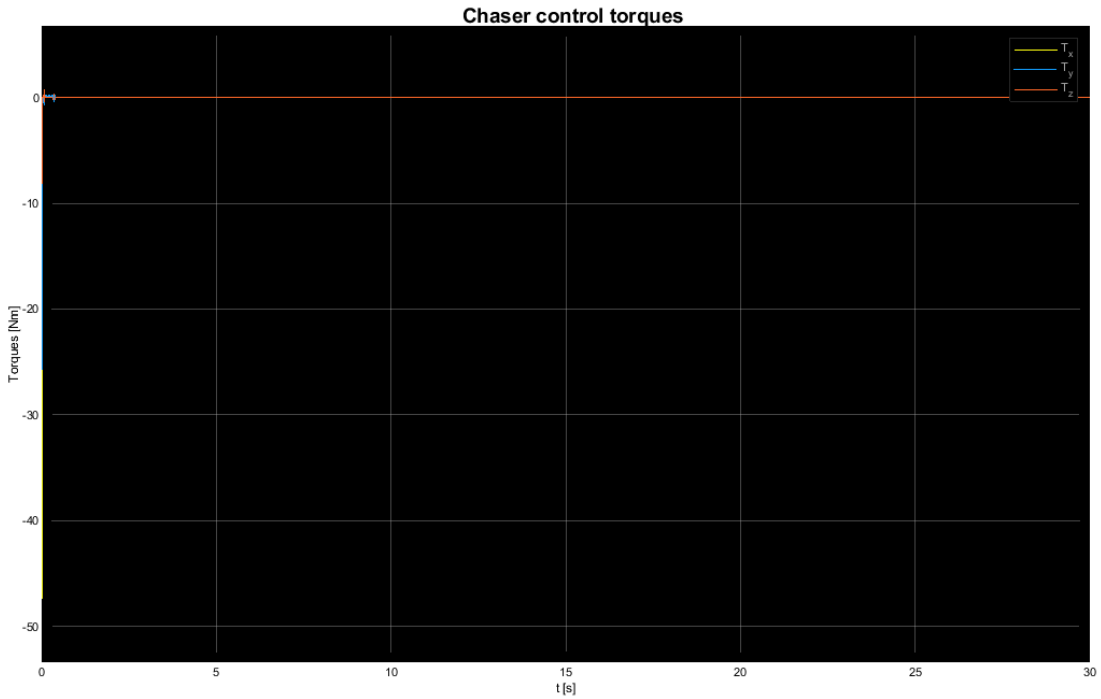

Figure 168: Chaser control torques, NN control.

4.4 Attitude control: new conditions

4.4.1 Simulation 1

Like did before, the Neural Network has been tested to several inputs coming from the Matlab environment. For the first simulation, the input vector was formed by this following characteristics:

- $\varphi = 20°$;
- $\theta = 15°$;
- $\psi = -6°$;
- $\omega_x = 0.5$ rad/s;
- $\omega_y = 0.5$ rad/s;
- $\omega_z = 0.5$ rad/s;
- Disturbance torque: 10^{-6} Nm.

Using this input vector in Simulink, and remaining fixed the simulation time, the obtained graphics have been pictured right below (Fig.83, 84 & 85):

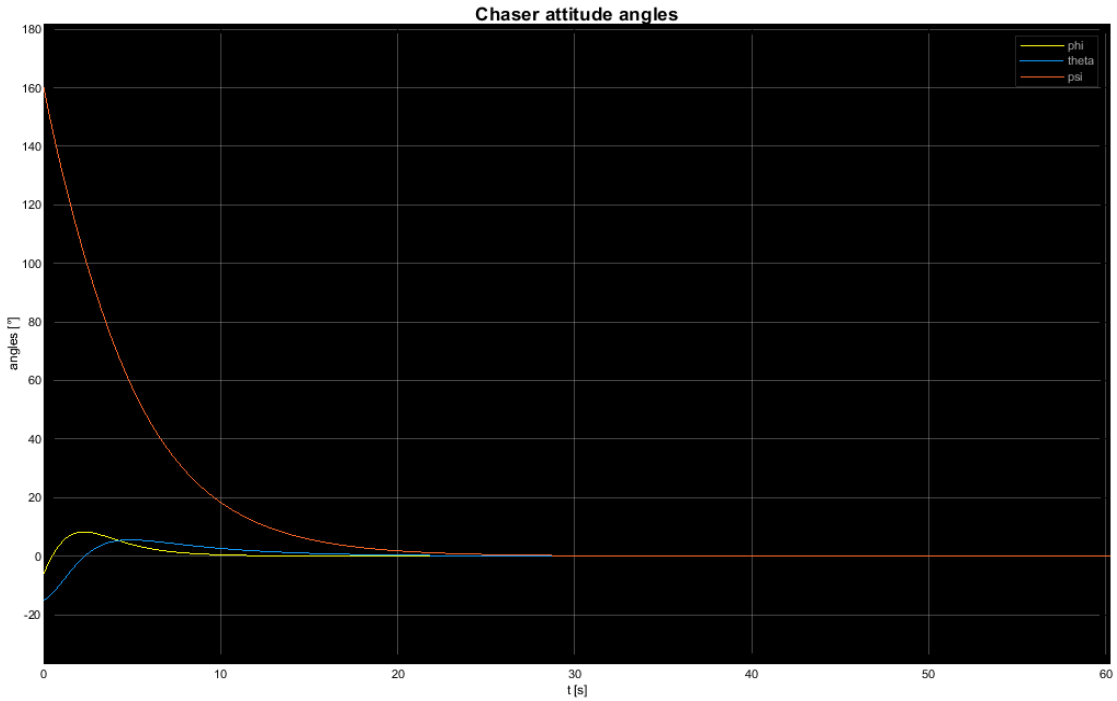

Figure 169: Chaser attitude angles, NN control.

