Design of an Unsupervised Machine Learning Approach to Fault Detection for CubeSat AOCS

Applied to LUMIO: Lunar Meteoroid Impact Observer

Master Thesis

Kasper De Smaele

Delft University of Technology

TU Delft

Design of an Unsupervised Machine Learning Approach to Fault Detection for CubeSat AOCS

Applied to LUMIO: Lunar Meteoroid Impact Observer

by

Kasper De Smaele

Student Number 4482433

to obtain the degree of Master of Science in Aerospace Engineering

at the Delft University of Technology,

to be defended publicly on Friday May 12, 2023 at 10:00 AM.

Supervisor:	Dr. Angelo Cervone
Duration:	October, 2022 - April, 2023
Faculty:	Aerospace Engineering, Delft
Department:	Space Systems Engineering

Thesis committee:	Dr. A. Cervone,	TU Delft, Astrodynamics & Space Missions
	Dr. S.M. Cazaux	TU Delft, Astrodynamics & Space Missions
	Dr. S. Speretta	TU Delft, Space Systems Engineering

Cover: Rendering of LUMIO Spacecraft observing the Lunar surface - courtesy of ESA (Modified)

TU Delft

Preface

This thesis is the final part of my now complete education as an Aerospace Engineer at TU Delft. After graduating from the BSc program in 2018, I set out to explore what it was like working as an engineer in the aerospace industry as well as in a global corporate environment. However, the urge to come back and complete the master program kept growing. After the past two challenging years with a steep academic and personal learning curve, an internship, a thesis, and so many interesting people and projects I can honestly say I do not regret the decision at all.

This thesis project is the result of 9 months of reviewing literature, speaking to experts, trying, failing, and trying again. I want to thank my supervisor, Dr. Angelo Cervone, for the opportunity, the guidance, and the belief during this project. From Deimos, I would also like to thank Giovanni Bay and Paulo Rosa for their support and insights on LUMIO provided in the early stages of this project.

I have endured a lot of hardship and adversity during the past two years, and would like to thank those who stood by me, believed in me, and encouraged me. I want to say thank you to all the people who supported me in one way or another in the past two years.

Specifically, to my Dad, who continues to believe in me and always encourages me to follow my dreams and have fun in life, a big thank you. The lessons you provide are invaluable. To Bart, Els, and Billie, thank you for the encouragement and always being willing and happy to help, and for treating me as one of your own. To my grandmother, Francine, thank you for continuing to push me to finish my education, and for providing the support when needed. ”*Aim high, the arrow drops as it flies*” is a saying that is engraved in my mind and guides me. I also want to thank Pieter, without whom this whole adventure would have been significantly harder. Last but absolutely not least, to Emma, I don’t know how I would have gotten to where I am today without you. As far as words can describe gratitude, thank you for literally being there all the time, laughing at my (sometimes low quality) jokes, for understanding the stress and late nights that come with a thesis, and for making the future look incredibly bright.

Kasper De Smaele
Delft, May 2023

Abstract

CubeSats suffer from low reliability and have little to no Fault Detection, Isolation and Recovery mechanisms onboard. Advanced CubeSat missions such as the Lunar Meteroid Impact Observer (LUMIO), will use more complex attitude determination and control systems, increasing the need for advanced fault detection. Traditionally this requires model-based fault detection methods which are complicated, computation heavy, and highly sensitive to disturbances. Machine learning has proven proficient at fault detection in several non-space related applications, but training data including spacecraft faults is not available. In this research an especially lightweight unsupervised learning method for fault detection is designed for the LUMIO attitude determination and control components. The result is a system capable of detecting artificially induced bias, calibration error, and drifting measurement faults on the scale of 0.1 mrad/s in the IMU, with no false alarms being raised. The method was tested on simulated LUMIO telemetry from the IMU and reaction wheels as well as on real spacecraft telemetry from the OPS-SAT sun sensor, star tracker, reaction wheels and IMU. In both cases, excellent fault detection and false alarm performance was observed indicating the potential of this method for application in CubeSat AOCS fault detection and isolation.

Executive Summary

This thesis describes the design of a novel model-based fault detection method, focussed on the AOCS systems of CubeSats. Specifically, it is applied to deep space CubeSats as these missions exhibit more complex architectures and a much lower risk tolerance compared to traditional CubeSats. In this thesis, the Lunar Meteroid Impact Observer (LUMIO) mission is used as a case study. Based on a failure analysis and trade study for the LUMIO AOCS, an unsupervised machine learning method for fault detection is designed in this thesis.

Literature Gap and Research Question

Based on a literature review performed on deep space CubeSats, spacecraft Fault Detection, Isolation and Recovery (FDIR), and the Lunar Meteroid Impact Observer (LUMIO) mission a gap was identified in the research. CubeSats have revolutionised access to space, with their low cost, a vast body of knowledge and experience, and off-the-shelf components readily available. In recent years, the CubeSat platform has been making its next steps in space exploration, venturing to the Moon, Mars and even asteroids.

However, the reliability of the platform is still very poor: 20% of CubeSat missions are dead-on-arrival, with as much as 40% of missions failing within the first 100 days in orbit. Additionally, this reliability decreases further as the mission profile becomes less typical, such as with deep space missions. Despite these statistics, the CubeSat platform typically incorporates only very basic FDIR mechanisms. This can be attributed to a lack of expertise, resources, and the general difficulty in designing a validated onboard FDIR system, or simply because the cost does not match the benefit in the case of low-cost missions. The deep space CubeSat missions however will need this type of fault detection system considering their more complex architecture, and the reduced risk tolerance compared to Earth orbiting missions.

Creating such an advanced yet accessible FDIR system which can operate on deep space CubeSats is a vast task, so as a starting point in this thesis the case study of LUMIO is used. Specifically, one of the most complex and critical subsystems is focused on: the Attitude and Orbit Control System (AOCS). The research objective is formulated as:

To contribute to the improvement of deep space CubeSat reliability and failure robustness by designing a model-based Fault Detection approach for LUMIO's AOCS.

This leads to the research questions:

1. **RQ1:** What are the most critical failure modes of the LUMIO AOCS subsystem which can be detected, isolated and recovered?
2. **RQ2:** How can the most critical faults be detected (and isolated) using a model-based method?
3. **RQ3:** How accurate is the proposed method at detecting faults in the LUMIO AOCS system?

Fault Analysis

In order to design a fault detection system, one needs to understand the faults which are most likely to be encountered and their effects. For that reason a Fault Tree Analysis (FTA) is performed for the LUMIO AOCS system, complemented by a Failure Mode Effects and Criticality Analysis (FMECA). These two analysis resulted in the criticality matrix shown in Table 1, which uncovered 53 feasible failures in the LUMIO AOCS of which 20 critical failure scenarios. Of these critical faults, 10 are related to incorrect or unavailable sensor data. Therefore, it is imperative that such faults are detected swiftly and accurately.

Severity	**SN**	**Probability (PN)**			
		1	**2**	**3**	**4**
		Extremely Remote	**Remote**	**Occasional**	**Probable**
Catastrophic	**4**				
Critical	**3**	STR.03 STR.04 STR.08 RW.03 RW.04 RW.05 RW.06 RW.07 RW.08 IMU.01 IMU.02 IMU.03 IMU.04 IMU.05 RCS.01 MT.01 MT.07 MT.08 AOCS.01	STR.01 STR.10 RW.01 STR.05 STR.11 RW.02 STR.09 STR.12 IMU.06 IMU.07 IMU.08 RCS.02 RCS.03 MT.02 MT.06 OBC.01	RCS.04 MT.04	
Major	**2**	STR.06 SADA.01	STR.02 RW.09 MT.03 SADA.02 SADA.03 SADA.04	RCS.05 MT.05	
Negligible	**1**	SS.01 SS.02 SS.03	STR.07 SS.04 SS.05		

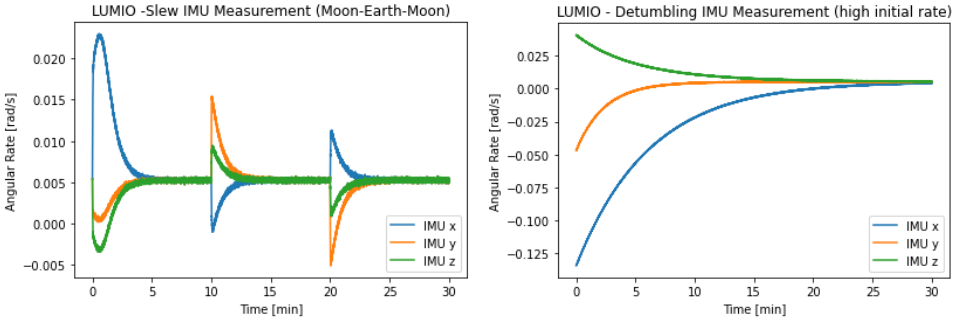
Table 1: Criticality Matrix for LUMIO AOCS (fault IDs refer to the FMECA IDs in Appendix B)

Based on the aforementioned analysis, a set of 35 FDIR requirements for LUMIO were generated. This requirements set served as the basis for a trade study of possible model-based fault detection methods, of which the most feasible were traded. Following concept exploration, weight and criteria setting, scoring, and a sensitivity analysis, the neural network approach to fault detection was selected as the winner.

Spacecraft Telemetry and Fault Simulation

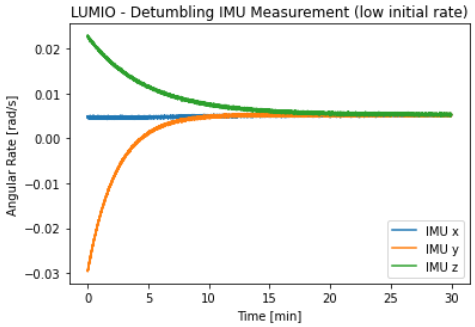
In order to train a neural network, qualitative spacecraft AOCS telemetry is needed as well as fault data. These are not readily available however, and are simulated for this thesis. The LUMIO IMU and Reaction Wheel (RW) telemetry is simulated by the design team at Politecnico di Milano for the four different operational scenarios shown in Figure 1 and Figure 2:

- **Slew**: 30 minute simulation of LUMIO slewing from Moon pointing to Earth pointing and then back to Moon acquisition. (Figure 1a)
- **Detumbling - high velocity**: simulation of detumbling from high initial angular rates (up to 0.15 rad/s on each axis) within 30 minutes. (Figure 1b)
- **Detumbling - low velocity**: simulation of detumbling from low initial angular rates (up to 0.03 rad/s on each axis) within 30 minutes. (Figure 1c)
- **Lunar Tracking**: the spacecraft holds the camera pointing steadily at the moon for a period of 7 days (Figure 2a and Figure 2b)



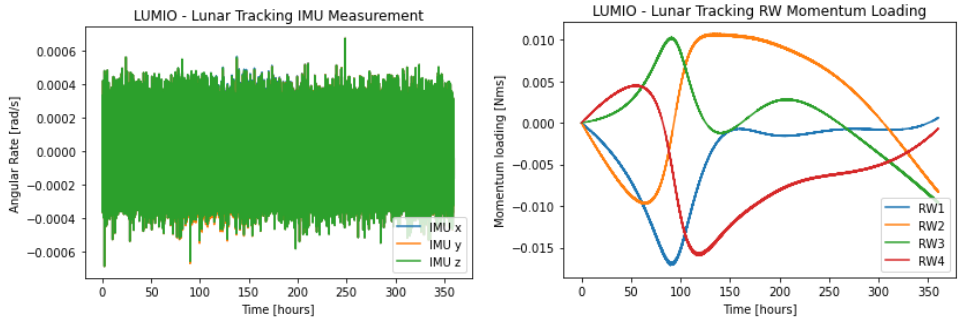
(a) Simulated LUMIO IMU Output: Slew manoeuvre, Moon-Earth-Moon within 30 minutes

(b) Simulated LUMIO IMU Output: high velocity detumbling



(c) Simulated LUMIO IMU Output: low velocity detumbling

Figure 1: Slew and Detumbling simulated IMU Data (source: Politecnico di Milano)



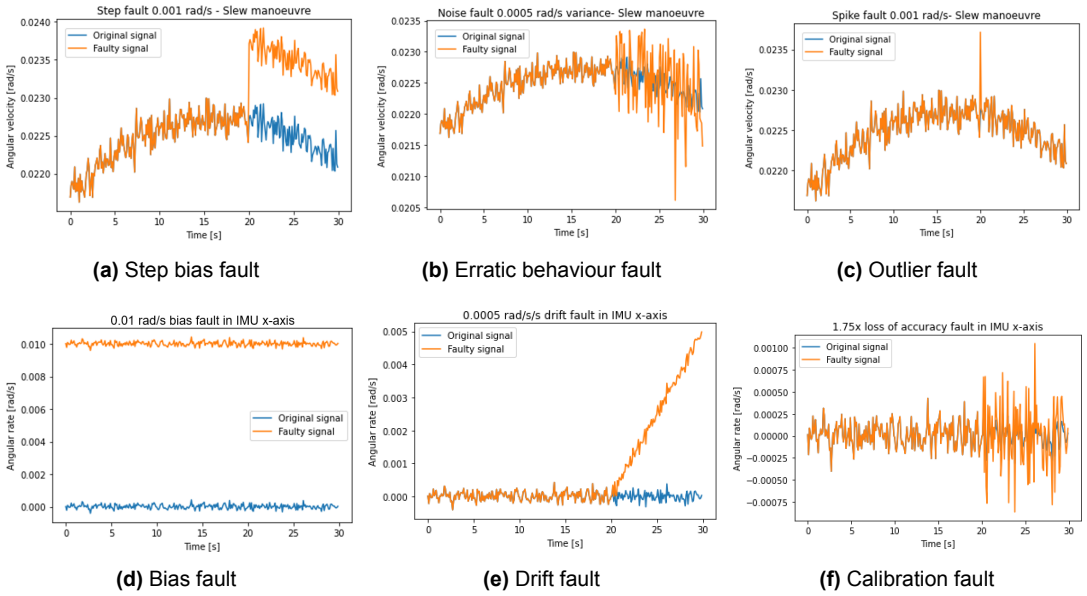
(a) Simulated LUMIO IMU rate measurement during 15 days of lunar tracking

(b) Simulated LUMIO Reaction Wheel momentum loading [Nms] during 15 days of lunar tracking

Figure 2: Simulated Lunar Tracking IMU and reaction wheel data (source: Politecnico di Milano)

Fault Engineering

A range of credible faults collected from literature are artificially introduced in the LUMIO telemetry. These faults are shown in Figure 3 for the LUMIO IMU data. These include both directly detectable fault features (step bias, outliers, erratic behaviour) as well as faults which require model-based approaches (bias, drift, calibration error) to detect. The faults, their potential sources, and the magnitude of the artificially introduced fault are shown in Table 2 and Table 3 for non-model based and model-based detectable faults respectively.



(a) Step bias fault **(b)** Erratic behaviour fault **(c)** Outlier fault

(d) Bias fault **(e)** Drift fault **(f)** Calibration fault

Figure 3: Examples of faults introduced into the LUMIO IMU Readings

Fault Type	**Potential root causes**	**Quantification**	**Reasoning**
Step Bias	Single Event Upset (SEU), ground loops, software bug	+0.002 rad/s	LUMIO pointing requirements
Erratic Behaviour	EM interference (external, internal), ADC/connector hardware fault, ground loops, thermal noise	0.01 rad/s STD	Standard deviation nominal noise
Outlier	SEU, processing/ sampling error	Spike +0.1 rad/s	+1 Order of Magnitude

Table 2: List of directly detectable IMU faults, sources, and quantification for LUMIO

Fault Type	Potential root causes	Quantification	Reasoning
Bias	Damage, ground loops, software bug	+0.01 rad/s	STIM Specifications
Signal drift	Temperature effects, ageing, interference, stress, calibration issues	0.0005 rad/s^2	LUMIO pointing requirements
Loss of accuracy	Calibration error, temperature effects	x1.75	LUMIO pointing requirements

Table 3: List of model-based detection IMU faults, sources and quantification for LUMIO

Since the LUMIO data is simulated, real satellite telemetry is also sourced from ESA's OPS-SAT in order to validate the fault detection method. This data includes the downlinked data from the IMU, Reaction Wheel, Sun Sensor, and Star Tracker.

Design of Neural Network Based Detection Method

After an exploration of neural network based fault detection concepts, the unsupervised learning approach using an autoencoder is chosen. The unsupervised learning approach is very applicable in this case as labelled fault data is not available, and overrepresentation (very little fault data compared to nominal data) is a very real problem in anomaly detection networks. The

fault detection method is schematically shown in Figure 4. The autoencoder network learns to accurately reconstruct nominal telemetry. When faced with faulty spacecraft telemetry, it is unable to reconstruct this data accurately and the reconstruction error will increase indicating the presence of a fault. The threshold is chosen such that false alarms are minimised while still detecting all faults.

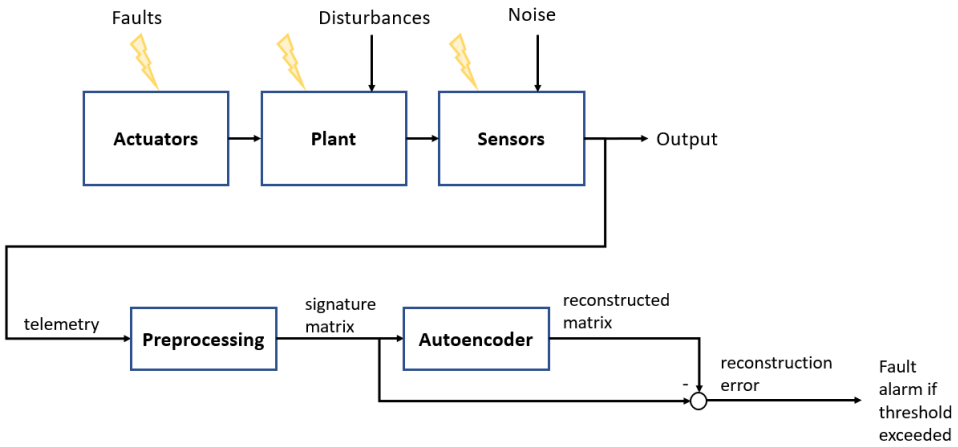


Figure 4: Fault detection process using autoencoder and reconstruction of signature matrices

The network was designed as a lightweight, simple architecture network which is able to run onboard CubeSats without consuming excessive resources. Following tuning of the network hyperparameters, the final design is shown in Table 4.

Parameter	Value
Number of encoding layers	2
Number of decoding layers	2
Neurons in encoding layer 1	260
Neurons in encoding layer 2	64
Activation function	ReLu
Activation function output	Sigmoid
Neurons in latent space	12
Neurons in decoding layer 1	64
Neurons in decoding layer 2	260
Dropout (all layers)	0.1
Number of training epochs	10
Batch size	75
Optimiser function	ADAM
Loss function	MSLE

Table 4: Autoencoder hyperparameters

Signature Matrix Method

To perform model-based fault detection, the signals are not directly fed into the proposed network but rather correlated first through a signature matrix. This signature matrix forms an image which correlates the rate measurements of the IMU with the momentum loading of the reaction wheels. The seven LUMIO measurements (3x IMU, 4x RW) form the 7x7 signature matrix shown as a heatmap in Figure 5. In this matrix every element m_{ij} at row i, column j, is the dot product of the two corresponding telemetry streams.

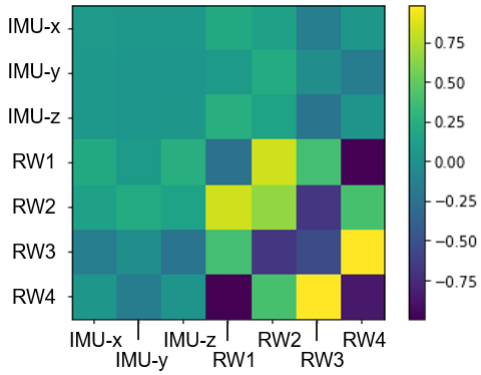
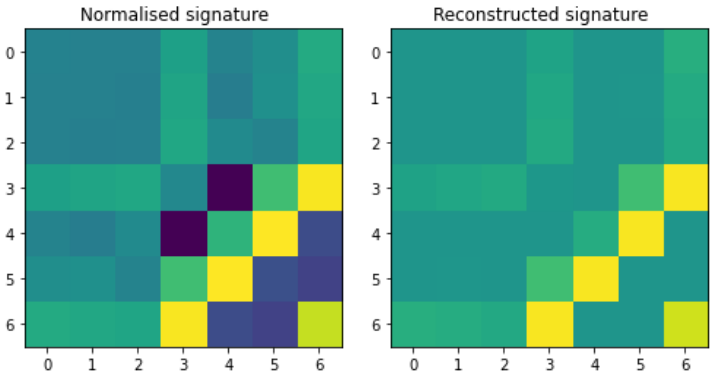


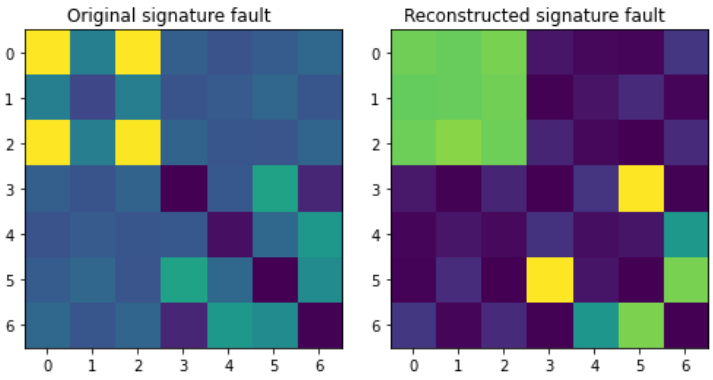
Figure 5: LUMIO Normalised Signature Matrix example with telemetry locations

The reconstruction of the matrix using the autoencoder network can be seen in Figure 6. Here, Figure 6c is generated from the same telemetry as the matrix shown in Figure 6a, but with a drift fault introduced. A non-nominal pattern is introduced by the fault in the upper left corner, and from Figure 6d it can be seen that the reconstruction is far from successful (4 orders of magnitude larger) compared to the reconstruction of the fault-free matrix seen in Figure 6b.



(a) IMU and Reaction Wheel telemetry signature matrix under nominal conditions

(b) Autoencoder reconstructed nominal signature matrix. $\epsilon_{rec} = 2.13x10^{-5}$ (MSLE)



(c) Signature matrix with induced drift fault in LUMIO IMU telemetry

(d) Autoencoder reconstructed fault signature matrix $\epsilon_{rec} = 0.611$ (MSLE)

Figure 6: Autoencoder reconstruction of nominal and faulty signature matrices (LUMIO, 7x7 matrices)

Results

Using the simulated telemetry and the engineered faults for LUMIO, the fault detection network was tested. It was found that all model-based faults were accurately detected. The signature matrix of each fault can be seen in Figure 7 and the reconstructions of these signature matrices can be seen in Figure 8. It can be seen that especially bias and drift faults are easily detected. The detection results per model-based fault are shown in Table 5.

Fault Type	ID	Measurement	Size	Detection LUMIO	Detection OPS-SAT
Bias	1	Angular rate x-axis	0.01 rad/s	✓	✓
	2	Angular rate y-axis		✓	✓
	3	Angular rate z-axis		✓	✓
	4	Angular rate x-axis + z-axis		✓	✓
Drift	5	Angular rate x-axis	0.0005 rad/s^2	✓	✓
	6	Angular rate y-axis		✓	✓
	7	Angular rate z-axis		✓	✓
	8	Angular rate x-axis + z-axis		✓	✓
Loss of Accuracy	9	Angular rate x-axis	x1.75	✓	✓
	10	Angular rate y-axis		✓	✓
	11	Angular rate z-axis		✓	✓
	12	Angular rate x-axis + z-axis		✓	✓

Table 5: Detection results IMU faults using signature matrices for LUMIO (IMU + RW, 7x7 matrices) and OPS-SAT (11x11 matrices)

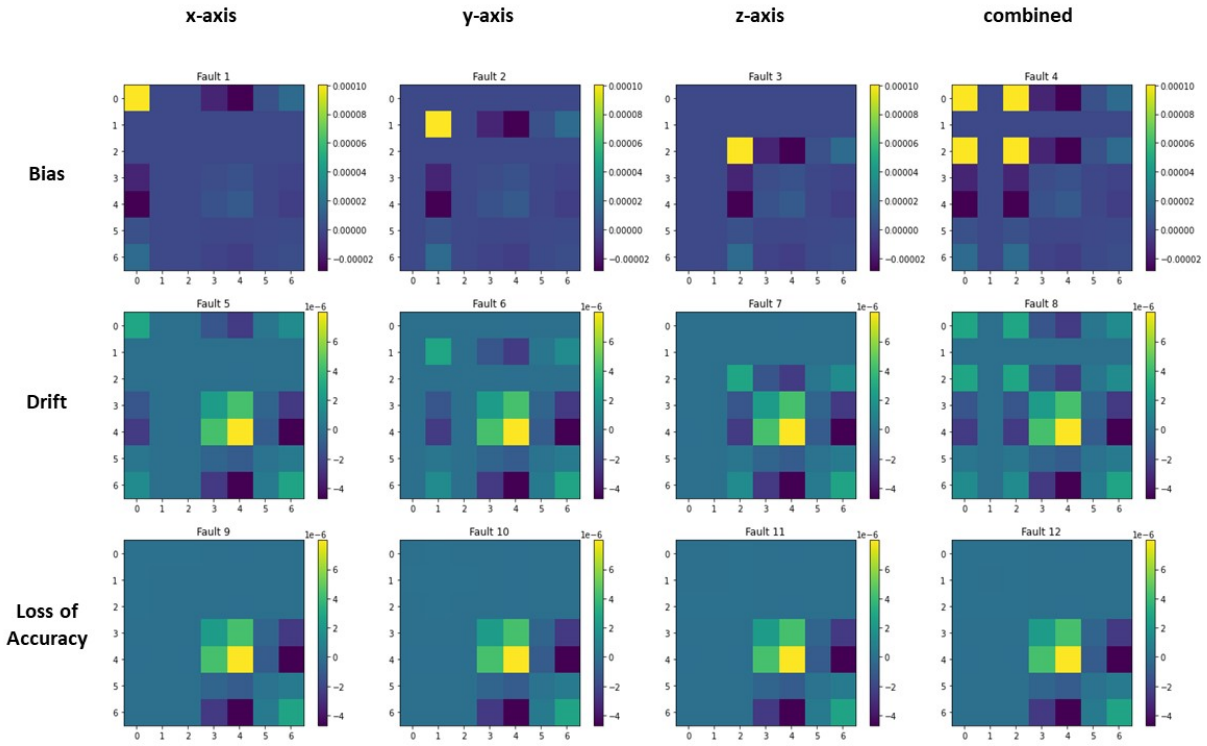


Figure 7: Signature matrices of bias, drift, and loss of accuracy faults introduced in LUMIO IMU signals

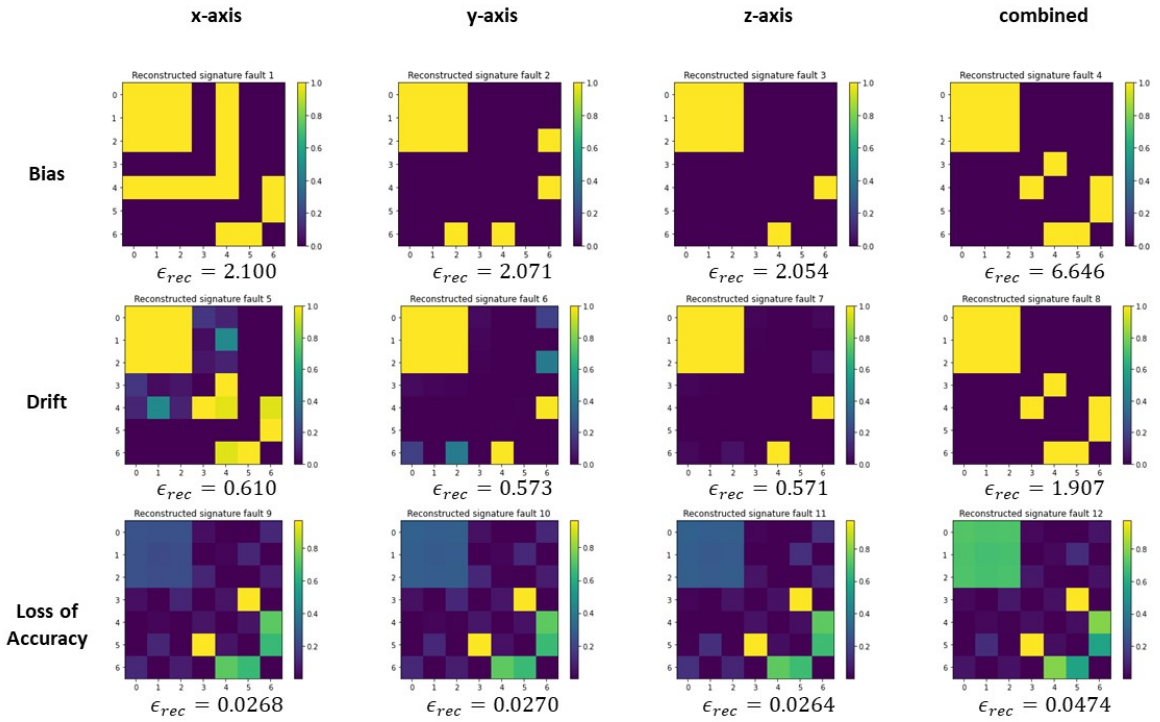
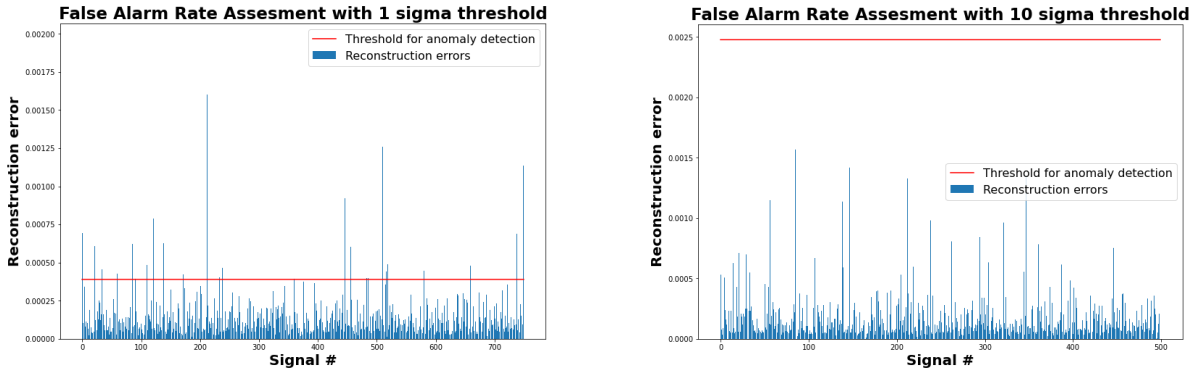


Figure 8: Signature matrices of Figure 7 reconstructed by autoencoder including MSLE reconstruction error. Detection threshold $\tau = 3.93x10^{-4}$

On the other hand, no false alarms were raised when the network was faced with a stream of 700 seconds of nominal telemetry, provided that the detection threshold is tuned accurately. This threshold is set by taking the standard deviation σ of the set of reconstruction errors of fault-free signals, and setting the threshold at a certain amount of standard deviations from the mean of this set of nominal values. As can be seen in Figure 9, at 1 standard deviation there are some risks of false alarms: a false alarm is triggered when the threshold is exceed three times in a row.However, a 10σ detection threshold provides the required fault detection performance while avoiding all risks of false alarms.



(a) False alarms with 1 σ detection threshold

(b) False alarms with 10 σ detection threshold

Figure 9: LUMIO False Alarm Rate in 700 seconds of nominal telemetry

OPS-SAT Fault Detection

As a final step, this system was tested on the real, processed, satellite telemetry from OPS-SAT. The results are shown under the 'OPS-SAT' column in Table 5, and the reconstruction

errors are found in Figure 10. This graph indicates that although the reconstruction errors are lower in absolute value than with LUMIO, they still exceed the detection threshold by at least one order of magnitude in the worst case. This indicates that even when faced with noisy, variable, real spacecraft telemetry the network exhibits good fault detection performance while maintaining no false alarms.

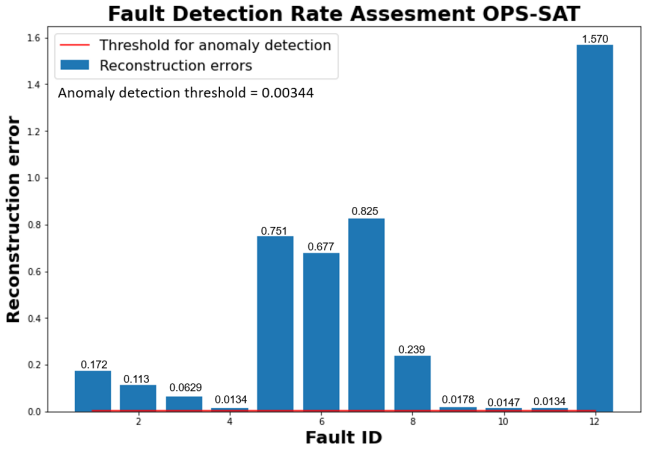


Figure 10: Fault detection results OPS-SAT Data

Computational Performance

A quantitative and qualitative estimation of the impact of using this network on onboard computational resources was made. Comparison to the current state of the art neural networks which have been flown on smallsats today show that this network is tens of thousands times lighter in the number of neurons compared to the heavy payload processing networks. Estimations based on the LUMIO OBC also showed that the network requires around 0.2% of the available computing power if run every 10 seconds. These two assessment indicate that the computational power required to run it will not be an issue for implementation onboard CubeSats.

Conclusion & Recommendations

In conclusion, it was found that a neural network based approach could be a promising model-based fault detection method, which is also accessible to CubeSat developers and able to be run on onboard hardware. The designed network is easily able to detect faults based on the reconstruction error of the signature matrices. When faced with a fault, the reconstruction error increases by two to four orders of magnitude, raising an alarm. Even those subtle faults such as drifts or small biases in the order of a few milliradians, which amount to at most 6 degrees of pointing offset for LUMIO, are detected in all signals of the IMU. It was found that the detection threshold can be tuned such that the system fault detection rate is 100% for the engineered fault set, while no false alarms are triggered.

When testing the method on real spacecraft telemetry coming from ESA's OPS-SAT, which includes telemetry from the sun senors, IMU, reaction wheels, and star trackers. The fault detection accuracy was again 100% for the engineered fault set while not triggering any false alarms. Therefore it is considered a highly promising method for fault detection (and possibly isolation) in CubeSats and other missions.

Based on this simple case study, further work recommended is focused on four areas:

- **Improve data quality:** using real, unprocessed spacecraft telemetry and real fault data would drastically improve the performance of the network.
- **Improve network capabilities:** the fault patterns in the signature matrices could be used to perform fault isolation. This requires real fault data or fault simulation on flight hardware.
- **Improve efficiency:** fine-tune network size and architecture for increased performance with reduced impact on computational resources.
- **Create proper validation setup:** in-flight testing on missions such as OPS-SAT would be preferable but are risky due to the purposeful introduction of faults in the system. Testing the system in a flatsat setup (spacecraft is electronically fully integrated) would be the next best option.

Contents

Preface i

Abstract ii

Executive Summary iii

List of Symbols xvi

List of Figures xvii

List of Tables xviii

List of Abbreviations xx

1 Introduction 1
- 1.1 Literature Gap . . . 1
- 1.2 Research Question and Objective . . . 2
- 1.3 LUMIO Case Study . . . 3
- 1.4 Thesis Outline . . . 3

2 Literature Study 4
- 2.1 Deep Space CubeSats . . . 4
- 2.2 LUMIO . . . 4
  - 2.2.1 Scientific Objectives . . . 5
  - 2.2.2 Mission Profile . . . 6
  - 2.2.3 Mission Phases . . . 6
  - 2.2.4 Architecture . . . 7
- 2.3 FDIR . . . 10
  - 2.3.1 Traditional Spacecraft FDIR . . . 10
  - 2.3.2 Overview of Model Based Methods . . . 11
  - 2.3.3 FDIR in Deep Space CubeSats . . . 13

3 LUMIO Fault Analysis 15
- 3.1 LUMIO Fault Tree Analysis . . . 15
  - 3.1.1 The FTA Methodology . . . 15
  - 3.1.2 LUMIO FTA . . . 16
  - 3.1.3 FTA Results . . . 18
- 3.2 FMECA . . . 19
  - 3.2.1 Scope of the FMECA . . . 20
  - 3.2.2 FMECA Results . . . 20
  - 3.2.3 Fault Register . . . 23
- 3.3 FDIR Requirements . . . 23
  - 3.3.1 General Requirements . . . 24
  - 3.3.2 Functional Requirements . . . 24
  - 3.3.3 Performance Requirements . . . 25
- 3.4 Trade Study . . . 25
  - 3.4.1 Design Options . . . 26
  - 3.4.2 Trade Criteria and Weights . . . 26

3.4.3 Trade Off Results . . . 28
3.5 Alternate Scoring and Critical Review . . . 28

4 Fault Data Simulation **29**
4.1 Fault Definition and Simulation Method . . . 29
4.1.1 Directly Detectable Faults (Non Model Based) . . . 30
4.1.2 Faults Requiring Cross Checks (Model Based) . . . 31
4.1.3 Other fault types . . . 32
4.2 LUMIO AOCS Telemetry Simulation . . . 33
4.2.1 GAFE Simulator . . . 33
4.2.2 Simulated LUMIO Telemetry: Politecnico di Milano . . . 35
4.3 Real Satellite Telemetry: OPS-SAT . . . 36

5 Design of Fault Detection Method **38**
5.1 Introduction to Neural Networks . . . 38
5.1.1 Weights and Bias . . . 39
5.1.2 Activation Function . . . 39
5.1.3 Loss Function . . . 40
5.1.4 Autoencoder Explained . . . 41
5.1.5 Effectiveness Metrics for Neural Networks . . . 42
5.2 Exploration of Fault Detection Methods Using Neural Networks . . . 43
5.2.1 Signal Level Fault Detection . . . 43
5.2.2 Neural Network Based Nonlinear Regression and Residual Generation 43
5.2.3 Neural Network Based Fault Classification . . . 44
5.2.4 Time Series Correlation . . . 44
5.2.5 Other Methods . . . 44
5.2.6 Challenges in Neural Network Based FD . . . 45
5.2.7 Selection of Unsupervised Learning for this Thesis . . . 45
5.3 Training Data . . . 46
5.3.1 Data Structure . . . 46
5.3.2 Normalisation . . . 46
5.3.3 Reserving Validation Data . . . 48
5.4 Design of the Autoencoder Network . . . 49
5.4.1 Design Philosophy . . . 49
5.4.2 Detection Mechanism . . . 49
5.4.3 Network Hyperparameter Tuning . . . 50
5.4.4 Final Network Architecture . . . 54

6 Results **55**
6.1 Detection of Signal Level Faults in LUMIO IMU . . . 55
6.1.1 False Alarm Rate . . . 56
6.2 Model-Based Fault Detection in LUMIO Data . . . 57
6.2.1 Signature Matrix Method . . . 58
6.2.2 Fault Detection Results LUMIO Data . . . 60
6.2.3 False Alarm Rate . . . 62
6.3 Model Based Fault Detection in OPS-SAT Telemetry . . . 63
6.3.1 Fault Detection Results OPS-SAT . . . 63
6.3.2 False Alarm Rate . . . 66
6.4 Analysis of Results . . . 67
6.4.1 Limitations . . . 67
6.5 Computational Resources . . . 67
6.5.1 Comparison to State of the Art . . . 68

6.5.2 Estimation of Number of Operations . . . 68
6.6 A Note on Verification and Validation Activities . . . 68

7 Conclusion **70**
7.1 Conclusion . . . 70
7.2 Answers to Research Questions . . . 71
7.3 Recommendations . . . 72
7.3.1 Improvement of Training and Fault Data . . . 72
7.3.2 Improvement of Network Capabilities . . . 72
7.3.3 Optimisation . . . 73
7.3.4 Validation Activities . . . 73

References **74**

A LUMIO Fault Trees **79**

B LUMIO AOCS FMECA **90**

C Critical Faults Register **100**

D LUMIO FDIR Requirements Analysis **103**
D.1 Relevant Mission and System Requirements . . . 103
D.2 Relevant AOCS Requirements . . . 103
D.3 Relevant Autonomy Requirements . . . 104
D.4 FDIR Requirements . . . 104
D.4.1 General Requirements . . . 104
D.4.2 Functional Requirements . . . 104
D.4.3 Performance Requirements . . . 109
D.4.4 Interface Requirements . . . 109

E Trade Off **111**
E.1 Concept Exploration . . . 111
E.2 Methods to Determine Weights . . . 113
E.2.1 Scoring . . . 113
E.2.2 Ranking . . . 113
E.2.3 Analytical Hierarchy Process . . . 114
E.3 Comparison to classical ranking . . . 115
E.4 Comparison to Pugh Matrix Scoring Method . . . 116

F LUMIO and OPS-SAT Faulty Signals **117**
F.1 LUMIO IMU Faults . . . 117
F.2 OPS-SAT IMU Faults . . . 119

G OPS-SAT Telemetry **122**
G.1 Quaternion Data . . . 122
G.2 Reaction Wheel Data . . . 124
G.3 IMU Data . . . 125
G.4 Sun Angle Telemetry . . . 127

List of Symbols

CN	Criticality Number	[-]
F_1	F1-score	[-]
$I_{sp,vac}$	Vacuum Specific Impulse	[s]
n_{th}	Detection threshold tuning factor	[-]
PN	Probability Number	[-]
R_E	Earth Radii	[km]
SN	Severity Number	[-]
ΔV	Change in orbital velocity	[m/s]
$\vec{X}_j^t$	Time series	[-]
ϵ_{rec}	Reconstruction error	[-]
μ_ϵ	Mean reconstruction error	[-]
σ_ϵ	Standard deviation of reconstruction errors	[-]
τ	Anomaly detection threshold	[-]

List of Figures

2.1 LUMIO operative phases. Source: Cervone et al. [4] . . . 6
2.2 LUMIO Phase A AOCS Architecture . . . 8
2.3 The LUMIO Spacecraft rendered in the Phase A design configuration. Source: Cervone et al. [4] . . . 9
2.4 Nonlinear System Fault Diagnosis Methods. Source: Sanchuan Xu [70] . . . 12

3.1 Demonstration of the Fault Tree Analysis methodology. Source: Bidner [2] . . . 16
3.2 Fault Tree for LUMIO IMU Fault . . . 18
3.3 LUMIO FDIR Design Option Tree (green/bold are those concepts selected for trade off) . . . 26

4.1 Fault examples - directly detectable faults . . . 31
4.2 Fault examples - model based detectable faults . . . 33
4.3 GAFE Simulated IMU Faults: following boot up ($60s$) and detumbling ($600s$) a noise fault occurs at $t = 1200s$ and random walk fault at $t = 4000s$ . . . 34
4.4 Slew and Detumbling simulated IMU Data (source: Politecnico di Milano) . . . 35
4.5 Simulated Lunar Tracking IMU and reaction wheel data (source: Politecnico di Milano) . . . 36
4.6 OPS SAT IMU telemetry downlinked during a 39 minute communication window (window 2) on 29 November 2022 . . . 37

5.1 Simple neural network layout with two hidden layers . . . 38
5.2 Basic workings of a single neuron in a neural network explained . . . 39
5.3 ReLU (left) and PReLU (right) activation functions. Source: He et al. [27] . . . 40
5.4 Sigmoid Activation Function. Source: Martin Thoma [61] . . . 40
5.5 Generic autoencoder network with 2 encoding layers, 2 decoding layers 2 and a latent representation layer . . . 41
5.6 An example of noisy handwriting of numbers reconstructed by an autoencoder. Source: Prashanth Venkataraman [63] . . . 42
5.7 Conversion from LUMIO AOCS Telemetry stream to data frames for training and testing . . . 46
5.8 Normalisation methods performance comparison . . . 48
5.9 Fault detection process using autoencoder and reconstruction of signature matrices . . . 49
5.10 Autoencoder loss and validated loss evolution over the training epochs . . . 51
5.11 Layer size tuning for autoencoder network . . . 51
5.12 Batch size tuning process . . . 52
5.13 Loss function selection . . . 53
5.14 Latent representation size tuning . . . 54

6.1 Fault locations in LUMIO IMU slew data. At each location, a step, noise and outlier fault are inserted once. . . . 55
6.2 Slew manoeuvre reconstruction errors LUMIO IMU (S = step, N = noise, O = outlier) . . . 56

6.3 False Alarm Rate Assessment for signal processing - 1σ threshold 57
6.4 LUMIO Normalised Signature Matrix example with telemetry locations 59
6.5 Autoencoder reconstruction process demonstrated for nominal signal 59
6.6 Autoencoder reconstruction of nominal and faulty signature matrices (LUMIO, 7x7 matrices) 60
6.7 Signature matrices of bias, drift, and loss of accuracy faults introduced in LUMIO IMU signals 61
6.8 Signature matrices of Figure 6.7 reconstructed by autoencoder including MSLE reconstruction error 61
6.9 Reconstruction errors for LUMIO faults compared to detection threshold 62
6.10 LUMIO False Alarm Rate in 700 seconds of nominal telemetry 63
6.11 OPS-SAT Signature Matrix 63
6.12 Signature matrices of bias, drift, and loss of accuracy faults introduced in OPS-SAT signals 64
6.13 Signature matrices of Figure 6.12 reconstructed by autoencoder, including reconstruction error ϵ_{rec} 64
6.14 Fault detection results OPS-SAT Data 66
6.15 OPS-SAT False Alarm Rate in 30 minutes of nominal telemetry 66

List of Tables

2.1 A summary of launched deep space CubeSat missions. All missions are of form factor 6U unless mentioned otherwise. 5
2.2 LUMIO Phase A ADCS sensors and actuators selection [4] 8

3.1 LUMIO FTA Feared Events List (Phases: 1 Parking, 2 Transfer, 3 Operations, 4 Disposal) 17
3.2 FMECA Probability Number quantified as defined in the ECSS-Q-ST-30-02C standard [14] 19
3.3 Failure mode severity levels as defined in the ECSS-Q-ST-30-02C standard [14], with additions from the US Nuclear Regulatory Commission Fault Tree Handbook [9] 19
3.4 Criticality Matrix. Source: ECSS [14] 20
3.5 Example of FMECA Item - IMU.08 IMU Measurement Drift 21
3.6 Criticality Matrix for LUMIO AOCS (fault IDs refer to the FMECA IDs in Appendix B) 22
3.7 Critical Fault F26 as an example from the critical fault register in Appendix C . 23
3.8 LUMIO FDIR General Requirements most relevant to this thesis 24
3.9 FDIR Functional Requirements LUMIO most relevant to this thesis 25
3.10 FDIR Performance Requirements LUMIO 25
3.11 Criteria weights determined using AHP 27
3.12 Trade Off with weights determined through AHP 27
3.13 Results of the trade study using alternate ranking and scoring mechanisms . . 28

4.1 List of directly detectable IMU faults, sources, and quantification for LUMIO . . 31
4.2 List of IMU faults not directly detectable, sources and quantification for LUMIO 32
4.3 OPS-SAT AOCS telemetry package content used in this thesis 37

5.1 Binary Classifier Network outcomes 42
5.2 Autoencoder hyperparameters 54

6.1 Example LUMIO signature matrix (lunar tracking, 50 second frame taken at t= 5000s): each number represents the dot product of the telemetry stream corresponding to its row and column respectively 58
6.2 Detection results IMU faults using signature matrices for LUMIO (IMU + RW, 7x7 matrices) 62
6.3 Detection results IMU faults using signature matrices for OPS-SAT (extended telemetry, 11x11 matrices) 65

List of Abbreviations

ADCS Attitude Determination and Control System.
AHP Analytical Hierarchy Process.
ANN Artificial Neural Networks.
AOCS Attitude and Orbit Control System.

