Autonomous Neuro-Fuzzy Solution for Fault Detection and Attitude Control of a 3U Cubesat

IAC-15-B4.6B.8

# AUTONOMOUS NEURO-FUZZY SOLUTION FOR FAULT DETECTION AND ATTITUDE CONTROL OF A 3U CUBESAT

**Lorenzo Feruglio**
Politecnico di Torino, Italy, lorenzo.feruglio@polito.it

**Loris Franchi**
Politecnico di Torino, Italy, loris.franchi@polito.it

**Raffaele Mozzillo**
Politecnico di Torino, Italy, raffaele.mozzillo@polito.it

**Fabrizio Stesina**
Politecnico di Torino, Italy, fabrizio.stesina@polito.it

**Sabrina Corpino**
Politecnico di Torino, Italy, sabrina.corpino@polito.it

In recent years, thanks to the increase of the know-how on machine-learning techniques and the advance of the computational capabilities of on-board processing, algorithms involving artificial intelligence (i.e. neural networks and fuzzy logics) have begun to spread even in the space applications. Nowadays, thanks to these reasons, the implementation of such techniques is becoming realizable even on smaller platforms, such as CubeSats. The paper presents an algorithm for the fault detection and for the fault-tolerant attitude control of a 3U CubeSat, developed in MathWorks Matlab & Simulink environment. This algorithm involves fuzzy logic and multi-layer feed-forward offline-trained neural network. It is utilized in a simulation of a CubeSat satellite placed in LEO, considering as available attitude control actuators three magnetic torquers and one reaction wheel. In particular, fuzzy logics are used for the fault detection and isolation, while the neural network is employed for adapting the control to the perturbation introduced by the fault. The simulation is performed considering the attitude of the satellite known without measurement error. In addition, the paper presents the system, simulator and algorithm architecture, with a particular focus on the design of fuzzy logics (connection and implication operators, rules and input/output qualificators) and the neural network architecture (number of layers, neurons per layer), threshold and activation functions, offline training algorithm and its data management. With respect to the offline training, a model predictive controller has been adopted as supervisor. In conclusion the paper presents the control torques, state variables and fuzzy output evolution, in the different faulty configurations. Results show that the implementation of the fuzzy logics joined with neural networks provide good robustness, stability and adaptability of the system, allowing to satisfy specified performance requirements even in the event of some malfunctioning of a system actuator.

## I. INTRODUCTION AND BACKGROUND

### I.I 3-STAR Mission as a technology test bed

3-STAR mission is a 3U CubeSat mission that is being developed by the CubeSat Team at Politecnico di Torino. It will take part in the GEOID constellation for the validation of the GENSO network through the HumSat communication payload. In addition, 3-STAR carries a remote sensing GNSS-based payload. This payload will open the door to several applications, from Earth monitoring to civil protection warning services.

In addition to the main mission objectives, 3-STAR will be used as a validation platform for different technologies currently being developed in the team's facilities. Among these, Artificial Intelligence (AI) based Autonomous Command and Data Handling System (A-C&DH) and Attitude Determination and Control System (A-ADCS) will be included.

This paper presents a joint effort between the GNC and C&DH teams to design an algorithm for attitude control robust enough to maintain acceptable performances even in the event of a failure of one magnetic torquer (MT).

The paper will be structured as follows. The remaining part of the introduction will cover the simulation environment, and will introduce the main two technologies used in the work: fuzzy logics and neural networks. The successive chapter will focus on magnetic

torquers technology, highlighting their typical features and describing the most common failures that usually affect them. Continuing, a chapter describing the failure detection will cover into details how fuzzy logics have been used to solve this type of problem. Then, a chapter on neural networks will describe the developed neural network architecture and the training methods. Simulations and results will provide insight on the behavior of the system. Lastly, conclusions and future directions will be presented.

## I.II Simulation Setup and model description

The simulation has been completely realized in MathWorks Matlab / Simulink environment (Figure 1). The top level blocks are traditional ones: environment, dynamics and kinematics for the 3-STAR mission, a controller block where the neural networks are deployed, an actuation block where the failures for the MT are injected, and a FDIR block that stores all the fuzzy logics needed in the simulation to perform the failure detection. The FDIR output is then fed as an input for the control magnetic torque distribution.

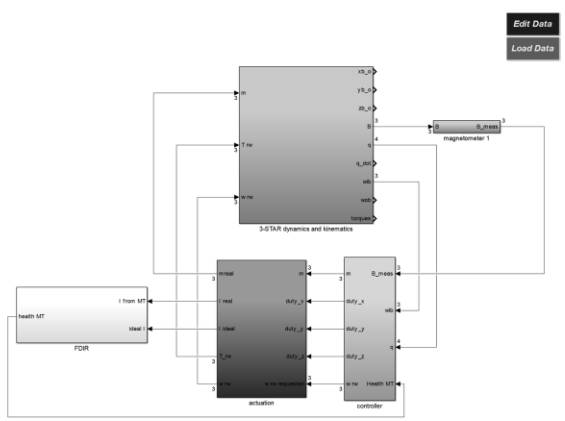


Figure 1 - Matlab / Simulink model

Main characteristics of the simulations are described. Since the paper presents a simulation not related to attitude determination algorithms, the attitude of the spacecraft is considered perfectly estimated. Magnetometer sensor, as well as on board actuators, are modeled with the most typical disturbances. Failure injection is executed by configuring a time of failure and a type, as described in the next chapter.

## I.III Simulation Models

We assume we can treat the satellite as a rigid body.

$$\dot{\omega}^b{}_{ib} = I^{-1}(-S(\omega^b_{ib})I\omega^b_{ib} + \tau^b_{ext} + \tau^b_{int}) \quad [1]$$

Where $S(k)$ is the skew-symmetric matrix and I represent the (constant in the body frame) inertia tensor, the torque $\tau_{ext} = [\tau_{ext1}\ \tau_{ext2}\ \tau_{ext3}\ ]^T$ is the total torque acting on the body and $\tau_{int} = [[\tau_{int1}\ \tau_{int2}\ \tau_{int3}\ ]^T$ is the internal torque acting on the body.