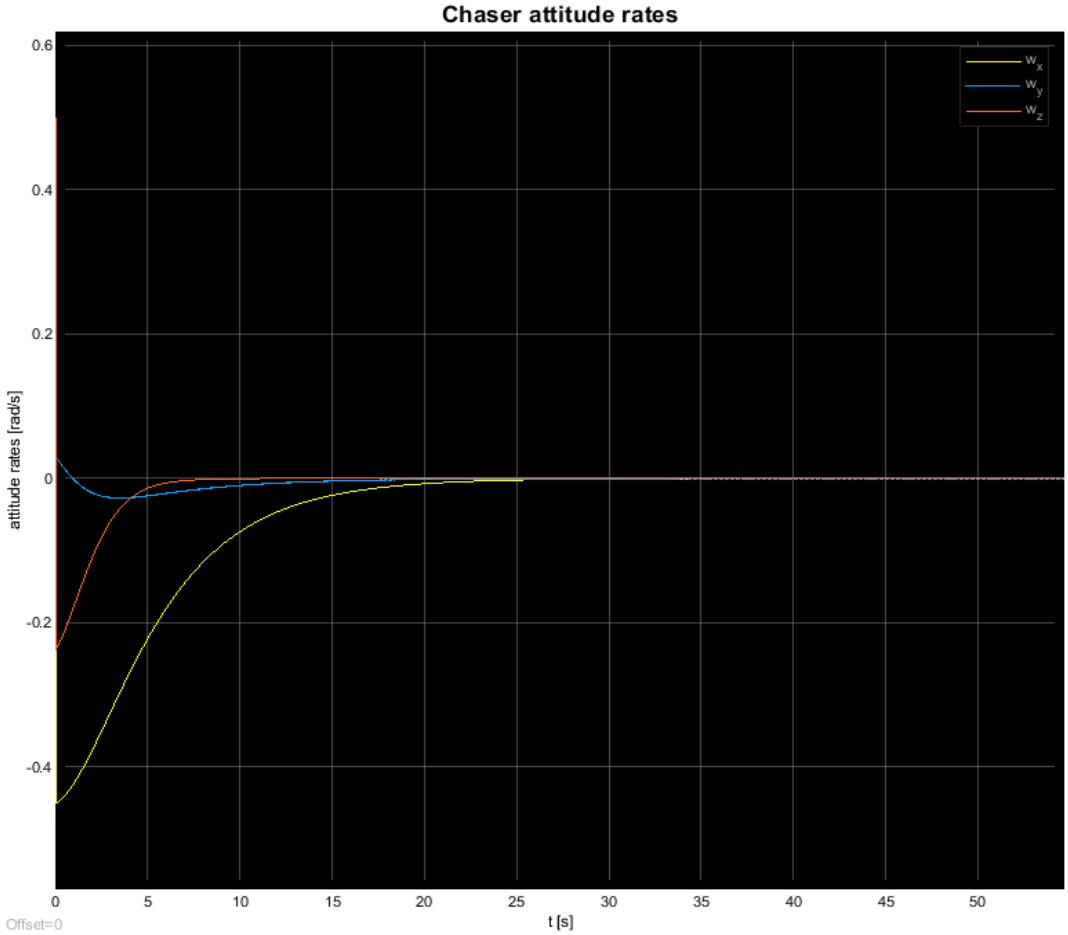

Figure 172: Chaser attitude rates, NN control.