CE Cross-Entropy.
CMG Control Moment Gyroscope.
CNN Convolutional Neural Network.
COTS Commerical Off-The-Shelf.
CPL Cognitive Programming Language.
CRC Cyclic Redundancy Check.

DBN Dynamic Bayesian Network.
DOT Design Option Tree.
DTE Direct-to-Earth.

ECSS European Cooperation for Space Standardization.
EMC Electromagnetic Compatibility.
EPS Electrical Power System.
ESA European Space Agency.

FAR False Alarm Rate.
FDIR Fault Detection, Isolation and Recovery.
FDR Fault Detection Rate.
FMECA Failure Mode Effects and Criticality Analysis.
FTA Fault Tree Analysis.

GAFE Generic AOCS/GNC Techniques & Design Framework for FDIR.
GCR Galactic Cosmic Rays.

HIM Halo Injection Manoeuvre.

IMU Inertial Measurement Unit.
ISL Inter-Satellite Link.

KLD Kullback-Leibler Divergence.

LEO Low Earth Orbit.
LEOP Launch and Early Orbit Phase.
LUMIO Lunar Meteroid Impact Observer.

MAE Mean Absolute Error.

MAPE Mean Absolute Percentage Error.
MSE Mean Squared Error.
MSLE Mean Squared Logarithmic Error.

NEO Near Earth Objects.

OBC Onboard Computer.
OBPDP Onboard Payload Data Processing.

RAMS Reliability, Availability, Maintainability, Safety.
RCS Reaction Control System.
RCT Reaction Control Thruster.
ReLU Rectified Linear Unit.

SADA Solar Array Drive Assembly.
SEB Single Event Burnout.
SEGR Single Event Gate Rupture.
SEU Single Event Upset.
SMIM Stable Manifold Injection Manoeuvre.
SVM Support Vector Machines.

TCM Trajectory Correction Manoeuvre.
TID Total Ionising Dose.
TLI Trans Lunar Injection.
TT&C Telemetry, Tracking & Command.

1 Introduction

The CubeSat platform has revolutionised access to space for commercial and academic developers in recent years. Low costs, many off the shelf components and publicly available information on designing, building, testing and operating CubeSats have all contributed to the form factor becoming a widely accepted solution for simple, Earth orbiting missions. However, the platform suffers from low reliability and high dead-on-arrival rates making them less suitable for deep space missions. Nevertheless, NASA's MarCO mission and Italy's LICIACube have demonstrated a successful implementations of the CubeSat in space missions beyond Earth orbit, which in turn increased interest in the use of these versatile platforms in such applications. To help achieve this expansion of what is possible with CubeSats, the reliability problem should be solved in part by implementing more accurate, efficient and accessible solutions to CubeSat Fault Detection, Isolation and Recovery.

This thesis is concerned with the development of a novel model-based fault detection method using neural networks for the Attitude Determination and Control Systems of the Lunar Micrometeoroid Impact Observer mission. This introduction includes the gap in the body of scientific knowledge which was identified in section 1.1, followed by a definition of the thesis research objective and question in section 1.2. The LUMIO case study is introduced in section 1.3 and finally the outline of this thesis is presented in section 1.4.

1.1. Literature Gap

The literature review preceding this thesis covered many pieces of academic work ranging from satellite FDIR to the LUMIO mission specific design and the deep space CubeSat missions landscape. Note that in this thesis the term 'deep space mission' is used to broadly describe those missions which do not orbit the Earth. This is because a mission then encounters a very different radiative, thermal, aerodynamic, gravitational, magnetic, and dynamic environment. These changes compared to Earth orbiting missions impact the requirements, design, cost, and risk tolerance of the mission. Note that the definition of deep space can differ in literature.

A key gap identified in academic work is the successful application of machine learning techniques for spacecraft fault detection. While machine learning has advanced rapidly in recent years, and found applications in many fields such as image recognition or language processing, it seems spacecraft designers are still reluctant to apply such methods to their systems outside of payload data processing. This is for a few key reasons:

- Insufficient (labelled) data available for training
- No flight heritage (which forms a vicious cycle)
- Onboard computational resources required to train and run
- Difficult to perform verification and validation due to non-deterministic behaviour

Research into machine learning applications show that neural networks can be succesfully applied in the field of anomaly detection. The potential benefits are numerous: networks can recognise patterns which cannot be seen with traditional analytical, statistical, or numerical methods thus detecting faults more accurately and rapidly while increasing spacecraft reliability and availability. The machine learning methods can make FDIR more accessible as it does not require expert knowledge of a subsystem or highly accurate models of nonlinear dynamics. It can therefore reduce the failure rate of these low cost missions developed by educational and commercial institutions with little to no prior experience. Once developed. machine learning methods can be easily transfered and scaled to other mission types. Therefore, the gap identified in literature is the design of a model-based FDIR method for deep space CubeSats which uses machine learning.

1.2. Research Question and Objective

Based on the aforementioned gap in literature, a research objective and the corresponding research questions are formulated. The research objective is a clear formulation of what is expected to be achieved by the end of this thesis. Based on the literature study and the discussions held with the LUMIO project team, the main research objectives is:

To contribute to the improvement of deep space CubeSat reliability and failure robustness by designing a model-based Fault Detection approach for LUMIO's AOCS.

This leads to the research questions:

1. **RQ1:** What are the most critical failure modes of the LUMIO AOCS subsystem which can be detected, isolated and recovered?
    - **RQ1.1:** What are all the feasible failure modes of the LUMIO AOCS subsystem?
    - **RQ1.2:** Which critical failure modes in other subsystems would lead to a malfunctioning in the AOCS subsystem?
    - **RQ1.3:** Which AOCS failure modes are most critical?
    - **RQ1.4:** Which AOCS failure modes can be reasonably detected, isolated and potentially recovered?
2. **RQ2:** How can the most critical faults be detected (and isolated) using a model-based method?
    - **RQ2.1:** Which model-based methods and models are available and what are their characteristics?
    - **RQ2.2:** Which method(s) and models are most suitable for the detection of the faults leading to failure modes identified in RQ1.3?
3. **RQ3:** How accurate is the proposed method at detecting faults in the LUMIO AOCS system?
    - **RQ3.1:** Which representative data is available for testing the fault detection accuracy of the method?
    - **RQ3.2:** How will the fault detection performance of the method be evaluated?

1.3. LUMIO Case Study

As CubeSats are deemed less reliable compared to their bigger counterparts, they were initially not considered for deep space missions. This is especially true considering the CubeSat reliability was shown to be even lower than average for less 'typical' mission profiles. This is certainly not desirable for pioneering CubeSats with the aim of travelling beyond Earth orbit. This inherent lack of reliability could be exacerbated for deep space missions due to the comparatively more hazardous environment with increased radiation levels, or the small body of knowledge and experience for these mission types. This raises concerns for the reliability of CubeSats beyond Earth orbit.

To contribute to this area, a case study was chosen in order to add realism to the development of a novel FDIR method and take into account real-life constraints of a spacecraft: Lunar Meteroid Impact Observer (LUMIO) is a 12U deep space CubeSat perfectly suited to this role. It is pioneering new technologies and architectures, and faces operational challenges different to LEO missions. Designing an advanced fault detection system for this spacecraft aims to lay the groundwork for improving CubeSat reliability using new techniques such as machine learning. It is especially applicable to those CubeSats being used to explore space beyond the Earth. A preliminary FDIR design for LUMIO was already created in prior work, but a gap was left in detecting complex AOCS faults as this required advanced models of LUMIO dynamics and expert knowledge of spacecraft attitude determination and control theory. By using the LUMIO case study, this thesis hopes to contribute to the project while demonstrating neural networks are an effective way to detect faults in nonlinear dynamic systems such as the LUMIO AOCS.

1.4. Thesis Outline

The approach taken in this research is reflected in the structure of the thesis: first the outcomes of the literature study relevant to this thesis are summarised in chapter 2. This includes a review of deep space CubeSat missions recently launched or soon-to-be launched, a summary of the LUMIO mission and a short overview of the Fault Detection, Isolation and Recovery body of knowledge.

Next, the LUMIO mission is analysed and the fault analysis is performed on the AOCS system in chapter 3. This involves a Fault Tree Analysis, a Failure Mechanics, Effects and Criticality Analysis and the generation of basic FDIR requirements and constraints for the design of a fault detection system. Based on the available model-based fault detection design options a trade study is performed in order to confirm the suitability of the neural network based approach for this application compared to other methods.

In chapter 4 the collection of data for training and testing the network from various sources is described, including (simulated) spacecraft telemetry. This chapter also discusses the typical faults to be expected and their potential sources, as well as the simulation of these faults in the telemetry.

The fault detection mechanism and the network itself are designed in chapter 5, where the hyperparameters are discussed and tuned. In chapter 6 the performance of the system is characterised through tests on LUMIO data using the signature matrix method, and a more realistic and extended test is performed on the OPS-SAT data using more telemetry from the AOCS system. Finally, the conclusion and recommendations follow in chapter 7.

2

Literature Study

In preparation of the thesis work, a literature study was performed. A short summary of the relevant topics and findings following this study are presented in this chapter, starting with the assessment of the current deep space CubeSat field in section 2.1. In section 2.2 the findings on the LUMIO mission profile and spacecraft architecture are presented, and finally the relevant information on spacecraft FDIR and the model-based FDIR methods are shown in section 2.3.

2.1. Deep Space CubeSats

Increasing interest has been shown in using the CubeSat platform for deep space applications. In Table 2.1 an overview of recent CubeSat missions flown to cislunar and deep space is displayed. Most of these have been launched as secondary mission with NASA's Artemis-I mission on November 16 2022. Prior to Artemis-I only two successfully completed deep space applications of the CubeSat were found in available public sources, as well as two currently ongoing missions. This shows the platform has only very recently gained credibility in this field. LICIACube, the companion mission for NASA's Double Asteroid Redirection Test (DART) completed its mission in 2022 [12], and the duo of MarCO CubeSats served succesfully as relay spacecraft around Mars for the Insight Lander [7]. The CAPSTONE mission was launched in June 2022 and is currently conducting its mission around the Moon [68]. Following the Artemis-I mission, one other CubeSat mission was launched and is currently exploring the Lunar poles: Lunar Flashlight [8].

It can also be seen from Table 2.1 that most missions are developed by agencies and educational institutions, and so far there has been little interest from commercial/amateur CubeSat developers in these missions. All but one (CAPSTONE) mission have chosen a 6U form factor meaning that the 12U format, which is also the chosen form factor for LUMIO, has very little flight heritage in this mission type.

2.2. LUMIO

LUMIO, the case study in this thesis, is a 12U CubeSat conceived by a team of academic and industrial partners participating in ESA's SYSNOVA competition, focusing on Lunar CubeSats for Exploration. The concept won the competition and went through two design phases in collaboration with ESA. At the time of writing it is undergoing its Phase B design (preliminary definition) phase of the ESA project lifecycle, however this thesis will use the completed Phase A design described by Cervone et al. [4] as a baseline.

Mission Name	Developer	Launch	Mission	Lifetime	Mass
MarCO (2 spacecraft)	NASA	May 2018	Data relay for Insight Mars lander [7]	1 year (cruise mostly)	13.5 kg (each)
LICIACube	Italian Space Agency	Nov 2021	Post impact observation for DART [12]	15 month cruise + 6 month ops	14 kg
CAPSTONE (12U)	NASA	Jun 2022	Demonstrate new navigation and lunar orbit [68]	10 months (4 transfer)	25 kg
Lunar IceCube	Morehead University	Nov 2022	Detect water in lunar exosphere and surface [38]	<2 years	14 kg
LUNAH-MAP	Arizona State	Nov 2022	Investigate hydrogen in moon's shadowy regions [25]	2 months	14 kg
LUNIR	Lockheed Martin	Nov 2022	Characterise lunar surface composition	Unknown	14 kg
OMOTENASHI	JAXA	Nov 2022	Demonstrator for semi-hard lunar landing [26]	Order of days	12.6 kg
NEA Scout	NASA	Nov 2022	Demonstrator for solar sail + image NEA [40]	2.5 years	12 kg
EQUULEUS	JAXA, University of Tokyo	Nov 2022	Observe Earth plasmasphere [18]	1+ year	11 kg
Biosentinel	NASA	Nov 2022	Characterise radiation effects on DNA repair [49]	18 months	13 kg
CUSP	NASA	Nov 2022	Measure space weather [23]	3 months	14 kg
Team Miles	Miles Space & Fluid	Nov 2022	Test long range radio communications for CubeSats	No target (distance defined)	14 kg
ARGOMOON	Argotec, ISA	Nov 2022	Record images of the ICPS in operation [11]	190 days	14 kg
Lunar Flashlight	NASA (JPL)	Dec 2022	Ice and volatiles mapping in lunar poles [8]	60 days	14 kg

Table 2.1: A summary of launched deep space CubeSat missions. All missions are of form factor 6U unless mentioned otherwise.

2.2.1. Scientific Objectives

A Near Earth Objects (NEO) is defined by NASA as an object with perihelion of less than 1.3 AU [42] and can form a significant threat to both Earth and space assets. Therefore, the monitoring of these objects is important for planetary defence in the first place, but also to protect spacecraft, space stations and potential future extraterrestrial bases. While observations of larger objects is possible from Earth, smaller objects in the sub-meter range are very difficult to monitor. Their impact however can still be catastrophic. On top of this, scientists are attempting to model micrometeroid (diameter of 10μm to 2mm) flux in the Earth-Moon system in order to test various hypothesis about the spatial distribution of impacts on the moon [4].

These tiny particles cannot be observed directly, but rather through secondary phenomena

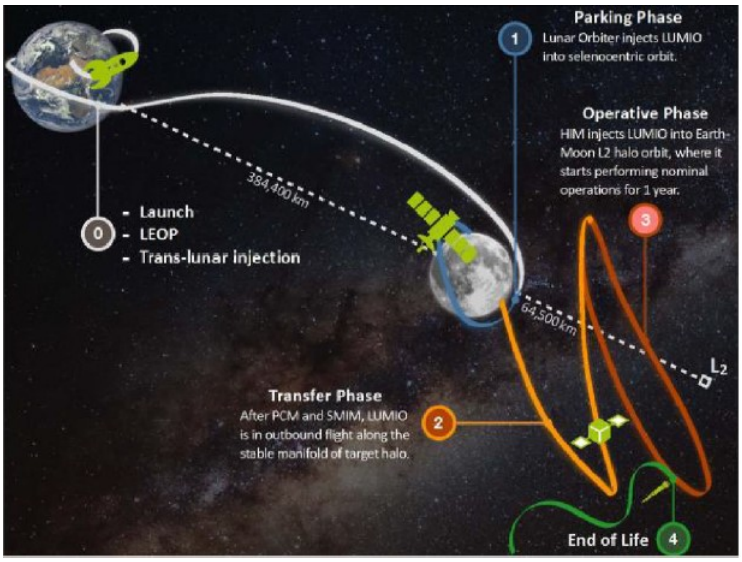


Figure 2.1: LUMIO operative phases. Source: Cervone et al. [4]

such as their impacts on the lunar surface. Despite the small scale of the particles, the high velocity impact releases a large amount of energy, which is partially released in the form of a flash. This flash can be observed by simply observing the Moon in the visible spectrum when the surface is not overly illuminated. However due to the tidal locking effect, Earth-based observations will only ever cover a single hemisphere of the Moon, with the other half (the ”far side” or ”dark side”) not being visible. This establishes the need for a space-based mission which is able to observe impact flashes on the far side: LUMIO.

The LUMIO science goal is phrased as *”Advance the understanding of how meteoroids evolve in the cislunar space by observing the flashes produced by their impacts with the Lunar surface.”* [4] This leads to the science question *”What are the spatial and temporal characteristics of meteoroids impacting the Lunar surface?”* [4]

2.2.2. Mission Profile

In order to answer the research question, visual observations of the entire lunar far side are required. Therefore the design study chose to place LUMIO in a halo orbit around the Earth-Moon L2 equilibrium point, around 60,000 km from the surface, for a nominal 1-year mission. This orbit is considered to be very stable which means little station keeping is required. Another benefit of this orbit is that the spacecraft is never eclipsed by the moon, which eases communication and navigation systems design. The environment itself is not considered particularly hazardous to the spacecraft, with the main risks coming from solar particles and Galactic Cosmic Rays (GCR).

2.2.3. Mission Phases

In order to reach the desired final orbit and perform its scientific mission, LUMIO will go through a number of phases from launch to disposal. These five phases are discussed below and shown in Figure 2.1.

Phase 0: Launch, LEOP and Trans-lunar Injection

While no launch provider has been selected, two main launch opportunities have been identified: Commercial Lunar Payload Services (CLPS) and Artemis-II. Both are moon-bound, but the Artemis-II mission injects LUMIO into a trans-Lunar orbit which is less suitable for the trans-

fer compared to the CLPS injection into selenocentric parking orbit. Therefore, the ΔV budget for the mission was generated based on the Artemis-II launch, and an optimised transfer strategy was designed to bring LUMIO into the desired operational orbit. This thesis will assume this case as the baseline.

During initial stage of the mission,the launch, Launch and Early Orbit Phase (LEOP), and Trans Lunar Injection (TLI), the spacecraft is stowed inside the CubeSat dispenser. Here, the ”kill switches” are pressed such that no power is supplied to the spacecraft and it is inactive while inside the transfer stage. This is to limit the risk to the primary mission and other CubeSat missions, in case a malfunction causes an unexpected event such as a sudden propellant or electrical discharge.

Phase 1: Parking
Once the spacecraft is released from the dispenser, the onboard systems are switched on, detumbling occurs and the solar arrays are deployed. Following this all systems are commissioned. Depending on the chosen launcher, the spacecraft is now in a trans-Lunar orbit (Artemis-II) or a lunar parking orbit (CLPS).

Phase 2: Transfer
Although the transfer strategy is different depending on which mission LUMIO is launched, there are a few main components to the transfer: Stable Manifold Injection Manoeuvre (SMIM) to approach the halo orbit, Trajectory Correction Manoeuvre (TCM) manoeuvres on the approach to L2, and finally a Halo Injection Manoeuvre (HIM). Notable is that for the Artemis-II case, the SMIM is much more demanding and therefor the total ΔV budget is larger compared to the CLPS launch: 201.8 m/s versus 119.5 m/s including margins. The mission is therefore designed assuming the worst case of an Artemis-II launch.

Phase 3: Operations
After successful completion of the transfer and injection into the L2 halo orbit, the operations can start. The primary mission, which observes the lunar far side for impact flashes, can only take place when the illumination of the disk is less than 50%, or around half of a lunar cycle (14.765 days). Therefore the operations are divided into a science cycle and navigation and engineering cycle:

1. **Science Cycle:** when less than 50% of the lunar surface is illuminated (14.765 days, around half of the lunar cycle), impact flashes can be observed and the spacecraft will be continuously performing its scientific mission.
2. **Navigation and Engineering Cycle:** For the remaining part of the Moon’s revolution around the Earth when the far side is illuminated too much for observations, the spacecraft will perform experiments such as autonomous optical navigation and ISL communication, as well as perform station keeping and wheel desaturation manoeuvres.

Phase 4: Disposal
After the nominal mission duration of 1 year, barring any extensions, the spacecraft is removed from the L2 halo orbit and placed into a disposal orbit and the systems are decommissioned.

2.2.4. Architecture

The spacecraft itself has gone through initial conceptual designs and passed through a Phase 0 design study at ESA’s CDF. Currently there is a completed Phase A design, and the phase B design is ongoing at the time of writing. While the choice of components may still be subject to change, the top-level system architecture is assumed to be definitive. This is important as the

overall spacecraft architecture is very relevant to the Fault Detection, Isolation and Recovery (FDIR) design of the spacecraft. Therefore a short description of LUMIO's subsystems will be given in this section, based on the latest status described by Cervone et al. [4].

Payload
The LUMIO CAM is the payload observing micrometeroid impacts on the moon. It consists of two CCD201 detectors each containing 1024x1024 pixels. The observations occur both in the visible and near infrared spectrum, which is made possible by the beam splitting optics which divide the signal into two channels. This LUMIO CAM data will also be used to demonstrate optical navigation (line of sight navigation), although this is experimental and the baseline navigation system uses radiometric ranging and tracking [5].

Attitude and Orbit Control System
The LUMIO AOCS system, which is the focus of this thesis, relies on sun sensors, star trackers and an IMU for attitude determination. The attitude control occurs through the four reaction wheels. Since the CubeSat operates in the Lunar environment it does not include the typical CubeSat magnetorquer actuators. A Reaction Control System (RCS) is also present for desaturation (see Propulsion System). The full AOCS component list can be seen in Table 2.2. The architecture of the LUMIO AOCS can be seen in Figure 2.2. Note that the AOCS processor is shown here as a dedicated processor because it is run on the redundant processing unit, but could also be run on the OBC in case one of the onboard processors fails [4].

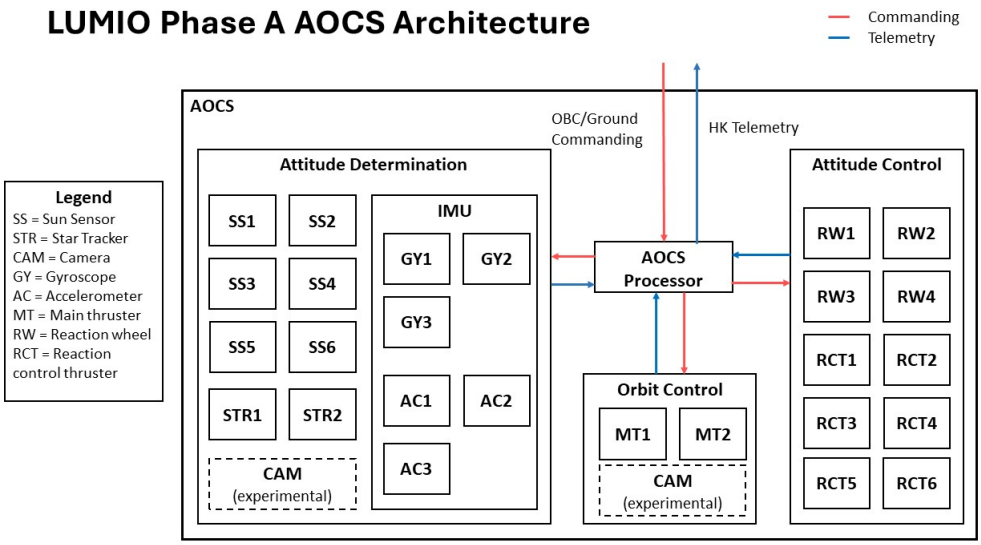


Figure 2.2: LUMIO Phase A AOCS Architecture

Sensors		
Type	Number	Supplier
Fine Sun Sensors	6	Lens R&D, MAUS
Star Tracker	2	Sodern, Auriga
IMU	1	ISISpace, SCG
Actuators		
Type	Number	Supplier
Reaction Wheels	4	Astrofein, RW25 SW50

Table 2.2: LUMIO Phase A ADCS sensors and actuators selection [4]

Propulsion System
Following the initial design iterations, the propulsion system of the LUMIO mission now refers to two separate systems. The first is the so-called ”main propulsion system” which will perform the orbital transfer from parking orbit to the final operational orbit and which will deliver the majority of the required ΔV for the mission. The second part of the system is the Reaction Control System (RCS) which performs the de-tumbling and the wheel desaturation [6].

Selection of the main propulsion system has not been finalised in the Phase A design: there are still two options to be considered in the Phase B study [4]: NanoAvionics EPSS and Bradford ECAPS HPGP, each of them slightly modified. Both are European mono-propellant units.

For the RCS system there are also two options remaining for further research in Phase B. These are the GomSpace 6DOF cold gas system and the (customised) Aurora Propulsion Technologies ARM water resistojet. The GomSpace option offers either six Reaction Control Thruster (RCT) with 1 mN or 10 mN thrust, a vacuum specific impulse $I_{sp,vac} = 50s$ and a wet mass of 802g [24]. The Aurora ARM resistojet also uses six thrusters with a thrust in the range of 0.6 to 4 mN, a wet mass around 1kg, and an $I_{sp,vac} = 100s$ [60].

Communication System
Communication between LUMIO and ground will use a direct-to-Earth approach while demonstrating a potential Inter-Satellite Link (ISL) during its mission. The communication system therefore consists of two COTS components: for the ISL radio the (customised) Syrlinks EWC31 ([58]) operating in the S-Band is chosen, while for the DTE link the C-DST radio operating in the X-Band and produced by IMT ([28]) is selected.

Data Handling System
LUMIO has three processing units: one ’main’ Onboard Computer (OBC) which controls the spacecraft, an Onboard Payload Data Processing (OBPDP) unit which processes and analyses payload data for transmission to ground. The third unit is a redundant unit which runs the AOCS algorithm, although this is not necessarily in need of a dedicated processor. The units are the IOBC of ISISpace [30] and the UniBap iX5.

Electrical Power System
The LUMIO Electrical Power System (EPS) in phase A is the ISISpace Modular EPS [31] which covers power conditioning, storage and distribution. This includes four battery packs of 45 Wh each. Power generation occurs using two solar arrays each mounted on a Solar Array Drive Assembly (SADA) which allows LUMIO to maintain precise lunar pointing during the science cycle while still generating power by tracking the Sun.

A rendered image of LUMIO in the Phase A design is shown in Figure 2.3.

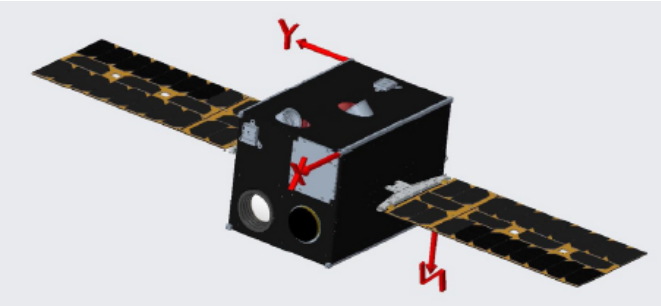


Figure 2.3: The LUMIO Spacecraft rendered in the Phase A design configuration. Source: Cervone et al. [4]

2.3. FDIR

In this section, a short review is performed of the state of industry practices in the field of Fault Detection, Isolation and Recovery (FDIR), and the state of the art. The most commonly used methods for (autonomous) fault detection are discussed, and those methods deemed most promising for future applications are summarised.

A note on terminology: faults, failures, detection and diagnosis
It should be noted that in this thesis, the terms fault and failure are often used. They are not interchangeable. Wander & Forstner [65] define a **fault** as "*an undesired deviation of at least one characteristic property of a system variable from an acceptable/nominal behaviour that leads to degraded overall system performance, malfunctions or failure of the system*". Their definition of a **failure** is "*a total cessation of a function via subsystem or the total system*". From this one can induce that a fault is a cause for a failure, whereas when a system performs in a fashion that is not acceptable (often due to a fault) it is labelled a failure.

As an example: a fault in the IMU could be a gyroscope with a drifting signal (due to radiation effects) which produces an erroneous spacecraft rate measurement. This fault, if left undetected and untreated could lead to an attitude determination and control failure. The spacecraft attitude can no longer be accurately measured, and therefore the desired attitude can no longer be achieved either. In the worst case, this failure can become a mission failure if power generation cannot occur due to the inability to point the solar panels, or if communication is inhibited by the antenna pointing away from the ground stations.

Also used repetitively are the terms fault diagnosis, fault isolation, and fault detection. They are often used in the same context, but are focused on different goals. In this thesis, Fault Detection (FD) is used when it concerns detecting that a fault has occurred, without specifically understanding which fault this is. Fault Isolation relates to classifying which fault has occurred and identifying the root cause, and the combination of Fault Detection and Isolation (FDI) is also referred to as fault diagnosis in literature. Fault recovery is only concerned with recovering the system after a fault has occurred, whether or not this fault has been isolated or not: FDR and FDIR can both occur.

2.3.1. Traditional Spacecraft FDIR

Since the dawn of spaceflight, FDIR has been a consideration in the design of reliable and available missions. Tipaldi and Bruenjes [62] state that goal of FDIR is in the first place to prevent mission loss, but also to prevent service loss as much as possible. It is considered a system-level discipline, and helps achieve a space mission Reliability, Availability, Maintainability, Safety (RAMS) objectives. These are defined by the European Cooperation for Space Standardization (ECSS) as follows [15]:

- **Reliability:** "the ability of an item to perform a required function under given conditions for a given time interval"
- **Availability:** "ability of an item to be in a state to perform a required function under given conditions at a given instant of time or over a given time interval, assuming that the required external resources are provided"
- **Maintainability:** "ease of performing maintenance on a product"
- **Safety:** "state where an acceptable level of risk is not exceeded"

The sources of the faults which the FDIR system has to deal with are categorised into three areas by Tipaldi et al. [62]:

- One or more failed components
- Incorrect control actions
- External disturbances

Furthermore, the types of failures that are considered in scope of the FDIR system are defined in the recently released SAVOIR FDIR Handbook [56] as:

- Hardware failures (random part failure, wear out, single event radiation induced failure, accumulated radiation induced failure)
- Software errors
- Operation errors (including human errors)

The most rudimentary approach to fault management was put in place in the 1960s: given a set of conditions which are fulfilled, the onboard computer performs a predefined set of actions to resolve the issue. If the issue is not solved after all programmed actions are taken, the spacecraft enters a Safe Mode and awaits human intervention [74].

Due to limited computational power on older space-grade processors, a large part of the onboard FDIR was done through hardware redundancy and hardware voting logic: the design included physical redundant components for the critical hardware components. This way fault detection through comparison was carried out. The added volume, mass, complexity and cost were traded against a higher robustness against failure and higher availability of the mission.

2.3.2. Overview of Model Based Methods

Most missions are not monitored 24/7 from the ground and some faults require rapid detection, isolation and a response to protect the spacecraft, hence a certain level of autonomy is required in the FDIR system.

The selection of a fault detection and isolation method typically depends on the mission objective, scale and complexity. Large communication satellites which require high availability might carry redundant hardware, or even be made fully redundant such as with the 'half satellite' concept once used by Thales Alenia Space. [43] However, this is expensive, adds complexity to the system and is not considered suitable for small, low cost missions such as CubeSats. Therefore smarter fault detection methods are employed: model-based fault detection and isolation.

The family of model-based methods for autonomous fault detection are based on many different principles such as simple signal analysis, analytical methods or knowledge based methods. The most interesting methods for consideration in this research are those used for fault diagnosis in non-linear systems. Here, many specific methods exist, as shown in (the non exhaustive) Figure 2.4 by Sanchuan Xu in 'A Survey of Knowledge-Based Intelligent Fault Diagnosis Techniques' [70]. A short summary of these methods is given below.

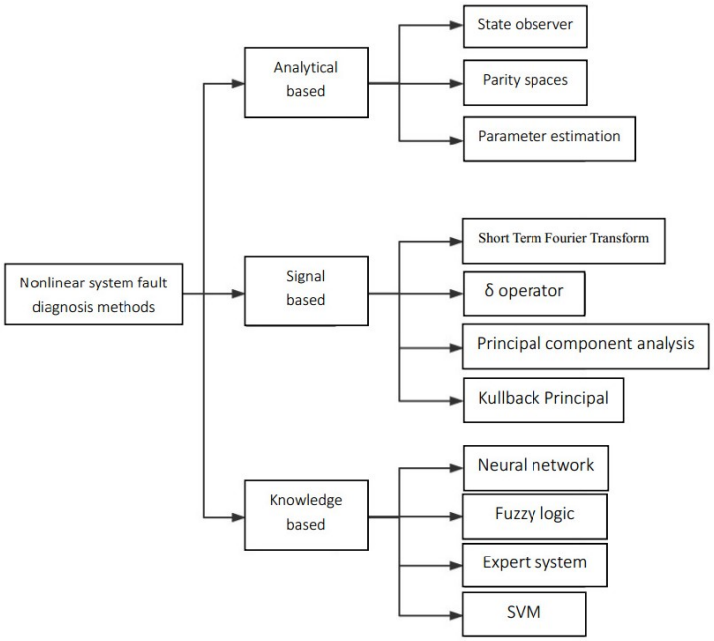


Figure 2.4: Nonlinear System Fault Diagnosis Methods. Source: Sanchuan Xu [70]

Signal Based Methods

Signal based methods, as the name implies, analyse the characteristics of a signal directly in order to detect and isolate faults. They have been used for decades and are simple, cost efficient and reliable. They are also able to detect and isolate faults at a very low level, which is useful in avoiding fault propagation due to erroneous sensor readings entering control loops. Their main drawback however is their limited use in dynamic systems, as the applicability is mostly in steady state conditions [45].

Signal processing methods can be as simple as fixed limit checking or magnitude checking of the signal, but also includes more dynamic methods such as variable limit checking, root mean square (RMS), Fourier transform, delta operator, principal component analysis and Kullback Principal [70]. They can be divided into three main categories according to Yin et al [71]:

- Statistical Methods
- Time Domain Methods
- Frequency Domain Methods

Analytical Methods

To counter the aforementioned issues of signal processing methods such as steady state applicability or limitations of analysing a single signal, the analytical methods can be used. These methods typically rely on obtaining an analytical or numerical model of the system in which faults should be detected. They are cost efficient and versatile, while also much more applicable in dynamic and non-linear systems. However a key issue with these methods is the need for highly reliable and accurate mathematical models, which lead to a large amount of computational resources required and some issues with numerical convergence [70]. Access to these models is typically limited and the expertise required to implement them is not always available to smallsat developers.

Within this set of analytical model-based FDIR methods, the most commonly used is the parity space method [62]. This method relies on generating an estimated system output based on

an observed system input, and then comparing the estimated output to the observed output, which leads to the residual and potential fautl detection. This generated residual can then be checked for fault features to perform isolation, given that the features and corresponding faults are known to the system. Typically a residual should be 0 or under a certain threshold to account for noise. Based on the comparison of multiple residuals a fault can be detected and isolated, even in the presence of noise, uncertainties and disturbances [70]. This method is widely used in spacecraft FDI and is considered a mature field according to Wander & Förstner [66].

Another analytical method is the state observer which is similar to the parity space method in that it uses an analytical model to estimate variables. However, the state observer approach uses the system output to estimate the system internal state, which it then compares to the observed system state [70]. It is known to be applied where the expected system states are well-defined and where mostly linear behaviour can be expected [48].

The third analytical method is parameter estimation, where values in a mathematical system model are estimated using different approaches such as least-squares estimation, maximum likelihood estimation and Bayesian estimation [71]. This method suffers from noise sensitivity however and requires high accuracy models [62].

Knowledge Based
Knowledge based systems use experience and expert knowledge to detect and isolate faults. These methods are very useful where one cannot describe the system with an analytical model [70]. They are also considered cost efficient, and highly accurate for classifying discrete events. However, they are computation heavy and until recently some knowledge based methods such as neural networks were considered very hard to fully verify and validate. In January 2021, ESA and the German Research Center for Artificial Intelligence (DFKI) established ESA_Lab@DFKI [1] demonstrating an increased willingness to incorporate these methods in space missions.

Typical knowledge based methods used in fault detection and isolation include Artificial Neural Networks (ANN), Fuzzy Logic, Expert Systems, Support Vector Machines (SVM), Cognitive Automation, and Dempster Shafer Evidence Theory. These methods all rely on expert knowledge or available data to train the systems on. In the case of spacecraft FDI, typically one uses operational data including faults from similar missions to train the systems, or simulated data based on knowledge about fault characteristics. The survey conducted by Tipaldi and Bruenjes [62] notes that these methods show a lot of potential for future missions, specifically the cognitive automation, SVMs and ANNs.

2.3.3. FDIR in Deep Space CubeSats

CubeSats are considered highly unreliable compared to their larger, more expensive counterparts. Langer and Bouwmeester [35] showed that dead-on-arrival and infant mortality effects dominate the reduced reliability: an overall CubeSat reliability following successful deployment of 75.62% to 87.09% is estimated with 95% confidence. That reliability drops to 58.94% to 73.24% (95% confidence) after 100 days in orbit. The research also showed that CubeSats in Low Earth Orbit (LEO) are less susceptible to wear out than geostationary satellites. This could be traced back to environmental reasons or to a lack of experience in developing these kinds of missions. In both cases, it does not bode well for deep space CubeSat reliability,

[1] https://www.esa.int/Enabling_Support/Preparing_for_the_Future/Discovery_and_Preparation/Artificial_intelligence_in_space#

which face harsh radiation environments as well as mission profiles and requirements which are not typical for Earth-orbiting CubeSats. Adding to this, deep space CubeSats are expected to be much more reliable due to the often unique launch opportunities they are given, meaning that a failed CubeSat cannot easily be replaced.

Current State of Practice
This implies the need for a capable FDIR system, where nowadays CubeSats perform little to no FDIR onboard [52]. The current practice for health management in Earth-orbiting CubeSat missions includes simple mechanisms such as [62]:

- **Watchdog timers**: requires a response from the OBC within a set time, otherwise it will assume it has encountered an issue and will reset it externally
- **Health monitoring**: sensors (voltage, temperature...) are used to ensure certain parameters do not exceed predefined limits, allowing for preventive actions to be taken before permanent damage is caused to the spacecraft
- **Information error detection**: methods such as the Cyclic Redundancy Check (CRC) are used to detect bit flips in memory or communication, typically caused by charged particles (SEU)

While the aforementioned ideas provide a basic level of protection against events processor malfunctioning, temperature anomalies, EPS faults, or charged particles strikes in the memory, they are not able to detect more complex faults with disastrous consequences such as inaccurate attitude readings or actuator faults. These complex issues require a model-based analysis of data in order to allow detection and isolation.

Model Based FDIR in CubeSats
While model-based FDIR could be advantageous to all CubeSat missions, the cost and effort associated with it as well as the added complexity in the design process leads to the belief that it will be most suitable for these deep space missions. As far as can be deduced from literature, only one succesfully flown spacecraft with onboard model-based fault detection remains to date: the Remote Agent Experiment on NASA's Deep Space 1 spacecraft [1]. While the experiment was considered successful, the modelling complexities and spacecraft constraints were considered too substantial at the time.

Another attempt was made at onboard model-based fault detection with the launch of 6U exoplanet hunting CubeSat ASTERIA (Arcsecond Space Telescope Enabling Research In Astrophysics) by JPL. The model-based fault detection experiment was planned to be performed onboard near the end of the mission [52]. The CubeSat, launched from the International Space Station and operational in LEO, used the popular Commerical Off-The-Shelf (COTS) Blue Canyon Technologies XACT attitude control system. A test was planned to demonstrate its model-based FDIR system with a number of seeded faults at the end of its mission. Given the risk to the mission, the demonstration could only be performed after completion of the primary and three extended missions of ASTERIA. However, perhaps slightly ironically, contact with the mission was lost during one of its three extensions and the fault detection experiment was not conducted. The cause of the mission failure is not known at this time, but it emphasises the need and value of more advanced CubeSat fault management systems.

Additionally, the relatively simple CubeSat architecture allows for the demonstration of promising, more advanced FDIR methods which may not have been implemented on large scale missions due to the lack of flight heritage or validation such as artificial intelligence methods.

3

LUMIO Fault Analysis

In order to detect and isolate faults in the spacecraft AOCS system, the design team should try to understand where potential faults could originate and what their effects are on the spacecraft as much as possible. The ECSS standards for Space Product Assurance [14] and the SAVOIR FDIR Handbook [56] provide guidelines in using the Fault Tree Analysis (FTA) and Failure Mode Effects and Criticality Analysis (FMECA) as a structured approach. In this chapter, the results of the LUMIO AOCS Fault Tree Analysis and FMECA are discussed. The outcomes of this analysis will serve as a starting point for the design of a model based fault detection system.

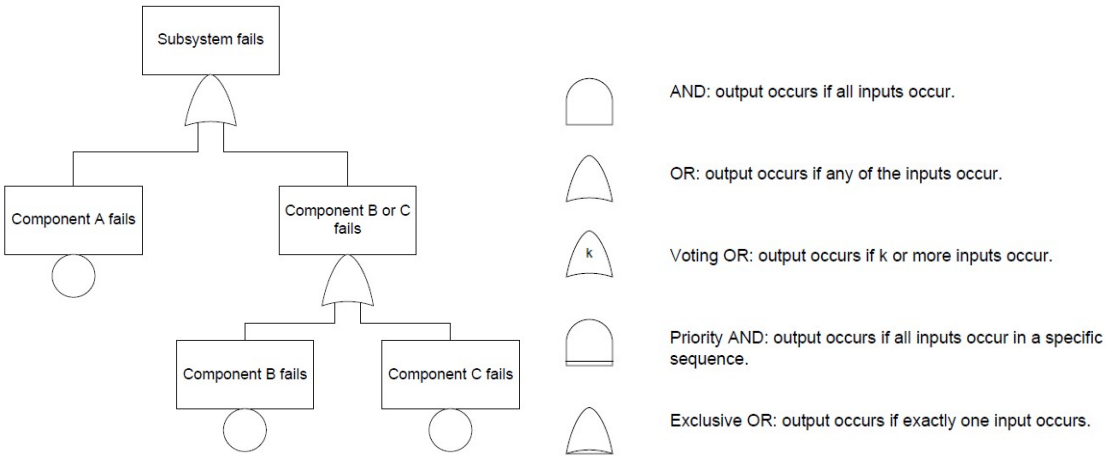
First the FTA is performed in section 3.1, followed by the FMECA in section 3.2. Then the requirements and constraints for the LUMIO FDIR system are explored in section 3.3. The trade study for model based fault detection concepts is presented in section 3.4, which is then critically reviewed in section 3.5.

3.1. LUMIO Fault Tree Analysis

As a starting point in identifying the critical and feasible fault scenarios for LUMIO, a Fault Tree Analysis (FTA) is generated. This FTA will focus on the AOCS system of LUMIO, while taking into account critical dependencies on other subsystems such as the power (EPS) and communications (TT&C) modules. The FTA will be performed separately for all operating phases of LUMIO, as the environment, risks and goals are different across the four phases.

3.1.1. The FTA Methodology

The FTA is a deductive analysis where a top level failure is identified for a system, based on which basic faults are then generated which would lead to this top level failure. It is described in a tree form with logic gates, as shown in Figure 3.1. Although a virtually infinite number of basic faults could lead to a top level event occurring, not all should be included in the FTA. The events which should be included are credible faults within the system boundaries [9]. The SAVOIR FDIR Handbook [56] dictates that the spacecraft FTA only concerns itself with hardware errors (random, wear out and radiation induced), software errors, and operational errors (human or other). Faults such as early life failures, vibration/Electromagnetic Compatibility (EMC)/space debris induced failures and outgassing effects are considered out of scope by the SAVOIR handbook and thus also in this thesis.



(a) Generic example of a Fault Tree Analysis **(b)** Fault Tree Gate Symbols

Figure 3.1: Demonstration of the Fault Tree Analysis methodology. Source: Bidner [2]

3.1.2. LUMIO FTA

The FTAs for LUMIO were developed based on a number of feared top level events per mission phase. These are shown in Table 3.1 and are assigned an identifier according to the logic: ”Subsystem.Error.” For example, AOCS.NAV refers to a feared event in the AOCS system where there are no or inaccurate navigation services. The subsystems and functions considered are Attitude and Orbit Control System (AOCS), Telemetry, Tracking & Command (TT&C), Electrical Power System (EPS), payload Camera (CAM), and Deployment (DEP).