The kinematic equations of motion of how the attitude quaternions change are given by

$$\dot{\eta} = -\frac{1}{2}\epsilon^T\omega^b_{ib} \quad [2]$$

$$\dot{\epsilon} = \frac{1}{2}(\eta I + S(\epsilon))\omega^b_{ib} \quad [3]$$

Where $\eta$ is the scalar component of the quaternion and $\epsilon$ are the three vector components.

Among the included torques are also included environment disturbance such as Earth gravitational field, atmospheric drag, satellite magnetic residual and solar radiation pressure, and internal reaction wheel gyroscopic torque.

# II. MAGNETIC TORQUERS FAILURES MODELING AND RECOVERY APPROACH

## II.I Magnetic torquers failures

Magnetic torquers provide a reliable way to control attitude, and are one of the most used technology for LEO CubeSats as, unlike other actuator options (such as thrusters), are usually cheaper, low power consuming, and lightweight.

Nevertheless, as all the hardware components in general, they can be affected by failures, whether temporary or definitive. The typical problems encountered in the reliability of magnetic torquers are four, and they sensibly alter the behavior of the actuator:

- Float: output is zero
- Lock in place: output is stuck to a value different than zero
- Hard-over: output assumes a ramp characteristic, until saturation
- Loss of efficiency: the behavior remains similar to unaffected torquer, but lower efficiency causes the response to be smaller in output

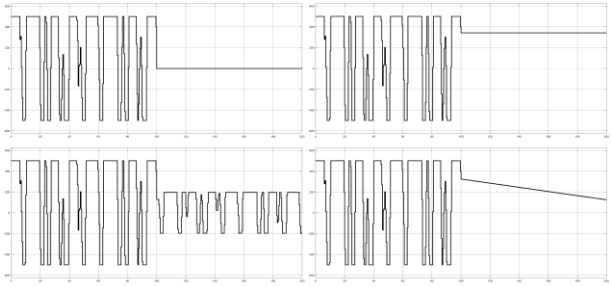

Figure 2 - Starting from top left, clockwise: float, lock in place, hard-over, loss of efficiency failures

Figure 2 illustrates examples of the most recurrent failures, where time of failure event is set at 100 seconds. One important thing to notice is that a hard-over failure

will result in a lock in place failure once the actuator output reaches saturation value.

## II.II Modeling

Magnetic torquers modeling takes into account the complete discretization of the real commands performed by the controller, as the frequency of the attitude control loop is 2 Hz. The effect of this behavior can be seen in Figure 2, where it can be noticed that the commands are not continuous-time signals. In addition, since this is a digital system, the quantization of the output for the MTs is also considered. PWM control logic is adopted: the duty cycle is commanded with a value between 0 and 999, which corresponds to a voltage from -Vmax to +Vmax.

The interaction between the earth magnetic field and the magnetic dipole moment generate the control torque, modelled as shown in formula 4:

$$\vec{T}_{control} = \vec{m} \times \vec{B}_{body} \quad [4]$$

Where $T_{control}$ the 3x1-control torque vector, *m* is the 3x1 magnetic torquer dipole moment and *B* is the 3x1 vector of the earth magnetic field expressed in body axis.

## II.III First recovery approach

In first recovery approach an optimal solution is obtained, from which the neural network will be trained.

The selected cost function for the optimization problem is involving the square of the torque error for each axis as in 5:

$$J = (T_{dx} - T_{ax})^2 + (T_{dy} - T_{ay})^2 + (T_{dz} - T_{az})^2 \quad [5]$$

Where $T_d$ is the desired torque from the control algorithm and $T_a$ is the actuated torque, which involved the actuator failure.

This cost function, from the Hessian and eigenvalue analysis applied in all fault types, results convex. This entails the good behavior for each kind of failure.

Considering an unconstrained quad prog optimization approach as finding a minimum for the multivariable function as searching the zero value of the gradient we obtain:

- Float

$$M_x = 0 \quad [6]$$

$$M_y = -\frac{Td_z B_x^2 - B_z Td_x B_x + Td_z B_y^2 - B_z Td_y B_y}{B_x(B_x^2 + B_y^2 + B_z^2)} \quad [7]$$

$$M_z = \frac{Td_y B_x^2 - B_y Td_x B_x + Td_y B_z^2 - B_y Td_z B_z}{B_x(B_x^2 + B_y^2 + B_z^2)} \quad [8]$$

- Lock in place

$$M_x = \overline{m_x} \quad [9]$$

$$M_y = -\frac{M_x B_x^2 B_y - Td_z B_x^2 + Td_x B_x B_z + M_x B_y^3 - Td_z B_y^2 + M_x B_y B_z^2 + Td_y B_y B_z}{B_x(B_x^2 + B_y^2 + B_z^2)} \quad [10]$$

$$M_z = \frac{Td_y B_x^2 - B_y Td_x B_x + Td_y B_z^2 - B_y Td_z B_z}{B_x(B_x^2 + B_y^2 + B_z^2)} \quad [11]$$

- Hard over

$$M_x = m_{sat} \quad [12]$$

$$M_y = -\frac{M_x B_x^2 B_y - Td_z B_x^2 + Td_x B_x B_z + M_x B_y^3 - Td_z B_y^2 + M_x B_y B_z^2 + Td_y B_y B_z}{B_x(B_x^2 + B_y^2 + B_z^2)} \quad [13]$$

$$M_z = \frac{Td_y B_x^2 - B_y Td_x B_x + Td_y B_z^2 - B_y Td_z B_z}{B_x(B_x^2 + B_y^2 + B_z^2)} \quad [14]$$

- Loss of efficiency

$$M_x = km_x \quad [15]$$

$$M_y = \frac{M_x B_x^2 B_y - Td_z B_x^2 + Td_x B_x B_z + M_x B_y^3 - Td_z B_y^2 + M_x B_y B_z^2 + Td_y B_y B_z}{B_x(B_x^2 + B_y^2 + B_z^2)} \quad [16]$$

$$M_z = \frac{(M_x B_x^2 B_z + Td_y B_x^2 - Td_x B_x B_y + M_x B_y^2 B_z - Td_z B_y B_z + M_x B_z^3 + Td_y B_z^2)}{B_x(B_x^2 + B_y^2 + B_z^2)} \quad [17]$$

# III. FAILURE DETECTION WITH FUZZY LOGICS

## III.I Understanding the problem

In order to define the algorithm, insights on the different types of failures have been examined, and for each one of them the most evident characteristics were defined.