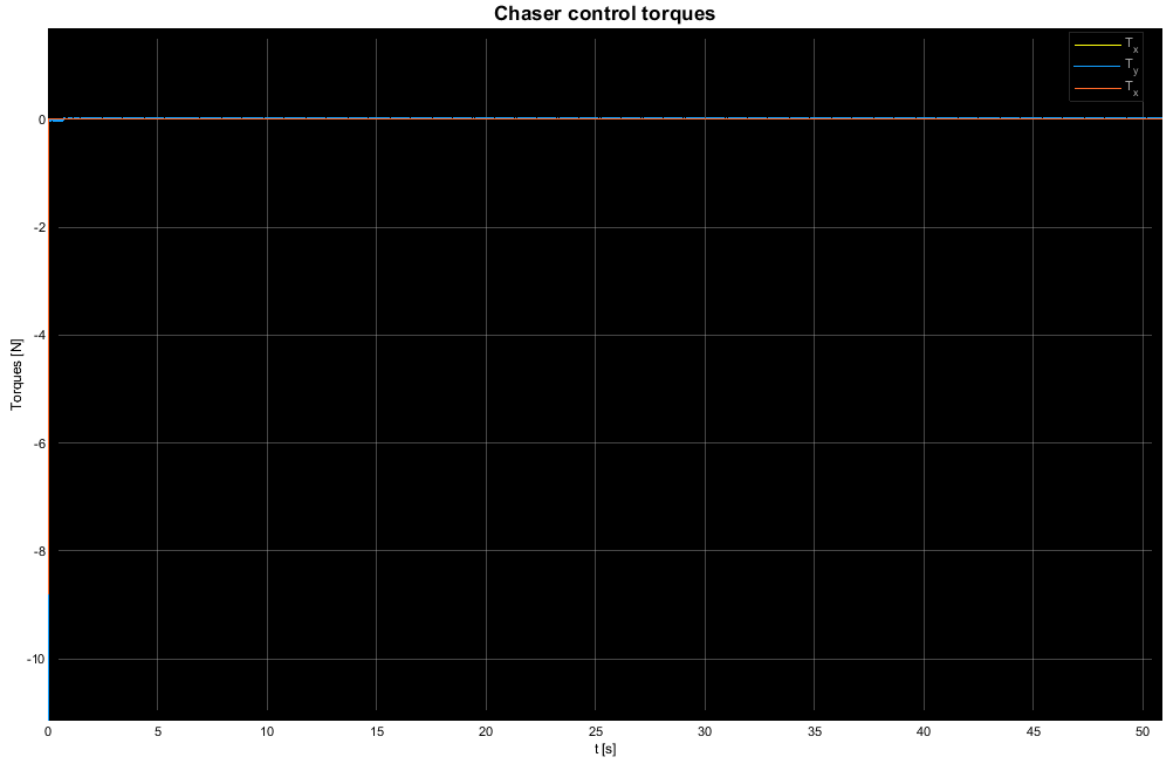

Figure 175: Chaser control torques, NN control.

It is possible to notice that, despite the presence of a disturbance torque, the NN control can still control in an efficient way both the angular velocities and the attitude angles. The simulation time has been of 1500 seconds, but in the graphics has been shown only until 50 seconds circa.

4.4.2 Simulation 2

The initial conditions have been again changed respect to the precedent simulation. In particular, in this case, it has been chosen the following input:

- $\varphi = -2°$;
- $\theta = 15°$;
- $\psi = 11°$;
- $\omega_x = 0.06$ rad/s;
- $\omega_y = 0.06$ rad/s;
- $\omega_z = 0.06$ rad/s;
- Disturbance torque: 10^{-5} Nm.

In this case, all the data have been changed respect to the precedent simulation. In particular, the roll angle has been set negative and the disturbance torque has been increased of an order of magnitude. The graphics of the angular velocities, angles and control torques using the NN control are respectively shown in the fig.87, fig.88 and fig.89:

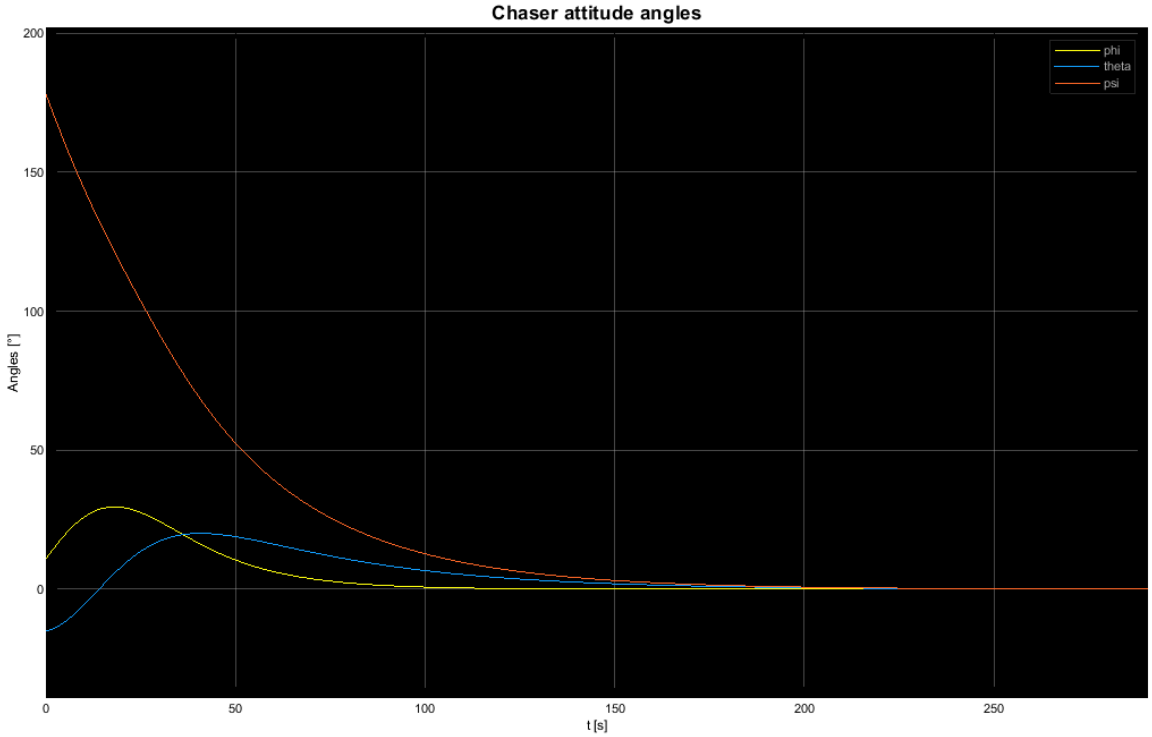

Figure 179: Chaser attitude angles, NN control.