These feared events were defined based on the Generic AOCS/GNC Techniques & Design Framework for FDIR (GAFE) credible failure lists [39], taking into account the guidelines set out by the SAVOIR FDIR Handbook [56] which limits the credible top-level events to hardware, software and operation errors as described in section 2.3. From there, the top-level functions to be performed by the LUMIO AOCS in each phase were determined: for phase 1 (parking and detumbling) the key AOCS function is clearly detumbling, and therefore a major feared event for this phase is DEP.DET ’detumbling not performed’, as shown in Table 3.1. Performing this function requires attitude determination and control, and electrical power, which traces back to ’generic’ fault trees which apply to the all mission phases: AOCS.NAC ’no attitude control’ and EPS.NPA ’no power available’.

Using a similar reasoning, the top level feared events for the other phases, shown in Table 3.1, were determined. As phase 2 (transfer) and 4 (disposal) consist of manoeuvres, the feared events mainly relate to failure to perform such manoeuvre (AOCS.TNP ’transfer not performed’). The feared events for phase 3 (operations) are focused on something going wrong in obtaining or processing the desired scientific output, or succesfully relaying it to ground (CAM.NSI ’no scientific imaging’, CAM.NSP ’no scientific product’, and AOCS.SK ’unable to keep station’).

Aside from the phase-specific feared events, some feared events are always relevant: lack of navigation, communication, and collision avoidance are required at all times. Arguably the spacecraft can detumble without ground communication and navigation but this does not allow transitioning to the next phase and is therefore also considered a feared event during the phase.

Feared Event			Phase			
ID	Name	Description	1	2	3	4
AOCS. NAV	No or inaccurate navigation	Navigation is required for all phases to ensure correct manoeuvre and validation of experiments	■	■	■	■
AOCS. NAC	No attitude control	Full 3 DOF attitude control is critical in ensuring solar array and antenna pointing during all phases	■	■	■	■
TTC. DTE	No communication with ground	Ground communication is mission critical during all phases	■	■	■	■
EPS. NPA	No power onboard	Power should be available at all times to operate any subsystem	■	■	■	■
DEP. DET	Detumbling not performed	Detumbling following release from the launcher is critical to establish ground contact and generate power	■			
AOCS. DES	Reaction wheels not (fully) desaturated	Reaction wheels should be desaturated such that sufficient momentum capacity is available for next transfers/pointing manoeuvres	■	■	■	■
AOCS. TNP	Transfer manoeuvre not correctly performed	The accurate execution of the transfer manoeuvre is necessary to put LUMIO in the required orbit for science and operations		■		■
CAM. NSI	No imaging of impact flashes	No imaging leads to mission failure			■	
CAM. NSP	No scientific product to ground	No scientific products lead to mission failure			■	
AOCS. SK	Spacecraft unable to keep station	The operational orbit cannot be maintained			■	
AOCS. COL	Collision with other spacecraft/object	Collision leads to mission loss	■	■	■	■

Table 3.1: LUMIO FTA Feared Events List (Phases: 1 Parking, 2 Transfer, 3 Operations, 4 Disposal)

3.1.3. FTA Results

For each top level event shown, a fault tree structure was created. Due to the number of fault trees they will not all be displayed here, but the full set can be found for the different feared events and mission phases in Appendix A. For clarity, a single example FTA will be shown and discussed in this section. The Inertial Measurement Unit Fault (AOCS.IMU) tree was chosen, shown in Figure 3.2, due to its relevance to the work performed later in this thesis. As can be seen here, the top level event identified is an IMU fault, meaning no angular rate readings are available to the AOCS system. This IMU fault top level event is one of the basic events in the No Attitude Control (AOCS.NAC) tree, which is a feared event in all mission phases.

The top-level IMU fault is caused by one of two causes: either the lack of IMU data being supplied, or the wrong data being supplied. The lack of data was deemed to be possible through a self-test failure, communication failure, out of range temperatures (which invalidates the reading externally, i.e. different to self-testing), and a complete unit shutdown. The erroneous data can be caused by erratic behaviour in the gyroscope signals, stale data, or drifting/biased/uncalibrated measurements. The circles underneath these events means they are considered basic events and no further root cause is investigated. The causes for these basic events are many and unpredictable, and could relate to software bugs, Single Event Upset (SEU), radiation lifetime effects (Total Ionising Dose (TID)), wear-out, or hardware failures.

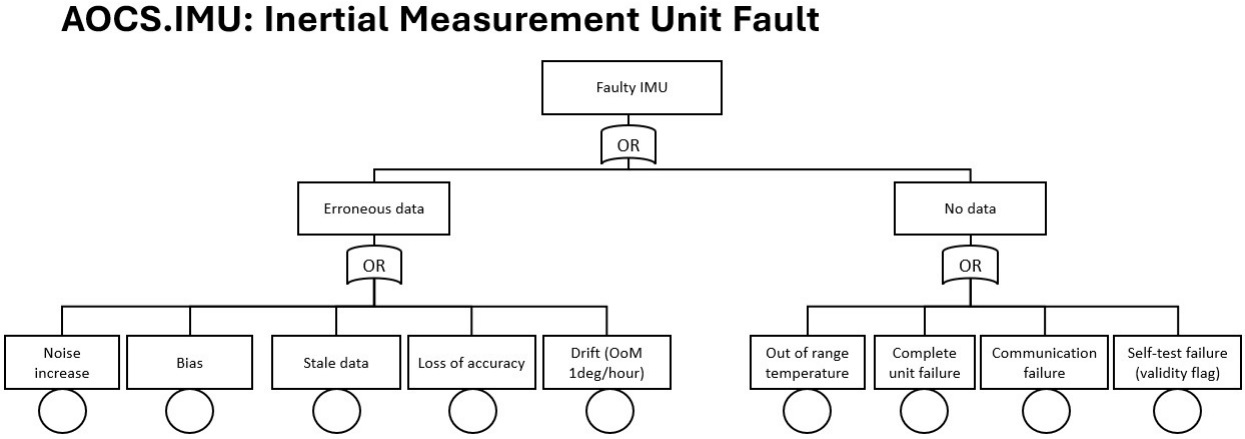


Figure 3.2: Fault Tree for LUMIO IMU Fault

The results of the full FTA are used to complement the FMECA in the next section. However, from the fault trees alone some conclusions can be drawn, discussed below.

High Interdependency Between Subsystems

Although not surprising, the FTA shows a very high degree of connection between subsystems, as well as a lot of dependencies. The AOCS system depends in a very large part on the proper functioning of the EPS and Onboard Computer (OBC), and in a lesser part on the Telemetry, Tracking & Command (TT&C) and thermal control subsystems. Therefore, a lot of major faults in these systems can easily propagate and manifest themselves in the AOCS system. It emphasises the need for a reliable integrated FDIR system for the entire spacecraft.

Little Single Point Failures

It is also clear from the FTA that the LUMIO phase A design has already actively considered and incorporated some measures to reduce the likelihood of these undesirable events. This is likely in part thanks to the previous work done by the LUMIO team in phase 0 and the work performed by Gelmi [22]. Most fault trees do not lead to a single point failure, and where it

does they are often due to unlikely external factors (excessive environmental disturbances) or they are acceptable risks on tried & tested units such as the Main Thrusters (MT).

Critical Sensor Fault Effects
Sensors provide a myriad of data to the AOCS algorithm, which is used to accurately control the spacecraft. The FTA analysis shows the detrimental effect sensor faults can have if left undetected and untreated. The additional challenge is that certain faults in sensors are incredibly difficult to detect without having multiple identical units to compare the sensor output to (n-modular voting). In CubeSats, which lack these multiple redundant units due to mass, volume or cost considerations, these faults are especially likely to propagate and cause high level undesirable effects. Knowing that the measured data is accurate and reliable and invalidating the data when it is not is a key step in improving the reliability of the spacecraft.

3.2. FMECA

The Failure Mode Effects and Criticality Analysis (FMECA) is an inductive fault analysis tool in which a specific failure mode of a unit is hypothesised, and then its effect on the overall system are analysed [14]. This analysis also describes the criticality of a failure mode, which is defined through the Criticality Number (CN):

$$CN = PN \times SN \tag{3.1}$$

Here the Probability Number (PN) and Severity Number (SN) are assigned to each failure mode by the analyst and/or experts and range from 1 (extremely remote likelihood or minor mission degradation) to 4 (probable event or catastrophic effects). These criteria for each level of probability or severity is quantified by the ECSS standard [14]. The definition of the PN is shown in Table 3.2 and the SN definition is shown in Table 3.3, with the sublevels of the SN2 being added from the US Nuclear Regulatory Commission Fault Tree Handbook [9].

Level	**Limits**	**PN**
Probable	P>0.1	4
Occasional	0.001 <P <0.1	3
Remote	1E-5 <P <0.001	2
Extremely remote	P <1E-5	1

Table 3.2: FMECA Probability Number quantified as defined in the ECSS-Q-ST-30-02C standard [14]

Severity category	**Severity level**	**Dependability effects**
Catastrophic	1	Failure propagation
Critical	2	Loss of Mission
	2a	A second fault causes transition to FDIR level 3 (system control SW affected)
	2b	A second fault causes transition to FDIR level 4 (safe mode requried)
	2c	A fault where the effects depend upon the situation at hand (e.g. fault in a non-active, redundant unit)
Major	3	Major mission degradation
Minor	4	Minor mission degradation

Table 3.3: Failure mode severity levels as defined in the ECSS-Q-ST-30-02C standard [14], with additions from the US Nuclear Regulatory Commission Fault Tree Handbook [9]

In the case of LUMIO this FMECA analysis was performed for the Phase 0 design in the thesis of Gelmi [22]. Based on this work, the Phase A FMECA is performed in this thesis for the AOCS only, as that is the focus of this research. The resulting full FMECA is too extensive to be discussed here and can be found in Appendix B for reference. A single example FMECA item will be discussed in the following section.

3.2.1. Scope of the FMECA

The FMECA could theoretically be expanded to an infinite number of scenarios which lead to failures. However, in this thesis the focus is on the LUMIO AOCS and its units, and the credible failures (hardware, software, operational) mentioned before. The FMECA is performed for each failure scenario in each LUMIO mission phase, although often times the phases are grouped into the dynamic phases (2 transfer, 4 disposal) and the static phases (1 parking, 3 operations) as they exhibit similar failure effects and mechanics. For example, a main thruster failure leads to the same effects (inability to perform manoeuvre) in both phase 2 and 4, and is of much higher severity in these phases compared to a similar failure occurring in phase 1 or 3 assuming it can be recovered or compensated. It should be noted that SN and PN scores are assigned qualitatively in this analysis due to the scope of this thesis as well as the lack of available quantitative data relating to probabilities or impact. It is recommended that in more advanced design stages, these scores are revised by the LUMIO system engineers and AOCS experts.

3.2.2. FMECA Results

In total 53 failures were considered across 9 different units of LUMIO: star trackers, reaction wheels, IMU, sun sensor, main thrusters, reaction control thrusters, AOCS processor, OBC, and Solar Array Drive Assembly (SADA). For each unit some credible faults were analysed per mission phase. The failure effect on the unit, subsystem and system were analysed and the detection method as well as compensation method (if available) were documented for each failure mechanic. Based on this, the severity and probability were assigned and the criticality was calculated. This criticality will serve as the selection criteria for critical items, i.a.w. the Criticality Matrix in Table 3.4. Here, the criticality number determines if the item is critical or not, with scores of 6 or higher being considered critical (orange/dark area) as well as those items with the lowest probability but a severity number of 4 (catastrophic).

Severity	**SN**	**PN**			
		1	**2**	**3**	**4**
Catastrophic	**4**	4	8	12	16
Critical	**3**	3	6	9	12
Major	**2**	2	4	6	8
Negligible	**1**	1	2	3	4

Table 3.4: Criticality Matrix. Source: ECSS [14]

Example FMECA Item

For clarification, a single example item of the FMECA is shown and discussed in Table 3.5: the IMU.08 'drift in measurements' fault. This fault was previously seen in the fault tree example in Figure 3.2 as one of the events leading to no IMU reading being available. The assumed failure in the FMECA is a drift in the rate measurements in the order of 1 degree per hour, based on the credible IMU faults listed in the GAFE methodology document [44]. It can be seen that the failure leads to inaccurate relative attitude measurements, but that the end effect is different for each mission phase. For phase 3 (operations) the inaccurate pointing will lead to a scientific product of lower quality, or the inability to continue the experiment as the LUMIO-

CAM is not pointed correctly. For phase 1 (parking) the effect is that detumbling cannot be performed accurately and the spacecraft will remain in a tumbling state, leading to potential mission loss due to no communication and deployment. Whether or not the mission is fully lost depends on the severity and direction of the tumble and the ability to recover the IMU. For phases 2 (transfer) and 4 (disposal) the risk is that with inaccurate IMU measurements the pointing accuracy is reduced and the manoeuvres are performed inaccurately. Depending on how the star tracker and sun sensors are able to correct this error, the inaccurate transfer could lead to mission loss.

ID	Block	Function	Assumed Failure	Phase	Failure Effect	Detection	Compensation	SN	PN	CN
IMU 08	IMU	Measure angular rate around 3 axes	Drift in measurements (1 deg/hour or 0.01G /hour)	3	Inaccurate relative attitude measurement ->Reduced pointing accuracy ->Lower quality or no science	Cross check angular rates with star tracker data, check acceleration with position, velocity, time	Power cycle IMU	2	2	4
				1	Inaccurate relative attitude measurement ->Risk of detumbling not being performed ->Mission loss			3	2	6
				2, 4	Inaccurate relative attitude measurement ->Reduced pointing accuracy ->Inaccurate transfer performed			3	2	6

Table 3.5: Example of FMECA Item - IMU.08 IMU Measurement Drift

The- traditional detection method is determined to be through comparison (cross check) of the IMU rates with other available rates such as from the star tracker depending on the accuracy required. Compensation can occur through power cycling of the IMU, which may include re-calibration depending on the component specifications. The scores are assigned based on the failure effect, and the probability of the failure occurring is designated as 'remote', which is a score of 2. This shows that the failure is not critical during the science phase, but is critical during the parking and transfer phases as well as the disposal phase. It is therefore important that this failure can be detected and compensated accurately.

Critical FMECA Items

The results of the FMECA is a list of 20 critical faults out of the 53 analysed, according to the criteria defined by the ECSS standard as shown in Table 3.4. Eighteen of these faults have a CN of 6, meaning they are considered highly critical but only just. A small lowering of the severity or likelihood would bring them to an acceptable level of criticality. The two items with a CN of 9 are the RCS and main thruster valves being stuck in the open position (FMECA ID RCS.04 and MT.04 respectively). Their relatively high likelihood over repeated operation in space combined with their severe consequence means they are highly critical. In order to reduce their criticality, the probability should be lowered by ensuring the use of systems with long-standing flight heritage in comparable missions, and testing the thrusters in representative vacuum conditions at low temperatures for different firing profiles. The severity of these two failures could be reduced by means of a redundant thruster branch which can be isolated from the faulty thruster branch. However, in a CubeSat the latter option is likely not possible due to mass and volume constraints.

The criticality of all 53 analysed FMECA items is shown in Table 3.6. It can be seen here that the majority of critical faults are critical due to their high severity, not due to their excessive probability. Also notable is that there are no faults with the highest severity or probability. This reflects the previous design and analysis work done in the LUMIO CDF and Phase 0 designs as well as the FMECA by Gelmi [22]. As can be seen from the large number of items with a SN of 3 and PN of 1, the faults with a critical severity have been made of extremely low probability in prior work through the introduction of redundancy or the adoption of flight-proven systems.

Severity	**SN**	**PN**			
		1	**2**	**3**	**4**
		Extremely Remote	**Remote**	**Occasional**	**Probable**
Catastrophic	**4**				
Critical	**3**	STR.03 STR.04 STR.08 RW.03 RW.04 RW.05 RW.06 RW.07 RW.08 IMU.01 IMU.02 IMU.03 IMU.04 IMU.05 RCS.01 MT.01 MT.07 MT.08 AOCS.01	STR.01 STR.10 RW.01 STR.05 STR.11 RW.02 STR.09 STR.12 IMU.06 IMU.07 IMU.08 RCS.02 RCS.03 MT.02 MT.06 OBC.01	RCS.04 MT.04	
Major	**2**	STR.06 SADA.01	STR.02 RW.09 MT.03 SADA.02 SADA.03 SADA.04	RCS.05 MT.05	
Negligible	**1**	SS.01 SS.02 SS.03	STR.07 SS.04 SS.05		

Table 3.6: Criticality Matrix for LUMIO AOCS (fault IDs refer to the FMECA IDs in Appendix B)

3.2.3. Fault Register

The identified critical faults are summarised in a Fault Register, which is included in Appendix C. This Fault Register also indicates what the possible detection method would be in this case. Only two of the twenty faults can be detected through 'simply' cross checking variables and system states (e.g. power or communication status). Detection of twelve of the critical faults relies on some form of model-based fault detection, requiring complex nonlinear dynamic models of the spacecraft, as well as thermodynamic models of the propellant tank and spacecraft. Finally the five other critical faults are presumed be detectable through signal processing, by measuring running average, variance and other signal characteristics. The faults are also assigned codes within the Fault Register which will help isolate specific faults and instigate appropriate recovery actions in the system.

Continuing the same example from the FTA shown in Figure 3.2 and the FMECA item shown in Table 3.5, the corresponding critical fault in the fault register can be seen in Table 3.7.

ID	Block	FMECA ID	Fault Name	Symptoms	'Simple' Cross Check Detection	Signal Detection	Model-Based Detection
F26	IMU	IMU.08	IMU reading inaccurate (drift, bias..)	IMU readings do not match absolute attitude readings and actuator inputs			

Table 3.7: Critical Fault F26 as an example from the critical fault register in Appendix C

The main area of application for the model-based fault detection methods is in the sensor reading faults. Uncalibrated sensors, drifting signals, biases, steps, outliers and noisy sensors require more intelligent detection methods. This is due to the fact that, as a first step, sensor data is assumed to be truthful and used for fault detection in the rest of the system. However, when the sensors themselves start to provide inaccurate data there is no way to know unless there are one or more duplicate measurements available. In the fault register, half of the critical faults (10 out of 20) are related to the incorrect or unavailable sensor data, all of which are critical to the LUMIO mission. This shows there is a need for an accurate, efficient and accessible model-based fault detection method for advanced CubeSat mission such as LUMIO.

3.3. FDIR Requirements

Having detailed the system architecture and potential faults as well as their effects, a set of preliminary FDIR constraints can be generated, which can be translated into a set of requirements. These requirements will ensure the chosen FDIR methods can be applied to the CubeSat platform, and will guide the trade study and its criteria in the next section of this chapter.

The LUMIO mission design is guided by a set of mission requirements, described in the Mission Requirements Document (MRD) [19] as well as a set of system requirements which are described in the System Requirements Document (SRD) [20]. From these requirements, those deemed especially relevant to the FDIR system were selected to form a basis for the FDIR system requirements. These FDIR requirements and constraints will determine the trade criteria in the next section, the full requirements set can be found in Appendix D.

Based on LUMIO requirements, 35 FDIR requirements were identified across four categories:

- **GEN** - General
- **FUN** - Functional: what should the FDIR be capable of doing?
- **PER** - Performance: how well should FDIR functions be accomplished under certain conditions?
- **INT** - Interface: how should the FDIR interact with other systems and functions?

Due to the scope of this thesis, only a few of the most relevant requirements out of the 35 FDIR requirements will be discussed here, a full overview can be found in Appendix D.

3.3.1. General Requirements

The general FDIR requirements, categorised by the 'GEN' identifier, most relevant to this thesis are shown in Table 3.8. The most important limitations to the system are that the FDIR system should be computationally lightweight such that no additional processors are required, that the system complexity is not increased through introduction of new hardware or resource requirements, and that verification is required before integration in the spacecraft. The FDIR system shall also make use of the onboard telemetry in order to accurately detect faults. This is especially relevant to the AOCS system, where valuable information can be gathered from cross-checking telemetry between units.

ID	Description	Rationale	Verification
GEN-010	The FDIR system shall not require any additional processors in the spacecraft	The FDIR system should not require any additional hardware (sensors, processors, actuators) to be implemented, and should consist of only software which can be run on any of the LUMIO processors.	Inspection
GEN-020	The FDIR system shall be verified before integration into the spacecraft	The FDIR system must be tested in order to ensure correct functioning before implementation on LUMIO hardware	Demonstration
GEN-030	The FDIR system shall not increase LUMIO system complexity	The FDIR system should not make the CubeSat satellite architecture more complex by requiring additional hardware (sensors, actuators, processors), power, or propellant	Analysis
GEN-070	The FDIR system shall be able to access the housekeeping telemetry onboard for fault detection and isolation	All housekeeping data onboard should be accessible for the FDIR system to ensure maximal coverage in fault detection and isolation	Demonstration

Table 3.8: LUMIO FDIR General Requirements most relevant to this thesis

3.3.2. Functional Requirements

The functional requirement most relevant to this thesis is shown in Table 3.9. FUN-010 dictates the possible fault sources which are in scope for a fault detection and isolation system. This is important as in the verification activities, representative faults are simulated based on the expected fault types and their features.

ID	Description	Rationale	Verification
FUN-010	The FDIR system shall detect and isolate hardware faults caused by: random faults, wear out, radiation	As defined by the SAVOIR FDIR HB-003 Iss2 rev0 to be in scope of the FDIR system	Analysis

Table 3.9: FDIR Functional Requirements LUMIO most relevant to this thesis

3.3.3. Performance Requirements

The four most relevant performance requirements are shown in Table 3.10. It can be seen here that the key performance drivers are focused on accurate fault detection, as well as avoidance of false alarms. It should also be noted that the FDIR system has a limited available onboard storage and working memory to perform its tasks, again emphasising the need for a highly lightweight system. Due to the early stage of the mission design the exact numbers are not yet determined, hence the 'TBD'. For the fault detection rate and the false alarm rate also no exact numbers are assigned yet as these will depend on the threshold selection and tuning during the design process. In general, missed fault detections should be avoided at all costs, while minimising the false alarms triggered in the system.

ID	Description	Rationale	Verification
PER-010	The FDIR system shall require at most TBD GB of onboard RAM	The FDIR system should not limit the computational resources available for nominal operations	Demonstration
PER-020	The FDIR system shall be storeable in at most TBD GB of onboard non-volatile memory	The memory available onboard is required for payload and housekeeping data storage	Inspection
PER-030	The FDIR system shall catch faults with a fault detection rate (FDR) of TBD %	The FDIR should be able to catch as many faults as possible	Analysis
PER-040	The FDIR system shall have a false alarm rate (FAR) as low as possible, and of no more than TBD % based on test data.	The FDIR should avoid unnecessary interruption of nominal, fault-free operations.	Analysis

Table 3.10: FDIR Performance Requirements LUMIO

3.4. Trade Study

In chapter 1 the research question focuses on novel model-based fault detection methods, and the literature study performed into available model based FDIR methods [10] provides plenty of design options. The task at hand is to determine which of the model based methods is most suitable for development of a novel fault detection method. To do this, the space systems engineering process for a trade off [47] is followed, with a design option tree (DOT), elimination of infeasible concepts and selection of the winning concept in a trade off using criteria based on the requirements outlined in the previous section. The weights for the criteria are determined through Analytical Hierarchy Process (AHP), and scoring is performed using the classical trade off method. Confirmation is given by using a different system to assign weights, as well as by comparison to the results of the trade off using a Pugh Matrix for scoring.

For brevity, only the DOT and outcome of the trade off will be discussed in this section. The full concept exploration and criteria determination process is described in Appendix E.

3.4.1. Design Options

The first step in selecting the appropriate method(s) for FDIR onboard LUMIO is to gain a solid understanding of the available options and their characteristics. A detailed summary of these methods was reviewed in the literature study [10], and some methods are shortly discussed in chapter 2. A more complete Design Option Tree (DOT) is shown in Figure 3.3, with the concepts which are considered feasible and included in the trade study indicated in green/bold. The options are discussed one by one in Appendix E along with a short motivation as to why they are considered feasible or infeasible and included in the trade off, or not.

In general, concept feasibility is linked to general characteristics of each design option and if they meet the previously discussed requirements. For example: hardware redundancy requires additional hardware, which already violates GEN-030 which states no additional hardware shall be introduced for FDIR purposes. Most concepts were eliminated due to the clear violation of FDIR requirements, or due to the limited applicability in CubeSat missions: expert systems are an example of this. The need for experts and the lack highly specific nature of the system is contradictory to the nature of the CubeSat platform.

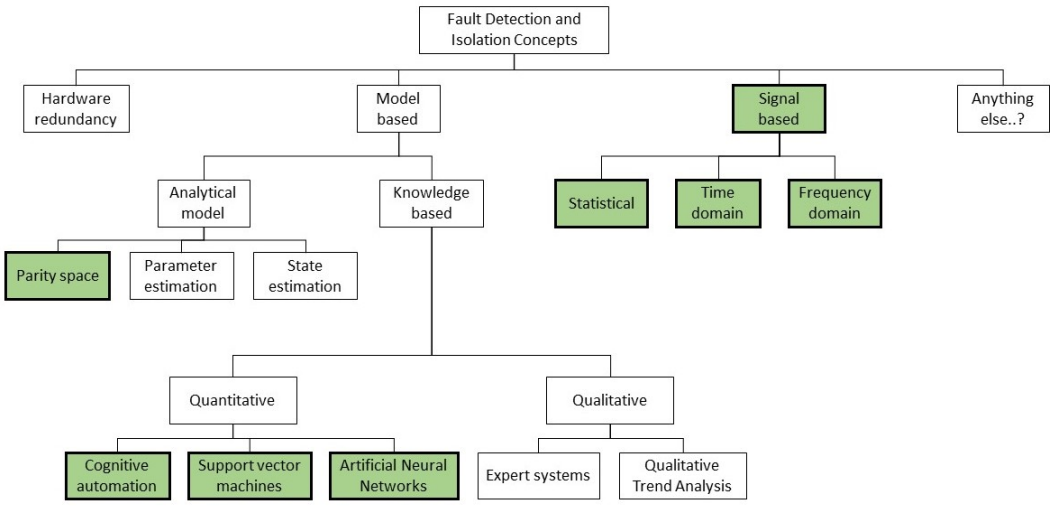


Figure 3.3: LUMIO FDIR Design Option Tree (green/bold are those concepts selected for trade off)

3.4.2. Trade Criteria and Weights

To perform an informed trade study, the criteria for the trade off should be well defined, mutually exclusive and based on the requirements defined in this chapter. Additionally, these criteria should be weighted such that their influence in the chosen solution reflects their importance in the design. Determination of these weights can be influenced by the subjectivity of the analyst, and therefore it is critical that the process is transparent and the reasoning documented.
The chosen criteria to trade of the FDIR method, with traceability to the relevant requirements are summarised below. The detailed traceability to the requirements and the rationale for selecting criteria can be found in Appendix E.

- Fault detection accuracy
- System complexity

- Model complexity
- Verification and validation feasibility
- Required computational resources
- Software based (killer)
- Thesis feasibility (killer)

It should be noted that aside from the five criteria identified from requirements, two killer requirements are taken into account. The first, software-based, is a reflection of the GEN-030 requirement which states no additional hardware can be introduced. This killer requirements ensures no design options are selected based on hardware voting or similar systems. The second, 'thesis feasibility', does not come from the FDIR constraints but rather from the limited scope and timeframe of this thesis. This is to ensure the chosen design option can be developed within the required timeframe and using the available resources and expertise.

The Analytical Hierarchy Process (AHP) was used to determine the weights described in Table 3.11. As can be seen the fault detection accuracy and verification and validation feasibility are considered most driving in selection of the system. The full ranking and AHP process is described in Appendix E.

Criteria	**Weight (AHP)**
V&V feasibility	1.300
Fault detection accuracy	1.387
System complexity	0.607
Model complexity	0.410
Computational resources	1.000

Table 3.11: Criteria weights determined using AHP

Criteria	**Weight**	**Signal processing**	**Parity Space**	**Cognitive Automation**	**Neural Networks**	**Support Vector Machine**	**Dynamic Bayesian Network**
Verification & validation feasibility	1.300	100	110	90	90	90	80
Fault detection accuracy	1.387	80	90	110	110	100	100
System complexity	0.607	110	90	90	100	100	90
Model complexity	0.410	120	90	110	110	100	80
Computational resources	1.000	110	90	80	120	90	100
	Results	0.99	0.96	0.96	1.05	0.92	0.91
Killer Req	Thesis	1	1	0	1	1	0
	Software	1	1	1	1	1	1
	Final Score	0.99	0.96	0.00	1.05	0.92	0.00

Table 3.12: Trade Off with weights determined through AHP

3.4.3. Trade Off Results

The classical trade off method relies on scoring each design option for each criterion. The scoring scale can be chosen rather arbitrarily as the results are normalised. In this case, scoring is done with a baseline score of 100 points for an acceptable solution, with solutions which underperform on a criterion receiving point deductions in steps of 10, and solutions which outperform the criterion receiving point awards in steps of 10. The results are summed and normalised, and the killer requirements are taken into account before achieving the final score, as seen in Table 3.12.

The results of the trade off show a single winner: neural networks. However, three other alternatives are too close to provide a conclusive result. Therefore to confirm or reject this outcome, this outcome is compared to that of a trade off with a different scoring system first, and following that a different weight designation system.

3.5. Alternate Scoring and Critical Review

The outcome of any trade off should be reviewed critically to ensure the chosen solution(s) are actually those that best meet the requirements. In this case there are a few close matches, which indicates that they may all form roughly equivalent solutions depending on the scoring or weights. Therefore the determination of these two factors which significantly influence the outcome will be performed using alternate methods in order to confirm or reject the trade off results. Another trade off is performed with the weights being changed from the AHP determined ones to a simple ranking system. A third trade is performed using the AHP determined weights but the scoring uses a Pugh matrix. The full outcomes of these methods can be found in Appendix E. The result of both ranking system and the Pugh matrix are shown in Table 3.13, both continue to confirm the Neural Network based methods as the winner.

	Signal Processing	Parity Space	Cognitive Automation	Neural Networks	Support Vector Machine	Dynamic Bayesian Network
Alternate Ranking	0.99	0.97	0	1.04	0.91	0
Alternate Scoring	0.93	-0.11	0	4.704	1.387	0

Table 3.13: Results of the trade study using alternate ranking and scoring mechanisms

This outcome supports the conclusion that neural network based approaches are most suitable in this specific case, but the other two candidates (SVM, Signal Processing) should not be completely ruled out just based on this study. In fact, for larger, more risk-averse spacecraft cognitive automation methods are considered more promising by some researchers [62]. However, based on the trade off in this chapter, this thesis will focus on developing a neural network based approach to fault detection in the LUMIO AOCS system. This is mainly thanks to its relative low computational requirements (relative to the other methods) in combination with the high fault detection accuracy characteristics and low model complexity.

4

Fault Data Simulation

In order to train, test and verify any neural network for fault detection, nominal and faulty data is required. The best case would be to have labelled sets of real spacecraft data with real anomalies. However, these are not readily available. Therefore other sources of training and testing data are investigated and discussed in this chapter, and the fault simulation method and characteristics is discussed for the LUMIO test case.

This chapter will start by summarising the faults which can be encountered in AOCS telemetry, and defining a suitable method of simulating these faults in section 4.1. Following this, sources of LUMIO AOCS telemetry for testing and training purposes will be investigated and a final source will be selected in section 4.2. Finally section 4.3 will describe the datasets obtained for validation using real spacecraft data from OPS-SAT.

4.1. Fault Definition and Simulation Method

In order to design and test a neural network based method for fault detection, a baseline definition should be set stating which faults should be detected. This is challenging since it is unknown exactly which faults can occur and how they will manifest. However, literature such as the NASA Fault Management Handbook [41] and the 'Fault Tolerant Flight Control and Guidance Systems' book by Guillaume J.J. Ducard [13] give some insight into typical faults one can expect in spaceflight as well as guidance systems actuators and sensors. From a fault detection perspective in sensor data, two broad classes of faults can be distinguished: those directly detectable without context: step faults, erratic behaviour, outliers; and the class of faults which require additional information to detect: sensor drift, constant bias, loss of accuracy. This section will discuss these faults, their potential sources, and quantify them for simulation in the context of the LUMIO.

The IMU will be the test case under consideration in this thesis due to its critical nature in the AOCS system, and the option to validate IMU signals compared to other AOCS telemetry such as absolute attitude data from star trackers and sun sensors. It also correlates directly to system inputs from actuators such as reaction wheels, or RCS thrusters. Based on the initial AOCS simulations performed by the prime contractor for LUMIO, Politecnico di Milano, the STIM 210 IMU [67] produced by Safran is considered the Phase B LUMIO IMU. Note that this is different from the Phase A ISISpace IMU because the simulated data is only available for the STIM 210 IMU, and not the Phase A IMU. The faults will be quantified based on the STIM 210 specifications. It should be noted that this IMU does have self diagnostics software built in, but these checks are limited to (as taken from the STIM 210 datasheet [67]):

- Check of internal references
- Check of gyros (error and overload)
- Check of internal temperatures
- Check of RAM and flash
- Check of supply voltage

There is no checking of the actual measurement content and especially no cross checking between measured quantities of the gyros compared to absolute attitude and actuator input.

4.1.1. Directly Detectable Faults (Non Model Based)

The directly detectable faults are those which become visible in the telemetry without needing context or models for detection. These faults are listed in Table 4.1, a visual example of the three faults is shown in Figure 4.1. The faults are quantified by setting the minimum detectable magnitude of the fault, based on the LUMIO pointing requirements.

Step Faults

The step bias as shown in Figure 4.1a is defined as a sudden constant offset in the signal, potentially caused by SEUs, ground loops, or software bugs. It is quantified based on LUMIO pointing requirements. LUMIO requires a pointing accuracy of 0.1 degree (0.0017 rad) and a pointing stabilisation of 79.9 arcsec/s (0,000387 rad/s) during lunar tracking [50]. Therefore, a step bias of 0.002 rad/s was deemed an acceptable fault size for detection. At this step bias size, integration errors will lead to violation of the pointing accuracy within seconds, by which time the fault is detected.

Erratic Behaviour

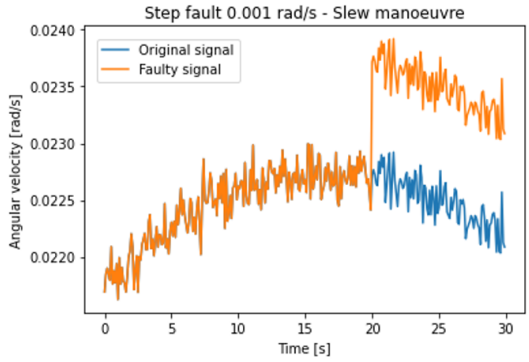
The erratic behaviour fault shown in Figure 4.1b is a (sudden) increase in signal noise. This can be the result of temperature effects such as thermal noise, but also thermal gradients and large variations in temperature such as those encountered in deep space. Other sources include environmentally induced electromagnetic interference (EMI), such as by radiation, or by internal electronics. Sensitive electronics can experience interference from onboard electronic units or even the electromagnetic fields of wires in the spacecraft amplifying each other [3]. Another source of this fault could be the ageing hardware, or a fault in a physical connector or analogue to digital converters due to shock and vibrations during launch and deployment. The fault is quantified by the standard deviation σ_{signal} of the signal, and is set to 0.01 rad/s based on the standard deviation of the nominal noise during the slew manoeuvre of 0.004 rad/s. The fault is simulated by sampling a Gaussian distribution for simplicity, although Markov noise could be used (such as in the GAFE simulator, see subsection 4.2.1) in the future to simulate more realistic, non-Gaussian noise.

Outliers

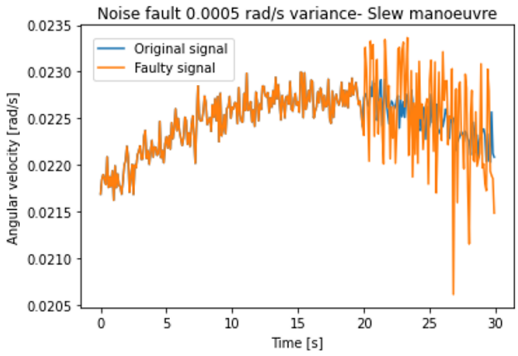
The outlier fault seen in Figure 4.1c is where a single datapoint deviates significantly from the rest of the signal. This fault can be the result of a charged particle striking the spacecraft (Single Event Upset (SEU)) or a sampling error in the sensor and processing units themselves. The quantification is difficult as outliers may present in different magnitudes. Therefore, it was assumed the outliers which should be detected shall be at least one Order of Magnitude (OoM) larger than the step bias faults, to account for noise and highly dynamic situations. This leads to outliers of 0.1 rad/s in the defined faults for this thesis.

Fault Type	FMECA ID	Potential root causes	Quantification	Reasoning
Step Bias	IMU.06	SEU, ground loops, software bug	+0.002 rad/s	LUMIO pointing requirements [50]
Erratic Behaviour	IMU.08	EM interference (external, internal), ADC/connector hardware fault, ground loops, thermal noise	0.01 rad/s STD	Standard deviation nominal noise
Outlier	IMU.06	SEU, processing/ sampling error	Spike +0.1 rad/s	+1 OoM

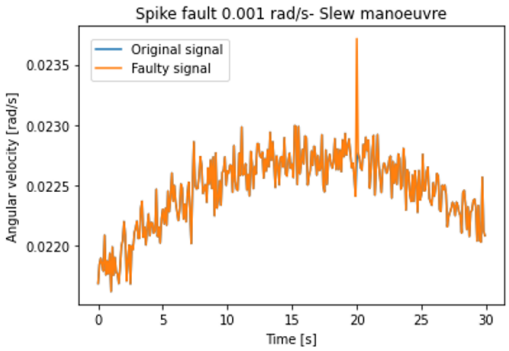
Table 4.1: List of directly detectable IMU faults, sources, and quantification for LUMIO



(a) Example of a step bias in the IMU x-axis angular rate reading



(b) Example of erratic behaviour in the IMU x-axis angular rate reading



(c) Example of an outlier ('spike fault') in the IMU x-axis angular rate reading

Figure 4.1: Fault examples - directly detectable faults

4.1.2. Faults Requiring Cross Checks (Model Based)

The second class of faults in AOCS telemetry are those which do not have distinguishable features in the signal as those shown in Figure 4.1. They exhibit the behaviour of the system under nominal conditions, but require a model based approach (or voting mechanisms) in order to detect them. These are summarised in Table 4.2 and shown in Figure 4.2.

Constant Bias
The first is a constant bias in the signal, seen in Figure 4.2a. This is a constant offset in the signal, present from the start of monitoring such that the 'step' feature is not seen such as in Figure 4.1a. It can occur due to a number of reasons, including ground loops, a software bug, or damage to the unit amongst other things. The quantification for the LUMIO test case is set at a bias of 0.01 rad/s. This magnitude implies that if detection occurs within 10 seconds a maximum error of 0.1 radians or 5.6 degrees is introduced in the real spacecraft attitude compared to the desired attitude.

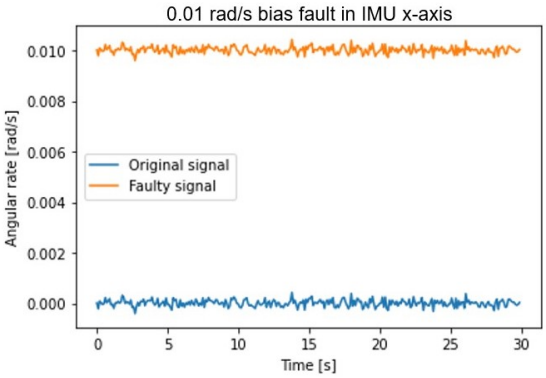
Signal Drift
The second is the drifting signal fault, shown in Figure 4.2b. This is the case where the measured value slowly deviates from the true value in a continuous manner. Real-life sources of this effect could be temperature effects and large temperature gradients, ageing, interference from other electronics in the spacecraft and calibration issues. This fault in the example of LUMIO is set to a drift of 0.0005 rad/s/s which equates to an error of 0.005 rad/s after 10 seconds, and a worst case accumulated pointing error of 0.0275 rad or 1.58 degrees at this point which is deemed acceptable.

Loss of Accuracy
The final fault type considered in this thesis is the loss of accuracy of the sensor seen in Figure 4.2c, in which the error between the measured value and the real value is dependent on the magnitude of this value. The larger the measured quantity, the larger the error. This can occur when sensor calibration is not performed or incorrectly performed. This fault was simulated using a scaling factor of 1.75, which means at the higher velocity slew rates of 0.02 rad/s seen in the LUMIO data (see subsection 4.2.2), this fault would induce a worst case error of 0.015 rad/s in the signal. This means a worst case offset of slightly less than 1 degrees after 10 seconds, indicating the scaling factor of 1.75 is an acceptable lower bound for this error.

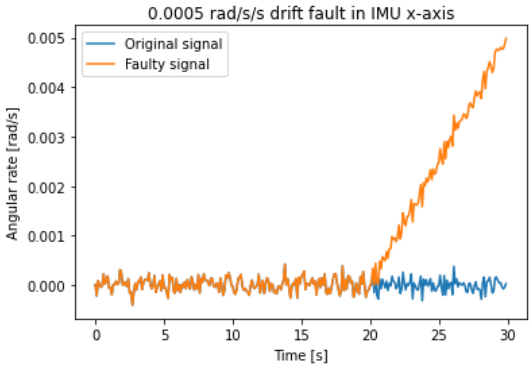
Fault Type	FMECA ID	Potential root causes	Quantification	Reasoning
Bias	IMU.06	Damage, ground loops, software bug	+0.01 rad/s	STIM Specifications
Signal drift	IMU.06	Temperature effects, ageing, interference, stress, calibration issues	0.0005 rad/s^2	LUMIO pointing requirements
Loss of accuracy	IMU.06	Calibration error, temperature effects	x1.75	LUMIO pointing requirements

Table 4.2: List of IMU faults not directly detectable, sources and quantification for LUMIO

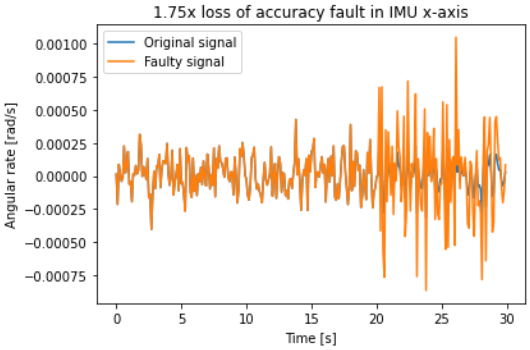
4.1.3. Other fault types
As mentioned before, it is not possible to determine every type of fault one will encounter beforehand, therefore in this thesis the most obvious and likely have been discussed. However, there are fault features which can be conceived which have not been discussed and which will not be treated in this thesis. This includes any imaginable combination of the above faults, such as a drifting signal which increases in noise or a biased signal which is also uncalibrated.



(a) Example of a constant bias in the IMU x-axis angular rate reading



(b) Example of a drift in the IMU x-axis angular rate reading



(c) Example of a loss of accuracy (calibration) in the IMU x-axis angular rate reading

Figure 4.2: Fault examples - model based detectable faults

Finally, for any of the aforementioned faults it can occur that these repeat intermittently. The pattern in the signal could for example not be just one step, but multiple steps. Erratic behaviour may cut in or out at times with low or high frequency. All these fault types could also be investigated, but it is hypothesised that if the fault detection mechanism can catch these subtle faults, these more obvious combined or period faults can also be easily detected.

4.2. LUMIO AOCS Telemetry Simulation

In order to design and test a fault detection method for the LUMIO AOCS, (simulated) spacecraft telemetry is needed. Two sources were investigated: the GAFE simulator and the simulation data obtained from the AOCS design team at Politecnico di Milano. These are both discussed in this section.

4.2.1. GAFE Simulator

Generic AOCS/GNC Techniques & Design Framework for FDIR (GAFE) is a powerful MATLAB tool developed jointly by ESA, Airbus, Astos, and IFR in order to support AOCS FDIR design specifically. [39] It is a highly advanced tool with incredible potential for simulating spacecraft AOCS nominal and faulty behaviour. Based on other simulators such as the Matlab CubeSat simulator or basic analytic models for spacecraft, this simulator can be considered as state of the art for AOCS FDIR design. Therefore, it was considered an excellent source of data for network training in this thesis as well.