- Float failure: output current and current rate of change are both zero.
- Lock in place failure: output current is *not* zero, but rate of change is.
- Hard-over failure: current rate of change is constant, therefore its derivative is zero.
- Loss of Efficiency failure: considering an estimated loss of efficiency parameter, the difference between the real current output and the ideal one multiplied by the estimated changed efficiency is zero. Considerations could have been made regarding the fact that both current curves have the same concavity, but it turned out to be not needed.

These considerations led directly to the definition of the algorithm, as the main input variables were defined and the insight on the needed rules was obtained.

## III.II Fuzzy logics configuration

### Input variables

The failure detection algorithm for the MTs has been designed using the Fuzzy Logics theory, and implemented using the Fuzzy Logic Designer toolbox in MathWorks Matlab environment (Figure 3).

As stated above, five different input variables were considered: current flow on the MTs, current rate of change, error between ideal current and effective one,

derivative of the current rate of change and finally the estimated loss of efficiency.

Input variables have been defined using three typical membership functions: triangular-shaped membership function (*trimf*), Z-shaped membership function (*zmf*) and S-shaped membership function (*smf*), appropriately tuned to the expected input behaviors.

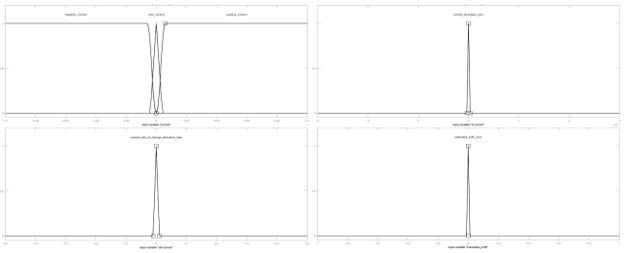

Figure 3 - Fuzzy logics input variables. Starting from top left, clockwise order: current, current derivative, difference between real MT output and ideal one multiplied by the estimated loss of efficiency, current rate of change derivative. Last input variable, the error, has been not included, but has the same graph as current derivative one.

### Output variables

The output variable has been designed to reflect directly the output of the decision rules of the fuzzy logic. In this case, five different and distinct output were needed, so the output has been designed by using five triangular membership functions, each one representing a type of failure (Figure 4).

In this way, the designed output variable provides directly an integer value for the detected failure of the MTs.

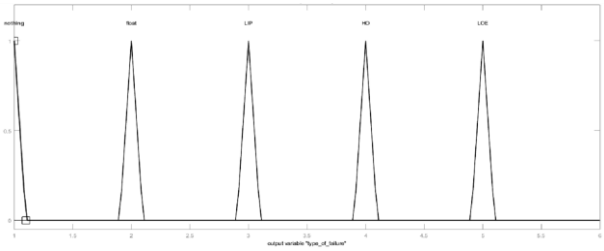


Figure 4 - Output variable: types of failure. Starting from the left, no failure (1), float (2), lock in place (3), hard-over (4), loss of efficiency (5) failures. Number in the parenthesis are the corresponding integer output values.

### Inference system and detection example

The detection of the MT failures has been done by using a five rule mamdani inference system (Figure 5).

Common traits of the designed rules are that for every type of failure the error is different than zero. The rule definition follows explicitly what has been defined above.

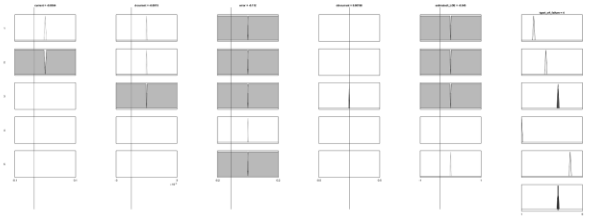

Figure 5 - Rule viewer for a case of hard-over failure. Current is non zero, and so is its derivative. Error is different than zero. Current rate of change derivative is zero. Therefore, the result of this set of inputs is a hard-over failure, as expected.

## IV. ANN AND APPLICATION IN CONTROL SYSTEM

Neural networks can be used in control systems with different approaches, such as inverse dynamics or supervised control, in function of the intrinsic system characteristics.

Principally we consider five types of neural controllers: supervised control, direct inverse control, internal model control, model reference control and unsupervised with off-line training or on-line training.

Due to the fact that the system is a nonlinear MIMO (multi-input multi-output) with open-loop instability and taking into account the computational cost we decided to use a supervised controller with off-line training.

In fact, is possible to teach to a neural network the correct actions by using an existing controller or human feedback.

Most traditional controllers are based around an operating point. These controllers will fail if there is any sort of uncertainty or change in the unknown plant. The advantages of neuro-control are that, if an uncertainty in the plant occurs, the ANN (Artificial neural network) will be able to adapt its parameters and maintain the plant controlled. In supervised control, a teacher provides correct actions for the neural network to learn.

In offline training the targets are provided by an existing controller data, the neural network adjusts its weights until the output from the ANN is similar to the controller one.

### IV.I ANN Training algorithm

The training process normally minimizes the output error through the application of an optimization method. These methods need to know, to some extent, how the net output varies with respect to a given neuron weight (Figure 6).

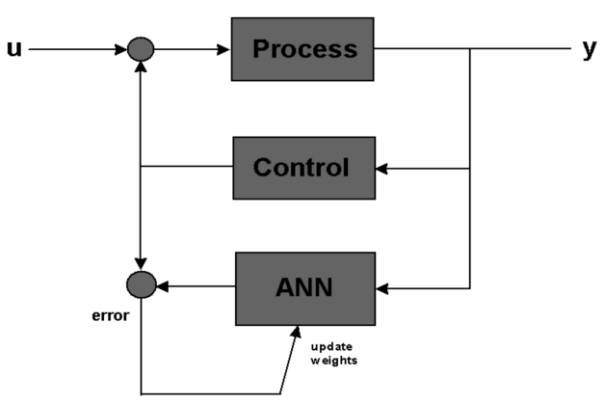


Figure 6 - Supervised Learning NN control

The problem of neural network learning can be seen as a function optimization problem, where we are trying to determine the best network parameters (weights and biases) in order to minimize network error.

This said, several function optimization techniques from numerical linear algebra can be directly applied to network learning, one of these techniques is Levenberg-Marquardt algorithm.