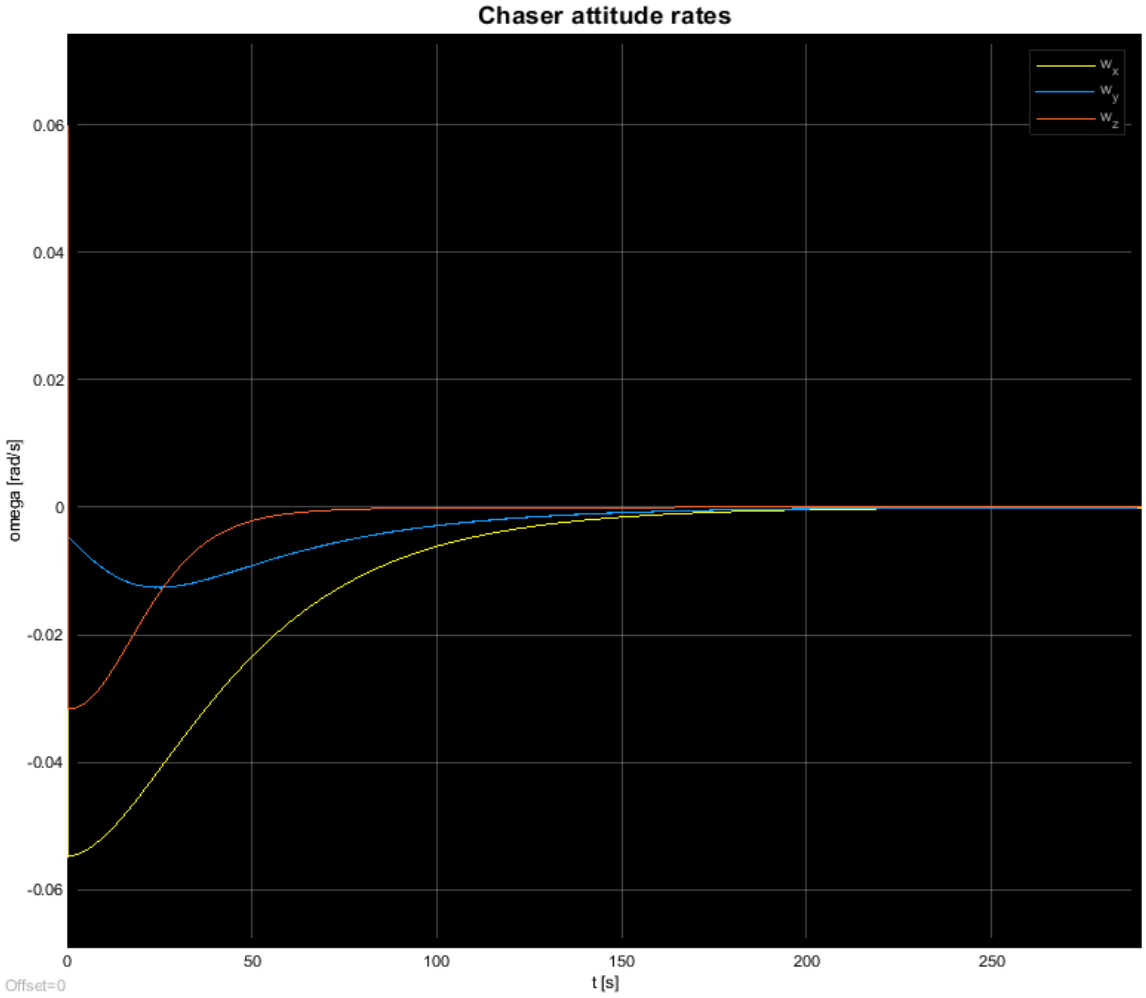

Figure 176: Chaser attitude rates, NN control.

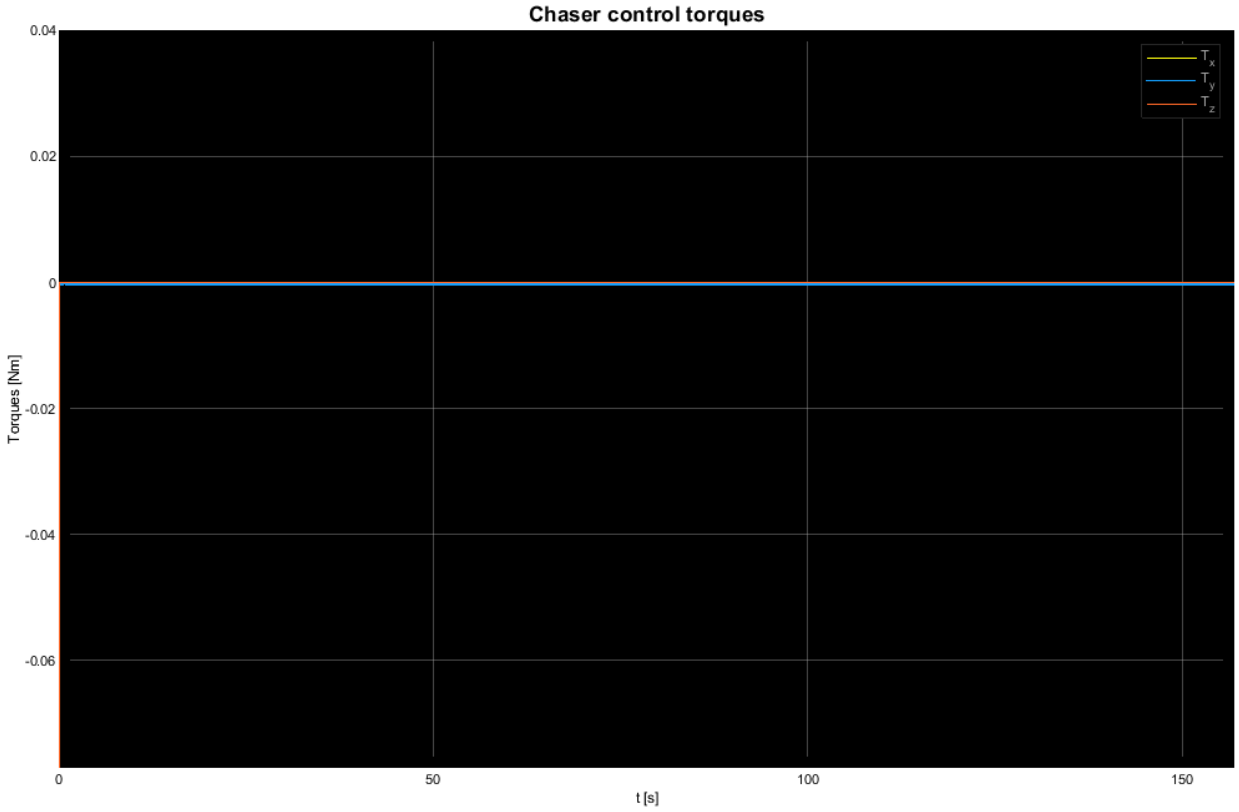

Figure 182: Chaser control torques, NN control.