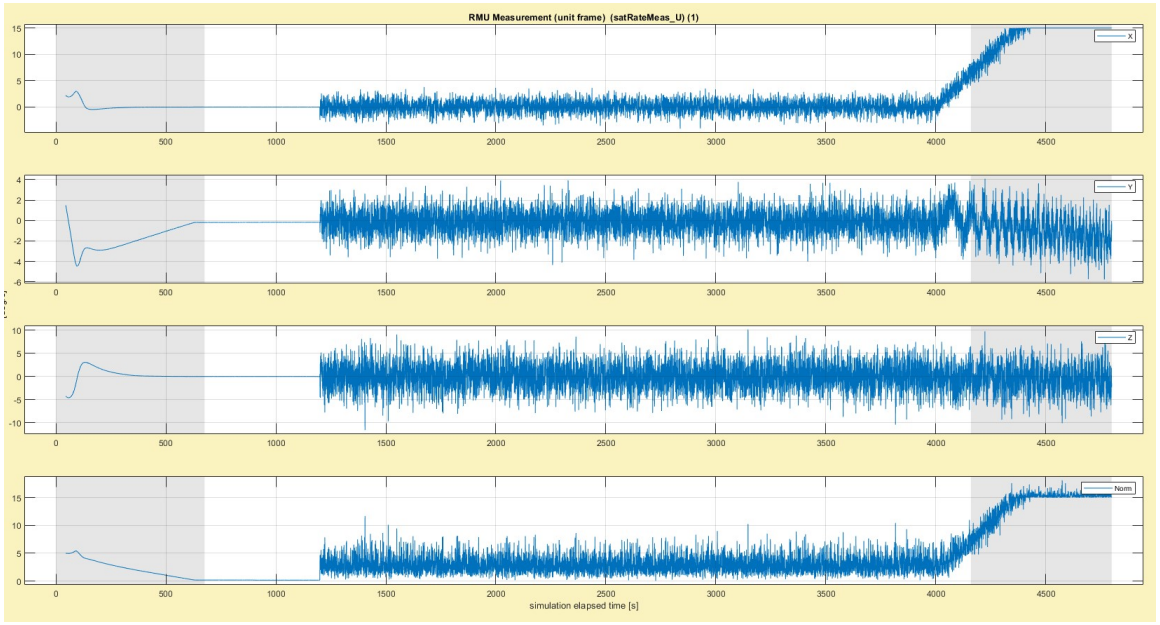


Figure 4.3: GAFE Simulated IMU Faults: following boot up ($60s$) and detumbling ($600s$) a noise fault occurs at $t = 1200s$ and random walk fault at $t = 4000s$

Tool Content
The simulator is extensive and includes realistic models of AOCS actuators and sensors, including but not limited to: reaction wheels, magnetorquers, thrusters, sun sensors, star trackers and Inertial Measurement Unit (IMU)s. The parameters of these units can be modified in order to set physical parameters (mass, volume, inertia), signal parameters (noise levels, sampling rates) as well as digital parameters (resolution, errors). Also incorporated in the tool are detailed models in order to model disturbances such as the aerodynamic drag, magnetic field interactions, gravitational effects from Earth, Sun and Moon, and solar radiation effects.

The AOCS system in GAFE is brought together through a predefined AOCS algorithm which takes the input data from sensors, and depending on the defined scenario controls the spacecraft accordingly. The FDIR system in the simulator watches over the units and controls a health manager which tells the AOCS algorithm which units are available and valid.

Fault Introduction
The tool allows for multiple highly realistic faults to be introduced at given times in different units. The generic fault types included are:

- Fixed Bias
- Random Walk (drift)
- Stale Data
- No Signal
- Non-Gaussian noise (erratic behaviour)

Unit-specific faults can be introduced, such as an increase in reaction wheel friction or temperature. An example of an IMU fault simulated using GAFE is shown in Figure 4.3.

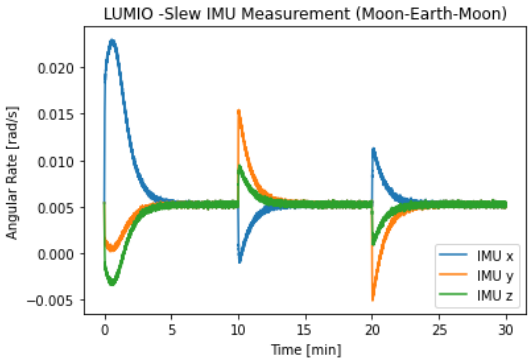
Usage of GAFE for this Thesis
Although this simulator seems to be the perfect candidate for use in this thesis, no accurate fault data could be generated. This is in part due to the non user-friendly interface, with a

large number of MATLAB scripts needing to be adjusted before the simulator works and with duplicate variables making it difficult to set specific conditions and perform verification. The simulator also did not show predictable, consistent or reproducible behaviour when the LUMIO spacecraft parameters were introduced, which due to lack of understanding of the underlying models could not be resolved. After extensive testing and consultation with experts, it was decided this simulator would not be used for the purpose of this thesis. However, in later stages and after further work by the developers, this tool is a very promising source of training data in the absence of real spacecraft telemetry.

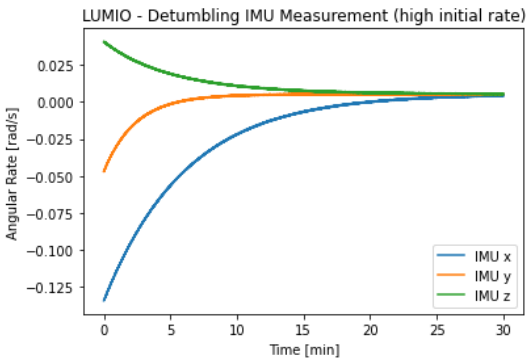
4.2.2. Simulated LUMIO Telemetry: Politecnico di Milano

The LUMIO project is currently undergoing its Phase B study with Politecnico di Milano as prime contractor on the project. Some initial AOCS simulations of LUMIO were obtained which include IMU rate measurements and Reaction Wheel momentum loading data for 4 different scenarios:

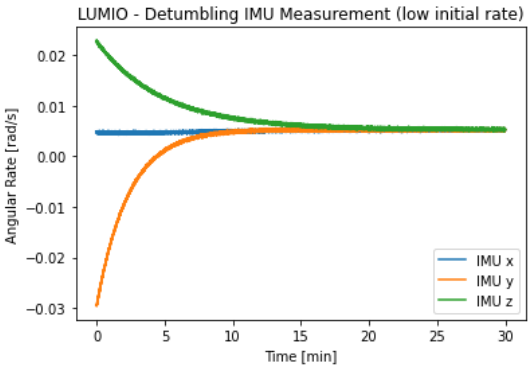
- **Slew**: 30 minute simulation of LUMIO slewing from Moon pointing to Earth pointing and then back to Moon acquisition. (Figure 4.4a)
- **Detumbling - high velocity**: simulation of detumbling from high initial angular rates (up to 0.15 rad/s on each axis) within 30 minutes. (Figure 4.4b)
- **Detumbling - low velocity**: simulation of detumbling from low initial angular rates (up to 0.03 rad/s on each axis) within 30 minutes. (Figure 4.4c)
- **Lunar Tracking**: the spacecraft holds the camera pointing steadily at the moon for a period of 7 days (Figure 4.5a and Figure 4.5b)



(a) Simulated LUMIO IMU Output: Slew manoeuvre, Moon-Earth-Moon within 30 minutes



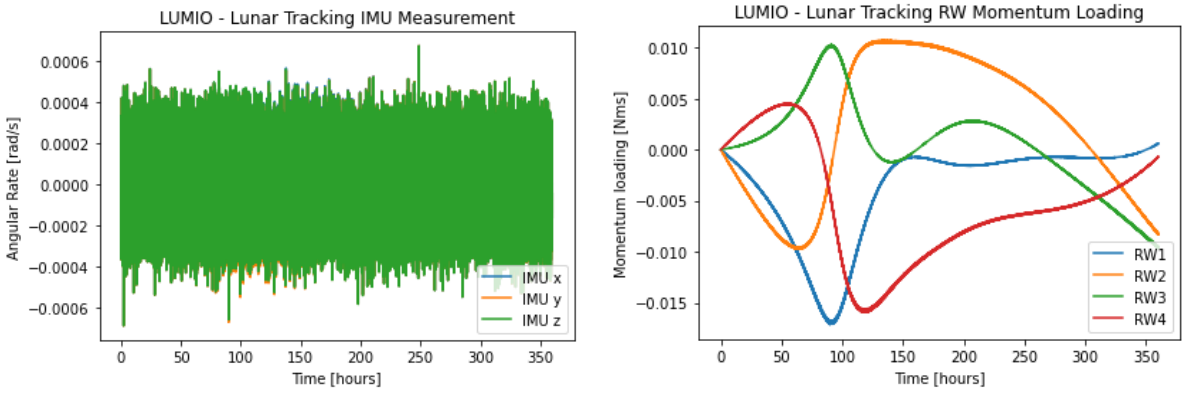
(b) Simulated LUMIO IMU Output: high velocity detumbling



(c) Simulated LUMIO IMU Output: low velocity detumbling

Figure 4.4: Slew and Detumbling simulated IMU Data (source: Politecnico di Milano)

The IMU data for slew and detumbling scenarios is plotted in Figure 4.4. The measurements are taken in the instrument relative reference frame of the spacecraft. This data is modelled based on real component performance, in this case the STIM210 IMU, and is sampled at a rate of 10Hz (reduced down to an assumed 1Hz by the AOCS algorithm). The simulation assumes a constant bias of 0.005 rad/s on the IMU data under nominal operating conditions, which is why the rate does not converge to 0 rad/s in all scenarios. The Lunar Tracking IMU and corresponding Reaction Wheel (RW) data are shown in Figure 4.5. Due to the early stage of the project, the RW data corresponding to the simulated IMU measurements is only available for the tracking scenario, not for the slew and detumbling scenario.



(a) Simulated LUMIO IMU rate measurement during 15 days of lunar tracking

(b) Simulated LUMIO Reaction Wheel momentum loading [Nms] during 15 days of lunar tracking

Figure 4.5: Simulated Lunar Tracking IMU and reaction wheel data (source: Politecnico di Milano)

4.3. Real Satellite Telemetry: OPS-SAT

As a final potential source of data, access was requested and granted to the telemetry of ESA's OPS-SAT. The OPS-SAT mission is based on a 3U CubeSat design and is a demonstrator satellite for testing ground control software under real flight conditions. The project is lead by ESA ESOC, and allows for rapid and flexible testing of new mission operations concepts [16]. Experimenters can request access to the reprogrammable spacecraft and are allowed to test concepts with no prior flight heritage and minimal preload testing. While for this thesis no in-flight testing is performed, the downlinked telemetry of the spacecraft functioning is valuable to test fault detection concepts on real flight data. As an example, a downlinked telemetry frame containing the three angular rates measured by the IMU is shown in Figure 4.6. The duration of these passes in view of a ground station is typically just under an hour.

For verification purposes, three windows were arbitrarily selected from recent telemetry, based on the fact that for these windows all AOCS data was available and complete. The windows are all taken on November 29 of 2022, at the following times:

- **Window 1:** 07:29:13 - 08:07:40
- **Window 2:** 10:38:07 - 11:17:03
- **Window 3:** 20:25:47 - 21:04:47

Each telemetry frame includes the data shown in Table 4.3. Before using the datasets, some processing of the data was required. Some series were missing certain datapoints compared to others, so comparison of the timestamps allowed to identify these. The missing datapoints were then artificially created and linear interpolation performed in order to fill the gaps. This is not thought to influence the quality or integrity of telemetry as there was never more than one

missing datapoint sequentially: every artificial datapoint was always based on two neighbouring real points. There were also at most three missing datapoints per window, out of over 180 measurements.

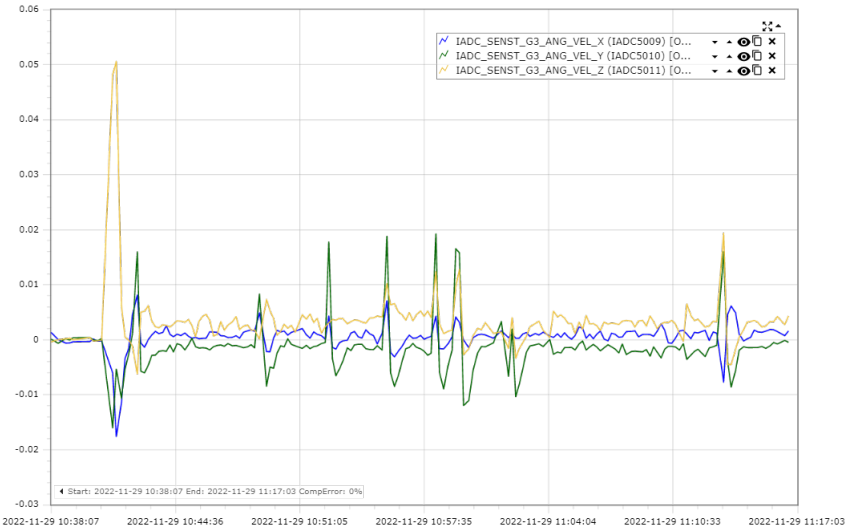


Figure 4.6: OPS SAT IMU telemetry downlinked during a 39 minute communication window (window 2) on 29 November 2022

Unit	Parameter	Unit
IMU	Angular rate x-axis	rad/s
IMU	Angular rate y-axis	rad/s
IMU	Angular rate z-axis	rad/s
RW	Wheel speed x-axis	RPM
RW	Wheel speed y-axis	RPM
RW	Wheel speed z-axis	RPM
STR	Quaternion 1 (Q1)	[-]
STR	Quaternion 2 (Q2)	[-]
STR	Quaternion 3 (Q3)	[-]
STR	Quaternion 4 (Q4)	[-]
SS	Sun Angle	Degrees

Table 4.3: OPS-SAT AOCS telemetry package content used in this thesis

5

Design of Fault Detection Method

In this chapter a fault detection neural network will be designed. Starting in section 5.1, the neural network concept and workings are introduced, as well as some typical metrics used. In section 5.2 different methods for fault detection using neural networks and their advantages and drawbacks are explored. Here, the challenges faced in using neural network based fault detection are also highlighted, and a motivated is given for the unsupervised learning approach used in this thesis. The training data structure and preprocessing steps are presented in section 5.3, after which the design process of the network itself is described in section 5.4. This includes the design philosophy, choices made and the tuning of the network hyperparameters.

5.1. Introduction to Neural Networks

While the full mathematical definition of a neural network will not be discussed here, a short description of the inner workings and commonly used terminology will be given. A neural network, as shown in Figure 5.1, consists entirely of neurons, with each neuron being part of a layer. In this example, the network has two (numerical) inputs, which form the input layer. The output layer of the network consists of four neurons. When the network is trained, each neuron is updated with weights and a bias, such that when the network receives the inputs it produces the desired outputs.

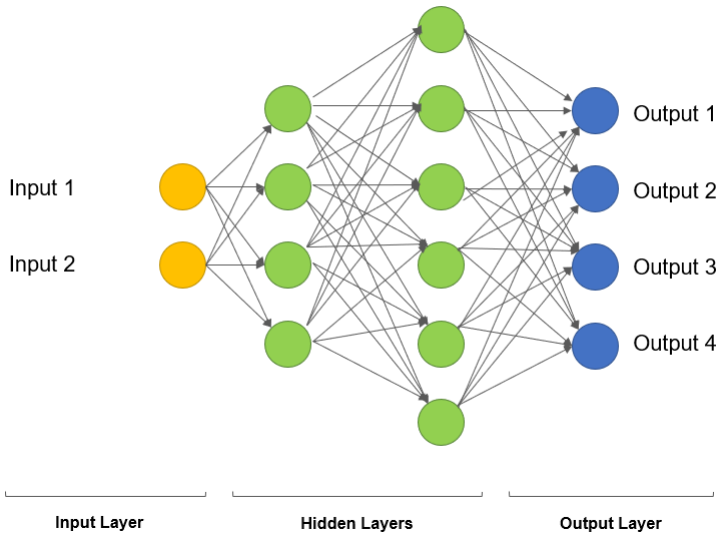


Figure 5.1: Simple neural network layout with two hidden layers

An example of a single neuron which is connected to three neurons of the previous layer is shown in Figure 5.2. It can be seen that the neuron takes a number of inputs, which can be the network input vector or the output from a previous hidden layer, and multiplies each input with a specific weight. The summation of these weights, together with a bias introduced in the neuron, are fed through an (often nonlinear) activation function which produces the output of the neuron. This output may be fed into the next layer of neurons or it may be the network output. The most important aspects of this process are shortly detailed below.

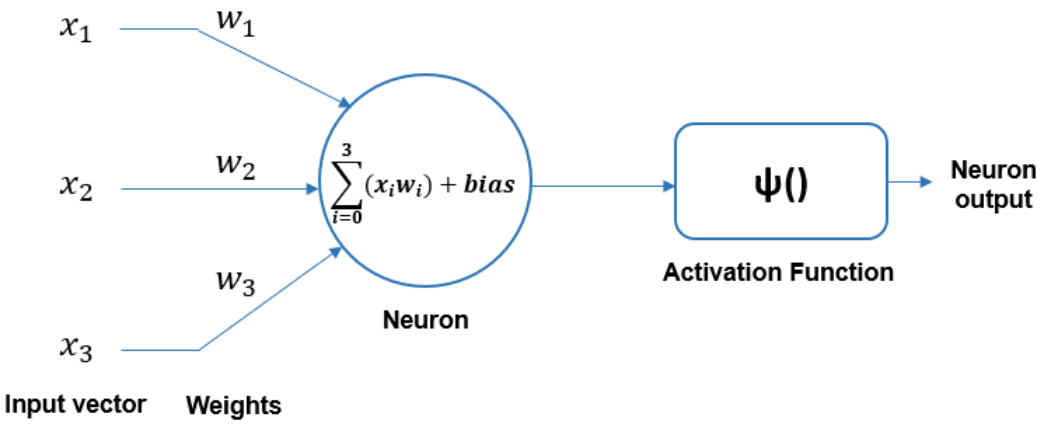


Figure 5.2: Basic workings of a single neuron in a neural network explained

5.1.1. Weights and Bias

The weights of a neuron are initialised stochastically at the start of the training process. They are then updated as training continues in order to produce a better output. Every neuron has their specific weights for every specific input. This means that a network with 2 layers of 8 neurons and an input of size 4 will have 4 x 8 x 8 = 256 unique weights. It can be seen that even for such a small network, the number of combinations of weights is endless. Therefore the process of setting these weights to produce the desired output is critical in setting up a useful network. For that reason an optimiser function is used during the training process. The optimiser function adjusts the weights and biases of the network based on the calculated gradients of the loss function (the loss function measures the difference between desired and actual output). The goal of the optimiser is to find the optimal set of weights and biases that will result in the lowest loss. The network is always trained in such a way that loss is minized as much as possible, whichever way loss is defined.

After summation of the input products, a bias is introduced for each neuron to help deal with inputs which are very small or contain a lot of zeroes (for example, dark images have a lot of pixels which are zeros). The bias is set for every single neuron and is also updated through the training process in a similar way to the weights.

5.1.2. Activation Function

The result of the input summation and the bias is then passed through activation function $\psi()$ in order to introduce nonlinearity. Without this function the summations of the inputs multiplied by the weights would lead to a linear relationship between the input and output. To allow the network to learn nonlinear behaviour, the nonlinear activation function is introduced. While there are many activation functions, two of the most popular for classification problems are used and discussed in this thesis: the Rectified Linear Unit (ReLU) and the Sigmoid function.

The ReLU, and its variation the Parametric ReLU (PReLU) are shown in Figure 5.3. It can be seen that the ReLU is a piece-wise function, which is linear for positive inputs and zero for negative inputs. A variant, which helps prevent the 'dying ReLU' problem (negative inputs lead to dead neurons which only output zero) is the PReLU. It has a learning parameter 'a' which can be tuned in the training process.

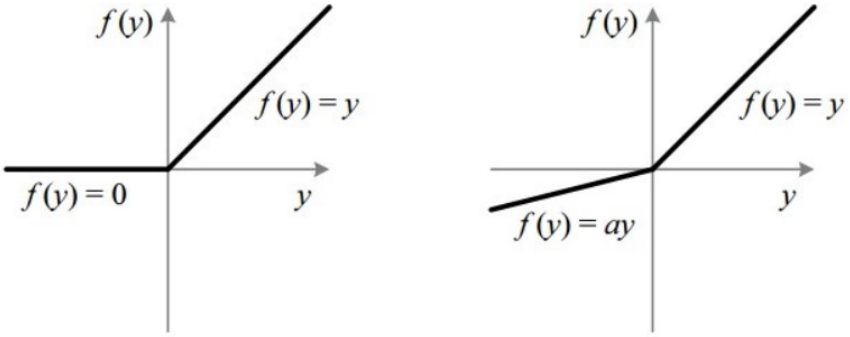


Figure 5.3: ReLU (left) and PReLU (right) activation functions. Source: He et al. [27]

The other nonlinear activation function used in this thesis is the Sigmoid function, shown in Figure 5.4. It can be seen here that this is a nonlinear function which translates its input to a range from 0 to 1 with a clear tendency to favour the extremes of this range as the input is further away from zero. This is highly useful for binary classification, where the output is expected to be close to zero or one, symbolising true or false or a similar classification.

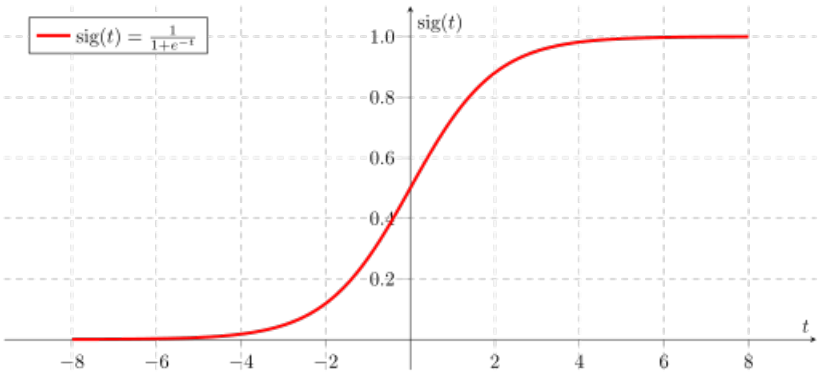


Figure 5.4: Sigmoid Activation Function. Source: Martin Thoma [61]

5.1.3. Loss Function

As mentioned before, the loss function is used to compare how well the neural network performs during training and help improve it. The neural network uses the loss function as its measure of how well it is learning certain patterns. Depending on the application of the network, a different function is chosen. Regressive loss functions are commonly used for continuous prediction (as is the case in this thesis) and include the Mean Absolute Error (MAE), Mean Squared Error (MSE), Mean Squared Logarithmic Error (MSLE), and Mean Absolute Percentage Error (MAPE). On the other hand, probabilistic loss functions are commonly used for prediction of probability distributions (e.g. evaluating a network for predicting the likelihood that a picture contains humans). This class of loss functions measure the difference between probability distributions, and include categorical cross entropy, binary cross entropy, and Kullback-Leibler Divergence.

It is essentially the way a network training process can understand how it is performing and how it should adapt its weights and biases in order to improve its performance. The loss curve is an important indicator in training as loss convergence indicates some form of learning is occurring [33]. Next to the loss, the validated loss is the loss calculated using unseen data, hence the 5% reserved data discussed in the previous section. This measure shows that the network learns to generalise well and does not just overfit to the training data.

5.1.4. Autoencoder Explained

For this thesis, an autoencoder network is used, which is a type of neural network that is used for unsupervised learning. This means that it is trained on data without any labels or annotations. This is especially beneficial for anomaly detection, since labelled data is not available in general and even if a labelled dataset were available the overrepresentation problem described previously in this chapter would prove challenging.

The idea behind an autoencoder is to encode input data into a lower-dimensional representation, the so-called latent space or latent representation, from which the network can reconstruct ('decode') the original data. During training, the network attempts to minimise the difference between the original input and the reconstructed output, calculated as the reconstruction error ϵ_{rec}. This forces the network to learn a compressed and optimised representation of the input data. Once trained, the autoencoder can be used to reconstruct new data points which are similar to the training data, but which it has not previously seen.

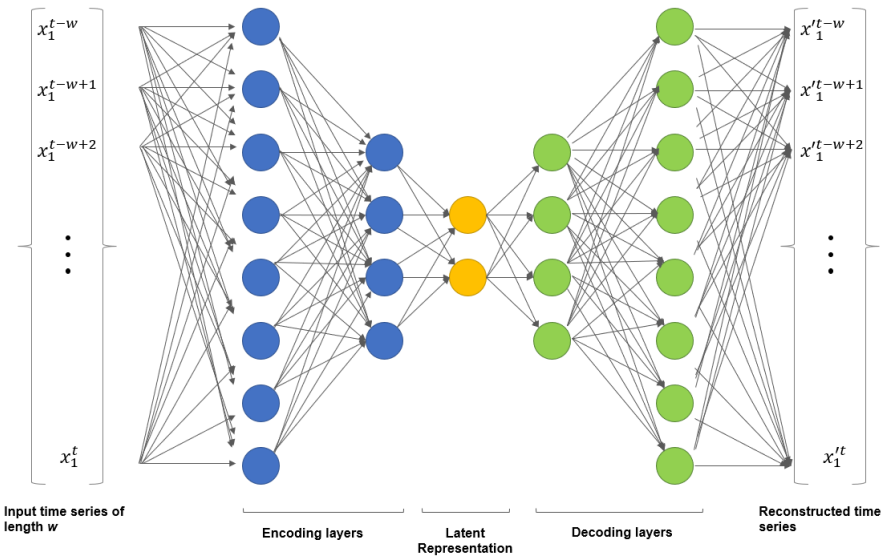


Figure 5.5: Generic autoencoder network with 2 encoding layers, 2 decoding layers 2 and a latent representation layer

Autoencoders have many applications, such as image and speech recognition, data compression, and anomaly detection. For clarity, an example of an autoencoder application is shown in Figure 5.6. Here the input is noisy handwriting pictures of numbers, which the autoencoder has learned to reconstruct. The output is the image reconstructed, but focusing only on the key features which were stored in its latent space. Thanks to this 'compressed' representation the image has been denoised, and is now suitable for processing.

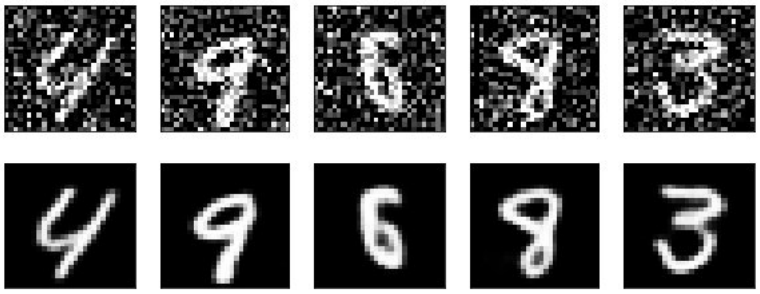


Figure 5.6: An example of noisy handwriting of numbers reconstructed by an autoencoder. Source: Prashanth Venkataraman [63]

5.1.5. Effectiveness Metrics for Neural Networks

In order to measure efficacy of a binary classifier networks and compare the results between different networks on a fair basis, a few standard metrics are first introduced which are commonly used in literature. These metrics are especially useful for binary classifier networks, since the performance not only depends on being able to classify something accurately (True Positive TP and True Negative TN), but also on avoiding inaccurate classifications (False Positive FP and False Negative FN). The definitions of these classifications are shown in Table 5.1. The performance metrics used in this thesis are shortly discussed below.

		Network Predicted	
		Fault	**No Fault**
Actual	**Fault**	True Positive	False Negative
	No Fault	False Positive	True Negative

Table 5.1: Binary Classifier Network outcomes

The first is recall, indicating how well the network can predict true positives (detect faults that are actually faults):

$$recall = \frac{TP}{TP + FN} \tag{5.1}$$

The second metric is the precision score, or the measure of how many of the classified positives (predicted faults) were true positives (actual faults):

$$precision = \frac{TP}{TP + FP} \tag{5.2}$$

These two metrics are often combined into an f-score or F1 score [73] which gives an overall indication of how well the network can classify its inputs.

$$F1 = \frac{2 * precision * recall}{precision + recall} \tag{5.3}$$

Aside from the general neural network metrics, some FDIR metrics are also commonly used: fault detection performance is oftentimes measured using the Fault Detection Rate (FDR). From the definition below, it can be seen that this is exactly the recall from Equation 5.1.

$$FDR = \frac{TP}{TP + FN} = recall \tag{5.4}$$

The opposite measure is the False Alarm Rate (FAR), which determines how easily the system classifies a non-faulty situation as faulty. It is defined as:

$$FAR = \frac{FP}{TN + FP} \tag{5.5}$$

The aforementioned metrics will be used where relevant in the design and testing of the fault detection network described in the next chapters.

5.2. Exploration of Fault Detection Methods Using Neural Networks

While artificial intelligence in general, and neural networks specifically have been a hot topic in research, the term itself covers a broad range of concepts and applications. A simple perceptron (a form of binary classifier) neural network was first proposed in 1943 already by Mcculloch and Pitts [64], whereas today neural networks exist in many forms and shapes each suited to their own applications.

There are multiple possible areas of application for neural networks in fault detection. Each application has their inherent challenges and benefits, as well as their degree of maturity. These design options are shortly discussed in this section. This includes some examples of supervised learning which uses labelled training data, as well as unsupervised learning which does not require labelled data.

5.2.1. Signal Level Fault Detection

In a spacecraft, it is desirable to detect and isolate a fault at the lowest level possible to avoid fault propagation and chain failures. To this end, signal processing is often applied to detect anomalies in a processed or unprocessed output signal of a unit, such as the reaction wheel tachometer, IMU gyroscopes or star tracker quaternions [71]. However, when looking at the different types of sensor faults described in chapter 4, this form of failure detection is only able to capture the class of 'directly detectable faults' such as frozen measurements, spikes, steps and erratic behaviour. The faults requiring cross checks such as a constant bias in the signal, a drift or loss of accuracy are not directly visible in the signal and require other measurements to compare to in cross checks.

Such a signal fault feature detection system in the form of a neural network is proposed by Ince et al. [29] for application to an electric motor. The raw data of a motor current signal is fed straight into a 1D Convolutional Neural Network (CNN) and feature extraction and fault classification happens immediately, showing an energy efficient method to perform real-time FDI. It is highly accurate with a recall of 97.8% and 97.0% precision, or an F1 score of 97.4% [29]. The main drawback here of course, is that this requires labelled training data from such a motor, which in turn requires the definition of fault types. If the system encounters a fault which was not included in training data because it was not expected or less likely, it will not be able to classify it. in the case of Ince et al. the data was obtained through (destructive) testing of the system, something which is not desirable for SmallSat developers with limited budget.

5.2.2. Neural Network Based Nonlinear Regression and Residual Generation

In smaller and modern spacecraft, hardware redundancy is often replaced with analytical redundancy [53]: instead of comparing identical measurements from different units to detect faults, one can estimate the measurement based on other available parameters and compare it directly to the real measurement to detect a fault. These model based FDIR methods require complex numerical models of the system to estimate outputs, inputs or system states: the so-called analytical methods. They also rely heavily on system observability and highly accurate model in order to be effective. These methods are computationally heavy and suffer from convergence problems [71]. The universal approximation theorem, proven by Hornik et

al. in 1989 [34] states that a feed-forward neural network with a single hidden layer containing a finite number of neurons can approximate any continuous function to an arbitrary degree of accuracy, given enough neurons. This implies that any of these complex numerical models could simply be replaced with a regressive neural network, tailored to suit the desired degree of accuracy. The residual generation process is then the same, allowing for a sufficiently large deviation of a residual to indicate the presence of a fault.

The presumed benefit of using such a regression network is to generate residuals at either lower computational cost, higher accuracy, the absence of convergence problems, or a combination of the aforementioned factors. The benefit depends on the design criteria, as a highly accurate network would likely be more computationally expensive than a numerical model of the same accuracy. It would however be highly accurate at detecting and isolating faults and not suffer from convergence issues, so it is up to the designer of the system to decide.

A recent example of such system is proposed by Yuandong et al. [72] in the form of a disturbance observer for Control Moment Gyroscope (CMG) to help detect and isolate faults. In these designs, the neural network does not directly perform fault diagnosis, but provides estimations on external disturbances to the FDIR system which leads to improved fault detection and isolation. The outcome of the paper showed the proposed scheme lead to a more accurate attitude tracking of the spacecraft, while reducing energy consumption of the algorithm.

5.2.3. Neural Network Based Fault Classification

A third potential application of neural networks would be in classifying faults in system states. A classification network could take as input the system variables such as states, operational modes, and system inputs and outputs to determine the presence of a fault. However, this is not considered beneficial in the instance of a CubeSat, as a simple truth table could cover this and be fully deterministic. A short example: if the spacecraft is in detumbling mode, the reaction wheel is in the 'ON' mode, and the communication status is 'OFF' for this reaction wheel, then this is an indication of a communication fault. A neural network to identify this kind of scenario is deemed overkill. The possibility of an integrated network which can perform system-level FDIR for more complex missions could be promising here, but no such system was found for spacecraft or similar systems in literature.

5.2.4. Time Series Correlation

An application which is able to detect more fault types than the signal processing methods has recently been proposed by Xiang et al. [69] This idea tries to relate behaviour in multiple time series to a certain set of predefined fault profiles on which it has been trained, i.e. supervised learning. A similar idea but with unsupervised learning is proposed by Zhang et al. [73] using the concept of signature matrices and convolution to correlate time series in a system and recognise behaviour that is anomalous over time. It should be noted that the work of Zhang et al. uses an autoencoder and achieves a recall score of 85% and a precision of 95%. In the work of Zhang et al, the F1 score can be calculated as 89.7%, which is acceptable when it comes to classifiers but inferior to the performance of the simple supervised learning approach used in the signal level fault detection concept by Ince et al [29], which showed an F1 score of 97.4%. The key benefit of the method introduced by Zhang et al is that it can deal with unpredictable or unexpected faults, unlike the signal level method.

5.2.5. Other Methods

There are other methods available, specifically related to certain fields of research such as genetics or natural language processing which will not be discussed here. There are also other,

more advanced neural network types recently developed such as the Spiking Neural Network which will not be considered in this thesis due to the complexity of the process. When it comes to the domain of spacecraft dynamics, navigation and control, a comprehensive review of available methods using neural networks is given by Silvestrini and Lavagna [54].

5.2.6. Challenges in Neural Network Based FD

Overrepresentation Problem

When it comes to creating an accurate neural network for fault diagnosis, one of the key challenges is to ensure it can generalise well based on the training data [54]. Typical satellite telemetry data will not contain a fault or anomaly the majority of the time if reliable units are used and the system is designed well. This implies that if the network never recognises an anomaly it is still close to 100% accurate, yet completely useless. To solve this issue, many solutions have been proposed and applied in the field. Some of the recently proposed solutions to the problem are:

- Unsupervised learning approaches using only nominal data: in order to recognise non-nominal behaviour, a neural network is trained to recognise nominal behaviour and thus reject non-nominal behaviour [54].
- Undersampling nominal data: in the case of supervised learning, one can undersample the nominal data (make it a smaller proportion of total dataset) in order to force the network to account for faulty situations more. The risk here is that useful training data is discarded [21].
- Oversampling faulty data: the same concept can be applied to the faulty data, making the faults a larger part of the training dataset. The risk here is that by copying training data, overfitting becomes more likely and generalisation is not achieved [21].

Data Labelling and Feature Engineering Challenge

Another challenge encountered is that in supervised learning systems, the data needs to be labelled by experts or features need to be engineered such that the network can recognise those features. In the first case, there is a need for expert knowledge of the features, in this case the fault features. Given the limited applications of large (12U) CubeSats in deep space, and the lack of operational data, features can be hard to generate. Therefore the system does not know what it is looking for.

This is especially true for classifier systems which aim to recognise and classify a fault, meaning the faults need to be known beforehand. As stated in the SAVOIR FDIR Handbook: ”*while the number of intended behaviours of a system under design is finite, the number of unintended behaviours is potentially infinite and managing this problem space is challenging by definition*” [56].

5.2.7. Selection of Unsupervised Learning for this Thesis

Based on the literature review performed prior to this thesis, it is concluded that while a lot of research has been performed into fault detection using supervised methods such as the works referenced in the previous sections, very little research focusing on unsupervised learning was found in the domain of fault detection. No research at all was found to focus on unsupervised learning for spacecraft fault detection. This is counter-intuitive considering the lack of labelled spacecraft data available, and the unpredictable nature of faults in space systems. This unsupervised learning is likely a promising method for performing intelligent fault detection and will be further investigated in this thesis.

5.3. Training Data

As a first step in building a fault detection neural network, qualitative training data from the LUMIO AOCS is required. This is sourced from Politecnico di Milano as described in chapter 4. Given these datastreams, they need to be processed before they can be used for training and testing a neural network. In this section, the process of transforming raw datastreams to normalised dataframes is described.

5.3.1. Data Structure

In order to use the LUMIO AOCS signals, the 'continuous' stream of telemetry coming from the IMU and Reaction Wheels needs to be prepared. The input vector of the network needs to be of fixed length, which is set to be 100 measurements. This is chosen assuming an AOCS algorithm sample rate of 2 Hz, which equates to 50 seconds of measurements. This is deemed long enough to notice faults within an acceptable timeframe (as soon as a few seconds) while still preserving the correlation with prior nominal data, and without requiring a massively demanding network. The design can easily be adapted accommodate larger or smaller input vectors as required, and the re-tuning of the network is straightforward.

The data is then stored per telemetry stream in a file containing frames. Each file represents one stream (e.g. RW 1 momentum) and contains n-100 frames where n is the number of datapoints in the telemetry stream. The process and structure is shown in Figure 5.7. The choice to update each frame by one datapoint at a time is in order to simulate the fault detection network receiving subsequent frames, with a refresh rate of 2 Hz. This way, faults can be detected in a matter of seconds. If the final implementation of the fault detection network receives telemetry at a lower refresh rate (the frames differ by multiple datapoints, or show no overlap at all) then the network will still be able to detect anomalous behaviour.

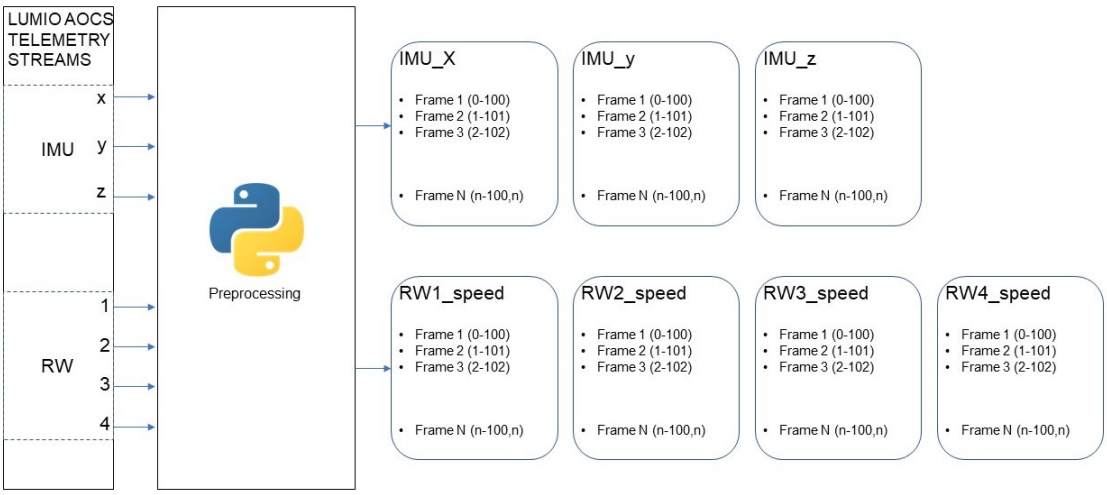


Figure 5.7: Conversion from LUMIO AOCS Telemetry stream to data frames for training and testing

5.3.2. Normalisation

With this time series split into manageable chunks of 100 data points, the next step is to normalise the data. Nola and Sevilla [55] already proved in 1997 that normalising data improves neural network performance significantly by a factor five to ten (estimation error reduction), and reduces computational resources required to run it by one order of magnitude. The latter is especially important for running these networks onboard small spacecraft with limited computing power. Another reason is that neural networks can presume that larger values carry more importance, depending on the loss function used. Therefore the normalisation is an ef-

fective way of reducing this bias in the network and allowing the training process to generalise behaviour better. Although many normalisation methods exist, only those methods commonly used in statistics methods are compared.

Rescaling Methods
There are many normalisation methods available for use in machine learning. Rescaling, commonly referred to as min-max scaling, is the process of rescaling the data to a specific range of values. This is typically in the range of 0 to 1 or -1 to 1. Min max scaling of dataset x to form normalised dataset x' is calculated as follows for every normalised datapoint x'_i :

$$x'_i = \frac{x_i - x_{min}}{x_{max} - x_{min}} \tag{5.6}$$

Here, x_{min} is the dataset minimum value, and x_{max} is the dataset maximum value. It can be seen that this method would be sensitive to a dataset with very far outliers.

Z-Score Normalisation
Z-Score Normalisation, also referred to as Standardisation, is the process of changing the dataset such that the mean becomes zero and the standard deviation σ is one. The formula for every datapoint x' to be normalised is then:

$$x' = \frac{x - \mu}{\sigma} \tag{5.7}$$

Where the mean and standard deviation are represented by μ and σ respectively. This normalisation tends to work well for datasets with a Gaussian distribution. It is less sensitive to outliers compared to rescaling, but can alter the shape of the distribution of the dataset.

Mean Normalisation
Mean normalisation also subtracts the mean from every datapoint in a set, but divides not by the standard deviation but by the feature range:

$$x' = \frac{x - \mu}{x_{max} - x_{min}} \tag{5.8}$$

The effect is again that the dataset has a zero mean, but that now the distribution shape is preserved. This also means the it is more sensitive to the outliers which affects the mean strongly.

Robust Scaling
Finally robust scaling subtracts the median and scales the data to the interquartile range instead of the full range. As the name suggests, this method offers more robustness to outlier as it typically uses the first and third quartile (25-75%) as the scaling range, and subtracts the median rather than the mean. The scaled data x' is then calculated as:

$$x' = \frac{x - median(x)}{Q_3(x) - Q_1(x)} \tag{5.9}$$

Here $Q_1(x)$ and $Q_3(x)$ represent the first and third quartiles respectively. Robust scaling can be applied where many outliers are present in a dataset, offsetting the mean.

Performance Comparison
In order to determine the most suitable method out of the four aforementioned scaling methods, a proof of concept test was run using a sample of the available data and some arbitrarily introduced faults as described in chapter 4 in order to assess and compare preliminary performance of each method. The results of this test are shown in Figure 5.8. The main measure for accuracy here is the F1 score, although it is desirable to pick high fault detection rate over low false alarm rate since the impact of missed detections poses a higher threat than a false alarm. From the graph it is clear that in this application mean normalisation outperforms the other methods with an F1 Score of 0.73, whereas the other networks show F1 scores in the range of 0.3 and 0.5. Notable is that rescaling offers a slightly better fault detection performance, but at a cost of the false alarm rate increasing tenfold. This indicates the network is not able to reconstruct nominal data very well using the rescaling method. Therefore the rescaling method is discarded and the mean normalisation will be used in further training and development. Note that these scores are based on a rough test and purely used for comparison of the normalisation methods. They do not indicate the real fault detection performance of this method.

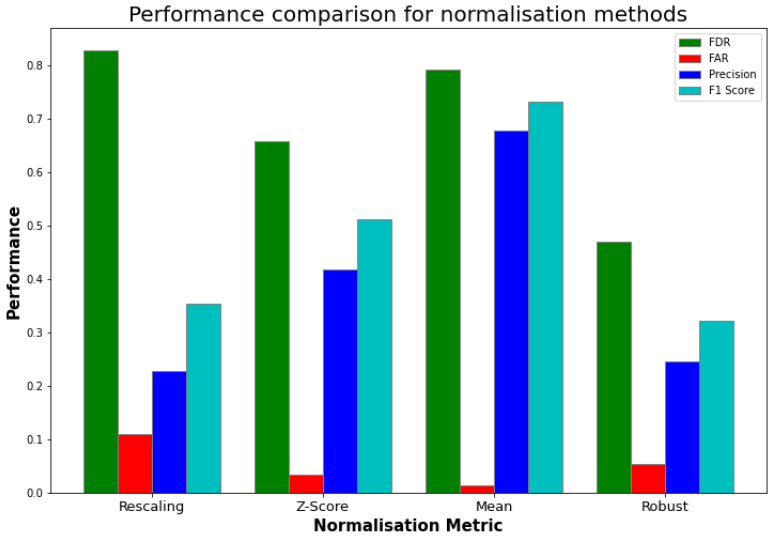


Figure 5.8: Normalisation methods performance comparison

5.3.3. Reserving Validation Data

The final step in preprocessing the data is to split the dataset into two: a training set and a validation set. The validation set is the 'unseen' data which the network should be able to reconstruct without having trained on it. It is one of the measures to ensure overfitting does not occur. When splitting this dataset, often times it is done by randomly selecting a percentage of the data. In order to ensure reproducible outputs, a built-in Keras function 'train_test_split' was used which allows the setting of a random state such that the 'random' selection of data is the same every time new tests are run. As a standard practice in this thesis, 5% of the training dataset is reserved for validation.

5.4. Design of the Autoencoder Network

Having defined the input data format and size, and the goal of the network (accurately reconstruct inputs) the network can be designed and tuned to improve performance.

5.4.1. Design Philosophy

As previously mentioned, unsupervised learning allows for a network to learn from nominal behaviour or patterns in order to detect anomalies in data. The autoencoder network is very proficient at this. The autoencoder learns to reconstruct its input accurately by first encoding the input into a latent representation which is smaller than the input vector, and then decoding the original input from this latent representation. For this thesis, considering the relatively small input vectors (100 datapoints) and lightweight network requirements, a simple autoencoder consisting of two encoding and two decoding layers will be designed. A schematic of the design is shown in Figure 5.5.

5.4.2. Detection Mechanism

The process for fault detection is illustrated in Figure 5.9. The autoencoder is trained on a nominal dataset, such as spacecraft telemetry under nominal conditions, and it is able to accurately reconstruct this type of data. However, when faulty signals come in which the autoencoder has not learnt to reconstruct accurately, the reconstruction error increases significantly and a fault is detected [73].