The Levenberg-Marquardt algorithm is a very simple, but robust, method for approximating a function.

It consists in solving the equation 17:

$$(J^T J + \mu I)\sigma = J^T e \qquad [18]$$

Where **J** is the Jacobian matrix for the system, $\mu$ is the Levenberg's damping factor, $\sigma$ is the weight update vector that we want to find and **e** is the error vector containing the output errors for each input vector used on training the network. $\sigma$ models how much we should change our network weights to achieve a better solution. The $J^T J$ matrix can also be known as the approximated Hessian.

In term of state sequential steps, the formulation becomes:

$$x_{k+1} = x_k - [J^T J + \mu I] J^T e \qquad [19]$$

The $\mu$ damping factor is adjusted at each iteration, and guides the optimization process. When the scalar μ is zero, this is just Newton's method, using the approximate Hessian matrix. When μ is large, this becomes gradient descent with a small step size. Newton's method is faster and more accurate near an error minimum.

Thus, μ is decreased after each successful step (reduction in performance function) and is increased only when a tentative step would increase the performance function.

## IV.II ANN architecture

For modelling the network it is necessary to perform a trade-off on the ANN architecture, in terms of:

- Type of network (MLP, RBF, HONN, ADALINE, NARX)
- Number of layers
- Number of neurons for layer
- Threshold function

Taking into account the computational cost (least number of neurons possible), training performance evolution and correct interpolation of the control behaviour, we obtain the follow ANN structure (shown in Figure 7), with:

- 3 output signals
- 4 hidden layers (3 with log-sigmoid function and 1 with linear function emulating a linear filter) with mirror neuron configuration, respectively from the first to the last, 15, 30, 30, 15 neurons
- 1 linear output layer

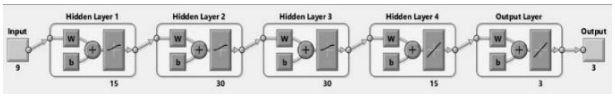


Figure 7 - Neural network architecture

## IV.III Network Training

The training, after weights and bias initialization, was performed off-line by collecting several data, such as attitude state $\{q_{2,4}, \dot{q}_{2,4}\}^T$ evolution, reference quaternion $q_{ref}$, control torque $\{m_x, m_y, m_z\}^T$, and failure detection value from a previous MPC (Model Predictive Control) controlled and recovered plant.

During the ANN configuration and training the data collection and organization is very important and results in a delicate phase.

The collected data was then used for the off-line training (70%), validation (15%) and test (15%) with Levenberg's back propagation method and considering Mean Square Error as performance index, obtaining the training performance summarized in Figure 8**Error! Reference source not found.**.

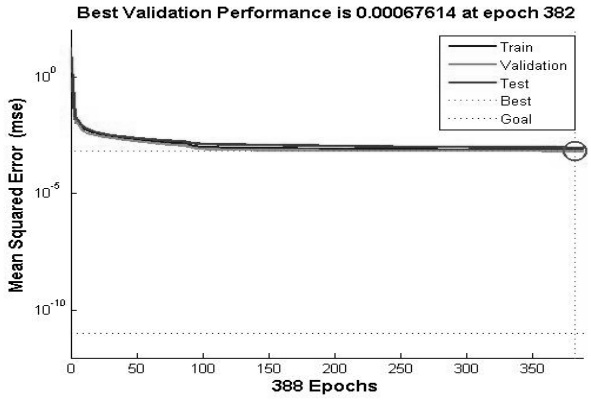


Figure 8 - Neural Training Performance

The mean square error between desired output and NN output, reached the value of 0.011742 after 291 epochs involving test and validation phases.

To avoid over-training, after some test cases the number of training epochs chosen was 300.

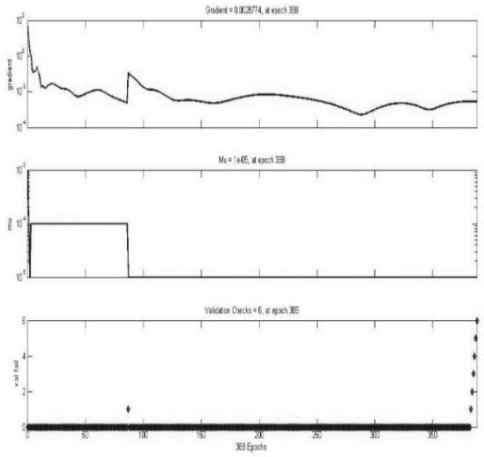

Figure 9 - Neural training state

In Figure 9 are reported the training states as gradient ( $g = J^T e$ ), $\mu$ parameter and validation.

A small value of the scalar $\mu$ verifies an optimal training.

Furthermore, the fact that the gradient reaches a small value entails the reaching of minimum error with respect to the variation of artificial neural network weights. In fact, in formula:

$$\Delta W_i = \eta \frac{\delta e}{\delta w_{io}} \qquad [19]$$

where W is the weight for each neuron and $\eta$ is the learning rate .Value of learning rate between 0.1 and 0.01 it is recommended to avoid slow learning or learning oscillations.

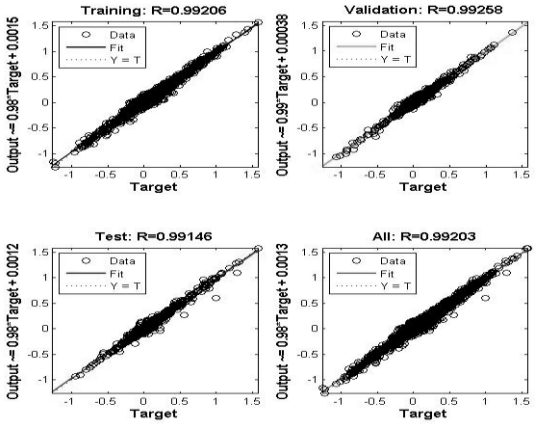


Figure 10 – Training Regression Vector

At last, Figure 10 is representing the evolution of the regression vector, which shows the relationship between the outputs of the network and the targets. If the training was perfect, the network outputs and the targets would be exactly equal, but the relationship is rarely perfect in practice.

If R = 1, this indicates that there is an exact linear relationship between outputs and targets. If R is close to zero, then there is no linear relationship between outputs and targets.