It is possible to notice that the target values are obtained in a very small interval, but the time of operations is increased until 250 seconds circa. This augmentation can be seen as a positive factor because it corresponds to a slower response of the system to the external output. Consequently, since the dynamics of the body is less relevant, the inertial forces acting on the structure will be minor with respect to the other case. In this way also the stress associated to the loading condition will be lower, resulting in a decrease of the possible vibrations during the motion too.

4.4.3. Simulation 3

Finally, it has been a last test on the Artificial Neural Network. Its efficiency has been submitted to this following conditions, but with adding a sun sensor:

- $\varphi = -2°$;
- $\theta = 15°$;
- $\psi = 11°$;
- $\omega_x = 0.06$ rad/s;
- $\omega_y = 0.06$ rad/s;
- $\omega_z = 0.06$ rad/s;
- Disturbance torque: 10^{-5} Nm;
- Sensor error: 10^{-4}.

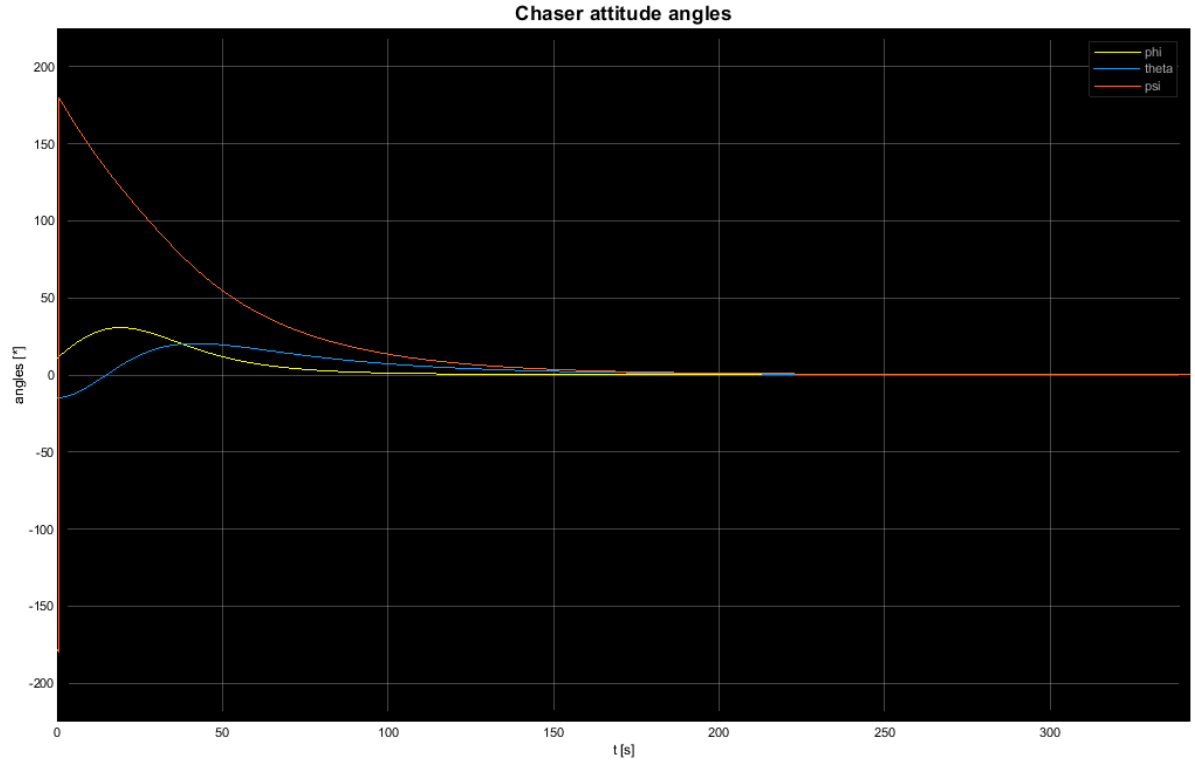

Figure 184: Chaser attitude angles, NN control.