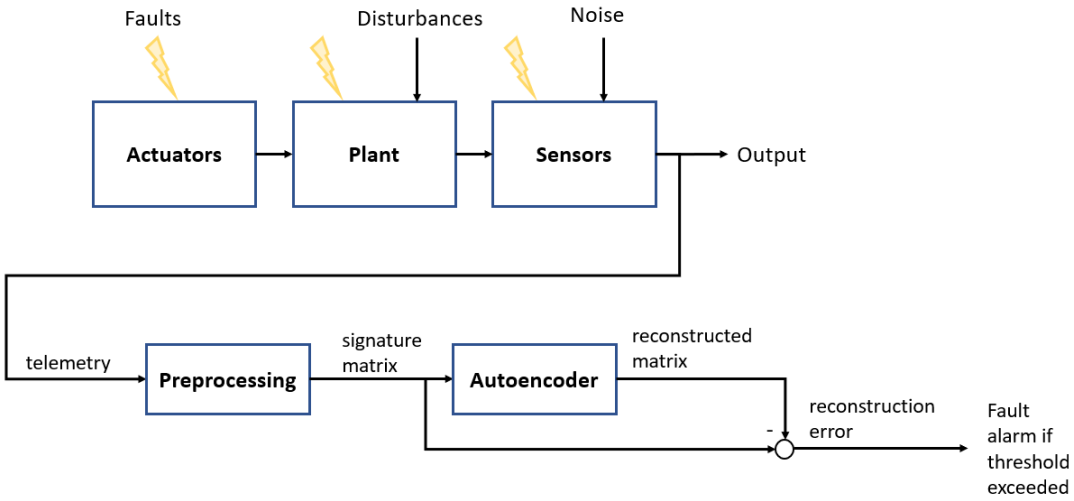


Figure 5.9: Fault detection process using autoencoder and reconstruction of signature matrices

The reconstruction errors therefore serves as anomaly score. An appropriate threshold should be selected for detecting these anomalies, which attempt to balance a low False Alarm Rate (FAR) and high Fault Detection Rate (FDR). The standard deviation of the nominal data's reconstruction errors can be used to set the threshold for anomaly detection. The reconstruction errors are calculated using a statistical error metric such as the Mean Squared Logarithmic Error (other metrics can be used and are investigated in subsection 5.4.3). The reconstruction error for the i-th signal is calculated as:

$$\epsilon_{rec,i} = \frac{1}{w}\sum_{k=1}^{w}(log(x_k+1) - log(x'_k+1))^2 \tag{5.10}$$

Where ϵ is the reconstruction error, w is the time series length, x_k is the k-th element of the input vector and x'_k is the k-th element of the reconstructed vector where $0 \leq k \leq w$. The fault

detection threshold is then determined by averaging the reconstruction errors for all nominal data, and n_{th} times the standard deviation of this set of errors is added to the average error to obtain the threshold. Here, n_{th} is a tuning factor which is determined after training such that FDR and FAR are optimised. The detection threshold τ for the entire system is then found as:

$$\tau = \mu_\epsilon + \sigma_\epsilon \times n_{th} \tag{5.11}$$

Where μ_ϵ is the average reconstruction error of all nominal signals and σ_ϵ is the standard deviation of all reconstruction errors of the nominal signals.

5.4.3. Network Hyperparameter Tuning

As the design of an autoencoder network is highly dependent on the shape and features of the input data, which it needs to represent in its latent space, there is no deterministic method of designing the network. Therefore, a first design is chosen to be a lightweight architecture with the capability of representing nonlinearities. Two encoding layers and two decoding layers each using the previously described ReLu activation function (Figure 5.3) are chosen and the basic autoencoder architecture is shown in Figure 5.5. The output activation function is then chosen as the Sigmoid function (Figure 5.4), which introduces nonlinearity in the network.

In the eyes of a machine learning expert, this network architecture could be considered quite simplistic compared to the typical spacecraft image processing autoencoders such as the one presented by Mohbat et al [32], which include convolution, pooling, and other advanced features. The main reason for this comparative simplicity is the input size of the fault detection network is about four orders of magnitude smaller compared to the typical remote sensing image (11x11 versus 1024x1024). It is also not required here that the input is reconstructed to perfection, the reconstruction error should just be minimised for training such that faulty signals generate a reconstruction error beyond the detection threshold. This is in line with the requirements presented in section 3.3 such that the detection system is simple and does not consume excessive onboard resources.

In an to attempt to achieve an optimal design, several of the network hyperparameters are tuned based on the produced reconstruction error. The hyperparameters of a network refer to those variables which describe the design of the network. This also includes the network architecture: number of layers, neurons, training epochs, batch size, optimiser functions, learning rates... There is no deterministic method in determining these parameters from the start, but rather experience and intuition along with rules-of-thumb provide a starting point from which these hyperparameters must then be further tuned to enhance performance [46]. The network will be initialised with a set of hyperparameters based on initial testing, and are then tuned to optimise performance. It is noted that for each application, tuning these parameters is required as performance may change with input size, dataset size and characteristics of the data.

Epoch Tuning

One key metric is the number of epochs a network is trained for. An epoch is counted everytime the dataset is completely fed through the network. At the end of an epoch, the loss (difference between desired and actual output, calculated as the MSLE) is plotted in order to see the improvement compared to the previous epoch, shown in Figure 5.10. In order to ensure the network is able to generalise and reconstruct unseen signals, the 5% of data reserved for validation is also tested which results in the validated loss shown in the plot. It can be seen that the loss and validated loss converge rapidly within a few epochs, indicating the network is learning in around 10 epochs. More epochs could lead to overfitting and therefore the 10 epochs will be maintained.

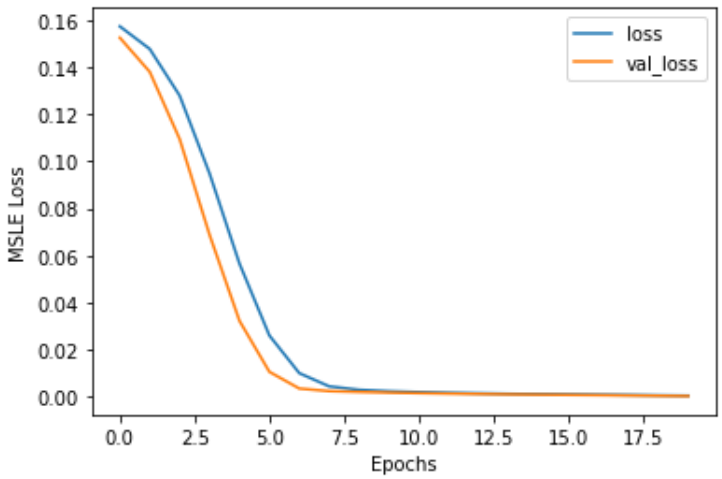


Figure 5.10: Autoencoder loss and validated loss evolution over the training epochs

Layer Size Tuning

By design choice, due to the lightweight requirements, two layers of neurons are used for both encoding and decoding. The number of neurons present in each layer will play an important role in network performance as well as computational efficiency. The autoencoder is designed as being symmetrical for simplicity, meaning the outer encoding and decoding layers have the same size, as do the inner encoding/decoding layers. The tuning occurred by trialling multiple configurations, with the outside layer size ranging from 80 (just below input size) to 300 neurons, and the inner layer ranging from 40 (over 2x the latent space size) to 140 neurons.

The upper limits of the layer size were chosen based on achieving a network with little need for computational resources. A neuron performs a single multiplication per input, and the additions per neuron equals the number of inputs plus one (the bias term), so the number of operations for a single pass through the encoder can be calculated. Every arithmetic operation takes one instruction on the processor in the case of the ARM9 processor, upon which the LUMIO processor is based [4]. The clock speed of this OBC is 400 MHz. In order to realistically run the fault detection every 10 seconds for example, one could state that this algorithm shall take up no more than 10% of resources in this timeframe, although preferably much less.

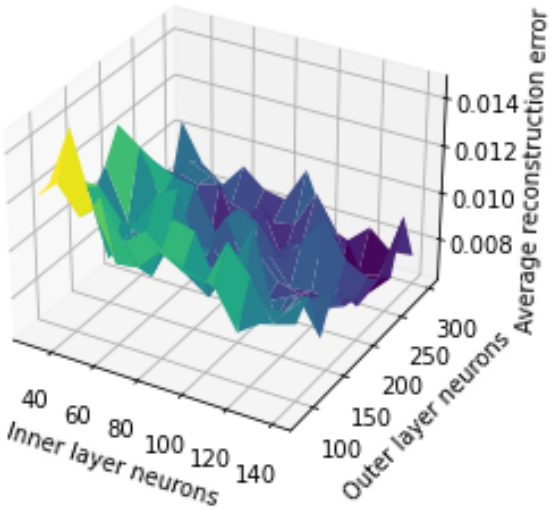


Figure 5.11: Layer size tuning for autoencoder network

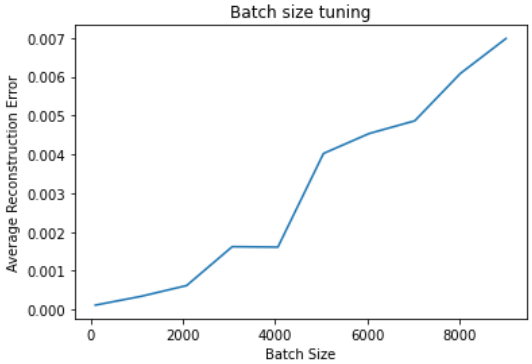
The maximum operations are then set to 50 million, which implies a maximum allowable layer size of 300 neurons in the outer layer and 140 in the inner layer. The minimum sizes are determined by the fact that the reconstruction error is seen to spike once the input layer size drops below the input size.

The results are shown in Figure 5.11. It can be seen that a network with a larger outer layer improves performance more than larger inner layer networks. A minimum reconstruction error was found at 260 neurons in the outer layer and 64 neurons in the inner layer.

Batch Size Tuning

Batch size refers to the amount of datapoints a network is fed at one time during a training epoch before updating the neuron weights. Large batch sizes can significantly reduce training time but may reduce accuracy [54]. Smaller batch sizes on the other hand are computationally more expensive and take exponentially longer to train but can improve results. A first, coarse, tuning of the batch size ranging from 1000 to 10,000 in steps of 1000 is shown in Figure 5.12a. It can be seen here that indeed the reconstruction error seems to decrease along with the batch size.

A more detailed finetuning is then performed of smaller batch sizes in the lower batch size region was run, the results are shown in Figure 5.12b. Tuning was performed for batch sizes ranging from 25 to 200 in steps of 25, with the resulting error being the average of training the network twice. This is done to account for the stochastic network initialisation, which can give slightly different results every time it is trained. From this finetuning it can be seen that the error continues to decrease significantly as the batch size decreases. The gains decrease as the batch size becomes smaller however, and are even seen to slightly increase for size 50. For this reason, it was chosen to set the batch size to 75 which allows for highly accurate reconstruction, yet trains 3x faster than the slightly more accurate 25 batch size training process. Smaller batch sizes could be further investigated but given the size of the dataset (120,000 datapoints) this is not considered feasible for training.



(a) Network reconstruction error vs batch size

(b) Batch size finetuning

Figure 5.12: Batch size tuning process

Loss Function

As described before, the loss function is the measure the network uses to determine if it is improving performance during training. In this case, where a numerical input of 100 points needs to be replicated as close as possible to the input dataset, it is logical to use an evaluation metric for a regression model. However some probabilistic measures used in machine learning classification problems will be investigated as well for completeness. The aforementioned methods, 4 regressive and 2 probabilistic, are defined as:

- Mean Squared Error (MSE) determines the average squared difference between the reconstructed and original signals.
- Mean Absolute Error (MAE) determines the average absolute difference between reconstructed and original signals.
- Mean Squared Logarithmic Error (MSLE) determines the average squared logarithmic difference between reconstructed and original signals.
- Mean Absolute Percentage Error (MAPE) determines the average percentage difference between reconstructed and original signals.
- Cross-Entropy (CE) determines the difference between probability distributions in classification problems. Categorical CE is typically used for multi-class classification, and Binary CE is typically used for binary classification problems, as is the case here.
- Kullback-Leibler Divergence (KLD) also measures probability distribution differences similar to the cross entropy metric, but also measures information loss. It is not particularly suitable to the reconstruction problem but is included for completeness.

The difference in performance between these methods is shown in Figure 5.13. For this application the MSLE loss function was used as it showed the best performance by far. This is not surprising, as the goal is to encourage accurate reconstruction of as many of the points of the signal as possible. The MAE and MSE also show good performance, yet one order of magnitude worse than the MSLE function. This is likely due to the logarithmic term in the MSLE metric which works very well in penalising small errors in the reconstruction.

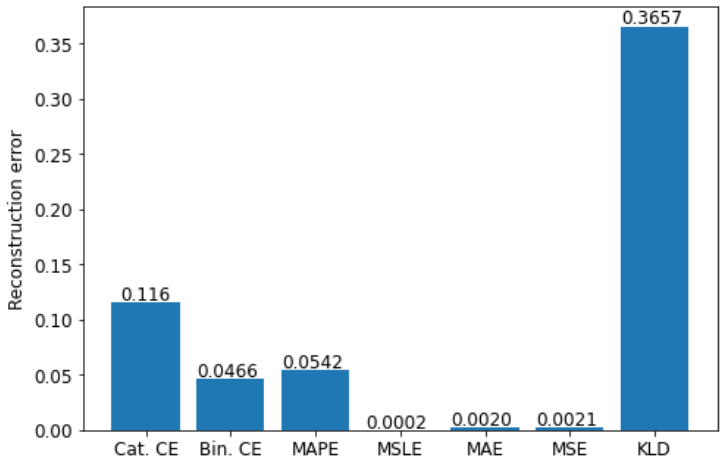


Figure 5.13: Loss function selection

Therefore, when assessing reconstruction error ϵ_{rec} between an original signal and the reconstructed signal, the MSLE will be used as shown in Equation 5.10.

Latent Representation Size Tuning

An autoencoder learns to reconstruct inputs based on the latent representation. Therefore the size of this representation is extremely influential in the performance of the network. As the input vector is 100 datapoints and the network should learn to recognise few key features out of the 100 points, it was chosen to tune the latent representation between 4 and 20 neurons. The lower limit of 4 was chosen as it is unlikely the network will be able to capture any features with less neurons. The upper limit of 20 is chosen since the second layer contains 64 neurons, and the reduction in information from 64 to 20 is roughly a factor three. Any more neurons in the latent space will likely not be able to capture more features from the second hidden layer.

The results of the tuning are shown in Figure 5.14. The error rapidly decreases with increasing latent space size up until around 12 neurons, after which the gain becomes relatively small. Therefore the latent representation size is set to 12 neurons.

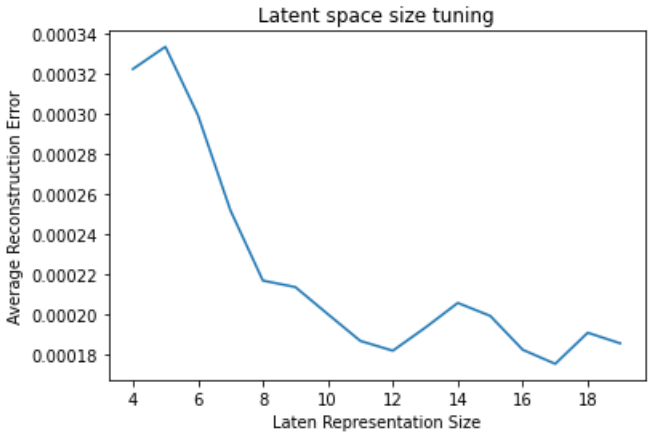


Figure 5.14: Latent representation size tuning

Optimiser

The optimiser function, which helps the network converge on an optimum, is the ADAM function. ADAM combines the advantage of the Adaptive Gradient Algorithm (AdaGrad) and the Root Mean Square Propagation (RMSProp) which have often been used in machine learning [33]. It is an improvement on the classical stochastic gradient descent process used before, and is widely considered the best optimiser function for most applications [51].

5.4.4. Final Network Architecture

Following the tuning of the hyperparameters mentioned before, the final network architecture is now designed and is summarised in Table 5.2. Aside from the tuned parameters, some design choices were made. Each layer has a dropout of 10%, meaning during training this amount of neurons in each layer are disregarded. This is often used to prevent overfitting and mutually dependent neurons, and is a standard practice in machine learning [57].

Parameter	Value
Number of encoding layers	2
Number of decoding layers	2
Neurons in encoding layer 1	260
Neurons in encoding layer 2	64
Activation function	ReLu
Activation function output	Sigmoid
Neurons in latent space	12
Neurons in decoding layer 1	64
Neurons in decoding layer 2	260
Dropout (all layers)	0.1
Number of training epochs	10
Batch size	75
Optimiser function	ADAM
Loss function	MSLE

Table 5.2: Autoencoder hyperparameters

6

Results

Having designed the network and tuned the hyperparameters, the fault detection capabilities are tested and discussed in this chapter. In section 6.1 the detection results of single signal level faults in LUMIO are presented, followed by the detection results of model-based faults in LUMIO in section 6.2. This is then validated using real satellite telemetry from OPS-SAT in section 6.3. The results are analysed and discussed shortly in section 6.4. Finally a short discussion about the computational performance of the network is found in section 6.5.

6.1. Detection of Signal Level Faults in LUMIO IMU

As a first test for this network, the signal level faults discussed in chapter 4 (step, noise and outliers) are introduced into the LUMIO IMU data. Although, as mentioned before, these fault types do not typically require advanced model-based detection methods, the supposed benefit of using the autoencoder fault detection system would be that a single network could replace a large number of different statistical, frequency, and time domain related measures which are often used to detect such faults. It would then provide a uniform 'black box' approach to AOCS fault detection, simplifying the FDIR design as well.

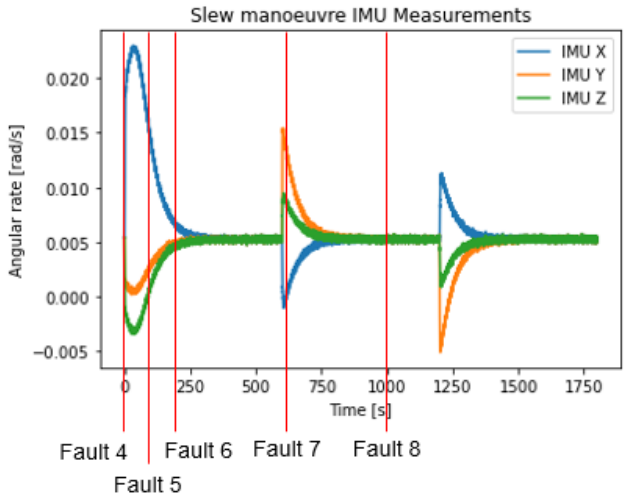


Figure 6.1: Fault locations in LUMIO IMU slew data. At each location, a step, noise and outlier fault are inserted once.

The step, noise, and outlier faults are introduced in the LUMIO IMU data at five strategically chosen locations across the data for slewing manoeuvres and tracking, shown in Figure 6.1.

The locations are chosen to ensure enough representative scenarios of all kinds are included. This includes highly dynamic scenarios with higher magnitudes and steeper gradients such as the slewing in Fault 4, Fault 5, and Fault 7 as well as more stable or even flat scenarios such as Fault 6 and Fault 8. At each location, each signal level fault type will be introduced separately and tested on the trained autoencoder network. This means in total 15 test faults will be fed into the network.

The results of this test are the reconstruction errors for each of these faulty signals shown in Figure 6.2. It can be seen that this type of network is not at all performant in detecting outliers in data: only two out of the five outlier faults were detected, and they only just reach the detection threshold. This is not surprising, considering a faulty signal with a single outlier is very similar to a nominal signal being fed into the network. Therefore the network is easily able to reconstruct this signal accurately and the reconstruction error will be very low, as only the outlier is inducing a (comparatively) small error.

The detection of erratic signal behaviour and step faults proves much better and all faults of this type were detected. Contrary to the outlier faults, here the error is much more persistent and therefore the reconstruction error will typically be higher compared to a nominal signal. The network has not learnt to reconstruct these signals, which leads to the network reconstructing a nominal signal from a faulty signal or a non-nominal signal from a faulty signal. Both will lead to a high reconstruction error, indicating the presence of a fault.

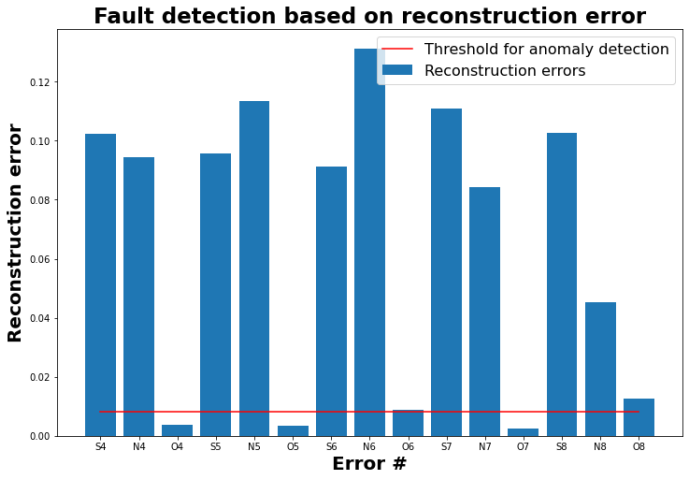


Figure 6.2: Slew manoeuvre reconstruction errors LUMIO IMU (S = step, N = noise, O = outlier)

6.1.1. False Alarm Rate

While Fault Detection Rate (FDR) is an important metric in assessing the performance of a fault detection system, one should avoid the false detection of faults in nominal situations (false positives) as well. False alarms risk initiating unnecessary recovery actions, may lead to the removal of healthy units from operation or may even trigger a safe mode where none is required. This reduces availability of the mission, which is highly undesirable for high autonomy spacecraft such as LUMIO which see little and irregular ground contact.

The metric to asses false alarm performance is the False Alarm Rate (FAR), which measures the amount of false positives (fault when there is none) out of a set of negative samples (nominal case). This set is a randomly chosen sequence of 1500 seconds which is fed into the network. Based on the work of Gelmi [22] regarding the LUMIO FDIR design, in order to trigger a false alarm one has to report a fault three times in a row when there is none present.

Therefore a single false positive is acceptable as it would not trigger recovery actions, given the next two results are also not false positives.

As displayed in Figure 6.3, out of the 1500 seconds, 6 false positives (0.4%) were triggered based on a 1σ detection threshold. This is acceptable and would not trigger any recovery actions. However, some of these reconstruction errors are very large even through the signal is nominal. Therefore ideally the detection threshold should be raised to avoid false alarms. Yet, as is shown in Figure 6.2 some of the lower reconstruction errors for faulty signals (N8) are in the same range (0.04) as those of nominal signals. This means that increasing the threshold means risking even more missed detections, but decreasing the threshold leads to a high false alarm rate. Both are undesirable and further confirm this method is not suitable for detecting such faults.

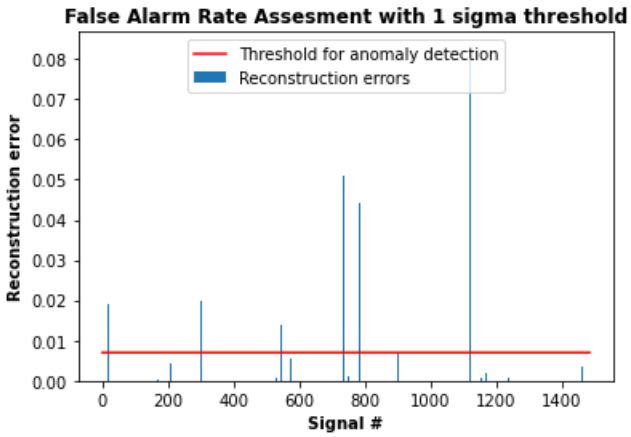


Figure 6.3: False Alarm Rate Assessment for signal processing - 1σ threshold

It should be noted that other methods exist to detect these faults such as running variance, running mean, derivative spikes and so on. Therefore this application of feeding signals straight into the network as-is will not be further explored and is not considered a good implementation of the autoencoder network in spacecraft fault detection.

6.2. Model-Based Fault Detection in LUMIO Data

A more promising application is using the autoencoder to capture more complex faults (drift, bias, loss of accuracy). As a specific example, fault F26 from the LUMIO fault register in Appendix C "*IMU reading inaccurate (drift, bias..)*" will be used here. Such fault cannot be caught from signal analysis alone as it does not have distinct features such as a step or an outlier spike. Therefore it is in need of comparison to other measurements. Currently, this is done through n-modular voting in larger spacecraft, and not performed at all in smaller spacecraft, or perhaps using a simple version of a model-based approach. The model based approach relies on complex numerical dynamic models of the spacecraft as well as a highly accurate state space representation. These methods are not immediately accessible to CubeSat developers and suffer from convergence issues [71]. These methods are also not very transferable between spacecraft as they are highly dependent on spacecraft architecture, physical characteristics, the environment it operates in, and the mission profile. Therefore in this section a solution is proposed, inspired by the work done by Zhang et al. [73] which uses signature matrices for fault detection in power plants.

6.2.1. Signature Matrix Method

The signature matrix as used in this thesis is a method to correlate telemetry time series to each other and form an abstract image of the entire AOCS system dynamics. From this image, the hypothesis is that an autoencoder network can learn what patterns constitute nominal behaviour and then detect anomalous behaviour. Please note that it is not to be confused with *time series signatures* from the theory of controlled differential equations, often used in machine learning applications for character recognition and language processing.

In time series data of the LUMIO AOCS, the signature matrix forms an image which correlates the rate measurements of the IMU with the momentum loading of the reaction wheels. These seven measurements form a 7x7 matrix, where every element of the signature matrix at position i, j is signature m_{ij}, which is the dot product of two time series X_i^t and X_j^t over a time period t:

$$m_{ij} = \vec{X_i^t} \cdot \vec{X_j^t} \tag{6.1}$$

The result of this operation is a 7x7 matrix, with an example shown in Table 6.1. As this large matrix of numbers is not very meaningful, the signature matrix will henceforth be represented as a heatmap, with colour scales indicating the magnitude of each cell. Such an example of a heatmap signature matrix for LUMIO IMU and Reaction Wheel is shown in Figure 6.4 along with the row and column corresponding to each of the time series. The heatmap representation is used in order to make the content of the matrix easier to interpret, rather than a 7x7 matrix of numbers. For clarification an example is given: in the matrix seen in Figure 6.4 it can be seen that element (4,5) represents the dot product of the RW1 and RW2 time series respectively, with the normalised outcome of this product being close to 0.75 (see colour scale).

	IMU-x	IMU-y	IMU-z	RW1	RW2	RW3	RW4
IMU-x	2.224228 E-08	2.224291 E-08	2.224204 E-08	1.270449 E-08	1.376678 E-08	-3.179112 E-09	-6.829204 E-09
IMU-y	2.224291 E-08	2.224367 E-08	2.224262 E-08	1.461977 E-08	1.645449 E-08	-3.827202 E-09	-8.248626 E-09
IMU-z	2.224204 E-08	2.224262 E-08	2.224181 E-08	1.195889 E-08	1.272047 E-08	-2.926812 E-09	-6.276636 E-09
RW1	1.270449 E-08	1.461977 E-08	1.195889 E-08	2.971280 E-05	4.169551 E-05	-1.005408 E-05	-2.201994 E-05
RW2	1.376678 E-08	1.645449 E-08	1.272047 E-08	4.169551 E-05	5.851193 E-05	-1.410892 E-05	-3.090079 E-05
RW3	-3.179112 E-09	-3.827202 E-09	-2.926812 E-09	-1.005408 E-05	-1.410892 E-05	3.402138 E-06	7.451123 E-06
RW4	-6.829204 E-09	-8.248626 E-09	-6.276636 E-09	-2.201994 E-05	-3.090079 E-05	7.451123 E-06	1.631933 E-05

Table 6.1: Example LUMIO signature matrix (lunar tracking, 50 second frame taken at t= 5000s): each number represents the dot product of the telemetry stream corresponding to its row and column respectively

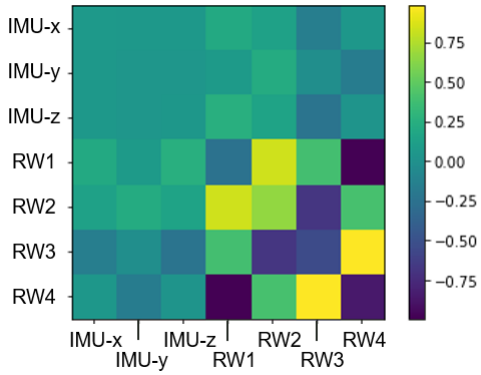
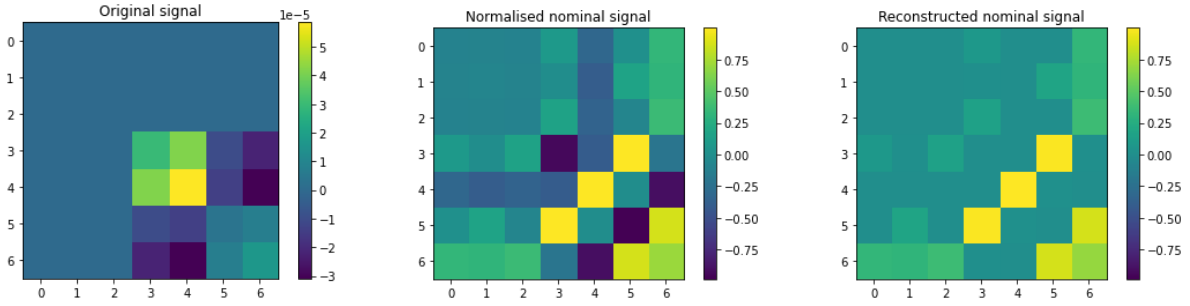


Figure 6.4: LUMIO Normalised Signature Matrix example with telemetry locations

This matrix can be used for fault detection by generating these signature matrices for all training data and training the autoencoder to accurately reconstruct them. This reconstruction process is visually demonstrated in Figure 6.5. The matrix is first normalised, as shown in Figure 6.5b to better express all the features in the matrix. Then the autoencoder reconstructs this matrix, shown in Figure 6.5c. It can be seen that visually, they are similar and the autoencoder managed to reconstruct the main features of the original matrix. This is also reflected in the reconstruction error ϵ_{rec} of $2.13x10^{-5}$ (MSLE).



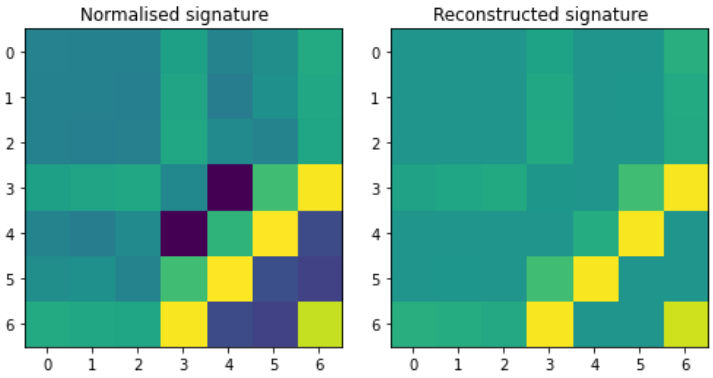
(a) Fault-free telemetry signature matrix LUMIO at arbitrarily chosen time

(b) Preprocessed (normalised) nominal signature matrix ready for processing by autoencoder

(c) Autoencoder reconstructed nominal signature matrix. $\epsilon_{rec} = 2.13x10^{-5}$ (MSLE)

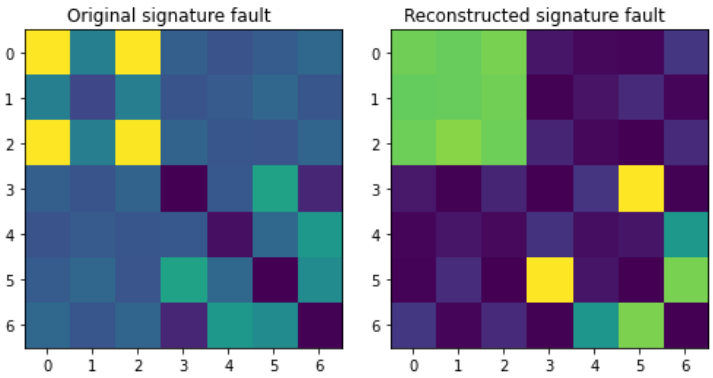
Figure 6.5: Autoencoder reconstruction process demonstrated for nominal signal

The network is proficient at reconstructing a signature matrix of the AOCS system in nominal conditions, such as another example signature matrix shown in Figure 6.6a. When a fault (in this example a drift in the measured angular rate of the x-axis and z-axis) is introduced into the exact same timeframe, as seen in Figure 6.6c, one can see the matrix changes significantly. The autoencoder is able to reconstruct nominal condition signature matrices accurately, as shown in Figure 6.6b, but cannot accurately do so for signature matrices where a fault is present, shown in Figure 6.6d: the reconstruction error is 4 orders of magnitude larger.



(a) IMU and Reaction Wheel telemetry signature matrix under nominal conditions

(b) Autoencoder reconstructed nominal signature matrix. $\epsilon_{rec} = 2.11x10^{-5}$ (MSLE)



(c) Signature matrix with induced drift fault in LUMIO IMU telemetry

(d) Autoencoder reconstructed fault signature matrix $\epsilon_{rec} = 0.611$ (MSLE)

Figure 6.6: Autoencoder reconstruction of nominal and faulty signature matrices (LUMIO, 7x7 matrices)

It can also be hypothesised that using these signature matrices and real fault data, patterns can be extracted from the matrices for fault isolation. To do this, one needs to know the signature of a specific fault. Due to the lack of available real-life fault data this is however a recommendation for future work and not performed in this thesis.

6.2.2. Fault Detection Results LUMIO Data

As a first step, this method is tested using the available LUMIO data from the three IMU channels and the four reaction wheel momentum loading datapoints. In the nominal LUMIO telemetry, a range of faults is introduced (one by one) as can be seen in Figure 6.7. It is clear that every type of fault introduced in the telemetry generates a specific pattern that is recognisably different from the nominal signature matrix shown in Figure 6.5a. The network then learns to reconstruct the nominal behaviour signature matrices in the training process, after which the detection threshold for anomalies is set using the 1σ approach outlined in section 5.4.

When reconstructing the signature matrices, it can be seen in Figure 6.8 that the reconstructions do not resemble the original matrices from Figure 6.7 at all. Not only visually, but also when calculating the reconstruction error ϵ_{rec}. The anomaly detection results are shown in Table 6.2. The reconstruction error for each fault and the threshold are visualised in Figure 6.9. Here it is seen that especially those simulated fault situations which manifest in multiple symptoms (fault 4, 8 and 12) show especially good detection performance compared to the single fault situations. This indicates the system is not easily confused by multiple symptoms presenting and can detect these situations without ever having encountered them before.

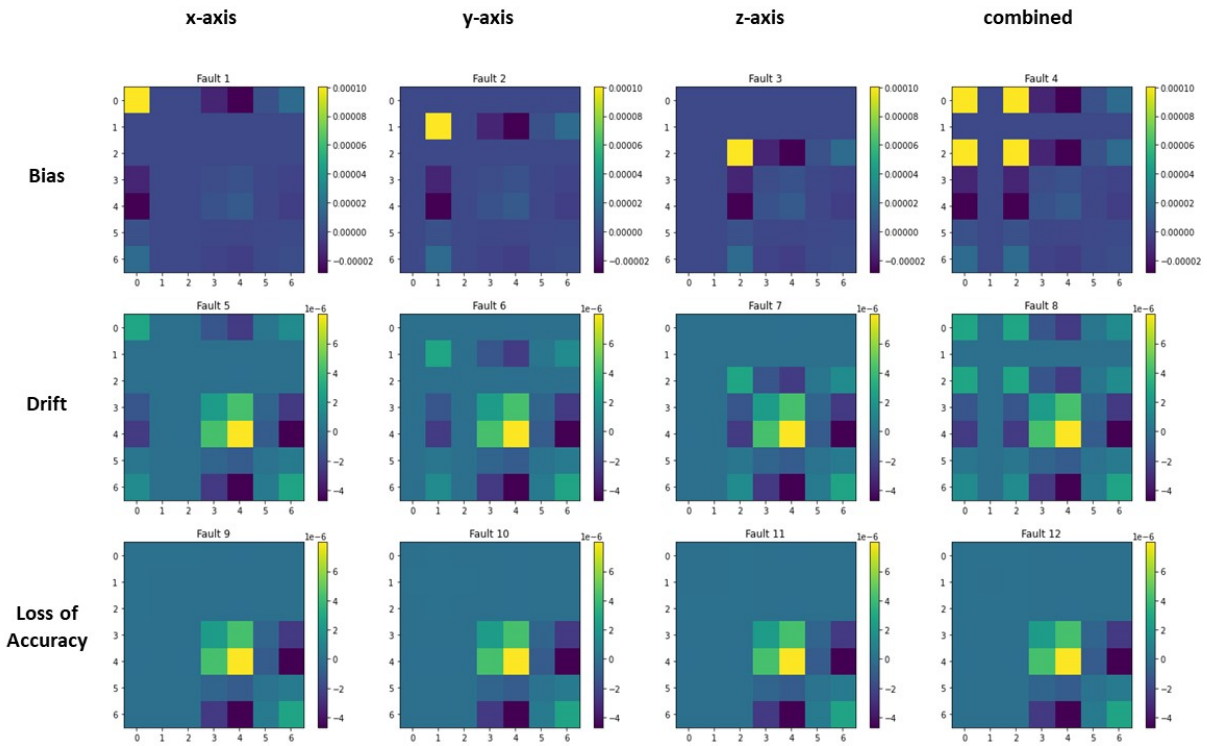


Figure 6.7: Signature matrices of bias, drift, and loss of accuracy faults introduced in LUMIO IMU signals

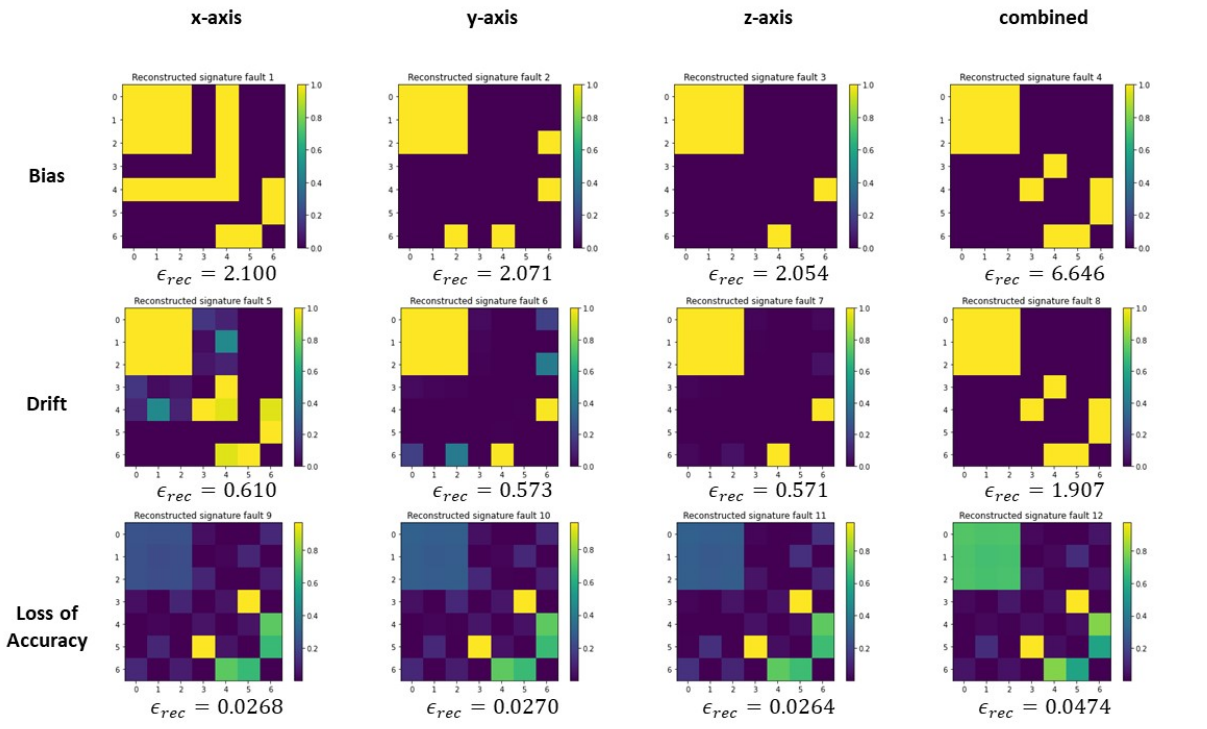


Figure 6.8: Signature matrices of Figure 6.7 reconstructed by autoencoder including MSLE reconstruction error

As single faults, the drift and bias faults are most easily detected when looking at the reconstruction error, the loss of accuracy faults are the hardest to detect using this method. This is not surprising when looking at the faulty signals and the subtle differences between the faulty

and nominal case in the loss of accuracy signals, as shown in Figure 4.2 of chapter 4. However, the error is still two orders of magnitude above detection threshold, meaning even at the subtle level of the loss of accuracy fault the detection is accurate.

Fault Type	ID	Measurement	Size	Detection
Bias	1	Angular rate x-axis	0.01 rad/s	✓
	2	Angular rate y-axis		✓
	3	Angular rate z-axis		✓
	4	Angular rate x-axis + z-axis		✓
Drift	5	Angular rate x-axis	0.0005 rad/s^2	✓
	6	Angular rate y-axis		✓
	7	Angular rate z-axis		✓
	8	Angular rate x-axis + z-axis		✓
Loss of Accuracy	9	Angular rate x-axis	x1.75	✓
	10	Angular rate y-axis		✓
	11	Angular rate z-axis		✓
	12	Angular rate x-axis + z-axis		✓

Table 6.2: Detection results IMU faults using signature matrices for LUMIO (IMU + RW, 7x7 matrices)

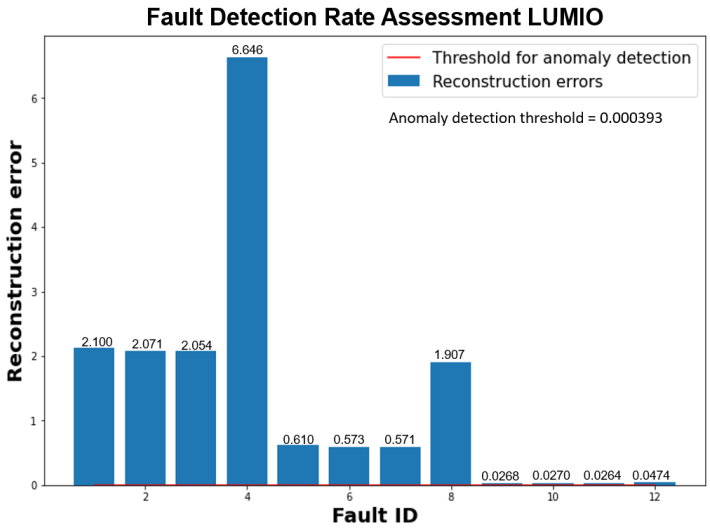
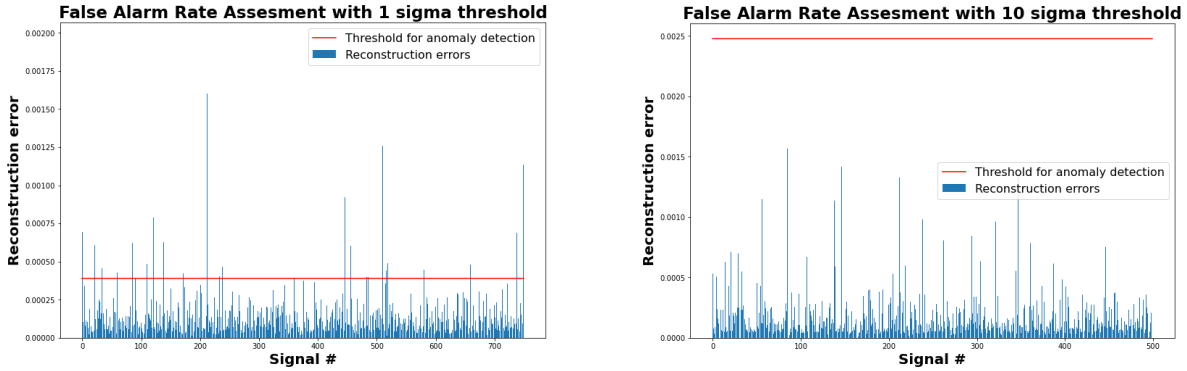


Figure 6.9: Reconstruction errors for LUMIO faults compared to detection threshold

6.2.3. False Alarm Rate

The results of 700 seconds worth of nominal telemetry being fed in the network are shown in Figure 6.10a. It can be seen that on 21 of the 700 instances (3%) a fault is falsely reported, but none occur in sequence of three in a row thus not triggering a false alarm.

Note that using the faults defined in Table 6.2, the fault detection rate is 100%, meaning the threshold could be increased to reduce false positive detection while still remaining accurate in true positive detection. The reconstruction errors for nominal telemetry are two to four orders of magnitude above the detection threshold: the lowest reconstruction error is 0.0264 for Fault 11, still an order of magnitude larger than the highest false alarm reconstruction error which is around 0.0015. Then, if the threshold is increased from 1σ to 10σ, the false alarms are avoided altogether as shown in Figure 6.10b while all faults are still detected.



(a) False alarms with 1 σ detection threshold

(b) False alarms with 10 σ detection threshold

Figure 6.10: LUMIO False Alarm Rate in 700 seconds of nominal telemetry

6.3. Model Based Fault Detection in OPS-SAT Telemetry

Although the previously described results are promising for highly accurate model-based fault detection, the LUMIO data is simulated and only represents a few operational scenarios. It cannot be considered as an accurate replacement for real satellite telemetry. As a next step in testing this fault detection method, the telemetry from ESA's OPS-SAT was used. Although the faults are still introduced artificially, the underlying data is now real AOCS telemetry from an orbiting CubeSat. The dataset and fault engineering performed is described in section 4.3, a detailed image of each fault can be found in Appendix F. The signature matrix and the corresponding telemetry locations in the matrix for OPS-SAT is shown in Figure 6.11.