In our analysis the regression value results greater than 0.9, therefore verifying a good fit of the output data set with respect to target data. At last, it is important to note that considering a different manoeuvre the off-line neural network will need modified training data (state, control couple) to guarantee the correct control operations.

## V. SIMULATIONS

### V.I Simulation Parameters

The developed controller and autonomous FDIR solution is tested using the 3-STAR mass proprieties and orbital parameters, involving only the stabilization phase as follows:

| ***Mass properties*** | | | | |
|---|---|---|---|---|
| *Property* | *Value* | | | *Unit* |
| Mass | 3 | | | *kg* |
| Inertia (Principal axes) | x<br>0.025 | y<br>0.025 | z<br>0.005 | $Kgm^2$ |
| Initial condition | Pitch<br>10 | Roll<br>0 | Yaw<br>10 | *deg* |
| Reference attitude | Pitch<br>0 | Roll<br>0 | Yaw<br>0 | *deg* |
| ***Orbital Parameters*** | | | | |
| RaanG | 0 | | | *deg* |
| Inclination | 98 | | | *deg* |
| Eccentricity | 0 | | | - |
| Perigee argument | 0 | | | *deg* |
| Mean anomaly | 0 | | | *deg* |
| Altitude | 800 | | | *km* |

*Table 1 - Simulation Parameters*

### V.II Simulations Results

In this section the simulation results are presented.

For each type of failure the detection value for the fuzzy network, the attitude of the spacecraft, the magnetic torquer dipole, the failed torquer dipole and the

control torque applied to the spacecraft are plotted. For a better comparison in term of magnetic dipole distribution, the nominal operational case is sub plotted for each simulated fault.

## Failure detection

Failure detection algorithm has been tested during different stages of the development. The bare Fuzzy Logic Algorithm showed good performance in detecting the following failure modes: float, lock in place and hard-over. During the testing of the loss of efficiency failure, several spurious peaks were detected, and this was related to the fact that, when the commanded value and the real actuator one were the same (such as when the actuator is commanded to be off), the fuzzy didn't properly detect the failure. Nonetheless, these peaks were instantaneous. This allowed the introduction of simple fixed average filtering algorithms to come up with a more stable detection of the failures, even during the loss of efficiency one (Figure 11).

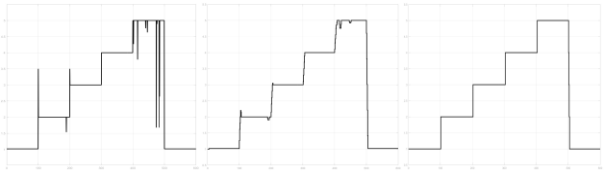

Figure 11 - Failure detection output for different types of failure, original detection logic output at the left. Errors in the detection are caused by instantaneous equivalence between the failed output state and the commanded one. As these errors are usually one sample long, they can be easily filtered out.

## Nominal operation

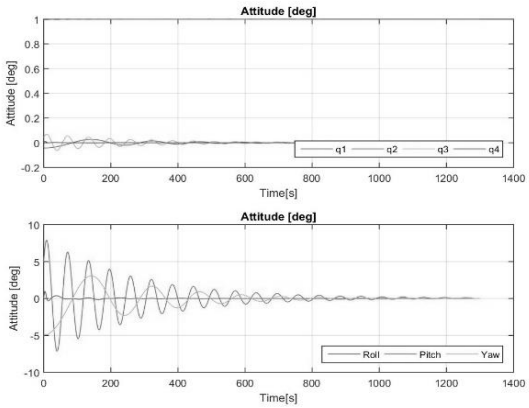


Figure 12 - Attitude evolution: Nominal case

The first case with absence of magnetic torque failure is shown from Figure 12 to Figure 14, which show the parameters evolution in case of torquer nominal condition. The neural attitude controller in condition of nominal operation is able to stabilize and tracking the attitude in presence of external disturbance. Is possible to notice the natural attitude oscillation due to the damping nature of the magnet torquer, which never goes to saturation values.

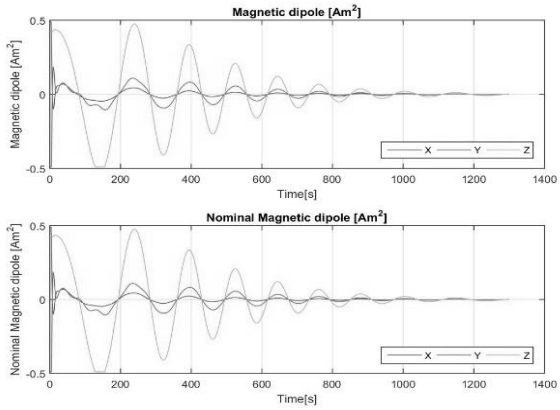


Figure 13 - Magnetic dipole evolution: Nominal case

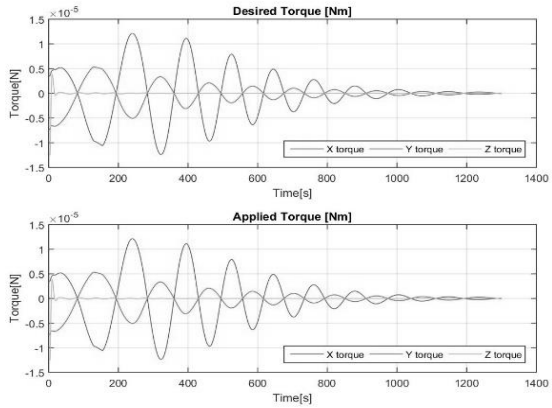


Figure 14 - Torque evolution: Nominal case

## Float Failure

The Float simulation results are shown from Figure 15 to Figure 18, which show the parameters evolution in case of fault failure on the torque along X body axis at 300 seconds from the simulation start.

When the failure occurs, the recovery controller is able to distribute the desired magnetic dipole on the y and z axis granting the desired control torque despite the zero value of the x torquer magnetic dipole and leading to the desired attitude stability and tracking behavior.