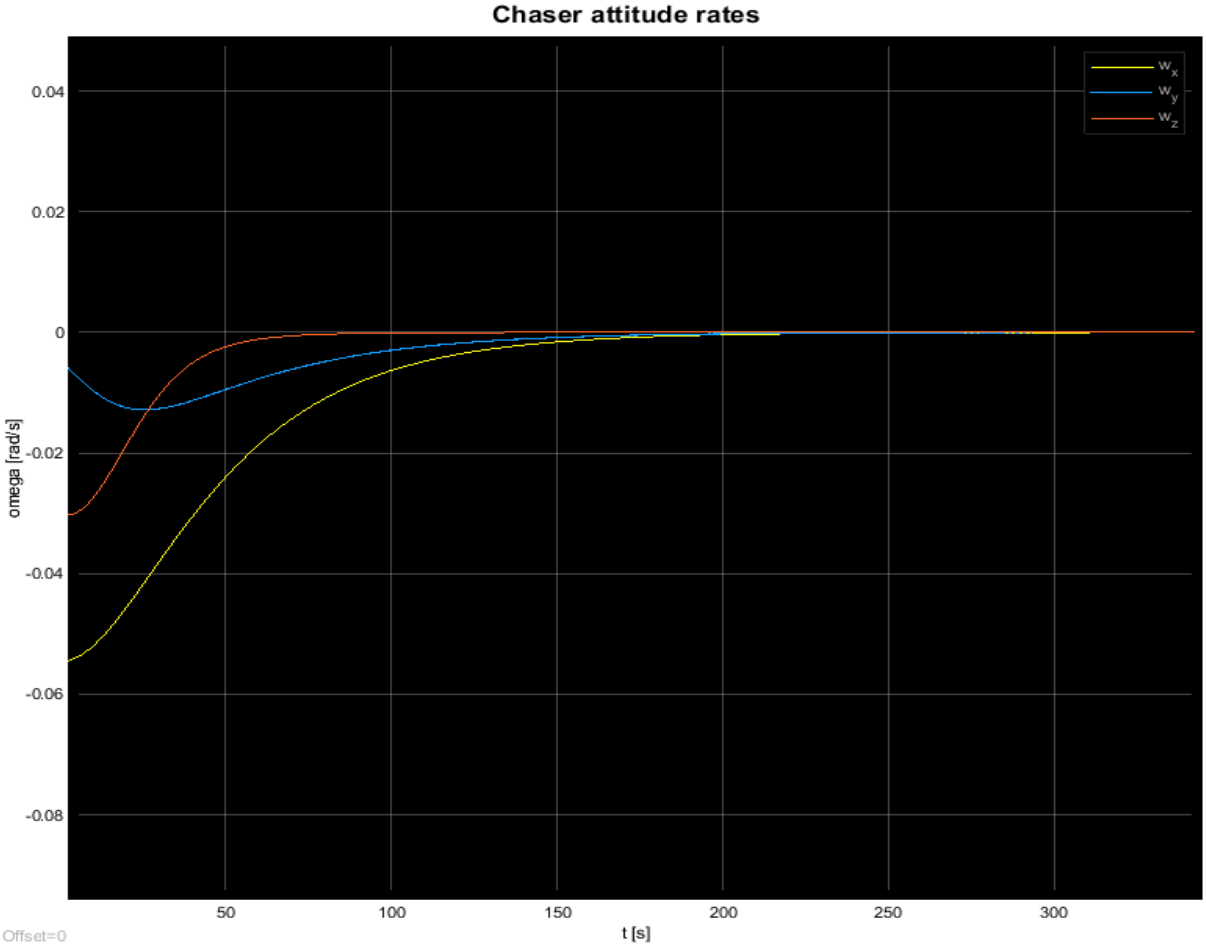

Figure 183: Chaser attitude rates, NN control.

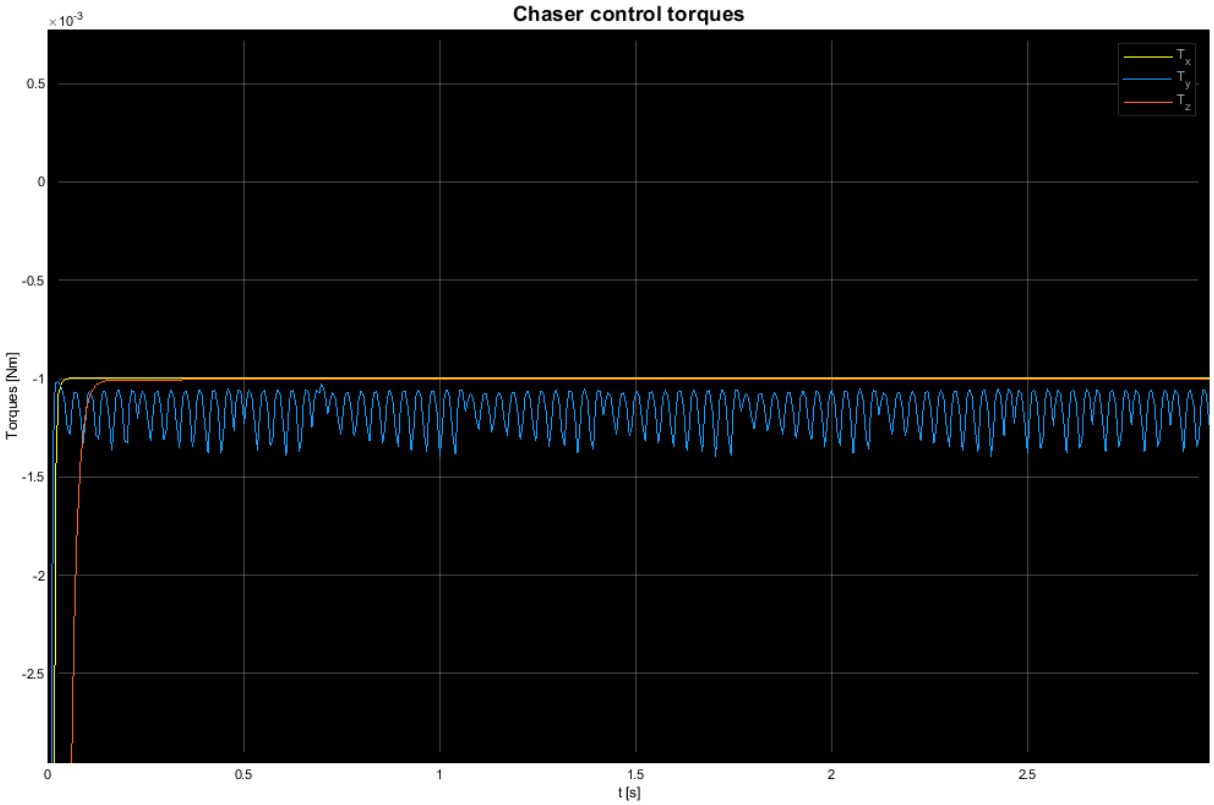

Figure 187: Chaser control torques, NN control.