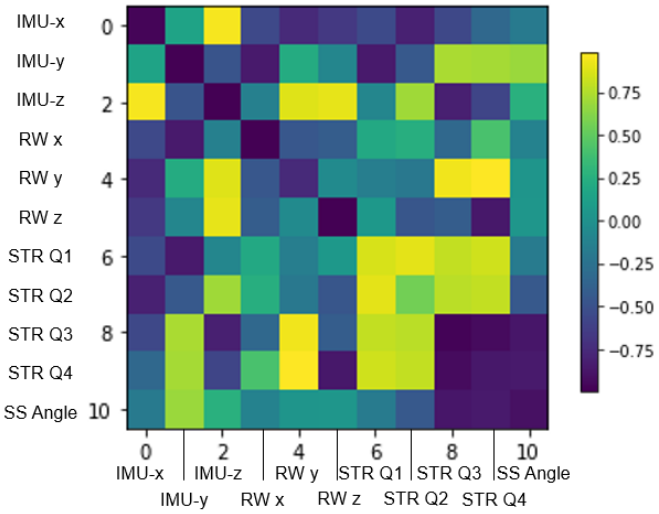


Figure 6.11: OPS-SAT Signature Matrix

6.3.1. Fault Detection Results OPS-SAT

In order to test detection capabilities of the signature matrix method for the model-based faults, the faults are introduced and the signature matrices generated, shown in Figure 6.12. The reconstruction errors for each of the faults as well as the detection threshold can be seen in Figure 6.14. The reconstructed matrices as well as the reconstruction errors are seen in Figure 6.13, and the results are shown in Table 6.3. It can be seen that the system performs as expected in detecting all of these faults, far above the detection threshold.

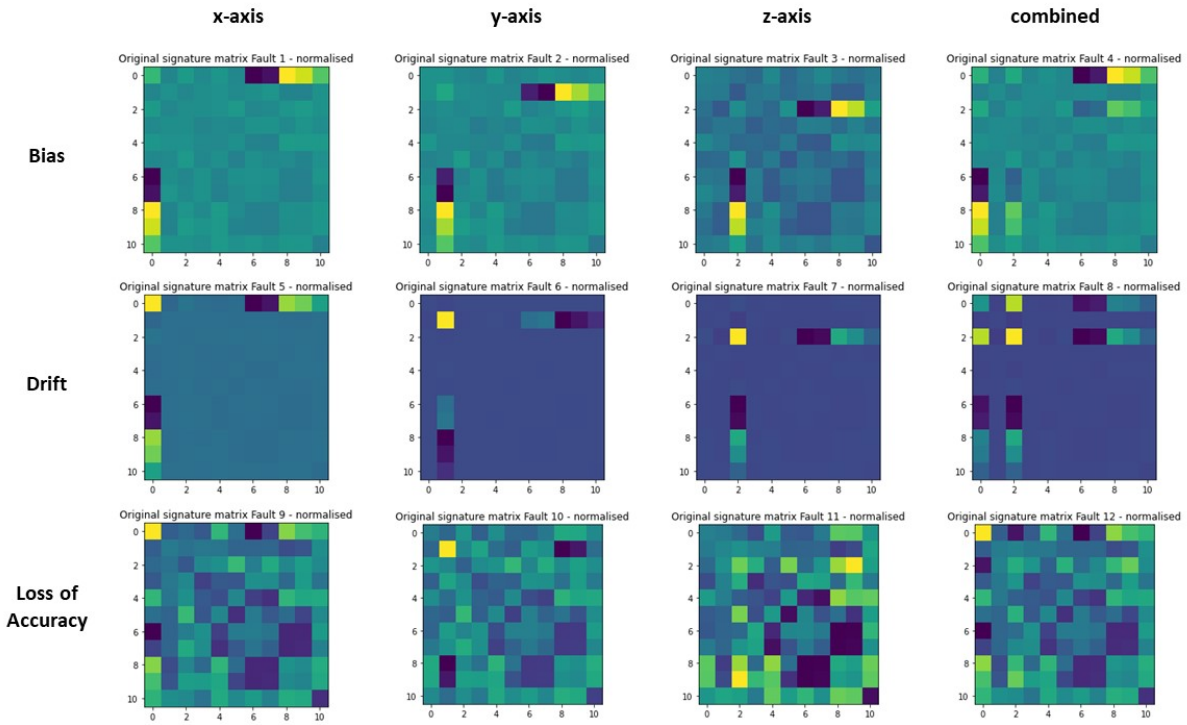


Figure 6.12: Signature matrices of bias, drift, and loss of accuracy faults introduced in OPS-SAT signals

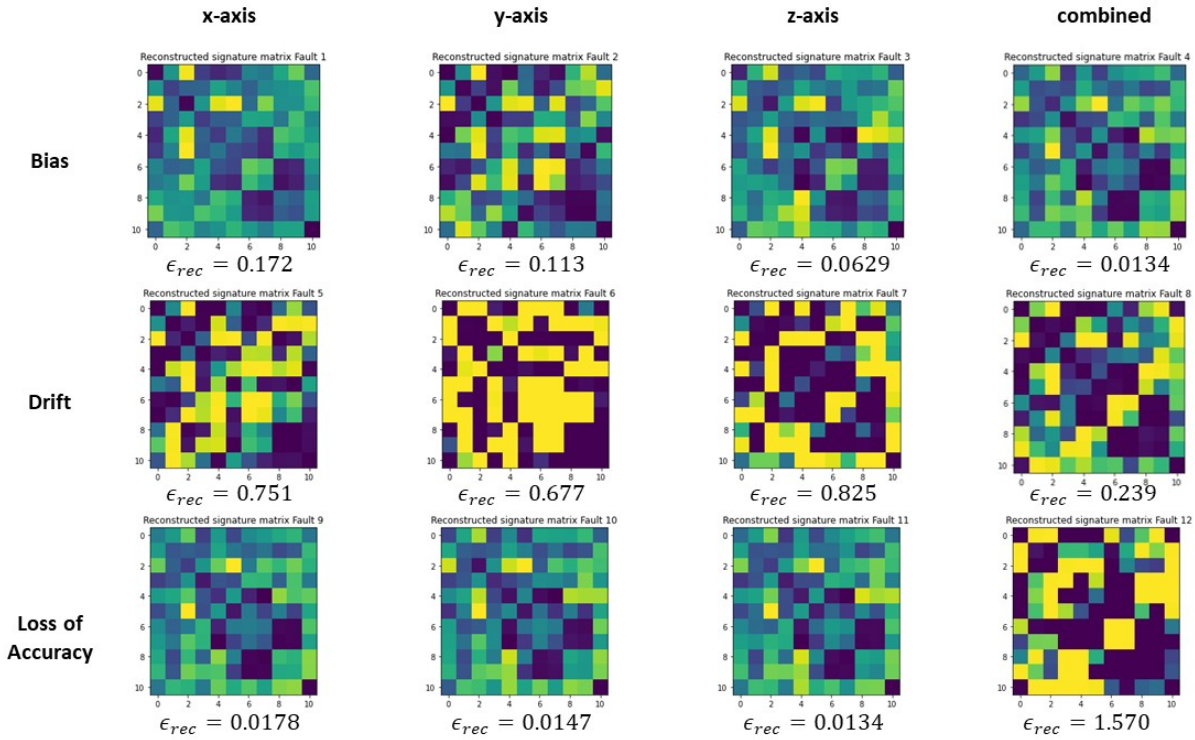


Figure 6.13: Signature matrices of Figure 6.12 reconstructed by autoencoder, including reconstruction error ϵ_{rec}

Fault Type	ID	Measurement	Size	Detection
Bias	1	Angular rate x-axis	0.01 rad/s	✓
	2	Angular rate y-axis		✓
	3	Angular rate z-axis		✓
	4	Angular rate x-axis + z-axis		✓
Drift	5	Angular rate x-axis	0.0005 rad/s^2	✓
	6	Angular rate y-axis		✓
	7	Angular rate z-axis		✓
	8	Angular rate x-axis + z-axis		✓
Loss of Accuracy	9	Angular rate x-axis	x1.75	✓
	10	Angular rate y-axis		✓
	11	Angular rate z-axis		✓
	12	Angular rate x-axis + z-axis		✓

Table 6.3: Detection results IMU faults using signature matrices for OPS-SAT (extended telemetry, 11x11 matrices)

Noticeable is that the detection is still easily performed: all faulty signals show a reconstruction error of at least one order of magnitude more than the detection threshold. However, compared to the LUMIO results the errors are much more close to the threshold. It is assumed that this can be attributed to three separate causes:

- **Same network, more input:** the network was not tuned to suit the OPS-SAT data inputs, which have a size of 121 elements whereas the LUMIO signatures were of size 49. Therefore, it makes sense that the reconstruction accuracy of nominal signals is worse, hence a higher reconstruction threshold.
- **Reconstruction errors are less sensitive to anomalies:** The increase in matrix size means the reconstruction error is influenced less easily if only a part of the matrix is reconstructed inaccurately. If 10 of the 49 values in the matrix were inaccurately reconstructed before, this will influence the calculation of the MSLE. However, if those same 10 points are reconstructed inaccurately out of 121 values, the MSLE will change by a smaller amount.
- **Different operational scenarios:** as the OPS-SAT data is taken from a more dynamic scenario, it is hypothesised that this is the cause for different fault types being detected better or worse. For example, the drift fault is consistently detected better than other faults in the OPS-SAT data, whereas this was the case for the bias fault in the LUMIO data. It could be that this is because in the more stable, steady LUMIO data a bias causes a much more explicit pattern in the signature matrix compared to the same bias fault in the OPS-SAT data.

However, it should be noted that despite the aforementioned issues, the system is shown to be robust and works even for an input double the size of the originally intended signature matrix.

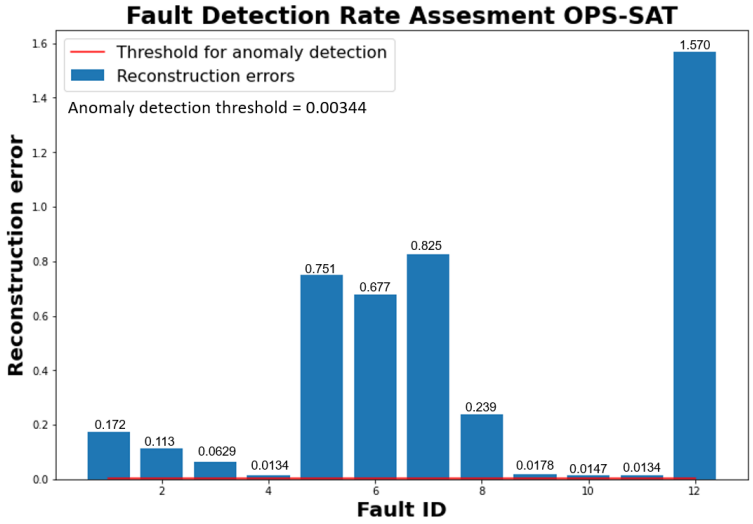
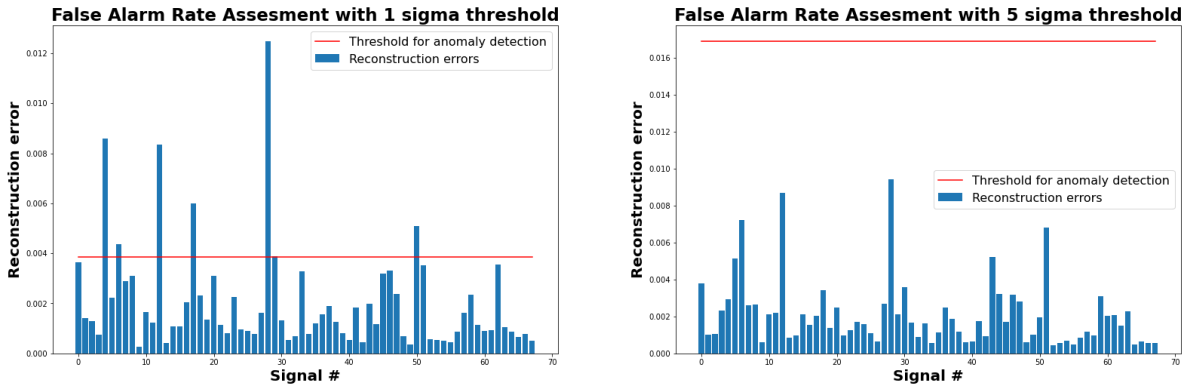


Figure 6.14: Fault detection results OPS-SAT Data

6.3.2. False Alarm Rate

Just as with LUMIO, the false alarm rate is also tested with OPS-SAT. Again a series of telemetry is fed through the network, with a false alarm being present if a fault is detected 3 times in a row. Given the reconstruction errors for faulty signals are now only one to two orders of magnitude higher than the nominal telemetry, the tuning of the threshold is a bit more sensitive. The initial 1σ approach shown in Figure 6.15a does not detect three false alarms in a row but still does have some single and double false positives. As a proof of concept a 5σ threshold is shown in Figure 6.15b which manages to remove any false alarm detection while still accurately detecting all faults. In practice, the value of the tuning factor n_{th} will have to be determined through trial and error, and may vary from 1σ to any number of standard deviations.



(a) False alarms with 1 σ detection threshold

(b) False alarms with 5 σ detection threshold

Figure 6.15: OPS-SAT False Alarm Rate in 30 minutes of nominal telemetry

6.4. Analysis of Results

As was discussed in section 6.1, the devised method of using an encoding-decoding neural network for fault detection in single signals is not considered a good use. The fault detection performance depends on how the fault manifests in the telemetry, and it is mostly unable to detect outliers. The false alarm rate is acceptable given a 1σ threshold, but reconstruction errors for nominal signals can be high and lead to further false alarm triggering. It is clear that this method is not suited to single signal fault detection.

When it comes to model-based fault detection in the LUMIO AOCS data, the results are much more promising. It can be seen from Figure 6.6 that the signature matrix indeed contains patterns which relate to nominal or non-nominal behaviour, and that the autoencoder can learn to reconstruct the key features of the nominal patterns easily. When trained on nominal matrices and then fed faulty signals, the reconstruction error immediately increases by a factor two to four orders of magnitude thus leading to highly accurate detection despite very subtle faults being presented. False alarms are absent at no cost to fault detection performance when selecting an appropriate threshold, which in the case of LUMIO was around 10σ.

As the data used for LUMIO is simulated, some real spacecraft telemetry from the OPS-SAT AOCS system is used as well. This showed that the method is effective even with 11 noisy telemetry streams from 4 different units (IMU, reaction wheels, star tracker and sun sensor) being used. The same kind of detection accuracy and false alarm rates were found as with LUMIO. Given the more complex dataset and less datapoints available the reconstruction error was not as high as with LUMIO for faulty situations, but still still strongly exceeding the detection threshold of 1σ and 5σ. It is able to detect all types of model-based faults in the data without raising false alarms and initiating unwarranted recovery actions.

6.4.1. Limitations

In order to fully understand these results, it is important to recognise the limitations of the proposed method as well. The accuracy of the fault detection approach depends on the quality of the training data, and if this data is fully representative of all viable mission scenarios. If not, a nominal operation could trigger high reconstruction errors and raise a false alarm. In the case of the IMU, this invalidates the rate reading, likely at a dynamic phase where accurate angular rates are critical for the AOCS to perform its task. This could be a risk to the mission, and needs to be addressed when training the network.

Another key limitation is that faults may express at different rates and scales, therefore the proposed approach of taking the past 50 seconds of telemetry may be too small to reveal gradually increasing faults, or may be too large to catch very short or intermittent faults. A potential solution is to generate a signature matrix of a small timeframe (e.g. 5s), a medium timeframe (e.g. 50s) and a large timeframe (e.g. 300s) in order to detect different types of faults.

6.5. Computational Resources

Onboard a spacecraft, resources are limited. This applies to the processing units as well, who deal with limited power, limited computational processing power, limited memory and finite amounts of time to run certain algorithms. A concern with model-based FDIR algorithms is their large and complex models which consume a lot of these resources, and can suffer from numerical convergence issues. Although neural networks do not suffer from the same convergence issues, it is true in general that computational power is a limiting factor for space applications [54]. Therefore, this thesis has considered this constraint from the start and de-

signed a simple, lightweight network.

It would be highly desirable to asses the viability of running such a neural network on a spacecraft processor and accurately predict the resource consumption (power, time, FLOPS). However, testing algorithms on the flight hardware is the only certain way to come up with a these numbers. Therefore, a qualitative estimation is performed instead, by comparison to the current state of the art. Following this a rough quantitative estimation of the number of operations is performed.

6.5.1. Comparison to State of the Art

The LUMIO processing unit in phase A is determined to be the ISISpace IOBC, and the payload data processor is the UNiBap iX5 [4]. These processors can be considered representative for use in CubeSat architectures, and have been applied in multiple missions. Ubotica, an Irish company specialising in AI and edge computing in spacecraft, has proven flight heritage using a processor with similar capabilities running a neural network (MobileNetV2) consisting of millions of neurons [36]. The system proposed in this thesis consists of only 660 neurons in the current architecture. Other than the network, the preprocessing work is relatively light, taking a 7x7 or 11x11 input image and normalising this. Although this is no full proof that the network can be run, it is reason to believe the resources required to run it are very small compared to the advanced networks which have been flown, or for example a high-accuracy AOCS model or parameter estimation algorithm.

On top of this, with more attention being paid to AI in space applications (such as with Ubotica) the hardware is becoming increasingly tailored for running these networks efficiently and quickly [54]. Therefore it is believed that the computational resources will not be a problem for this fault detection mechanism.

6.5.2. Estimation of Number of Operations

As a confirmation, a rough estimation was made of the computational resources required. The number of operations required to run a single pass of a network can be calculated assuming each neuron performs one multiplication for every input (weight x input), and a total number of additions which is one greater than the number of inputs (sum of weighted inputs + bias). Knowing this and the layer topology, an estimated $8.08x10^6$ operations are required to run a single pass excluding preprocessing.

The LUMIO OBC is based on the ARM9 Processor [30], with a clock speed of 400 MHz and one instruction required to perform addition or multiplication. This implies that the network (excluding preprocessing and other activities) would take 2% of the processor resources if run every second. However, depending on the needs the algorithm could be run once every 10 seconds which requires only 0.2% of the computing power. This shows that the designed network is sufficiently lightweight that it will not impact computing requirements or scheduling of other tasks in the processor.

6.6. A Note on Verification and Validation Activities

Machine learning based methods are notoriously hard to perform verification and validation activities on [54]. They may be prone to bias and overfitting without this being noticeable when testing with the training data. It is difficult to predict how the network will perform when faced with real, unseen, satellite data, and even then there may be some special cases in which the network shows undesired behaviour.

In this thesis, verification was performed using the simulated LUMIO data and the OPS-SAT telemetry. However, this is not sufficiently thorough for application in space. Due to the lacking data and the limited timeframe available for this thesis further verification data is not sought but should be pursued in the future.

Validation of such systems require a real, integrated, CubeSat AOCS system in a representative environment. Flatsat testing, where the systems are integrated electronically but not necessarily mechanically, could be a good starting point for this type of testing. However, the only way to perform real validation would be on a demonstrator as the complex relations between the AOCS sensors and actuators as well as the spacecraft dynamics are near impossible to accurately simulate on ground. On top of this, faults are needed to test the system. These faults can come from many sources and are unpredictable. A representative set of test faults which cover as many root causes as possible can be thought of by experts, but there will never be 100% certainty that all bases are covered. It is one of the key challenges in FDIR, and it is a challenge that is not inherent to the designed autoencoder but to all FDIR systems.

7

Conclusion

7.1. Conclusion

As a starting point of this thesis, the research objective was defined as:

To contribute to the improvement of deep space CubeSat reliability and failure robustness by designing a model-based FDIR approach for LUMIO's AOCS subsystem.

A Fault Tree Analysis and FMECA on the Phase A design of the LUMIO spacecraft AOCS revealed 53 faults of which 20 critical faults which could put the mission at risk. Of these 20, half cannot be detected using basic fault detection mechanisms such as signal processing, but rather requires a model-based approach. A trade study of these model-based methods revealed a neural network based approach could be suitable for performing this task, given its high accuracy and low complexity compared to intricate nonlinear dynamics models. Additionally, it was found that neural network based applications have not been implemented in many missions due to their lack of flight heritage and the difficulty in verifying and validating them. A CubeSat is a highly suitable platform for demonstrating such methods and removing these barriers, laying the groundwork for potential application of neural network based fault detection in other mission types.

A lightweight autoencoder network was designed and tuned for reconstructing its inputs without supervision. It was found that when training such autoencoder on nominal LUMIO telemetry from IMU and reaction wheels using signature matrices, it can detect faults easily based on the reconstruction error. Even those subtle faults such as drifts or small biases in the order of milliradians, which induce up to 6 degrees of pointing offset, are detected in all signals of the LUMIO IMU. It was found that the detection threshold can be tuned such that the system fault detection rate is 100% for the engineered fault set, while no false alarms are triggered.

The concept was tested on real spacecraft telemetry coming from ESA's OPS-SAT, which includes telemetry from the sun senors, IMU, reaction wheels, and star trackers. The fault detection accuracy was again 100% for the engineered fault set while not triggering any false alarms. The reconstruction errors were 1-2 orders of magnitude closer to the threshold compared to the LUMIO errors, due to the larger input vector (11x11 compared to 7x7). The autoencoder based fault detection method is therefore considered a highly promising method for fault detection (and possibly isolation) in CubeSats and other missions, replacing complex, inaccessible, and sometimes unreliable nonlinear numerical dynamic models.

7.2. Answers to Research Questions

At the start of this thesis, three key research questions were asked which have been answered.

RQ1: What are the most critical failure modes of the LUMIO AOCS subsystem which can be detected, isolated and recovered?

The LUMIO AOCS system Fault Tree Analysis and Failure Mechanics, Effects and Criticality Analysis have revealed a number of failure modes with causes both internal and external to the subsystem. It was found that in going from the Phase 0 design to the Phase A design most critical faults were already mitigated. However, some are left such as AOCS unit faults during critical phases of the missions (detumbling, transfer). It was found that any inaccurate sensor reading which was not detected as faulty could risk the entire mission at any of these stages by providing incorrect information to the AOCS algorithm. This includes the angular rate readings from the gyroscopes, tachometer reading from each reaction wheel, sun angle readings from the sun sensors and quaternions supplied by the star tracker. Recovery options are limited to power cycling (IMU, reaction wheel tachometer) or switching to a redundant unit where possible (star tracker, sun sensor).

RQ2: How can the most critical faults be detected and isolated using model- based methods?

There are many model based fault detection methods available, each relying on different principles. Based on a concept exploration and trade study performed the outcome was that neural network based methods are considered promising and versatile option for detecting these faults. Given the lack of available fault data from spacecraft, it was decided that an unsupervised learning approach should be adopted in the form of a self-reconstructing encoder-decoder. This autoencoder learns to reconstruct nominal spacecraft data signatures in an accurate manner (low reconstruction error). When faced with faulty spacecraft data the autoencoder is not able to accurately reconstruct it and the reconstruction error increases by several orders of magnitude, indicating a fault is present in the system. This method uses the correlations within the data by processing it into a signature matrix first, where every element is the dot product of two telemetry streams.

It is hypothesised that these signature matrices can likely be used for fault isolation as each fault will present with a unique pattern. This was however not further investigated in this thesis.

RQ3: How accurate is the proposed method at detecting faults in the LUMIO AOCS system?

When tested on simple fault features in simulated LUMIO IMU and reaction wheel signals the system was able to detect steps of 0.002 rad/s and erratic behaviour with a standard deviation starting from 0.01 rad/s. However, the system was relatively inaccurate in detecting outliers of 0.1 rad/s and was not considered a good use for this application as simple signal processing methods can also detect these faults.

The signature matrix method performed much better: it was able to accurately detect all bias, drift and loss of accuracy faults introduced in the angular rate measurements. These faults were sized based on LUMIO pointing requirements and the early detection by the autoencoder assures the fault never leads to more than 6 degrees pointing offset from the desired attitude at the time of detection. It was found that since the reconstruction error for faulty signals is at

least two orders of magnitude larger than the nominal signals, the anomaly detection threshold can be tuned to avoid false alarms alltogether, making this system a highly reliable and accurate fault detection mechanism.

Since the LUMIO data was simulated, the system was tested on an extended telemetry set of OPS-SAT, an ESA built mission for demonstration of ground control software and missions operations concepts. When constructing signature matrices from OPS-SAT star tracker, IMU, reaction wheel and sun sensor data the fault detection performance was still 100% accurate using the same faults as with LUMIO. The false alarm rate could also be reduced by tuning the detection threshold, showing the autoencoder's promise in replacing complex, inaccessible and computationally expensive model-based fault detection methods for system with nonlinear dynamics.

7.3. Recommendations

Based on the outcomes of this thesis, a multitude of recommendations for future work can be made. These will be divided into four main focus areas: improvement of training and fault data, improvement of network capabilities, optimisation of performance for real-time fault detection, and validation activities.

7.3.1. Improvement of Training and Fault Data

A neural network's performance is highly dependent on the quality and diversity of data available for training. In this thesis, this was a significant challenge, as raw CubeSat AOCS data including real-life faults are not readily available. Therefore simulated data or downlinked telemetry from another satellite (OPS-SAT) were used. However, the main idea behind the designed network is that if real CubeSat data becomes available from deep space CubeSats flying today, the network can be trained on this and validation can be performed while also improving performance.

On the other hand, real spacecraft faults are unpredictable and cannot be modelled. If a fault occurs onboard a spacecraft with FDIR systems, it is detected and potentially recovered. This data would be very interesting to use in testing the network better and validating that it can indeed catch real spacecraft fault occurrences. For that reason, a recommendation is that in the long term operators of CubeSat platforms which are commercial or educational missions share all available telemetry that is not related to their specific mission and confidential or privileged.

7.3.2. Improvement of Network Capabilities

The network is now capable of detecting when a fault occurs which, using simple signal processing, would not have been detected. It was also shown that if a single fault manifests in multiple symptoms, this network still recognises the fault. However the second step in the FDIR process, fault isolation, is not yet performed. It is hypothesised that given the signature matrices one can use the signature matrix in a faulty case to isolate which unit or telemetry stream the fault is coming from. Based on this, different neural network based approaches could also be used to isolate the exact fault: some faults have a very specific signature in their measurements and should readily be recognised. Given these two steps together (advanced detection and isolation), the recovery process can be initiated with very high certainty that the correct fault has been identified and the appropriate recovery action can be taken.

The detection method could also be extended to detect mismatches in actuator output and commands. The signature matrix could include the commanded torques from reaction wheels

or desired pointing angles from the AOCS algorithm. If a discrepancy is present due to an offset or fault in the actuator, the system could potentially capture this as well which allows for rapid detection and protection of the actuators before critical damage is done to it.

Another promising application of this method is to use it in detecting faults which present in multiple symptoms. In prior work, and in general in FDIR, the assumption is made that at any given time only one fault occurs simultaneously. This makes sense considering the extremely small probability of these faults occurring, let alone within similar timeframes. However, a single fault can manifest in multiple symptoms. The method for detection proposed in this thesis is highly suited to detecting these faults, and the signature matrix can be used to couple them to their root cause if labelled datasets are available.

7.3.3. Optimisation

Finally, as with many neural networks, optimisation of a number of factors is still possible. For example, a detailed study into possible network architectures and hyperparameters is required, and a number of techniques such as convolution and pooling can be applied to further improve the autoencoder performance. Another recommendation is to quantify and optimise the power use and computational resources required to run this algorithm, as spacecraft onboard resources are typically limited and it is desirable to be able to run the FDIR system at any time without impacting other onboard services.

7.3.4. Validation Activities

The validation of a machine learning system such as the one designed in this thesis is difficult as the behaviour is not deterministic. Therefore the key recommendation is to obtain real data from integrated flatsat testing (all components are electronically integrated and communicate) as well as obtaining real fault data such that representative tests can be conducted. Real validation will only come from testing the system onboard a spacecraft in a representative environment, i.e. orbit. The recommendation is to further test the system using real telemetry from CubeSats where available, and then demonstrate the fault detection mechanism on a demonstrator such as the OPS-SAT platform. Testing will consists of first allowing the system to demonstrate it does not raise false alarms in nominal cases (assuming no faults occur during this test) and then artificially feeding a fault into the telemetry.

References

[1] Douglas Bernard et al. “Design of the Remote Agent experiment for spacecraft autonomy”. In: IEEE Aerospace Conference Proceedings, Apr. 1998. DOI: 10.1109/AERO.1998.687914.

[2] Felix Bidner. *Fault Tree Analysis of the HERMES CubeSat*. Tech. rep. Mar. 2010.

[3] Xu Botao et al. “Electromagnetic Isolation and Electrical Protection in Spacecraft General Assembly”. In: *MATEC Web of Conferences* 173 (Jan. 2018). DOI: 10.1051/matecconf/201817301037.

[4] Angelo Cervone et al. “LUMIO: A CubeSat for observing and characterizing micro-meteoroid impacts on the Lunar far side”. In: *Acta Astronautica* 195 (June 2022), pp. 309–317. DOI: 10.1016/j.actaastro.2022.03.032.

[5] Angelo Cervone et al. “Phase A Design of the LUMIO Spacecraft: a CubeSat for Observing and Characterizing Micro-Meteoroid Impacts on the Lunar Far Side”. In: 71st International Astronautical Congress Proceedings, Oct. 2020.

[6] Angelo Cervone et al. “Selection of the propulsion system for the LUMIO mission”. In: 72nd International Astronautical Congress, Oct. 2021.

[7] Nacer Chahat, Emmanuel Decrossas, and M. Michael Kobayashi. “Mars Cube One”. In: *CubeSat Antenna Design*. Ed. by Nacer Chahat. 1st ed. Wiley, Dec. 2020, pp. 35–89. DOI: 10.1002/9781119692720.ch2.

[8] Barbara A. Cohen et al. “Lunar Flashlight: Illuminating the Lunar South Pole”. In: *IEEE Aerospace and Electronic Systems Magazine* 35.3 (2020), pp. 46–52. DOI: 10.1109/MAES.2019.2950746.

[9] US Nuclear Regulatory Commission. *NUREG-0492, ”Fault Tree Handbook”.* 1981.

[10] Kasper De Smaele. *Deep Space CubeSat Fault Detection Isolation and Recovery - Literature Study in preparation of Master Thesis*. 2022.

[11] Valerio Di Tana et al. “ArgoMoon: There is a Nano-Eyewitness on the SLS”. In: *IEEE Aerospace and Electronic Systems Magazine* 34.4 (Apr. 2019), pp. 30–36. DOI: 10.1109/MAES.2019.2911138.

[12] E. Dotto et al. “LICIACube - The Light Italian Cubesat for Imaging of Asteroids In support of the NASA DART mission towards asteroid (65803) Didymos”. In: *Planetary and Space Science* 199 (May 2021). DOI: 10.1016/j.pss.2021.105185.

[13] Guillaume J.J. Ducard. *Fault-tolerant Flight Control and Guidance Systems*. Advances in Industrial Control. London: Springer London, 2009. DOI: 10.1007/978-1-84882-561-1.

[14] European Cooperation for Space Standardization. *ECSS-Q-ST-30-02C6 Space Product Assurance - Failure Modes, Effects and Criticality Analysis*. Mar. 2009.

[15] European Cooperation for Space Standardization. *ECSS-S-ST-00-01C Glossary of Terms*. Oct. 2012.

[16] David Evans and Alexander Lange. “OPS-SAT: Operational Concept for ESA’s First Mission Dedicated to Operational Technology”. In: SpaceOps Conference Proceedings, May 2016. DOI: 10.2514/6.2016-2354.

[17] Yerui Fan et al. “A Bearing Fault Diagnosis Using a Support Vector Machine Optimised by the Self-Regulating Particle Swarm”. In: *Shock and Vibration* vol. 2020 (Mar. 2020), pp. 1–11. DOI: 10.1155/2020/9096852.

[18] Ryu Funase et al. “Mission to Earth–Moon Lagrange Point by a 6U CubeSat: EQUULEUS”. In: *IEEE Aerospace and Electronic Systems Magazine* 35.3 (Mar. 2020), pp. 30–44. DOI: 10.1109/MAES.2019.2955577.

[19] G. Merisio, C. Giordano, V. Franzese, F. Topputo. *LUMIO Phase A Mission Requirements Document Issue 1 Rev7*. Tech. rep. Mar. 2021.

[20] G. Merisio, C. Giordano, V. Franzese, K. Woroniak, E. Bertels, A. Cervone, S. Speretta. *LUMIO Phase A System Requirements Document Issue 1 Rev3*. Tech. rep. Feb. 2021.

[21] Vaishali Ganganwar. “An overview of classification algorithms for imbalanced datasets”. In: *International Journal of Emerging Technology and Advanced Engineering* 2 (Apr. 2012), pp. 42–47.

[22] Samuele Gelmi. *Fault Detection Isolation and Recovery for LUMIO mission*. Delft University of Technology MSc. Thesis. 2019.

[23] Don E George. “The CuSP Interplanetary CubeSat Mission.” In: 13th Annual CubeSat Developers Workshop, 2016. DOI: 10.13140/RG.2.1.2995.2406. (Visited on 09/15/2022).

[24] GomSpace. *NanoProp 6DOF Flyer*. URL: https://gomspace.com/shop/subsystems/attitude-orbit-control-systems/nanoprop-6u-propulsion.aspx (visited on 05/08/2022).

[25] Craig Hardgrove et al. “The Lunar Polar Hydrogen Mapper CubeSat Mission”. In: *IEEE Aerospace and Electronic Systems Magazine* 35.3 (Mar. 2020), pp. 54–69. DOI: 10.1109/MAES.2019.2950747.

[26] Tatsuaki Hashimoto et al. “Nano Semihard Moon Lander: OMOTENASHI”. en. In: *IEEE Aerospace and Electronic Systems Magazine* 34.9 (Sept. 2019), pp. 20–30. DOI: 10.1109/MAES.2019.2923311.

[27] Kaiming He et al. *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification*. 2015. arXiv: 1502.01852.

[28] IMT. *C-DST X-Band Transponder Specifications*. URL: https://www.imtsrl.it/products/x-band-transponder (visited on 06/08/2022).

[29] Turker Ince et al. “Real-Time Motor Fault Detection by 1D Convolutional Neural Networks”. In: *IEEE Transactions on Industrial Electronics* 63 (Nov. 2016). DOI: 10.1109/TIE.2016.2582729.

[30] ISISpace. *ISIS On Board Computer Specifications*. URL: https://www.isispace.nl/product/on-board-computer/ (visited on 06/08/2022).

[31] ISISpace. *Modular Electrical Power System Specifications*. URL: https://www.isispace.nl/product/modular-electrical-power-system/ (visited on 06/08/2022).

[32] Numan Khurshid et al. “A Residual-Dyad Encoder Discriminator Network for Remote Sensing Image Matching”. In: *IEEE Transactions on Geoscience and Remote Sensing* (2019), pp. 1–14. DOI: 10.1109/TGRS.2019.2951820.

[33] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *International Conference on Learning Representations Proceedings* (2015). eprint: 1412.6980.

[34] Halber White Kurt Hornik Maxwell Stinchcombe. “Multilayer Feedforward Networks are Universal Approximators”. In: *Neural Networks* 2 (1989), pp. 359–366.

[35] Martin Langer and Jasper Bouwmeester. “Reliability of CubeSats – Statistical Data, Developers’ Beliefs and the Way Forward”. In: *Proceedings of the AIAA/USU Conference on Small Satellites* (2016). SSC16-X-2.

[36] Vasileios Leon et al. “Towards Employing FPGA and ASIP Acceleration to Enable Onboard AI/ML in Space Applications”. In: *30th International Conference on Very Large Scale Integration* (2022), pp. 1–4. DOI: 10.1109/VLSI-SoC54400.2022.9939566.

[37] Uri Lerner et al. “Bayesian Fault Detection and Diagnosis in Dynamic Systems”. In: *Proceedings of the Seventeenth National Conference on Artificial Intelligence* (2000), pp. 531–537.

[38] Benjamin K. Malphrus et al. “The Lunar IceCube EM-1 Mission: Prospecting the Moon for Water Ice”. In: *IEEE Aerospace and Electronic Systems Magazine* 34.4 (Apr. 2019), pp. 6–14. DOI: 10.1109/MAES.2019.2909384.

[39] Marc Hirth, Haifeng Su, Domenico Reggio, Patrick Bergner. *Generic AOCS/GNC Techniques & Design Framework for FDIR - User Manual*. Tech. rep. Aug. 2018.

[40] Leslie McNutt et al. “Near-Earth Asteroid (NEA) Scout”. In: *AIAA SPACE 2014 Conference and Exposition*. American Institute of Aeronautics and Astronautics, Aug. 2014. DOI: 10.2514/6.2014-4435.

[41] NASA. *Fault Management Handbook - Draft 2*. NASA-HDNK-1002. Apr. 2012.

[42] NASA Center for Near Earth Object Studies. *NEO Basics*. URL: https://cneos.jpl.nasa.gov/about/neo_groups.html (visited on 04/08/2022).

[43] Xavier Olive. “FDI(R) for satellites: How to deal with high availability and robustness in the space domain?” In: *International Journal of Applied Mathematics and Computer Science* 22.1 (Mar. 2012), pp. 99–107. DOI: 10.2478/v10006-012-0007-8.

[44] Patrick Bergner, Andre Posch, Domenico Reggio. *Generic AOCS/GNC Techniques & Design Framework for FDIR - GAFE Methodology*. Tech. rep. June 2018.

[45] R. J. Patton and J. Chen. “Review of parity space approaches to fault diagnosis for aerospace systems”. In: *Journal of Guidance, Control, and Dynamics* 17.2 (Mar. 1994), pp. 278–285. DOI: 10.2514/3.21194.

[46] Philipp Probst, Anne-Laure Boulesteix, and Bernd Bischl. “Tunability: Importance of Hyperparameters of Machine Learning Algorithms”. In: *Journal of Machine Learning Research* 20 (2019), pp. 1–32.

[47] Prof. Dr. Eberhard Gill. *AE4S12 Space Systems Engineering Lecture Slides*. Tech. rep. Sept. 2021.

[48] Hu Qinglei, Zhang Xinxin, and Niu Guanglin. “Observer-based fault tolerant control and experiment verification for rigid spacecraft”. In: *Aerospace Science and Technology* 92 (Sept. 2019), pp. 373–386. DOI: 10.1016/j.ast.2019.06.013.

[49] Antonio J. Ricco et al. “BioSentinel: A 6U Nanosatellite for Deep-Space Biological Science”. In: *IEEE Aerospace and Electronic Systems Magazine* 35.3 (Mar. 2020), pp. 6–18. DOI: 10.1109/MAES.2019.2953760.

[50] A Romero-Calvo, J D Biggs, and F Topputo. “Attitude Control for the LUMIO CubeSat in Deep Space”. In: *70th International Astronautical Congress Proceedings* (2019).

[51] Sebastian Ruder. *An overview of gradient descent optimization algorithms*. 2017. arXiv: 1609.04747 [cs.LG].

[52] Maurice Prather Ryan Mackey; Allen Nikora; Cornelia Altenbuchner; Robert Bocchino; Michael Sievers; Lorraine Fesq; Ksenia O. Kolcio Matthew J. Litke. “On-Board Model Based Fault Diagnosis for CubeSat Attitude Control Subsystem: Flight Data Results”. In: Mar. 2021. DOI: 10.1109/AERO50100.2021.9438342.

[53] Javier SANZ LOBO et al. “Design of a Model-Based Failure Detection Isolation and Recovery System for Cubesats”. In: 8th European Conference for Aeronautics and Space Sciences. Madrid, Spain, 1-4 july 2019, 2019. DOI: 10.13009/EUCASS2019-702.

[54] Stefano Silvestrini and Michèle Lavagna. “Deep Learning and Artificial Neural Networks for Spacecraft Dynamics, Navigation and Control”. In: *Drones* 6.10 (Sept. 2022), p. 270. DOI: 10.3390/drones6100270.

[55] J. Sola and J. Sevilla. “Importance of input data normalization for the application of neural networks to complex industrial problems”. In: *IEEE Transactions on Nuclear Science* 44.3 (June 1997), pp. 1464–1468. DOI: 10.1109/23.589532.

[56] Spave Avionics Open Interface Architecture. “SAVOIR FDIR Handbook”. en. In: *European Space Software Repository* 2 (Nov. 2019).

[57] Nitish Srivastava et al. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *Journal of Machine Learning Research* 15 (June 2014), pp. 1929–1958.

[58] Syrlinks. *S-Band Transponder EWC31 Specifications*. URL: https://www.syrlinks.com/en/space/nano-satellite/s-band-transponder-ewc31 (visited on 06/08/2022).

[59] Hamed Taherdoost. “Decision Making Using the Analytic Hierarchy Process (AHP); A Step by Step Approach”. In: *International Journal of Economics and Management Systems* 2 (2017).

[60] Aurora Propulsion Technologies. *ARM A0 Flyer*. URL: https://auroraptfi/thrusters/#ARM (visited on 05/08/2022).

[61] Martin Thoma. *Sigmoid Function*. 2014. URL: https://upload.wikimedia.org/wikipedia/commons/5/53/Sigmoid-function-2.svg.

[62] Massimo Tipaldi and Bernhard Bruenjes. “Spacecraft health monitoring and management systems”. In: *2014 IEEE Metrology for Aerospace (MetroAeroSpace)*. Benevento, Italy: IEEE, May 2014, pp. 68–72. DOI: 10.1109/MetroAeroSpace.2014.6865896.

[63] Prashanth Venkataraman. *Image Denoising Using Convolutional Autoencoder*. July 2022. arXiv: 2207.11771.

[64] W. Pitts W. Mcculloch. “A Logical Calculus of Ideas Immanent in Nervous Activity”. In: *Bulletin of Mathematical Biophysics* 5 (1943), pp. 115–133. DOI: 10.1007/BF02478259..

[65] A Wander and R Förstner. “Innovative Fault Detection, Isolation and Recovery Strategies On-Board Spacecraft: State of the Art and Research Challenges”. In: Control and Fault-Tolerant Systems (SysTol), 2013 Conference, 2012, p. 9. DOI: 10.1109/SysTol.2013.6693950.

[66] Alexandra Wander and Roger Forstner. “Innovative fault detection, isolation and recovery on-board spacecraft: Study and implementation using cognitive automation”. In: 2013, pp. 336–341. DOI: 10.1109/SysTol.2013.6693950.

[67] Safran Group Website. *STIM210 Multi Axis Gyro Module - Datasheet TS1545 rev.22*. URL: https://sensonor.azurewebsites.net/media/vxudyo3g/ts1545-r22-datasheet-stim210.pdf (visited on 03/21/2023).

[68] Austin Williams and Rebecca Rogers. “Leaving No CAPSTONE Unturned: How a CubeSat Pathfinder Will Enable the Lunar Gateway Ecosystem”. In: 34th Small Satellite Conference Proceedings, 2020.

[69] Gang Xiang et al. “Intelligent Fault Diagnosis for Inertial Measurement Unit through Deep Residual Convolutional Neural Network and Short-Time Fourier Transform”. In: *Machines* 10 (Sept. 2022), p. 851. DOI: https://doi.org/10.3390/machines10085 1.

[70] Sanchuan Xu. “A Survey of Knowledge-Based Intelligent Fault Diagnosis Techniques”. In: *Journal of Physics: Conference Series* 1187.3 (2019). DOI: 10.1088/1742-6596/ 1187/3/032006.

[71] Shen Yin et al. “A Review on Recent Development of Spacecraft Attitude Fault Tolerant Control System”. In: *IEEE Transactions on Industrial Electronics* 63.5 (May 2016), pp. 3311–3320. DOI: 10.1109/TIE.2016.2530789.

[72] Xiaodong Shao Yuandong LI Qinglei HU. “Neural network-based fault diagnosis for spacecraft with single-gimbal control moment gyros”. In: *Chinese Journal of Aeronautics* 35 (7 2022), pp. 261–273.

[73] Chuxu Zhang et al. “A Deep Neural Network for Unsupervised Anomaly Detection and Diagnosis in Multivariate Time Series Data”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* (July 2019). DOI: 10.1609/aaai.v33i01.33011409.

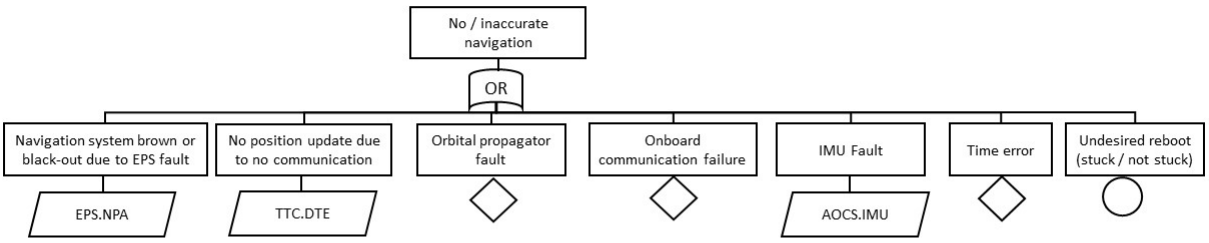
[74] Ali Zolghadri. “Advanced model-based FDIR techniques for aerospace systems: Today challenges and opportunities”. In: *Progress in Aerospace Sciences* 53 (Aug. 2012), pp. 18–29. DOI: 10.1016/j.paerosci.2012.02.004.