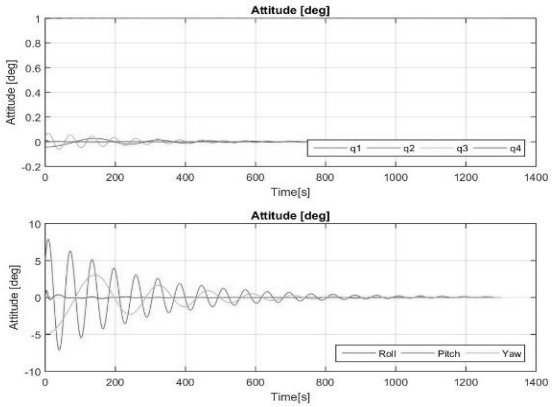


Figure 15 - Attitude evolution: Float case

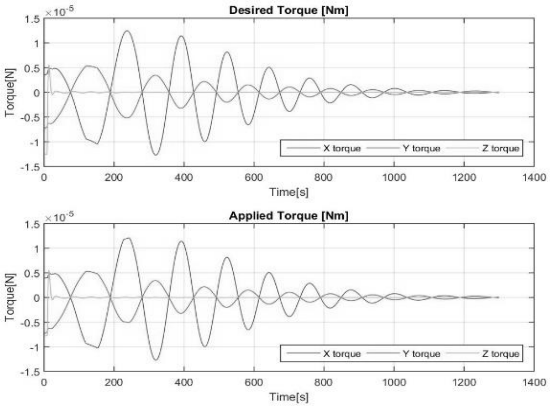


Figure 18 - Torque evolution: Float vs Nominal case

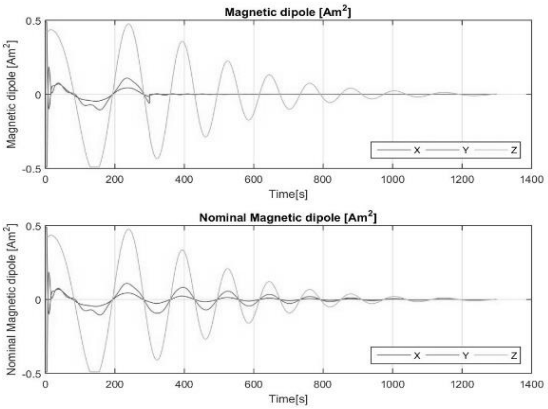


Figure 16 - Magnetic dipole: Float vs Nominal case

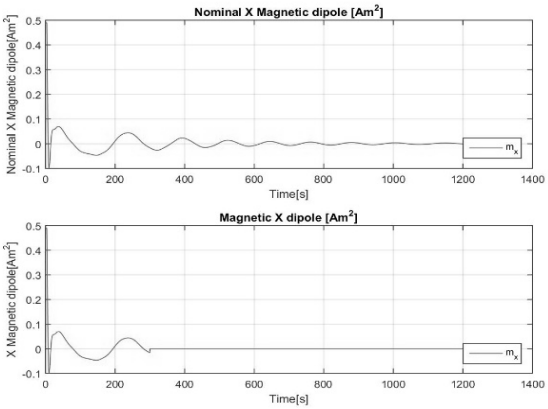


Figure 17 - X dipole evolution: Float vs Nominal case

## Hard Over

The Hard Over simulation results are shown from Figure 19 to Figure 22, which shown the parameters evolution in case of Hard over failure with saturation limit of $m_{sat} = 0.5\ Am^2$ along the x body axis on the torquer along $x$ body axis at 300 seconds from the simulation starts. In this case, the neural controller was not able to recover the attitude due to the high and nonlinear behavior involved in the saturation torque of the x torquer.

When the failure accurse the system become instable. In case of Hard Over detection, during the linear behavior, is possible to turn off the failed torquer switching from hard over to fault failure mode and guaranteeing the attitude control.

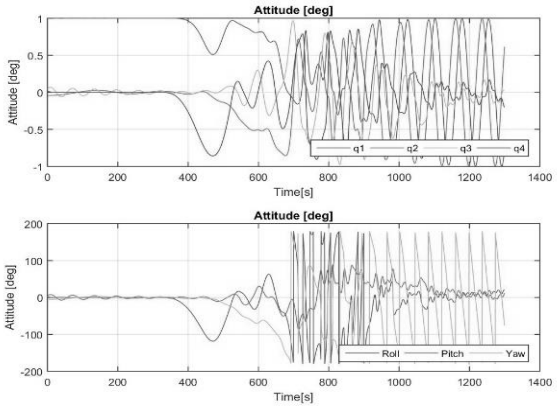


Figure 19 - Attitude evolution: HO vs Nominal case

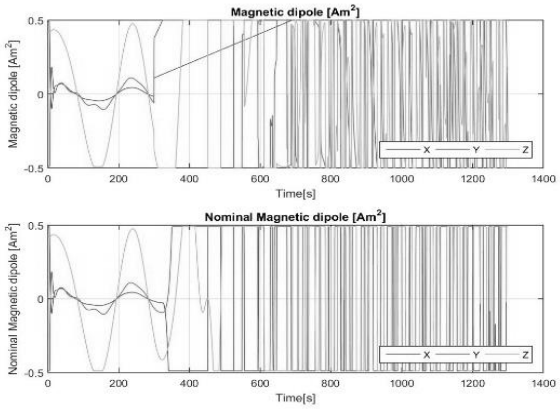


Figure 20 - Magnetic dipole: HO vs Nominal case

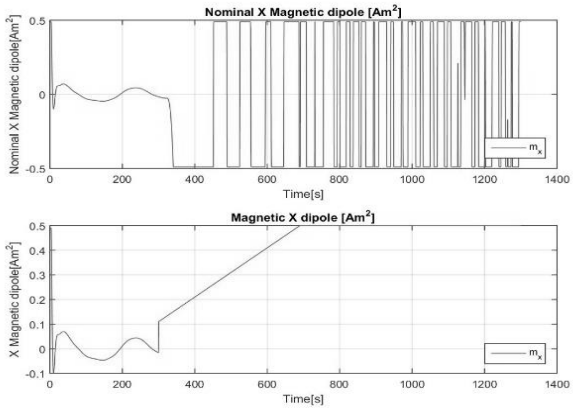


Figure 21 - X dipole evolution: HO vs Nominal case

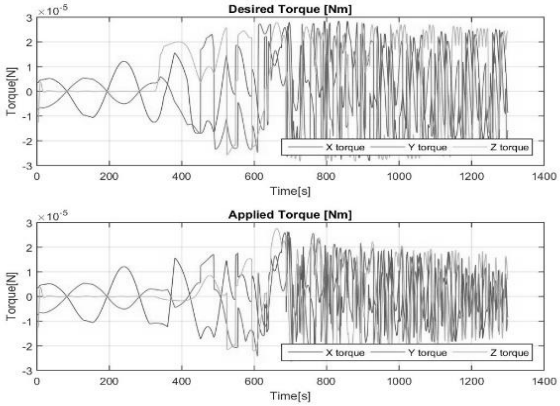


Figure 22 - Torque evolution: HO vs Nominal case

## Loss of efficiency

The Loss of efficiency simulation results are shown from Figure 23 to Figure 26, which show the parameters evolution in case of Loss of efficiency torque failure (involving constant gain of $k = 0.003$) applied to x axis torquer .