The error of this sensor plays a major role in the Chaser attitude rate plot and the one of the control torques (Fig. 89 & 91), while it does not effect in a consistent way the other parameter (Fig. 90). In fact, because of the presence of the sensor, these graphics have both shown some oscillations. Luckily, these fluctuations are small, so they did not effect very significantly on the proximity operation of the satellite, allowing the spacecraft to complete the docking manoeuvre.

4.5 Discussion

In the precedent chapters, there have been presented various scenarios that the satellite may be facing. So, it has been proven that the NN trained with an LQR controller has an equivalent capacity of control the spacecraft, both for the translation and the rotational motion. From the precedent simulations, it is important to underline two aspects:

1) Despite the adding of a disturbance force and/or torque (and a sun sensor too), the network has efficiently learned how to fight them and still control the spacecraft in order to complete the docking operation. The only presence of a sun error is linked to a fluctuation of
2) For particular parameters, like the attitude angles, the NN controller has the capability of working as fast as the LQR and reach the right values in less than 30 seconds.

5. Conclusion

Like said from the beginning, the principal of this dissertation was to develop a new control strategy in order to fulfil a rendez-vous and docking manoeuvre between two 6U CubeSats, respectively a Chaser and a cooperative Target, whom operate in a LEO. The description of this phase has been entrusted to the initial chapter, where it is possible to find some useful information about the components that can be used in the design of ADCS and GN&C of a small satellite. This sorting of instruments has been useful in order to stabilize the boundaries and the limits which the Chaser could achieve.

The principal problem has been to carry the Chaser from one point in the space to a desired one, with a determined value of velocity too. In order to achieve this goal, it have been utilized two techniques. The first one based the control of the spacecraft on a LQR controller; conversely, the second one take advantage of a new technology; the Artificial Neural Networks. This ANN have been trained in a Matlab/Simulink environment with the LQR controller itselves.

Using the toolbox provided by Mathworks, it has been possible to train the network as desired, and follow the entire process, selecting each parameter at will, like the number of hidden layers that constitute the network. The input and the output data utilized to teach the nets came from the Matlab workspace. During this phase, it has been important not to overtrain the network, i.e. the NN loses the capacity to generalize the problem which lead to a poor performance, or undertrain, that it has been possible to avoid training the network a reasonable number of time. Moreover, it has been crucial to decide the right amount of hidden layers and the dimension of the training data set. For the case studied in this thesis, for the translational motion it has been necessary only one training of the network to obtain excellent results. Differently, the rotational motion required several training sets to achieve good performances.

As described in the simulation chapter, it is possible to notice that NN control are a good alternative to the most common utilized technique. In order to achieve a good results, it is important to: train the net efficiently and with large data set; avoid overtraining and/or undertraining; pick the right number of hidden layers, in order to optimize the performance. More complex is the network, better it can deal with external output, but with a higher computational time. If all this requests are fulfilled, the network can adapt to every kind of external outputs and work with and extremely precision and accuracy.

For future works, in order to increase performances, it can be useful to train the ANN with a different kind of controller and training it with a larger data set.

Bibliography

Astrom, K. J. (2002). *PID Control.*

Atiya, A. F. (1990). *An unsupervised learning technique for artificial neural networks.* Elsevier.

Bevilacqua, R., Lehmann, T., & Romano, M. (2011). Development and experimentation of LQR/APF guidance and control for autonomous proximity manuevers of multiple spacecraft. *Acta Astronautica*, 1260-1275.

Borders, W. A., Akima, H., Fukami, S., Moriya, S., Kurihara, S., Horio, Y., . . . Ohno, H. (2017). *Analogue spin–orbit torque device for artificial-neural-network-based associative memory operation.* The Japan Society of Applied Physics.

Chaturvedi, D. (2008). *Soft Computing.* Springer, Berlin, Heidelberg.

Haykin, S. (2011). *Neural networks and learning machines.* Pearson Education.

Kingma, D. P., Rezend, D. J., Mohamed, S., & Welling, M. (2015). *Semi-supervised Learning with Deep Generative Models.*

McCall, J. (2005). Genetic algorithms for modelling and optimisation. *Journal of Computational and Applied Mathematics*, 205-222.

McCamish, S. B., Romano, M., & Yun, X. (2009). Autonomous Distributed Control of Simultaneous Multiple Spacecraft Proximity Maneuvers. *IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING*, 630-643.

Modi, B., & Jethva, H. (2016). *Reinforcement Learning with Neural Networks: A Survey.* Springer International Publishing Switzerland.

Sabatini, B. L., & Regehr, W. G. (1996). *Timing of neurotransmission at fast synapses in the mammalian brain.* PubMed.

Singhala, P., Shah, D., & Bhavikkumar, P. (2014). Temperature Control using Fuzzy Logic. *International Journal of Instrumentation and Control Systems*.

Sivanandam, S., & Deepa, S. (2008). *Introduction to Genetic Algorithms.* Springer.