A

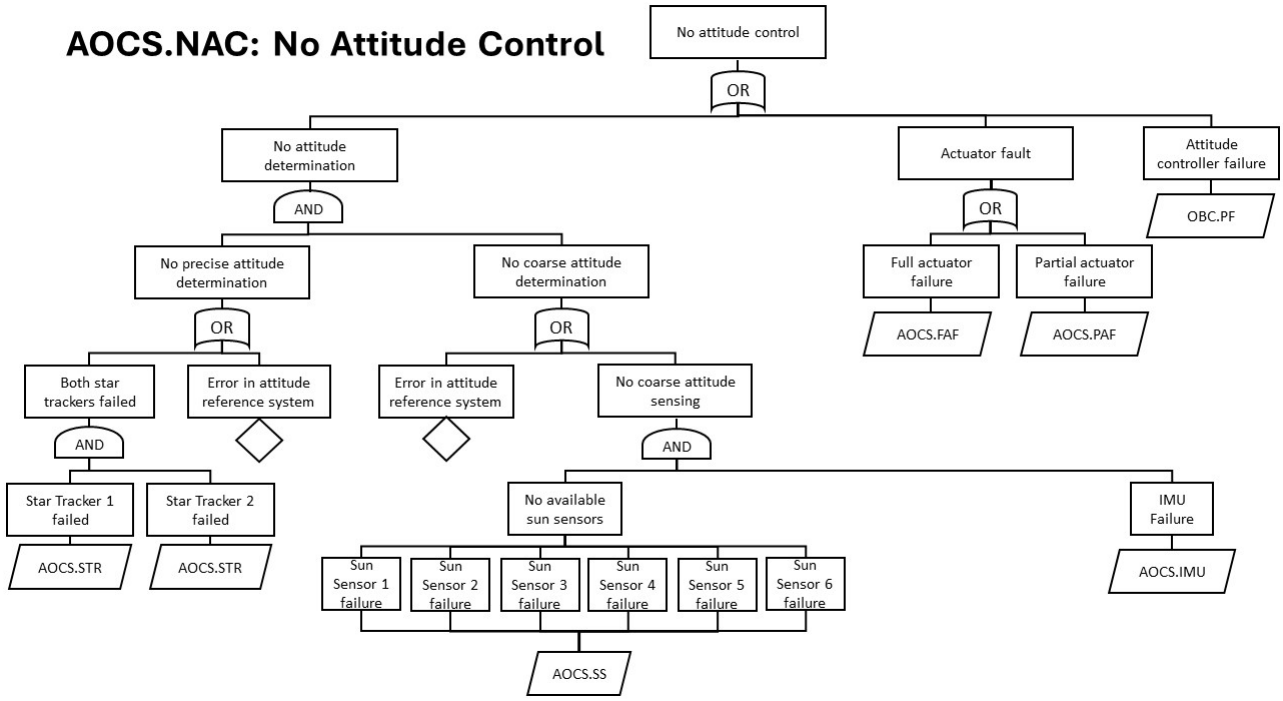
LUMIO Fault Trees

AOCS.NAV: No navigation / inaccurate navigation

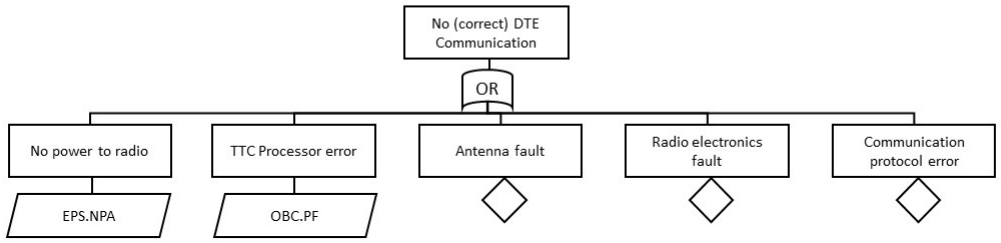
Assuming navigation is performed based on radiometric tracking – i.e. requiring a connection with the ground segment for ranging



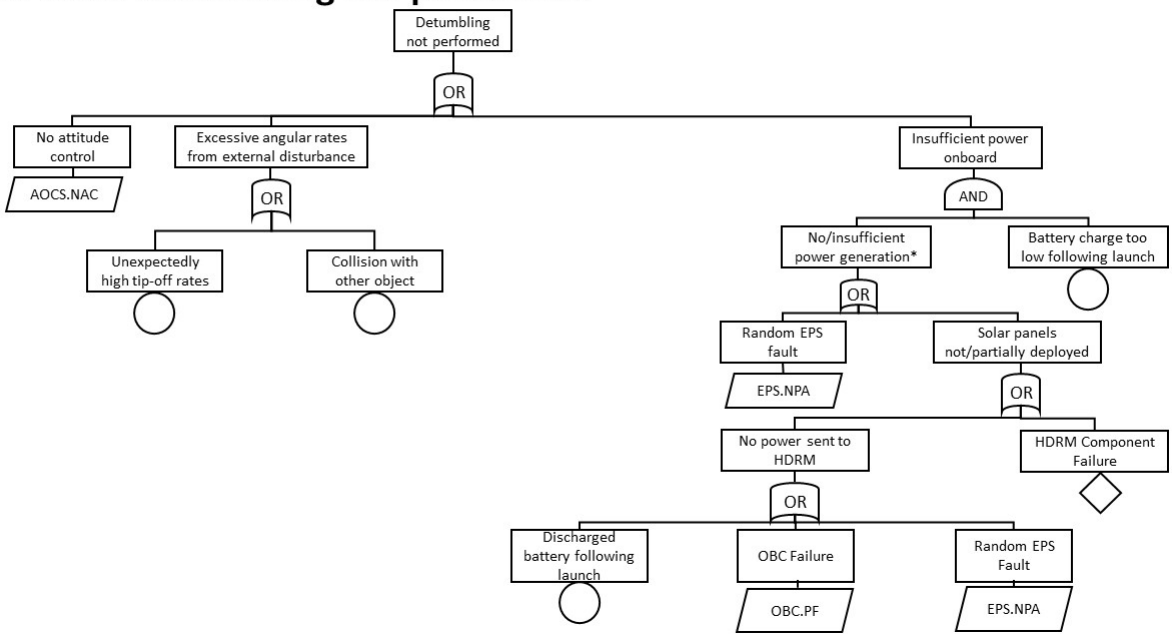
AOCS.NAC: No Attitude Control



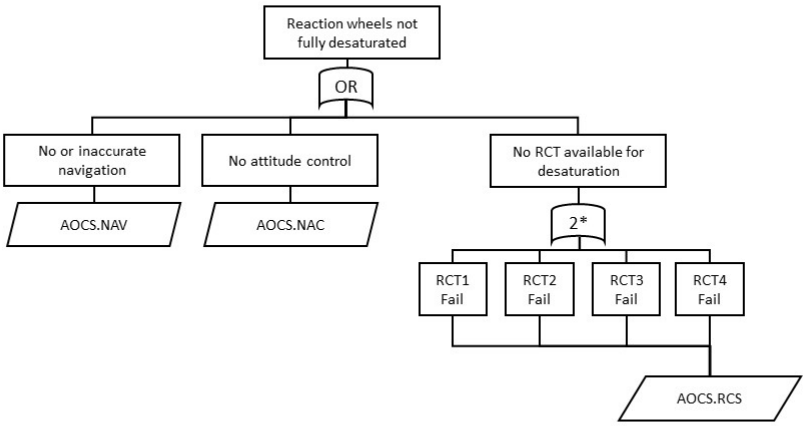
TTC.DTE: No (correct) direct to Earth communication



DEP.DET: Detumbling not performed

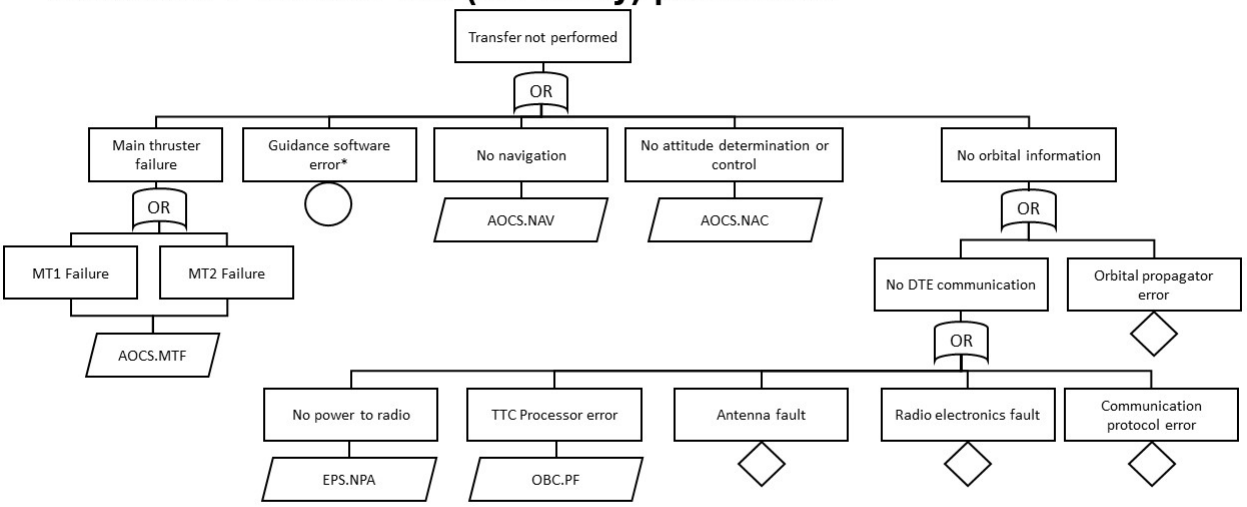


AOCS.DES: Reaction Wheels not fully desaturated



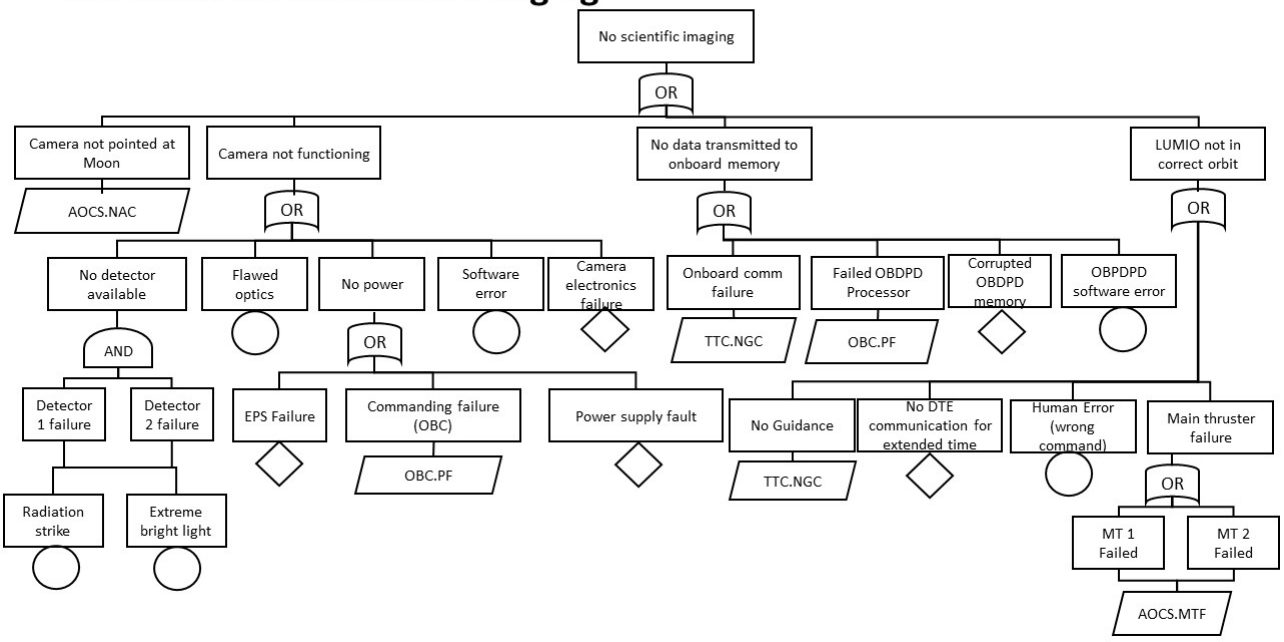
*depending on the RCS configuration, momentum about a specific axis may not be desaturated when 2 or more RCTs fail.

AOCS.TNP: Transfer not (correctly) performed

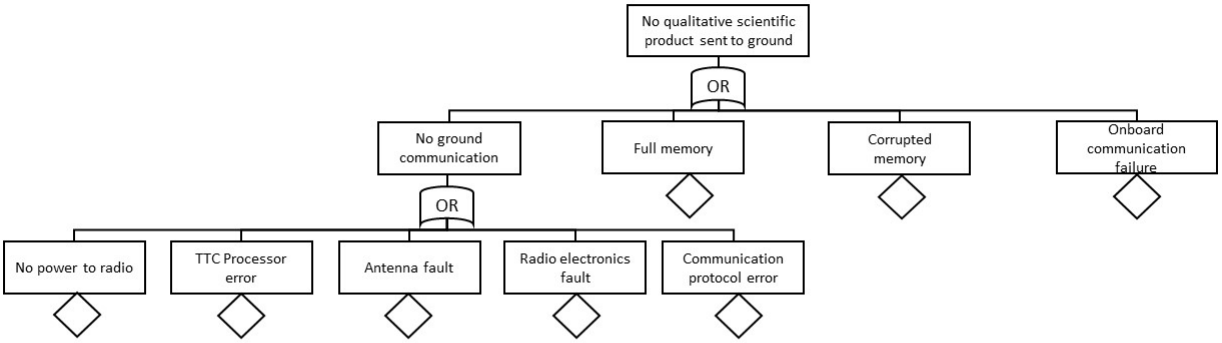


*assuming guidance calculations are performed onboard

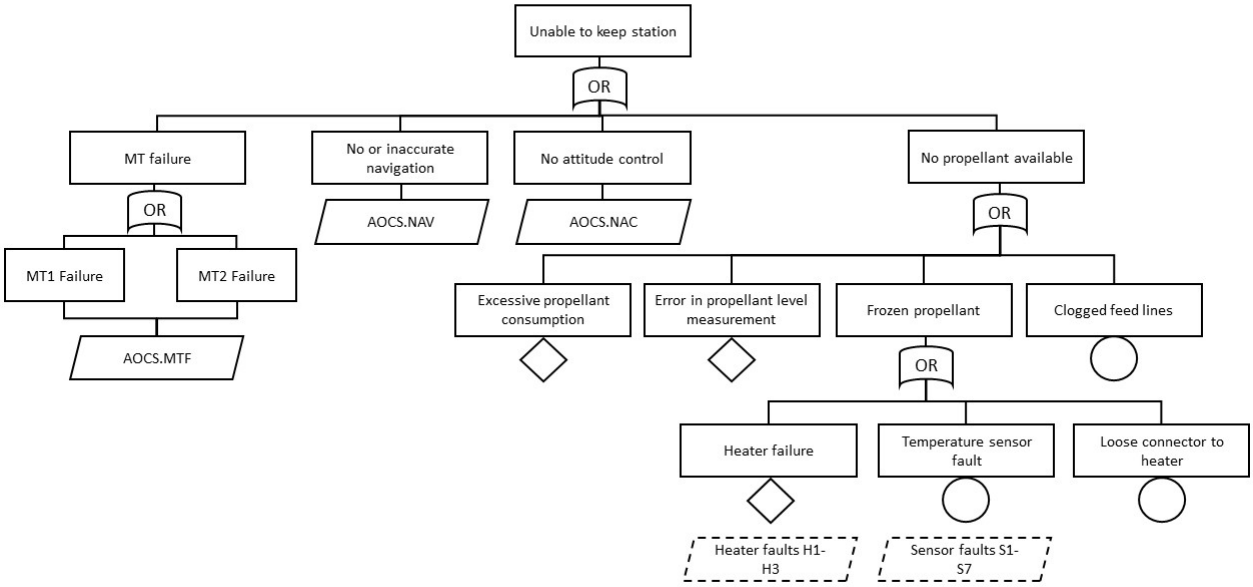
CAM.NSI: No Scientific Imaging



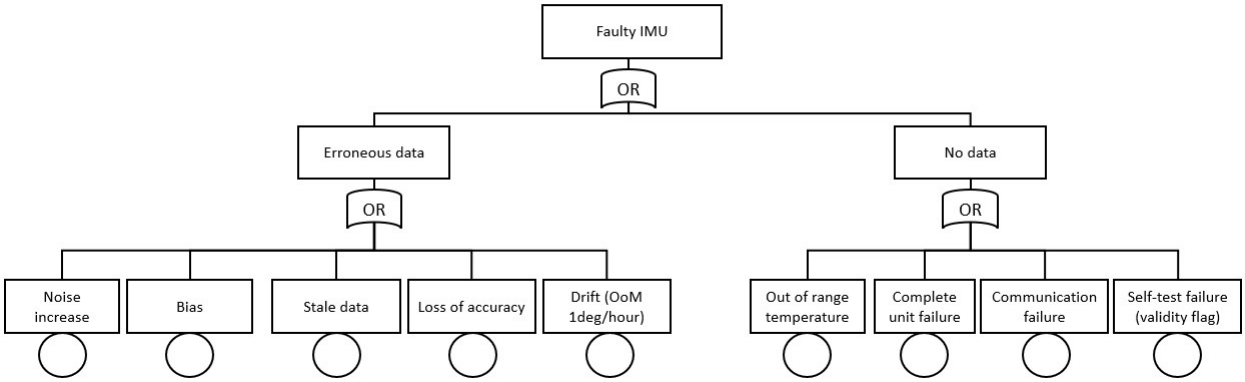
CAM.NDG: No Scientific Data sent to ground



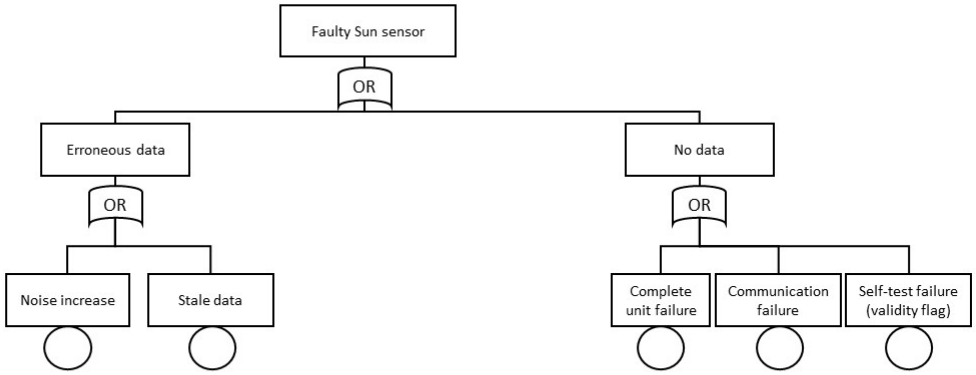
AOCS.SK: Spacecraft unable to keep station during operations



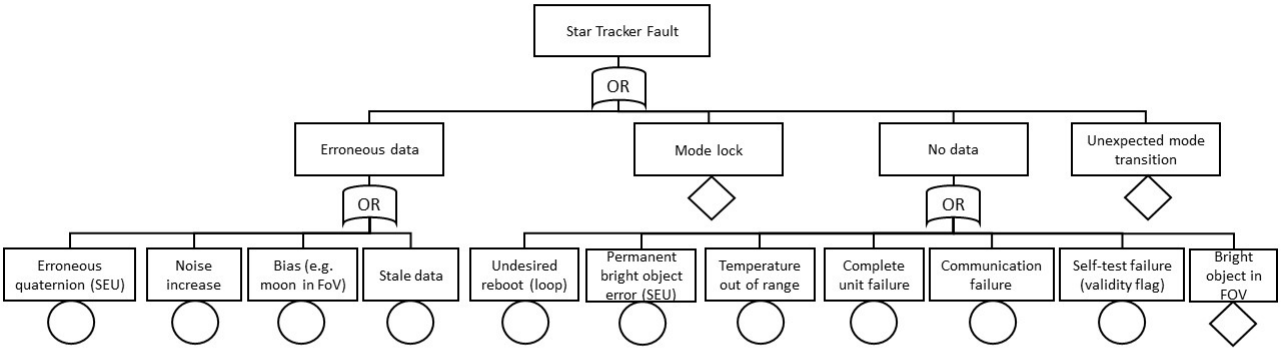
AOCS.IMU: Inertial Measurement Unit Fault



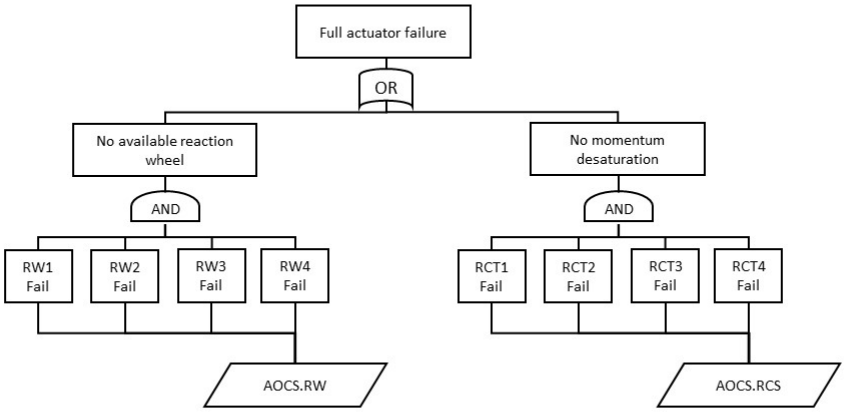
AOCS.SS: Sun Sensor Fault



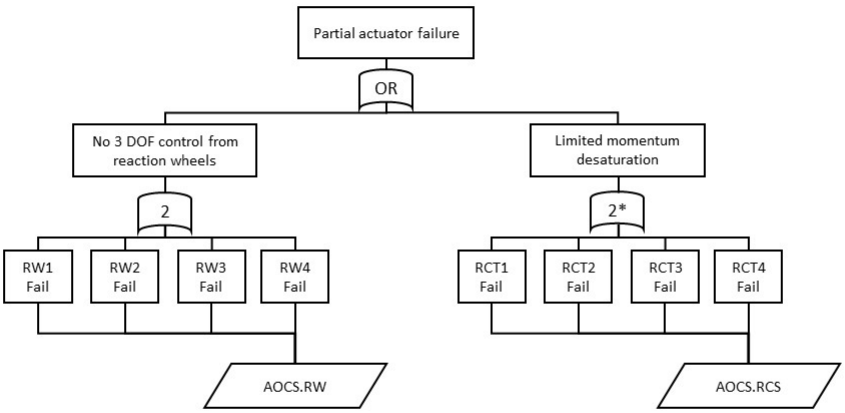
AOCS.STR: Star Tracker Fault



AOCS.FAF: Full Actuator Failure

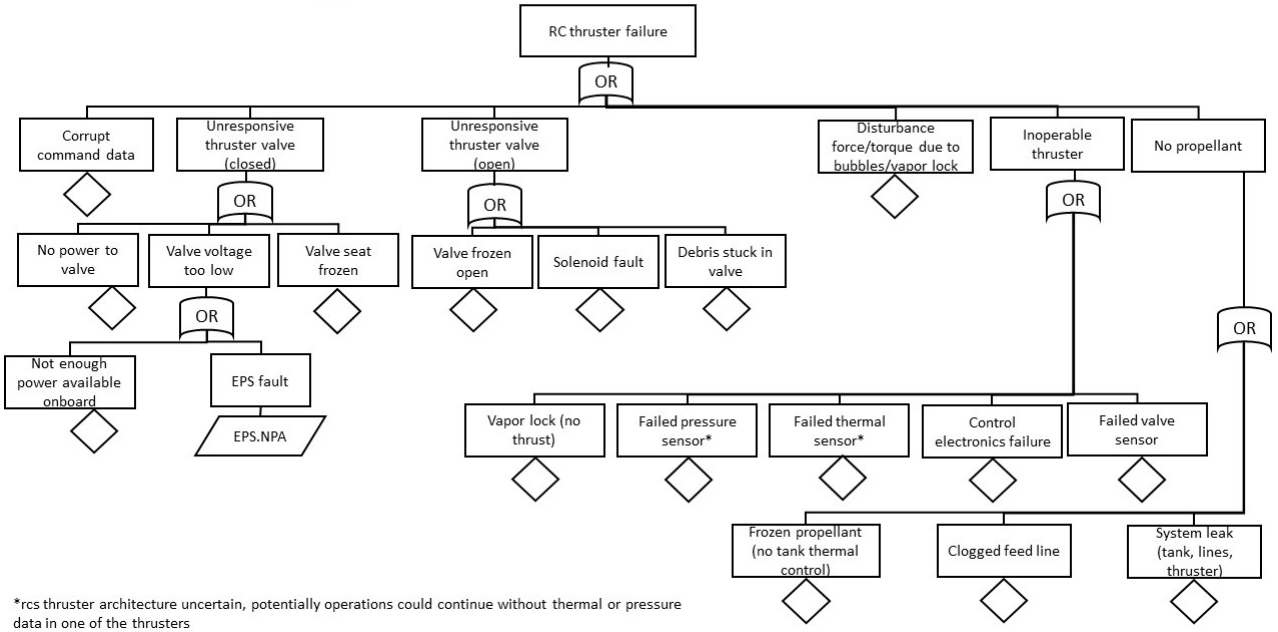


AOCS.FAF: Partial Actuator Failure



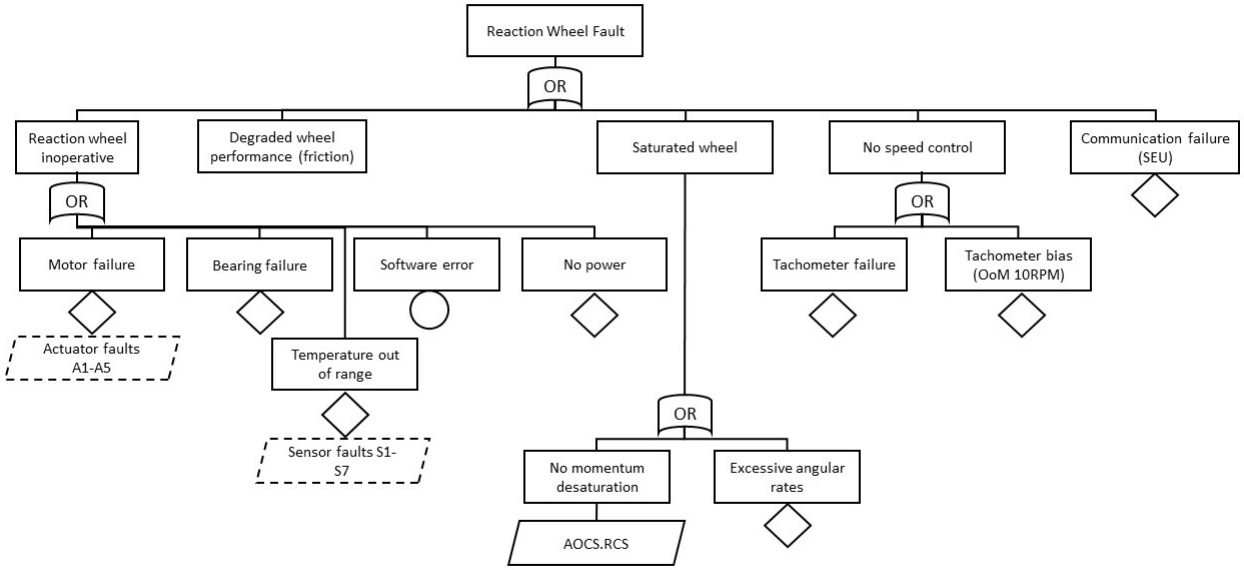
*depending on the RCS configuration, momentum about a specific axis may not be desaturated when 2 or more RCTs fail.

AOCS.RCS: Single RC Thruster Failure

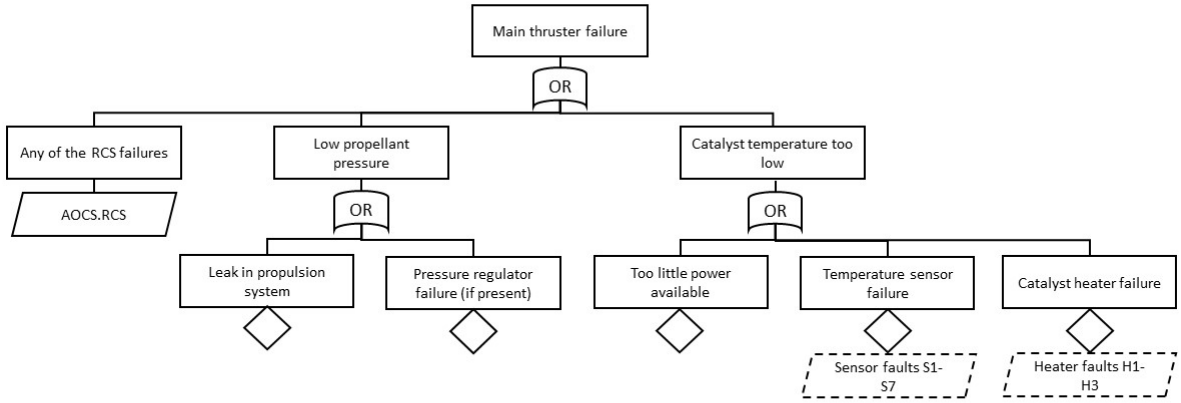


*rcs thruster architecture uncertain, potentially operations could continue without thermal or pressure data in one of the thrusters

AOCS.RW: Reaction Wheel Fault

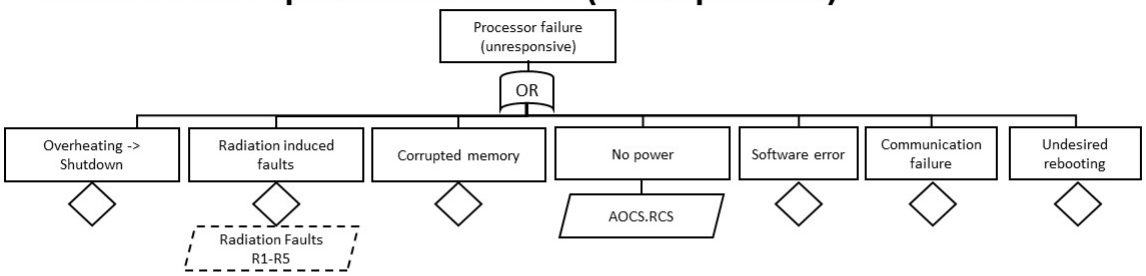


AOCS.MTF: Main Thruster Failure

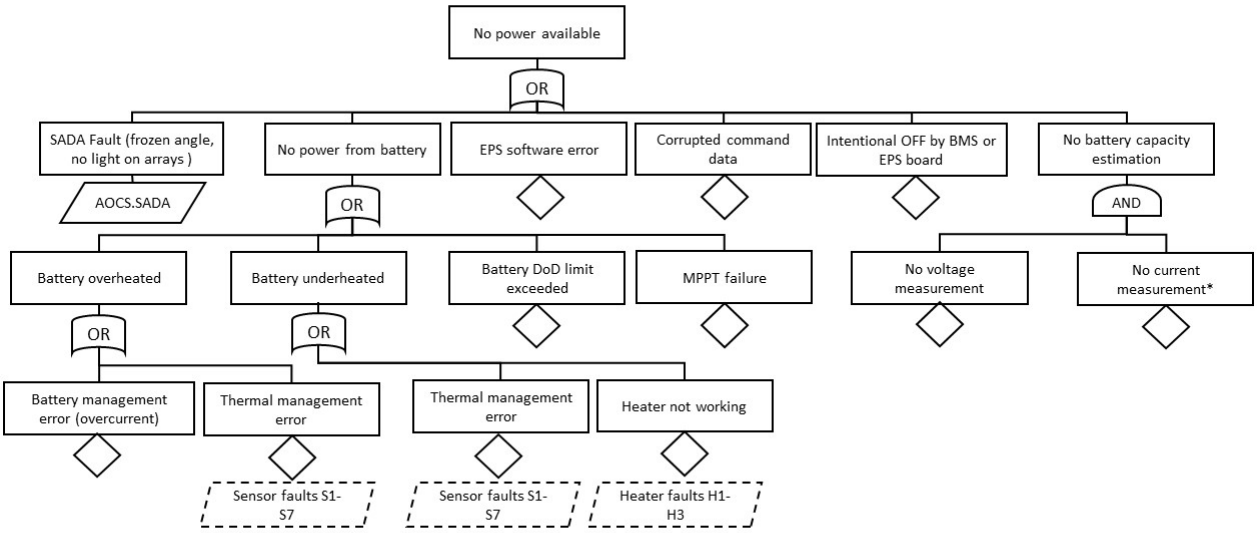


Note: due to limited information about the propulsion system architectures on LUMIO, the Main Thruster FTA is based on the RCS FTA (AOCS.RCS) with the addition of pressurisation and catalyst heating faults.

OBC.PF: main processor failure (unresponsive)

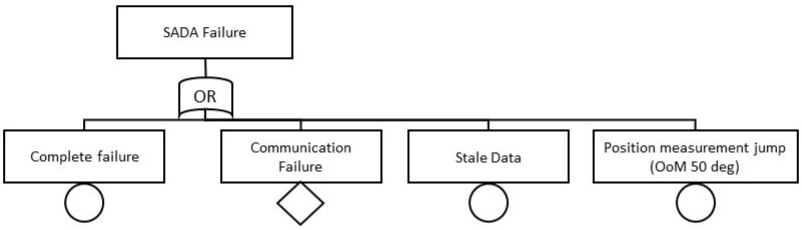


EPS.NPA: No Power Available from EPS

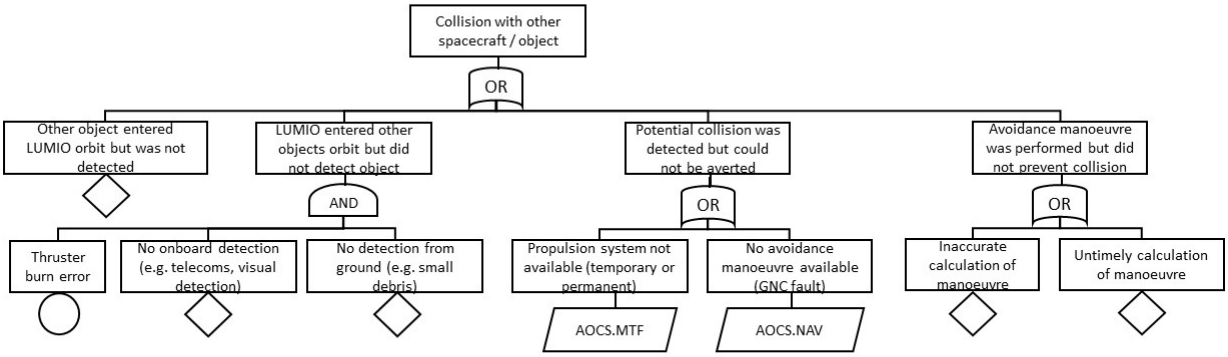


*If coulomb counting used

AOCS.SADA: Solar array drive assembly failure



AOCS.COL: Collision with other spacecraft/object



B

LUMIO AOCS FMECA

ID No	Item/Block	Function Description	Assumed Failure Mode	Mission Phase/AOCS mode	Failure Effect on Unit/Subsys/Sys	Failure Detection (Sensor, AOCS FDI, System FDI)	Compensation	Remarks	SN	PN	CN
STR.01	Star Tracker	Supply absolute attitude quaternion	Self test failure	1 - Parking	No valid validity flag -> No absolute attitude from star tracker -> No absolute attitude acquisition -> No pointing (aside from sun pointing) -> No communication -> Mission loss	Check validity flag	1. Power cycle 2. Switch to redundant unit	IMU Could temporarily provide attitude based on latest estimate but propagation errors make this a temporary solution. SN is assigned based on time criticality (e.g. during operations, time is available for power cycling and potential unit switching, during transfer this may not be the case).	3	2	6
				2 - Transfer	No valid validity flag -> No absolute attitude from star tracker -> No absolute attitude acquisition -> Inaccurate transfer performed or unable to start manoeuvre				3	2	6
				3 - Operations	No valid validity flag -> No absolute attitude from star tracker -> No fine pointing -> No science and risk of no communication				1	2	2
				4 - EOL	No valid validity flag -> No absolute attitude from star tracker -> Inaccurate EOL manoeuvre performed or unable to start				2	2	4
STR.02	Star Tracker	Supply absolute attitude quaternion	Complete failure	1 - Parking	No absolute attitude from star tracker -> No absolute attitude acquisition -> No pointing (aside from sun pointing) -> No	Star tracker flag set to '0' on operational status, and communication status, no incoming data from unit	1. Power cycle 2. Switch to redundant unit		3	1	3
				2 - Transfer	No absolute attitude from star tracker -> No absolute attitude acquisition -> Inaccurate transfer performed or unable to start				3	1	3
				3 - Operations	No absolute attitude from star tracker -> No fine pointing -> No science and risk of no communication				1	1	1
				4 - EOL	No absolute attitude from star tracker -> Inaccurate EOL manoeuvre performed or unable to start manoeuvre				2	2	4
STR.03	Star Tracker	Supply absolute attitude quaternion	Communication failure	1 - Parking	Star tracker operational but cannot interface with AOCS or CDHS system -> No absolute attitude -> No pointing (aside from sun pointing) -> No communication -> Mission loss	Star tracker set to operational on equip. manager, and PCDU power status = ON but communication flag is 0	1. Power cycle 2. Switch to redundant unit		3	1	3
				2 - Transfer	Star tracker operational but cannot interface with AOCS or CDHS system -> No absolute attitude -> No pointing (aside from sun pointing) -> Transfer inaccurate or not performed ->				3	1	3
				3 - Operations	Star tracker operational but cannot interface with AOCS or CDHS system -> No absolute attitude -> No fine pointing -> No science and risk of no communication				2	1	2
				4 - EOL	Star tracker operational but cannot interface with AOCS or CDHS system -> No absolute attitude -> No pointing (aside from sun pointing) -> EOL manoeuvre not performed or				2	1	2
STR.04	Star Tracker	Supply absolute attitude quaternion	Frequent undesired rebooting (SEU, software)	1 - Parking	No absolute attitude -> No pointing (aside from sun pointing) -> No communication -> Mission loss	Star tracker power and communication flags simultaneous go to '0', followed by initialisation and same phenomenon	1. Power cycle 2. Switch to redundant unit	Potential detection through star tracker telemetry (mode = 'init' or similar) is also possible	3	1	3
				2 - Transfer	No absolute attitude -> No pointing (aside from sun pointing) -> Transfer inaccurate or not performed -> Mission loss				3	1	3
				3 - Operations	No absolute attitude from star tracker -> No fine pointing -> No science and risk of no communication				2	1	2
				4 - EOL	No absolute attitude -> No pointing (aside from sun pointing) -> EOL manoeuvre not performed or inaccurate				2	1	2

ID No	Item/Block	Function Description	Assumed Failure Mode	Mission Phase/AOCS mode	Failure Effect on Unit/Subsys/Sys	Failure Detection (Sensor, AOCS FDI, System FDI)	Compensation	Remarks	SN	PN	CN
STR.05	Star Tracker	Supply absolute attitude quaternion	Persistent bright object error (SEU)	1 - Parking	No absolute attitude -> No pointing (aside from sun pointing) -> No communication -> Mission loss	Star tracker throws bright object error/low confidence index is provided, even when attitude is changed or tracker is known to be oriented away from main bright objects (Sun, Moon, Earth)	1. Use IMU for dead reckoning while power cycling faulty Star Tracker 2. Switch to redundant Star Tracker (point away from bright objects)		3	2	6
				2 - Transfer	No absolute attitude -> No pointing (aside from sun pointing) -> Transfer inaccurate or not performed -> Mission loss				3	2	6
				3 - Operations	No absolute attitude from star tracker -> No fine pointing -> No science and risk of no communication				2	2	4
				4 - EOL	No absolute attitude -> No pointing (aside from sun pointing) -> EOL manoeuvre not performed or inaccurate				2	2	4
STR.06	Star Tracker	Supply absolute attitude quaternion	Temperature above operating limits (insufficient heat	3 - Operations	No absolute attitude from star tracker -> No fine pointing -> No science (operations) and risk of no communication (all phases)	Temperature sensor above limit	1. Switch off & cool (reorient if required) 2. Switch to redundant unit		2	1	2
				2, 4 (transfers)	No absolute attitude from star tracker -> No fine pointing -> Transfer not performed or inaccurate		Run other unit in hot redundancy so that overheating tracker can be shut off during transfer phases		2	1	2
STR.07	Star Tracker	Supply absolute attitude quaternion	Stuck in same mode	All	Star tracker in undesired mode (e.g. tracking, acquisition) -> risk of no attitude data from unit -> risk of no pointing	Star tracker does not change to requested mode when commanded (mode manager)	1. Wait, command again 2. Power cycle 3. Switch to redundant unit if unsuccesfull	Fault not considered for separate phases as it is assumed mode changes are not time critical (e.g. during transfer)	1	2	2
STR.08	Star Tracker	Supply absolute attitude quaternion	Unexpected mode transition	1, 3 (static phases)	Star tracker in undesired mode -> Risk of no attitude data from unit -> Risk of no pointing	Star tracker mode changes in telemetry without mode manager reflecting this change	1. Command back to desired mode and wait 2. Power cycle 3. Switch to redundant unit		1	1	1
				2, 4 (transfers)	Star tracker in undesired mode -> Risk of no attitude data from unit -> Risk of no pointing -> Risk of transfer not able to start or				3	1	3
STR.09	Star Tracker	Supply absolute attitude quaternion	Erroneous quaternion supplied (SEU)	1 - Parking	Attitude measurement incorrect -> spacecraft pointing incorrect -> Risk of no power and communication -> Mission	1. Sensor signal processing 2. Cross check with redundant star tracker and/or IMU	1. Power cycle 2. Switch to redundant unit	If an SEU causes a sudden change (spike, step, hard-over) in a quaternion, the fault is detected internally. If the fault is more subtle (drift, bias, loss of accuracy) cross checks should detect it.	3	2	6
				2 - Transfer	Attitude measurement incorrect -> spacecraft pointing incorrect -> Risk of incorrect transfer being executed-> Mission				3	2	6
				3 - Operations	Attitude measurement incorrect -> spacecraft pointing incorrect -> Risk of no power, science and communication ->				2	2	4
				4 - EOL	Attitude measurement incorrect -> spacecraft pointing incorrect -> Risk of incorrect EOL transfer being executed ->				2	2	4
STR.10	Star Tracker	Supply absolute attitude quaternion	Stale data	1, 3 (static phases)	No attitude update -> spacecraft attitude unknown -> Risk of no power, science and communication	Check data timestamp	1. Power cycle 2. Switch to redundant unit		2	2	4
				2, 4 (transfers)	No attitude update -> spacecraft attitude unknown -> Transfer cannot be started				3	2	6
STR.11	Star Tracker	Supply absolute attitude quaternion	Excessive noise in signal	1, 3 (static phases)	Inaccurate attitude data -> Reduced pointing accuracy -> Risk of no communication, reduced power generation and no	Increased data variance	1. Power cycle 2. Switch to redundant unit		1	2	2
				2, 4 (transfers)	Inaccurate attitude data -> Reduced pointing accuracy -> Risk of transfer performed inaccurately (to a degree where it cannot				3	2	6

ID No	Item/Block	Function Description	Assumed Failure Mode	Mission Phase/AOCS mode	Failure Effect on Unit/Subsys/Sys	Failure Detection (Sensor, AOCS FDI, System FDI)	Compensation	Remarks	SN	PN	CN
STR.12	Star Tracker	Supply absolute attitude quaternion	Biased measurement (e.g. bright object in FoV)	1, 3 (static phases)	Inaccurate attitude data -> Reduced pointing accuracy -> Risk of no comms, reduced power genereation and no science	1. Check Star tracker confidence index 2. Cross check with IMU	1. Power cycle 2. Switch to redundant unit		2	2	4
				2, 4 (transfers)	Inaccurate attitude data -> Reduced pointing accuracy -> Inaccurate transfer performed or unable to start manoeuvre	1. Check Star tracker confidence index 2. Cross check with hot redundant unit/IMU			3	2	6
RW.01	Reaction Wheel	Fine attitude control	Tachometer fault - no reading	1, 3 (static phases)	No wheel speed reading -> No control of torque changes on wheel -> No attitude control using this RW -> Limited pointing of spacecraft -> Risk of no science and communication	1. Sensor signal processing 2. Cross check tachometer data with attitude measurements	1. Reset tachometer/speed control electronics 2. Switch to redundant wheel configuration		2	2	4
				2, 4 (transfers)	No wheel speed reading -> No control of torque changes on wheel -> No attitude control using this RW -> Limited pointing of spacecraft -> Risk of transfer not performed or inaccurate				3	2	6
RW.02	Reaction Wheel	Fine attitude control	Tachometer bias	1, 3 (static phases)	Biased wheel speed reading -> Inaccurate torque control -> Inaccurate spacecraft pointing -> Risk of no science and	1. Sensor signal processing 2. Cross check tachometer data with attitude measurements	1. Reset tachometer/speed control electronics 2. Switch to redundant wheel configuration		2	2	4
				2, 4 (transfers)	Biased wheel speed reading -> Inaccurate torque control -> Inaccurate spacecraft pointing -> Risk of transfer not				3	2	6
RW.03	Reaction Wheel	Fine attitude control	Self test failure (validity flag)	1, 3 (static phases)	No valid validity flag -> No attitude control using this wheel -> Limited pointing of spacecraft -> Risk of no communication or	Check validity flag	1. Power cycle wheel 2. Switch to redundant configuration		2	1	2
				2, 4 (transfers)	No valid validity flag -> No attitude control using this wheel -> Limited pointing of spacecraft -> Risk of transfer not performed				3	1	3
RW.04	Reaction Wheel	Fine attitude control	Communication failure	1, 3 (static phases)	No communication with reaction wheel -> No attitude control using this wheel -> Limited pointing of spacecraft -> Risk of reduced science and communications	Star tracker power flag and operational flag '1' but communication flag '0'	1. Power cycle wheel 2. Switch to redundant configuration		2	1	2
				2, 4 (transfers)	No communication with reaction wheel -> No attitude control using this wheel -> Limited pointing of spacecraft -> Risk of transfer not performed or inaccurate				3	1	3
RW.05	Reaction Wheel	Fine attitude control	Temperature measurement error (SEU, outlier)	1, 3 (static phases)	Unknown wheel temperature -> Risk of seizing, excessive friction and wear -> Wheel failure -> Large sudden torque applied to spacecraft -> No communication or science and	Temp sensor signal processing	1. Power cycle wheel 2. Switch to redundant configuration		2	1	2
				2, 4 (transfers)	Unknown wheel temperature -> Risk of seizing, excessive friction and wear -> Wheel failure -> Large sudden torque applied to spacecraft -> Transfer not performed or abrupt				3	1	3
RW.06	Reaction Wheel	Fine attitude control	Temperature above operating limits	1, 3 (static phases)	Excessive wheel temperature -> Need to desaturate and shut down wheel -> Limited pointing -> Risk of no science and	Check temperature sensor	1. Desaturate momentum on wheel 2. Switch off wheel and allow to cool (reorient spacecraft if		2	1	2
				2, 4 (transfers)	Excessive wheel temperature -> Need to desaturate and shut down wheel -> Limited pointing -> Risk of transfer not				3	1	3
RW.07	Reaction Wheel	Fine attitude control	Temperature below operating limits	1 - Parking	Wheel shut down to prevent damage -> Limited pointing -> Risk of delayed detumbling	Check temperature sensor	1. Desaturate momentum on wheel 2. Switch off wheel and turn on heater (reorient spacecraft if required) 3. Restart and return to operation		1	1	1
				2, 4 (transfers)	Wheel shut down to prevent damage -> Limited pointing -> Risk of transfernot performed / inaccurate				3	1	3
				3 - Operations	Wheel shut down to prevent damage -> Limited pointing -> Risk of temporary no science and comms				1	1	1