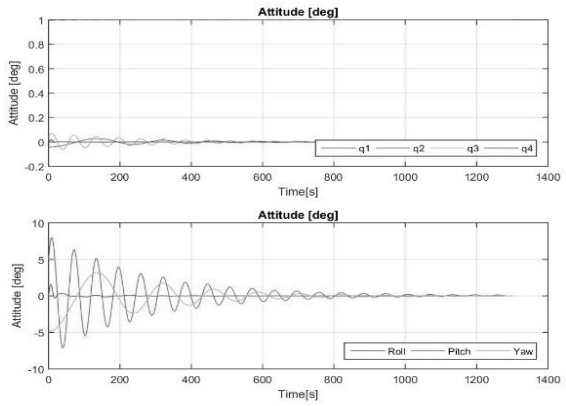


Figure 23 - Attitude evolution: LOE vs Nominal case

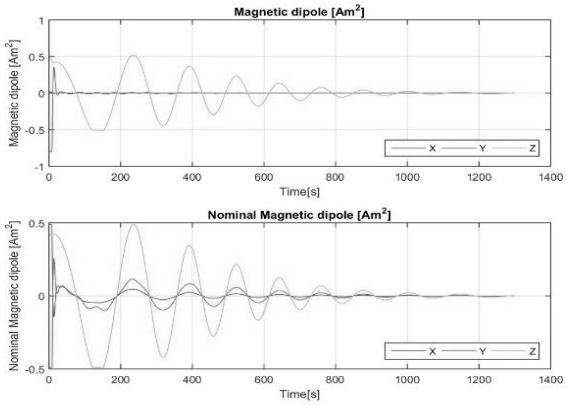


Figure 24 - Magnetic dipole: LOE vs Nominal case

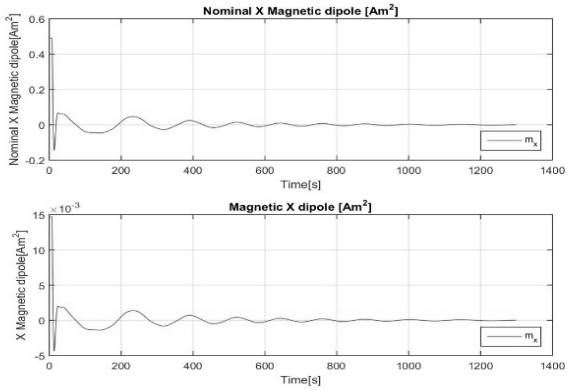


Figure 25 - X dipole evolution: LOE vs Nominal case

Due to the low value of the applied torque, and to a better understanding of the control behavior the failure recovery begins when the simulation starts. When the failure occurs, the recovery controller is able to distribute the desired magnetic dipole given by the y and z torquers guaranteeing the desired control torque despite the loss of efficiency of the x torquer magnetic dipole.

This leads to a still operational attitude controller with good tracking and stability performances.

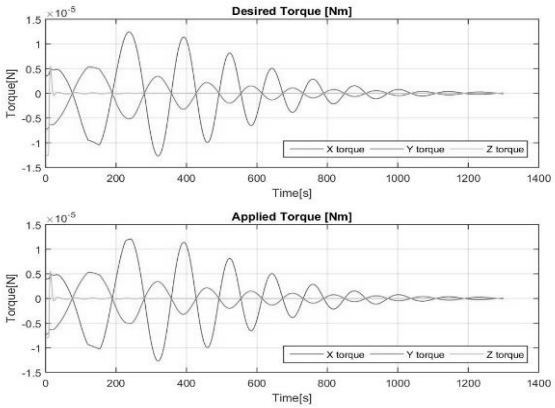


Figure 26 - Torque evolution: LOE vs Nominal case

## Lock in Place

The Lock in Place simulation results are shown from Figure 27 to Figure 30, which show the parameters evolution in case of a Lock in Place failure involving a magnetic dipole of $m_{LIP} = 0.1\ Am^2$ on the torquer along $x$ body axis at 300 seconds from the simulation start.

When the failure occurs, the system results able to recover the attitude and generate the desired control torque despite the constant activation of the x-torquer.

Is important to notice that the Lock in Place value have a threshold value over which the attitude control is not guaranteed. In fact high values of Lock in Place tend to the hard-over behavior leading to instability of the system.