Stoorvogel, A. (2000). *The H∞ control problem: a state space approach.*

Turabieh, H., Mafarja, M., & Li, X. (2019). Iterated feature selection algorithms with layered recurrent neural network for software fault prediction. *Expert Systems With Applications*, 27-42.

Wojsznis, W., Mehta, A., Wojsznis, P., Thiele, D., & Blevins, T. (2007). Multi-objective optimization for model predictive control. *ISA Transaction*, 351-361.

Yadava, D., Hosangadi, R., Krishna, S., Paliwal, P., & Jain, A. (2018). *Attitude Control of a Nanosatellite system using Reinforcement Learning and Neural Networks.* IEEE.

Yang, Y. (2004). Quaternion-Based LQR Spacecraft Control Design Is a Robust Pole Assignment Design. *Journal of Aerospace Engineering*.

Sitography

Figure 34: https://www.jeremyjordan.me/content/images/2017/07/Screen-Shot-2017-07-26-at-1.44.58-PM.png

Figure 40: https://www.researchgate.net/profile/Aviroop_Dutt_Mazumder/publication/51780656/figure/fig2/AS:339488044797956@1457951665496/Unsupervised-learning-in-a-typical-artificial-neural-network-model-adapted-from-Haykin.png

Figure 1: https://ars.els-cdn.com/content/image/1-s2.0-S1000936113001787-gr1.jpg

Figure 10: https://www.cubesatshop.com/wp-content/uploads/2018/05/NCTR-M012-Magnetorquer-Rod.png

Figure 11: https://www.cubesatshop.com/product/nctr-m002-magnetorquer-rod/

Figure 2: https://www.cubesatshop.com/product/nanossoc-d60-digital-sun-sensor/

Figure 22: https://www.mathworks.com/help/robust/gs/pic8.gif

Figure 23: https://dewesoft.pro/images/uploads/PIDfront.PNG

Figure 24: https://www.tutorialspoint.com/fuzzy_logic/images/fuzzy_logic_control_architecture.jpg

Figure 25: https://res.mdpi.com/applsci/applsci-07-01272/article_deploy/html/images/applsci-07-01272-g006.png

Figure 3: https://www.cubesatshop.com/product/nss-cubesat-sun-sensor/

Figure 33: https://qph.fs.quoracdn.net/main-qimg-98dfd80df18bc6804ce919e73247562f

Figure 35: https://cdn-images-1.medium.com/max/1200/1*K6s4Li0fTl1pSX4-WPBMMA.jpeg

Figure 38: https://image.slidesharecdn.com/week3bann-150512031518-lva1-app6892/95/artificial-neural-network-31-638.jpg?cb=1431400611

Figure 39: https://www.semanticscholar.org/paper/Reinforcement-Learning-Neural-Network-to-the-of-Huang-Cao/b603452fd76e12ebe27de698e2d91e58b4da45f9

Figure 4: https://www.cubesatshop.com/product/nss-magnetometer/

Figure 41: https://image.slidesharecdn.com/kid0kzysmglshqnnyxma-signature-7f7e9932c1ac6134e9030bbdd4cfa1c5606f58d572ef5d7da58609957cc1fc67-poli-160617131027/95/neural-networks-introducton-46-638.jpg?cb=1467147466

Figure 5: https://satsearch.co/products/gomspace-nano-sense-m315-magnetometer

Figure 6: https://www.sensonor.com/media/1331/stim300.jpg?center=0.38333333333333336,0.34824902723735407&mode=crop&heightratio=0.943181818181818181818181818182&quality=60&width=600&rnd=131939282610000000

Figure 7: https://satsearch.co/products/new-space-systems-nsgy-001

Figure 8: https://www.cubesatshop.com/product/mai-400-reaction-wheel/

Figure 9: https://www.cubesatshop.com/wp-content/uploads/2016/06/CubeWheel-large.png

Figure 12: https://www.cubesatshop.com/product/nss-gps-receiver/

Figure 13: https://n-avionics.com/wp-content/uploads/2018/07/piNAV-L1-new.jpg

Figure 14: https://www.cubesatshop.com/product/chameleon-imager/

Figure 15: http://www.xcam.co.uk/sites/default/files/styles/product_image/public/C3D%20board_3.jpg?itok=ih-jeKY3

Figure 16: https://satsearch.co/products/hyperion-technologies-pm400

Figure 17: https://www.cubesatshop.com/wp-content/uploads/2017/04/IFM-Module.png

Figure 18: https://n-avionics.com/wp-content/uploads/2018/07/EPSS-2-new.jpg

Figure 19: https://satsearch.co/products/hyperion-technologies-pm200

Figure 26: https://www.researchgate.net/profile/Linus_Aloo/publication/303910207/figure/fig4/AS:371848786268163@1465667067327/Linear-Quadratic-Regulator-LQR-control-system.png

Figure 47: https://ascelibrary.org/cms/attachment/18827/522354/1.gif

Figure 48: https://physics.aps.org/assets/3a10123d-35b1-4d88-8dc9-549ad83db2b1/e91_1.png

Figure 50: https://planetary.s3.amazonaws.com/assets/images/z_changeover/diagram_orbit_ephemeris_definitions.png

Figure 51: https://i.stack.imgur.com/q9SAq.png

Figure 52: https://www.tutorialspoint.com/control_systems/images/open_loop.jpg

Figure 53: https://4.bp.blogspot.com/-HxZhCQYwn-8/VVRUXtuAH5I/AAAAAAAAemM/jPvyaiXqwdA/s1600/feedback-control-system-11-3-15.gif