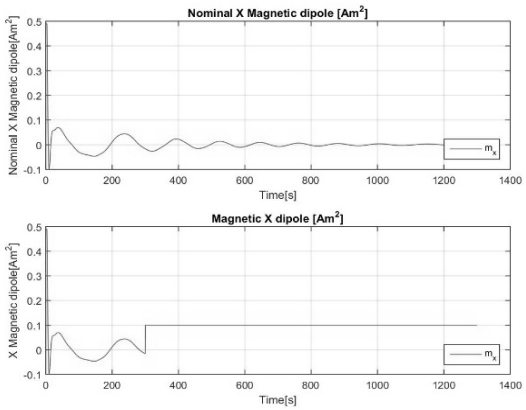


Figure 27 - X dipole evolution: LIP vs Nominal case

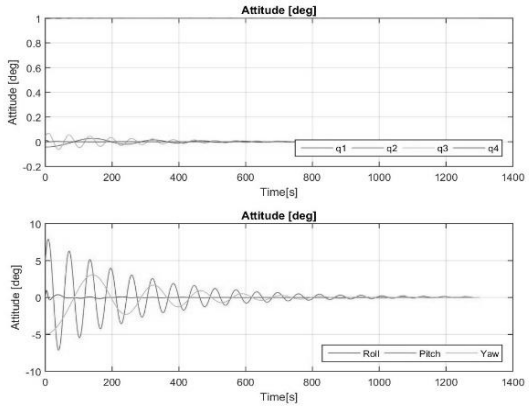


Figure 28 - Attitude evolution: LIP vs Nominal case

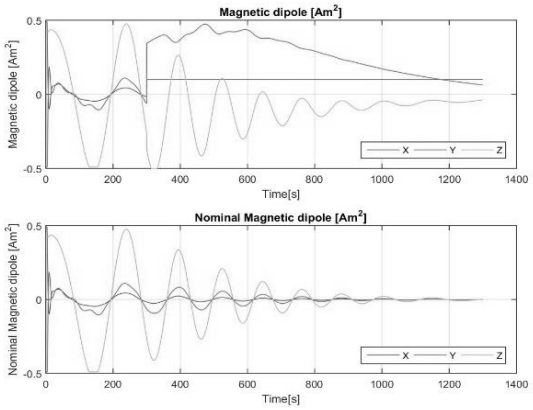


Figure 29 - Magnetic dipole: LIP vs Nominal case

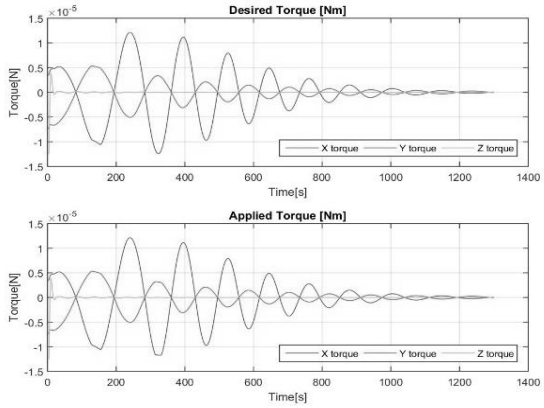


Figure 30 - Torque evolution: LIP vs Nominal case

## VI. CONCLUSIONS

An autonomous neuro-fuzzy solution for ADCS fault detection and recovery has been developed. The simulated system in Matlab and Simulink environment was the 3U CubeSat 3-STAR, with an attitude control

configuration of 3 magnetic torquers and one reaction wheel.

State of art failure injection on magnetic torquers has been implemented, and typical failure modes such as floats, lock in place, hard-over and loss of efficiency were considered. A detection algorithm based on fuzzy logics has been developed, while a neural network architecture was employed as adaptive attitude control.

The fuzzy logic setup was not complex, as only five rules and five input variables were enough to fully capture the behaviour of the faulty actuator. By using a look-up table, the implementation of this algorithm on resource-constrained on-board processors is feasible.

The fuzzy network left errors in the detection caused by instantaneous equivalence between the failed output state and the commanded one. As these errors are usually one sample long a fixed mean filter can easily filter them, aiding the neural network of the handling of the failure type and the definition of the recovery solution without signal oscillation.

The neural network controller, due to an initial trade-off between structure-training methods vs computational cost, was chosen as feed forward off line trained with back propagation algorithm. This entails that, aiming to obtain a single adaptive and recovery controller, the neural network was trained with a pre-existing controller and recovery data.

Nonetheless, a definitive network able to redistribute the desired magnetic dipole in function of the fuzzy network output was developed.

The proposed solution maintains desired performances of the attitude control recovery except for the case of the hard-over and near saturation lock in place failures.

In these cases, the controller was not able to guarantee the pointing performance due to the higher constant torque applied in the two orthogonal axis by the failed magnetorquer. In case of hard-over failure, if the magnetorquer can be turned off, the failure will be converted to a float one and this will ensure attitude recovery and the mission safety. Aiming to obtain a general fault model, given the ACS configuration, future research will provide the study of single reaction wheel and reaction wheel plus single torquer failures.

Once the computational cost of the implemented algorithm will be evaluated, and given the increasing capabilities of the COTS processors used in CubeSats, the generalization of the fuzzy logic detection algorithm or the use other types of off-line neural network based controllers (such as model reference adaptive control), and on line adaptive solution (such as MADALINE or RBF networks) will be investigated.

## VII. REFERENCES


[1] James R. Wertz and Wiley J. Larson: “Spacecraft attitude determination and control”, First Edition, Space Technology Series,
Space Technology Library, Springer, 1978

[2] James R. Wertz and Wiley J. Larson: “Space Mission Analysis and Design”, Third Edition,
Space Technology Series, Space Technology Library, Microcosm Inc, Kluwer Academic Publishers, 2005

[3] F. Landis Markley, John L. Crassidis:
“Fundamentals of Spacecraft Attitude Determination and Control”, First Edition, Space Technology Series, Space Technology Library, Springer-Verlag London, 2013

[4] Ali Zolghadri David Henry Jérôme Cieslak Denis Efimov Philippe Goupilailure: “Fault Diagnosis and Fault-Tolerant Control and Guidance for Aerospace Vehicles”, Advances in Industrial Control, Springer, 2013

[5] Bai, Y., & Wang, D.: “Fundamentals of Fuzzy Logic Control – Fuzzy Sets, Fuzzy Rules and Defuzzifications”, 1982.

[6] Li, H., & Gatland, H.: “A new methodology for designing a fuzzy logic controller”. Systems, Man and Cybernetics, IEEE, 25(3), 505–512, 1995.

[7] Nathalia, J., & Bemal, P: “A Model-based Fault Recovery for the Attitude Control Subsystem of a Satellite using Magnetic Torquers”, Concordia University, 2008.

[8] Singiresu S. Rao: “Engineering Optimization-Theory and Practice”, Fourth Edition, John Wiley & Sons, Inc., 2009

[9] Dr. Abebe Geletu: “Solving Optimization Problems using the Matlab Optimization Toolbox - a Tutorial”, 2007

[10] Shigeo Abe: “Neural networks and fuzzy systems Theory and Applications”, First Edition, Springer Science and Business Media LLC, 1997

[11] Mark Hudson Beale Martin T. Hagan, Howard B. Demuth: “Neural Network Toolbox: User’s Guide”,
The MathWorks, Inc., 2014