ID No	Item/Block	Function Description	Assumed Failure Mode	Mission Phase/AOCS mode	Failure Effect on Unit/Subsyst/Sys	Failure Detection (Sensor, AOCS FDI, System FDI)	Compensation	Remarks	SN	PN	CN
RW.08	Reaction Wheel	Fine attitude control	Complete failure	1, 3 (static phases)	Reaction wheel unavailable -> Limited pointing -> Risk of no science and comms	RW flag set to '0' on operational status and communication status, no incoming data from unit	1. Power cycle unit 2. Switch to redundant configuration		2	1	2
				2, 4 (transfers)	Reaction wheel unavailable -> Limited pointing -> Risk of transfer not performed or inaccurate				3	1	3
RW.09	Reaction Wheel	Fine attitude control	Excessive noise in signal	1, 3 (static phases)	Inaccurate wheel speed reading -> Inaccurate torque control -> Inaccurate spacecraft pointing -> Risk of no science and	1. Sensor signal processing 2. Cross check tachometer data with attitude measurements	1. Reset tachometer/speed control electronics 2. Switch to redundant wheel configuration				
				2, 4 (transfers)	Inaccurate wheel speed reading -> Inaccurate torque control -> Inaccurate spacecraft pointing -> Risk of transfer not						
IMU.01	Inertial Measurement Unit	Measure angular rate and linear acceleration on and around 3 axes	Self test failure (validity flag)	1 - Parking	No validity flag -> No relative attitude measurements -> Risk of detumbling not possible -> Mission failure	Check validity flag	Power cycle IMU		3	1	3
				3 - Operations	No validity flag -> No relative attitude measurements -> Reliance on star trackers and sun sensors for pointing				1	1	1
				2, 4 (transfers)	No validity flag -> No relative attitude measurements -> Risk of inaccurate transfer performed				3	1	3
IMU.02	Inertial Measurement Unit	Measure angular rate and linear acceleration on and around 3 axes	Complete failure	1 - Parking	No relative attitude measurements -> Risk of detumbling not possible -> Mission failure	IMU flag set to '0' on operational status, power status, and communication status	Power cycle IMU		2	1	2
				3 - Operations	No relative attitude measurements -> Reliance on star trackers and sun sensors for pointing				1	1	1
				2, 4 (transfers)	No relative attitude measurements -> Risk of inaccurate transfer performed				3	1	3
IMU.03	Inertial Measurement Unit	Measure angular rate and linear acceleration on and around 3 axes	Communication failure	1 - Parking	No relative attitude measurements -> Risk of detumbling not possible -> Mission failure	IMU power flag and operational flag '1 'but communication flag '0'	Power cycle IMU		2	1	2
				3 - Operations	No relative attitude measurements -> Reliance on star trackers and sun sensors for pointing				1	1	1
				2, 4 (transfers)	No relative attitude measurements -> Risk of inaccurate transfer performed				3	1	3
IMU.04	Inertial Measurement Unit	Measure angular rate and linear acceleration on and around 3 axes	Temperature above operating limits (insufficient heat	1 - Parking	No relative attitude measurements -> Risk of detumbling not possible -> Mission failure	Temperature sensor above limit	Switch off & cool (reorient if required)		2	1	2
				3 - Operations	No relative attitude measurements -> Reliance on star trackers and sun sensors for pointing				1	1	1
				2, 4 (transfers)	No relative attitude measurements -> Risk of inaccurate transfer performed				3	1	3
IMU.05	Inertial Measurement Unit	Measure angular rate and linear acceleration on and around 3 axes	Temperature below operating limits	1 - Parking	No relative attitude measurements -> Risk of detumbling not possible -> Mission failure	Temperature sensor below limit	Turn on heater (reorient if required)		2	1	2
				3 - Operations	No relative attitude measurements -> Reliance on star trackers and sun sensors for pointing				1	1	1
				2, 4 (transfers)	No relative attitude measurements -> Risk of inaccurate transfer performed				3	1	3

ID No	Item/Block	Function Description	Assumed Failure Mode	Mission Phase/AOCS mode	Failure Effect on Unit/Subsys/Sys	Failure Detection (Sensor, AOCS FDI, System FDI)	Compensation	Remarks	SN	PN	CN
IMU.06	Inertial Measurement Unit	Measure angular rate and linear acceleration on and around 3 axes	Excessive signal noise (gyro, accelerometer or both)	1 - Parking	Inaccurate relative attitude measurement -> Risk of detumbling not being performed -> Mission loss	Increased data variance in static conditions	Power cycle IMU		3	2	6
				2, 4 (transfers)	Inaccurate relative attitude measurement -> Reduced pointing accuracy -> Inaccurate transfer performed	1. Increased data variance in static conditions 2. Cross check with actuation of RW, thrusters			3	2	6
				3 - Operations	Inaccurate relative attitude measurement -> Reduced pointing accuracy -> Lower quality or no science	Increased data variance in static conditions			1	2	2
IMU.07	Inertial Measurement Unit	Measure angular rate and linear acceleration on and around 3 axes	Stale data (gyro, accelerometer or both)	1 - Parking	No relative attitude measurements -> Risk of detumbling not possible -> Mission failure	Check timestamp of data	Power cycle IMU		3	2	6
				2, 4 (transfers)	No relative attitude measurement -> Reliance on star trackers and sun sensors for pointing -> Reduced transfer accuracy				3	2	6
				3 - Operations	No relative attitude measurement -> Reliance on star trackers and sun sensors for pointing -> Reduced pointing accuracy -> Lower quality or no science				1	2	2
IMU.08	Inertial Measurement Unit	Measure angular rate and linear acceleration on and around 3 axes	Drift in measurements (1 deg/hour or 0.01G/hour)	1 - Parking	Inaccurate relative attitude measurement -> Risk of detumbling not being performed -> Mission loss	Cross check angular rates with star tracker data, check acceleration with position, velocity, time	Power cycle IMU		3	2	6
				2, 4 (transfers)	Inaccurate relative attitude measurement -> Reduced pointing accuracy -> Inaccurate transfer performed				3	2	6
				3 - Operations	Inaccurate relative attitude measurement -> Reduced pointing accuracy -> Lower quality or no science				1	2	2
SS.01	Sun Sensor	Supply sun vector	Self test failure (validity flag)	All	No sun vector -> Risk of reduced power generation	AOCS checks validity flag	Power cycle SS		1	1	1
SS.02	Sun Sensor	Supply sun vector	Complete failure	All	No sun vector -> Risk of reduced power generation	SS flag set to '0' on operational status, power status, and communication status	Power cycle SS	Assume there is always a redundant sensor with the Sun in LoS	1	1	1
SS.03	Sun Sensor	Supply sun vector	Communication failure	All	No sun vector -> Risk of reduced power generation	SS power flag and operational flag '1' but communication flag	Power cycle SS		1	1	1
SS.04	Sun	Supply sun	Excessive	All	No sun vector -> Risk of reduced power generation	Signal variance increase	Power cycle SS		1	2	2
SS.05	Sun	Supply sun	Stale data	All	No sun vector -> Risk of reduced power generation	Check timestamp of data	Power cycle SS		1	2	2
RCS.01	Reaction Control System	Desaturate reaction wheel momentum	Thrust disturbances on a single RCT (vapor lock, bubbles)	1 - Parking	Inaccurate thrusting -> Disturbance torques -> Undesired angular rates -> Momentum buildup -> No detumbling	Commanded thrust does not match real thrust in magnitude or direction (IMU data)	(assume RCS system has at least 1 redundant RCT unit for 3DOF control) 1. Shut down malfunctioning RCT 2. Switch to redundant RCT 3. Remove undesired angular rates induced by faulty RCT	Depending on the thruster configuration and severity of disturbance	3	1	3
				2, 4 (transfers)	Inaccurate thrusting -> Disturbance torques -> Undesired angular rates -> Momentum buildup -> No pointing capability -> Transfer not performed or inaccurate				3	1	3
				3 - Operations	Inaccurate thrusting -> Disturbance torques -> Undesired angular rates -> Momentum buildup -> No pointing capability -				2	1	2
RCS.02	Reaction Control System	Desaturate reaction wheel momentum	RCS not thrusting due to no propellant available (leaks,	1 - Parking	No RCS thrusting -> No momentum desaturation following detumbling -> Not ready to perform transfer -> Mission loss	Dropping propellant pressure / line pressure (if measurement available) IMU registers angular rate from desaturation RWs without RCT compensating (i.e. increasing	Heating of tank, cycling of valves		3	2	6
				2, 4 (transfers)	No RCS thrusting -> No momentum desaturation between transfer burns -> Risk of not completing transfer				2	2	4
				3 - Operations	No RCS thrusting -> No momentum desaturation -> Leads to reduced/no pointing capability -> Reduced mission lifetime				2	2	4

ID No	Item/Block	Function Description	Assumed Failure Mode	Mission Phase/AOCS mode	Failure Effect on Unit/Subsys/Sys	Failure Detection (Sensor, AOCS FDI, System FDI)	Compensation	Remarks	SN	PN	CN
RCS.03	Reaction Control System	Desaturate reaction wheel momentum	Failed thruster - unit failure	1 - Parking	No thrusting using this thruster -> Limited momentum desaturation following detumbling -> Risk of not being able to	1. Self health check 2. IMU anuglar rates register only RW input and increasing rates	1. Shut down malfunctioning RCT 2. Switch to redundant RCT		3	2	6
				2, 4 (transfers)	No thrusting using this thruster -> Limited momentum desaturation -> Risk of not completing transfers				3	2	6
				3 - Operations	No thrusting using this thruster -> Limited momentum desaturation -> Risk of reduced pointing capability and accuracy -> Risk of reduced mission lifetime				1	2	2
RCS.04	Reaction Control System	Desaturate reaction wheel momentum	Thruster valve stuck open	All	Thruster keeps thrusting -> Increased spacecraft rotational rate -> Loss of navigation and attitude control -> Mission loss	1. Signal processing valve voltages 2. Cross check angular rates with thruster status	Close thruster branch or line using upstream valve (if available)		3	3	9
RCS.05	Reaction Control System	Desaturate reaction wheel momentum	Thruster valve stuck closed	1 - Parking	No thrusting using this thruster -> Limited momentum desaturation following detumbling -> Risk of not being able to	1. Signal processing valve voltages 2. Cross check angular rates with thruster status	1. Power cycling (?) 2. Mark thruster failed and use redundant thruster		2	3	6
				2, 4 (transfers)	No thrusting using this thruster -> Limited momentum desaturation -> Risk of not completing transfers				3	2	6
				3 - Operations	No thrusting using this thruster -> Limited momentum desaturation -> Risk of reduced pointing capability and accuracy -> Risk of reduced mission lifetime				2	2	4
MT.01	Main Thruster	Provide orbit control	Vapor lock - thrust disturbances	Phase 2 - Transfer	Inaccurate thrusting -> Manoeuvre performed incorrectly -> Spacecraft on wrong trajectory -> Risk of excessive propellant consumption or mission loss	Spacecraft position, time velocity parameters do no match expected parameters following burn	1. If possible, find stable thrust level for this MT 2. Regulate thrust of other thruster unit to match 3. Use RWs to compensate for additional torques introduced		3	1	3
				Phase 3 - Operations	Inaccurate thrusting -> Disturbance torques -> Inaccurate manoeuvres during station keeping -> Mission on wrong orbit -> Science compromised	Spacecraft position, time velocity parameters do no match expected parameters following burn			3	1	3
				Phase 4 - EOL	Inaccurate thrusting -> End of life manoeuvre performed incorrectly -> Mission may endanger other missions	Spacecraft position, time velocity parameters do no match expected parameters following burn			3	1	3
MT.02	Main Thruster	Provide orbit control	No propellant available (leaks, clogged lines, frozen)	Phase 2 - Transfer	No thrusting -> No manoeuvres performed -> Transfer to Moon not completed -> Mission loss	1. Pressure sensor signal processing 2. Cross check with IMU data	1. Heat propellant tank 2. Power cycle main propellant system		3	2	6
				Phase 3 - Operations	No thrusting -> No manoeuvres performed -> No station keeping -> Early end of mission	1. Pressure sensor signal processing 2. Cross check with IMU data		Assuming RCS could not (partially) perform of station keeping	2	2	4
				Phase 4 - EOL	No thrusting -> Unable to perform EOL manoeuvre -> LUMIO does not comply with regulations	1. Pressure sensor signal processing 2. Cross check with IMU data		Assuming RCS could not (partially) perform of EOL manoeuvre	2	2	4

ID No	Item/Block	Function Description	Assumed Failure Mode	Mission Phase/AOCS mode	Failure Effect on Unit/Subsys/Sys	Failure Detection (Sensor, AOCS FDI, System FDI)	Compensation	Remarks	SN	PN	CN
MT.03	Main Thruster	Provide orbit control	Failed thruster unit failure	Phase 2 - Transfer	No thrusting -> No manoeuvres performed -> Transfer to Moon not completed -> Mission loss	MT Communication flag set to 0, no readings from system, no response when commanded	Power cycle		3	1	3
				Phase 3 - Operations	No thrusting -> No manoeuvres performed -> No station keeping -> Early end of mission	MT Communication flag set to 0, no readings from system, no response when commanded	Power cycle		2	2	4
				Phase 4 - EOL	No thrusting -> Unable to perform EOL manoeuvre -> LUMIO does not comply with regulations	MT Communication flag set to 0, no readings from system, no response when commanded	Power cycle		2	2	4
MT.04	Main Thruster	Provide orbit control	Thruster valve stuck open	Phase 2 - Transfer	Undesired thrusting -> Trajectory deviation and disturbance torques on spacecraft -> No pointing and orbit control -> Mission loss	Cross check commanded state and valve state (if signal can show real valve state) and IMU measurements	Close thruster branch or line using upstream valve (if available) and go into Safe Mode if spacecraft is safe (no thrusting +		3	3	9
				Phase 3 - Operations	Undesired thrusting -> Trajectory deviation and disturbance torques on spacecraft -> No pointing and orbit control -> Early mission end	Cross check commanded state and valve state (if signal can show real valve state) and IMU measurements	Close thruster branch or line using upstream valve (if available) and go into Safe Mode if spacecraft is safe (no thrusting +		3	3	9
				Phase 4 - EOL	Undesired thrusting -> Mission ends up in trajectory other than planned graveyard orbit	Cross check commanded state and valve state (if signal can show real valve state) and IMU measurements	Close thruster branch and go into safe mode		2	3	6
MT.05	Main Thruster	Provide orbit control	Thruster valve stuck closed	Phase 2 - Transfer	No thrusting -> No manoeuvres performed -> Transfer to Moon not completed -> Mission loss	Cross check valve voltage, command to thruster, propellant data, and IMU voltage	1. Power cycle thruster and retry 2. Safe mode		2	3	6
				Phase 3 - Operations	No thrusting -> No manoeuvres performed -> No station keeping -> Early end of mission	Cross check valve voltage, command to thruster, propellant data, and IMU voltage	1. Power cycle thruster and retry 2. Safe mode	Assume if one of the two MT units is lost, stationkeeping can be done with the other	1	3	3
				Phase 4 - EOL	No thrusting -> Unable to perform EOL manoeuvre -> LUMIO does not comply with regulations	Cross check valve voltage, command to thruster, propellant data, and IMU voltage	1. Power cycle thruster and retry 2. Safe mode	Assume RCS or other MT unit is able to remove LUMIO from operational orbit	1	3	3
MT.06	Main Thruster	Provide orbit control	Catalyst heater failure	Phase 2 - Transfer	No preheating -> Thruster impulse significantly reduced -> Delta V required to get to operational orbit cannot be imparted -> Mission loss	Cross check power status catalyst heater and temperature sensor MT	1. Power cycle thruster and retry 2. Safe mode		3	2	6
				Phase 3 - Operations	No preheating -> Thruster impulse significantly reduced -> Mission lifetime reduced	Cross check power status catalyst heater and temperature sensor MT	1. Power cycle thruster and retry 2. Safe mode		1	2	2
				Phase 4 - EOL	No preheating -> Thruster impulse signficantly reduced -> Intended graveyard orbit cannot be reached	Cross check power status catalyst heater and temperature sensor MT	1. Power cycle thruster and retry 2. Safe mode		1	2	2

ID No	Item/Block	Function Description	Assumed Failure Mode	Mission Phase/AOCS mode	Failure Effect on Unit/Subsys/Sys	Failure Detection (Sensor, AOCS FDI, System FDI)	Compensation	Remarks	SN	PN	CN
MT.07	Main Thruster	Provide orbit control	Temperature sensor fault (SEU)	Phase 2 - Transfer	Unkown operating conditions in thruster -> Reduced thruster performance -> Risk of not reaching operational orbit -> Risk of mission loss	1. Signal processing temperature sensor 2. Cross check power consumption and temperature sensor	1. Power cycle thruster and retry 2. Safe mode		3	1	3
				Phase 3 - Operations	Unkown operating conditions in thruster -> Reduced thruster performance -> No or reduced station keeping -> Reduced mission lifetime	1. Signal processing temperature sensor 2. Cross check power consumption and temperature sensor	1. Power cycle thruster and retry 2. Safe mode		2	1	2
				Phase 4 - EOL	Unkown operating conditions in thruster -> Reduced thruster performance -> Risk of not reaching graveyard orbit	1. Signal processing temperature sensor 2. Cross check power consumption and temperature sensor	1. Power cycle thruster and retry 2. Safe mode		2	1	2
MT.08	Main Thruster	Provide orbit control	Tank heater failure (burn through, loose connection, imperfect attachment)	Phase 2 - Transfer	No propellant (and other unit) heating -> Frozen propellant, potential other outages -> No thrusting -> Desired operational orbit not reached -> Mission loss	Signal processing heater power input and temperature sensor	Switch to redundant tank heater unit (assumed present in Phase A design - highly likely according to AC)		3	1	3
				Phase 3 - Operations	No propellant (and other unit) heating -> Frozen propellant, potential other outages -> No thrusting -> No stationkeeping ->	Signal processing heater power input and temperature sensor			2	1	2
				Phase 4 - EOL	No propellant (and other unit) heating -> Frozen propellant, potential other outages -> No thrusting -> EOL manoeuvre not	Signal processing heater power input and temperature sensor			2	1	2
AOCS.01	AOCS Processor	Determine and control attitude and orbit	Frequent undesired rebooting (SEU, software)	Phase 2 - Transfer	No attitude + orbit determination and control -> Transfer not performed -> Mission loss	Watchdog timer	Switch function to OBC processor (temporary or permanent)				
				Phase 3 - Operations	No attitude + orbit determination and control -> No pointing and stationkeeping -> Reduced mission lifetime	Watchdog timer					
				Phase 4 - EOL	No attitude + orbit determination and control -> No EOL manoeuvre performed	Watchdog timer					
OBC.01	Onboard Computer	Processing, communication, monitoring, control of satellite	Unexpected shutdown/reboot (SEU, overheating, power, software)	Phase 2 - Transfer	No commanding -> AOCS shutdown -> Transfer not completed -> Risk of mission failure	Watchdog timer	Switch function to AOCS processor (temporary or permanent)		3	2	6
				Phase 3 - Operations	No commanding -> AOCS shutdown -> Temporary loss of control -> Reduced availability	Watchdog timer			1	3	3
				Phase 4 - EOL	No commanding -> AOCS shutdown -> Temporary loss of control -> Risk of not reaching graveyard orbit	Watchdog timer			1	3	3

ID No	Item/Block	Function Description	Assumed Failure Mode	Mission Phase/AOCS mode	Failure Effect on Unit/Subsys/Sys	Failure Detection (Sensor, AOCS FDI, System FDI)	Compensation	Remarks	SN	PN	CN
SADA.01	Solar Array Drive	Orient solar arrays as commanded	Complete failure	All	No rotating of solar arrays w.r.t. body fixed frame -> Reduced power generation / additional requirements for attitude pointing which may limit operations	Cross check power generation data with expected levels and with solar array health	Power cycle and retry		2	1	2
SADA.02	Solar Array Drive	Orient solar arrays as commanded	Communication failure	All	No measurement and commanding of SADA -> Risk of reduced power generation + additional requirements for attitude pointing which may limit operations	SADA power flag and operational flag '1' but communication flag '0'	1. Power cycle SADA and retry 2. Power cycle communication bus and retry		2	2	4
SADA.03	Solar Array Drive	Orient solar arrays as commanded	Position measurement data stale	All	No information on solar panel orientation -> Risk of reduced power generation	Check data timestamp	1. Power cycle SADA		2	2	4
SADA.04	Solar Array Drive Assembly	Orient solar arrays as commanded	Position measurement jump	All	Inaccurate solar panel orientation information -> Risk of reduced power generation	1. Sensor signal processing 2. Cross check power generation with solar panel orientation	1. Power cycle SADA 2. Use relation between power produced and angle as a coarse measure for SADA angle		2	2	4

C

Critical Faults Register

ID	Block	FMECA ID	Fault Name	Symptoms	'Simple' Cross Check Detection	Signal Detection	Model-Based Detection
F1	Star Tracker	STR.01	Self test failure	Validity flag 0			
F5	Star Tracker	STR.05	Erroneous bright object (persistent)	Low confidence index even when oriented away from bright object		Check confidence index	Check orientation of tracker w.r.t known bright objects (ephemeris based?)
F8	Star Tracker	STR.09 + STR.12	Erroneous data (SEU) or biased measurement	Quaternion data from this STR does not match data from IMU and redundant STR (if available)			
F9	Star Tracker	STR.10	Stale unit data	Timestamp of data does not change			
F10	Star Tracker	STR.11	Erratic star tracker data	Increased variance in data			
F11	Reaction Wheel	RW.01	No tachometer reading	Signal from tachometer not present or NaN			
F12	Reaction Wheel	RW.02	Tachometer reading inaccurate (bias, drift..)	Data does not match spacecraft dynamics			
F24	IMU	IMU.06	Excessive noise in IMU signal	Increased variance in data			
F25	IMU	IMU.07	Stale IMU data	Timestamp of data does not change			
F26	IMU	IMU.08	IMU reading inaccurate (drift, bias..)	IMU readings do not match absolute attitude readings and actuator inputs			
F33	Reaction Control System	RCS.02	No thrust when commanded due to vapor lock	Thruster valve open, propellant available, IMU does not reflect any acceleration	Valve vs propellant state		IMU data vs thrust commanded
F34	Reaction Control System	RCS.03	Thruster unit failure	Thruster unresponsive when commanded, expected thrust input missing	thruster operational state, power state		IMU vs commanded thrust

ID	Block	FMECA ID	Fault Name	Symptoms	'Simple' Cross Check Detection	Signal Detection	Residual Detection
F35	Reaction Control System	RCS.04	Thruster valve stuck open	Valve sensor set to open, thrust input on system when not commanded (IMU)		Valve signal processing (if sensor present for valve state)	IMU angular rates vs expected system input
F36	Reaction Control System	RCS.05	Thruster valve stuck closed	Sufficient power applied to valve and valve state does not change OR torque is not imparted upon spacecraft as expected	Valve state vs power applied		Command vs IMU data
F38	Main Thruster	MT.02	No propellant	No pressure in tank, thruster not outputting thrust when opened (IMU)		Tank pressure	IMU vs commanded thrust
F40	Main Thruster	MT.04	Thruster valve stuck open	Valve sensor set to open, thrust input on system when not commanded (IMU)		Valve signal processing (if sensor present for valve state)	IMU vs commanded thrust
F41	Main Thruster	MT.05	Thruster valve stuck closed	Sufficient power applied to valve and valve state does not change OR delta V is not imparted upon spacecraft as expected	Valve state vs power applied		IMU vs commanded thrust
F42	Main Thruster	MT.06	Catalyst heater failure	Temperature in MT is not increasing during heating, circuit does not consume power	Power status vs command for heating		Temperature sensor vs power
F43	OBC	OBC.01	Unexpected shutdown	General shutdown of spacecraft onboard services	No services available, OBC unresponsive		

D

LUMIO FDIR Requirements Analysis

D.1. Relevant Mission and System Requirements

From the LUMIO SRD, two key system requirements were highlighted as being specifically relevant to the FDIR design:

SYS.030: Availability without ground contact
The system shall be able to continue scientific operations (TBC) and keep the satellite in a thermally and power safe condition for 10 (TBC) days without ground contact

Relevance: It is partially the task of the FDIR system to ensure the spacecraft can identify, isolate and recover from a failure if it occurs, while also ensuring the FDIR does not trigger an unnecessary safe mode during those 10 days.

SYS.170: Immunity to destructive radiation events
*All subsystems shall be immune to destructive events (SEL, SEB, SEGR) with LETth > 37.5 MeV*cm2/mg*

Relevance: For the design of the FDIR system, it can be assumed that events of these nature will not occur under the mentioned LET threshold. This severely reduces the likelihood of certain faults caused by radiation such as Single Event Burnout (SEB) and Single Event Gate Rupture (SEGR) in the AOCS hardware.

D.2. Relevant AOCS Requirements

The AOCS system requirements are also relevant to the AOCS FDIR, and one specific requirement was deemed especially important for the FDIR system and LUMIO in general:

ADC.020: ADCS Capabilities in Safe Mode
The ADCS is required to manoeuvre the solar arrays to Safe Mode. The slew manoeuvre shall be completed in less than 30 min. The pointing accuracy of the solar arrays to the Sun in Safe Mode is required to be less than 15 deg half angle cone.

Relevance: The FDIR has to ensure this capability is available in safe mode, even if a fault in the AOCS triggered the safe mode.

D.3. Relevant Autonomy Requirements

As a functional and advanced FDIR system ties into a properly functioning autonomous system, there are multiple autonomy requirements which are considered relevant to the FDIR design. These are listed in Table D.1.

D.4. FDIR Requirements

Based on the aforementioned analysis of the LUMIO requirements, and the basic guidelines for FDIR requirements in the SAVOIR FDIR Handbook [56] a set of preliminary constraints, 'FDIR Requirements', are devised. The requirements are classified into four groups with an identifier for each group:

- **GEN** - General
- **FUN** - Functional: what should the FDIR be capable of doing?
- **PER** - Performance: how well should the FDIR accomplish its functions under certain conditions?
- **INT** - Interface: how should the FDIR interact with other systems and functions?

D.4.1. General Requirements

The general FDIR requirements are noted in Table D.2.

D.4.2. Functional Requirements

The generated set of functional FDIR requirements are noted in Table D.3 and Table D.4.

ID	Requirement	Relevance
AUT.030	The LUMIO mission shall have F2 autonomy level according to ECSS standard definition (ECSS-E-ST-70-11C31).	The F2 autonomy level requires for a failed function to be restored within a mission specified interval of time by the FDIR system. This is driving for the FDIR design.
AUT.060	The spacecraft shall execute an autonomous detumbling after orbit injection achieving within 20 minutes.	This requires the FDIR to deal with potential AOCS faults during detumbling and restore the AOCS functions such that autonomous detumbling is achieved in the timeframe.
AUT.070	The spacecraft shall perform autonomous station keeping about the operative orbit.	This requires the FDIR to deal with potential AOCS faults during operations in order to restore the station keeping ability autonomously.
AUT.090	For all mission phases, the spacecraft shall have the autonomous capability to maintain the required attitude and to perform attitude manoeuvres during lack of contact with ground segment of at least 10 days.	FDIR must deal with potential AOCS faults without ground intervention in all mission phases and restore the system to a operational state which allows for safe operations for at least 10 days.
AUT.100	The AOCS subsystem shall be able to maintain during Safe Mode the solar arrays pointing to the the Sun.	The FDIR system must be designed such that in case of reasonably expectable faults, the sun-pointing ability of the spacecraft is available.
AUT.101	The AOCS subsystem shall be able to use during Safe Mode the onboard resources: Sun Sensors, IMU, SADA, OBDH, TT&C, heaters, RW	The FDIR can be designed knowing that in safe mode these should be available.
AUT.102	The AOCS subsystem shall be able to ensure power generation	Even in case of an AOCS fault, the FDIR system should be able to restore operations of the AOCS and, failing that, ensure power generation by sun pointing.
AUT.103	The AOCS subsystem shall be able to maintain during safe mode communication with the ground segment	FDIR must ensure that TTC and AOCS faults which trigger a safe mode can be isolated and/or recovered such that ground contact is maintained.
AUT.110	The spacecraft shall implement failure detection, isolation and recovery mechanisms in order to meet the required onboard autonomy levels	The basic requirement which calls for a sophisticated FDIR system on LUMIO

Table D.1: Relevant LUMIO Autonomy Requirements [20]

ID	Description	Rationale	Verification
GEN-010	The FDIR system shall not require any additional processors in the spacecraft	The FDIR system should not require any additional hardware (sensors, processors, actuators) to be implemented, and should consist of only software which can be run on any of the LUMIO processors.	Inspection
GEN-020	The FDIR system shall be verified before integration into the spacecraft	The FDIR system must be tested in order to ensure correct functioning before implementation on LUMIO hardware	Demonstration
GEN-030	The FDIR system shall not increase LUMIO system complexity	The FDIR system should not make the CubeSat satellite architecture more complex by requiring additional hardware (sensors, actuators, processors), power, or propellant	Analysis
GEN-040	The FDIR system development shall not require the development of spacecraft/AOCS models	The development, verification and validation of model is a time-consuming activity and several verified models are available in literature	Inspection
GEN-050	The FDIR should be fully reconfigurable by ground commands.	Limits, checks, recovery actions, and anything else related to the FDIR system should be able to be modified in-flight by ground crews at any phase	Demonstration
GEN-060	Fault management shall be handled in a hierarchical manner such that resolution is sought on the lowest possible level.	Standard FDIR approach, if a failure is not resolved on one level the next level is triggered until ground is the instance handling the fault.	Test
GEN-070	The FDIR system shall use the available housekeeping telemetry onboard for fault detection and isolation	All housekeeping data onboard should be accessible for the FDIR system to ensure maximal coverage in fault detection and isolation	Demonstration

Table D.2: LUMIO FDIR General Requirements

ID	Description	Rationale	Verification
FUN-010	The FDIR system shall detect and isolate hardware faults caused by: random faults, wear out, radiation	As defined by the SAVOIR FDIR HB-003 Iss2 rev0 to be in scope of the FDIR system	Analysis: not all faults can be tested but a representative set can be used to test and analyse FDIR system performance
FUN-020	[deleted]		
FUN-030	The FDIR system shall detect and isolate single-error faults caused by operator errors.		Test
FUN-040	The FDIR system shall be able to function with the spacecraft in any operational configuration	The FDIR system should still accurately detect faults after units have been reconfigured or removed from the system.	Test: verify the FDIR system functions for each operational mode, and for each of the major configurations
FUN-050	The FDIR system shall be able to independently reconfigure the spacecraft from a configuration with a fault to a fault-free operational configuration when such configuration is available	Basic definition of FDIR	Test: introduce a fault in the system that requires reconfiguration and verify that the FDIR autonomously recovers the fault
FUN-051	The maximum duration of an onboard reconfiguration shall be deterministic	To ensure reconfiguration can happen within a specified timeframe	Test
FUN-052	All onboard reconfigurations shall end with an unambiguously known and observable state of all involved elements	To ensure operations can continue autonomously after reconfiguration with all units in the intended state	Test
FUN-060	The FDIR system shall report any fault detections, isolation and recovery actions through telemetry to the ground segment	To track and understand FDIR actions	Demonstration: verify that when a fault is introduced and detected, the FDIR system reports the detection of that fault

Table D.3: FDIR Functional Requirements LUMIO (continued on next page)

ID	Description	Rationale	Verification
FUN-070	The FDIR system shall trigger a Safe Mode when a mission critical fault is identified	To ensure the safety of the system in case the FDIR system is not able to autonomously resolve the issue.	Analysis: demonstrate safe mode activation for a range of the most likely critical faults
FUN-071	Safe mode shall be defined as a condition in which an uninterrupted power supply is available, a thermally safe attitude is maintained and communications with the ground are guaranteed	Definition of safe mode	Analysis
FUN-072	Recovery from safe mode shall only be possible by command from ground	To ensure the situation allows for exit of safe mode	Demonstration
FUN-073	The spacecraft state variables shall be properly re-initialised for execution of the safe mode	To ensure no residual values from previous modes endanger safe mode transition or recovery	Test
FUN-074	The transition to safe mode, once started, shall not be interruptibe	GAFE FDIR BP 403	
FUN-080	The FDIR system shall catch an in-scope onboard fault in such a way that failure propagation to another unit/subsystem is avoided, if feasible	To avoid failure propagation where possible. Unfeasible scenarios are those where the failure propagates faster than the fastest possible detection rate.	Test: introduce the faults to be detected and confirm that the FDIR system detects and isolates the fault before it propagates and causes critical failures
FUN-090	The FDIR should be fully reconfigurable by ground commands.	Limits, checks, actions should be able to be modified in-flight by ground crews at any phase	Test
FUN-100	The FDIR system shall not trigger recovery sequences based on a single reading	Multiple samples should always be considered, and where possible redundant readings should be compared such that unnecessary recovery procedure triggering is avoided	Demonstration
FUN-110	The FDIR system shall avoid continuous reporting of the same anomaly if the anomaly cannot be fixed autonomously	To avoid an endless loop of detection, isolation and recovery, the system shall remove faulty units from operation if the fault keeps returning	Test

Table D.4: FDIR Functional Requirements LUMIO (continued)

D.4.3. Performance Requirements

The FDIR performance requirements can be seen in Table D.5.

ID	Description	Rationale	Verification
PER-010	The FDIR system shall require at most TBD GB of onboard RAM	The FDIR system should not limit the computational resources available for nominal operations	Demonstration
PER-020	The FDIR system shall be storeable in at most TBD GB of onboard non-volatile memory	The memory available onboard is required for payload and housekeeping data storage	Inspection
PER-030	The FDIR system shall catch faults with a fault detection rate (FDR) of TBD %	The FDIR should be able to catch as many faults as possible	Analysis: based on a predefined set of labelled test data, the FDIR system should achieve an acceptable FDR and FAR
PER-040	The FDIR system shall have a false alarm rate (FAR) as low as possible, and of no more than TBD % based on test data.	The FDIR should avoid unnecessary interruption of nominal, fault-free operations.	
PER-050	[deleted]		
PER-060	The FDIR system shall be able to recover L0 to L2 faults in TBD seconds.	To restore nominal operations onboard the spacecraft	Test: for each of the recovery actions, a test shall be run to verify the timely execution

Table D.5: FDIR Performance Requirements LUMIO

D.4.4. Interface Requirements

The interface requirements can be found in Table D.6.

ID	Description	Rationale	Verification
INT-010	The FDIR system shall be able to access all onboard telemetry required for fault detection	Telemetry is the basis of onboard fault detection	Demonstration: show that the required telemetry is available to the FDIR system
INT-011	The FDIR system shall receive data from any operational sensor	The FDIR uses measured quantity to detect faults	
INT-012	The FDIR system shall receive any command sent by the OBC to a subsystem	The FDIR system should know what the desired states of systems are in order to compare to the actual state.	
INT-013	The FDIR system shall have access to the relevant process outputs	This includes attitude vectors, calculated parameters, and others.	
INT-020	The FDIR system shall be able to command resets and ON/OFF actions on all subsystems and their components	Required in order to perform reconfiguration actions	Test: test that the FDIR system is able to switch ON/OFF every component and subsystem in scope of the FDIR
INT-030	The FDIR system shall be able to mark components as healthy or failed.	If an unrecoverable fault occurs in a unit, it should be removed from operation and not be included again	Test: test that the FDIR system can remove a failed component from the operations and mark it as failed in the spacecraft register

Table D.6: FDIR Interface Requirements LUMIO

E

Trade Off

E.1. Concept Exploration

Hardware Redundancy with voting

This concept is the most traditional method of fault detection, isolation and recovery: add extra hardware to compare outputs and add redundancy to recover faults. This is however an expensive and complex concept which is often applied only for mission critical systems on large missions such as interplanetary explorers, or those missions which require long lifetimes such as geostationary communications satellites. For a CubeSat it is not considered feasible due to limited budget, and volume and mass constraints. Therefore it will not be further considered.

Plausibility Testing

Plausibility testing checks if physical laws are upheld to detect faults. While relatively simple and cost efficient, it has limited applicability in complex systems and cannot be used for isolation of faults. It requires for a fault to lead to loss of plausibility, which may not always be the case. It is therefore not considered a feasible option for this thesis.

Signal Processing

Signal processing will use the signals directly and analyse them using different methods described in chapter 2 to detect known fault features. While it cannot detect all types of faults, it has the advantage of detecting most faults at low level and avoiding fault propagation, while also being very cost efficient. The main drawback is the limitation in dynamic systems, where advanced detection methods are required. Nevertheless it is considered a viable option for this thesis, with its simplicity being very complimentary to the CubeSat concept.

Parity Space

The parity space method uses algebraic representation of the system to generate models and detect faults. This is a commonly used concept and, as long as a model is available, very accurate. Its main drawback is the computational power required for real-time fault detection as well as the complexity that goes along with developing the parity relations. It is however definitely a feasible option for CubeSat FDIR.

Parameter Estimation

Parameter estimation is used to estimate values in a mathematical model, after which they are compared to measured variables. This method is extensively sensitive to noise however and require detailed model availability, which makes it unsuitable for AOCS telemetry which can be noisy and include disturbances while being very complex to model. Therefore it is not

considered here.

Observer Based
The observer based methods use a state observer to estimate variables based on the output, and it is quite similar to the parity space method. However, it is mainly used in cases where expected system states are well-defined. It is also computationally heavy and real-time computing can become very costly for complex systems such as the AOCS. Nonlinearity is also a difficulty for this method, adding another challenge and more computational cost to this method. For this reason, it is not considered further.

Cognitive Automation
As one of the knowledge based methods, it is more advanced and less used in practice. Through a language called Cognitive Programming Language (CPL) it allows the analyst to model the FDI system and automate it. It is considered highly accurate for large datasets, while performing well in real-time applications. However, as with most knowledge based methods, the performance is highly dependent on the quality of the data present. It will be further evaluated as a potential method for fault detection.

Neural Networks
Neural networks have been used in many applications in recent years, and have also been suggested for use in fault detection and isolation. However, so far there is no evidence of a neural network based FDI being used onboard CubeSats. This could be beneficial as it can combine the detection and isolation of many types of faults (discrete, continuous) across many functions and units without an accurate model. However, the data to train these networks will need to be gathered and standardised. The fault data could be simulated however, leaving this as an option for further investigation.

Support Vector Machines
Support Vector Machines (SVM)s is a type of supervised learning algorithm that can be used for classification or regression tasks. Based on spacecraft parameters, one could classify whether a system is functioning normally or experiencing a fault based on input features that describe the system's behaviour. They are especially robust and capable of handling nonlinear relationships as well as high-dimensional data [17]. It is thus considered a feasible option for implementation on CubeSats.

Dynamic Bayesian Network
The Dynamic Bayesian Network (DBN) is an extension of the Bayesian network, a probabilistic graphical model used to represent and reason about the relationships between variables. It can mix discrete and continuous variables, as well as handle simultaneous failures. However it is very complex and inference is difficult. [37] Therefor it is not considered feasible for CubeSat implementation.

Expert System
With expert systems, one relies on the human expertise to perform the fault detection. Therefore, experts are inherently required to develop this system, which is not suitable at all for CubeSat development which should be accessible to non-experts. Additionally, it cannot be easily adapted across different missions. These reasons mean it will not be further considered.

Qualitative Trend Analysis
As a method which detects patterns, fault detection can be performed relatively accurately.

However, once again data quality is a driver for performance and this method is oftentimes not considered very precise and prone to incorrect classifications of data for various reasons. The method will not be further considered here.

E.2. Methods to Determine Weights

As mentioned before the determination of the weights is tricky and should attempt to avoid as much bias from the process as possible. There are numerous ways to select weights, in this thesis three main methods will be used and compared:

1. Assigning a score on a scale of 1 to X (e.g rate each on a scale of 1 to 10)
2. Ranking the criteria, with the most important receiving the highest score (e.g. 1 to 5)
3. Analytical Hierarchy Process (AHP) (comparing the relative importance of each criteria to obtain weights)

E.2.1. Scoring

With 'simple' scoring, the designation of a weight is self explanatory, and for this trade a scale of 1 to 5 was arbitrarily chosen as it is deemed to be granular enough.

Criteria	**Weight**	**Driving Requirements**	**Rationale**
Thesis feasibility	KILLER	MSc. Thesis	The system should be designed within the timeframe and scope of a Master thesis project
Software based	KILLER	GEN-010	The FDIR system should be implemented in software onboard LUMIO
Verification & validation feasibility	5	GEN-020	The FDIR system should be verified and validated before operations using a suitable model of the LUMIO satellite and AOCS system
Fault detection accuracy	4	PER-030, PER-040	The FDIR system should achieve at least a predefined FDR and FAR rate
System complexity / cost?	3	GEN-030	The FDIR system should limit additional system complexity (e.g. hardware redundnacy) and costs
Model complexity	1	GEN-040	The models required for the FDIR system (CubeSat, AOCS) and its validation should be reasonably available and not require excessive work in defining and verifying these models
Computational resources	3	PER-010, PER-020	The FDIR system should be able to run onboard a CubeSat without limiting the resources and power available to other onboard operations

E.2.2. Ranking

With ranking, the advantage is that one has to prioritise certain criteria over others. Therefor, it cannot be that all characteristics are equally important. Since there are 5 criteria which are not killer, the ranking is done from 1 to 5. The weights resulting from this method are shown in Table E.1.

Criteria	Weight	Rationale
Verification & validation feasibility	5	No matter how efficient or accurate the FDIR, if it cannot be validated it is not fit for use.
Fault detection accuracy	4	The accuracy in detecting faults is one of the most important characteristics of an FDIR system
System complexity / cost?	2	System complexity should be minimised, but not at the expense of accuracy or efficiency
Model complexity	1	Model complexity should be minimised but is less important compared to the operational and validation criteria
Computational resources	3	Onboard computational resources are scarce and should be treated as such

Table E.1: Criteria Weighting using ranking method

E.2.3. Analytical Hierarchy Process

Finally, the AHP method as described by Taherdoost [59] is used as a comparison. Here, all criteria are listed as the first row and column of a matrix, as shown in Table E.2. The criteria pair is rated based on how important the criteria in the row is compared to that in the column. The matrix that results is then used to calculate the weights, which is the eigenvector of the matrix. The consistency is then checked using the consistency ratio CR = CI/RI where RI is the random consistency index (1.1159 for n=5 as seen in [59]) and CI is the consistency index

$$CI = \frac{\lambda_{max} - n}{n - 1} \tag{E.1}$$

Here n is the number of criteria (n=5) and λ_{max} is the maximum eigenvalue of the matrix. The comparison in this trade off yields a CR of 0.01, whereas valid results require a CR lower than 0.01, well below the proposed limit of 0.1 for a valid comparison [59].

Finding the eigenvector of the comparison matrix leads to the weights

$$\begin{bmatrix} w_1 & 1.3 \\ w_2 & 1.387 \\ w_3 & 0.607 \\ w_4 & 0.41 \\ w_5 & 1 \end{bmatrix}$$

AHP MATRIX	Verification & validation feasibility	Fault detection accuracy	System complexity / cost?	Model complexity	Computational resources
Verification & validation feasibility	1.0	1.1	2.0	2.5	1.5
Fault detection accuracy	0.9	1.0	2.0	3.0	2.0
System complexity / cost?	0.5	0.5	1.0	1.5	0.5
Model complexity	0.4	0.3	0.7	1.0	0.3
Computational resources	0.7	0.5	2.0	3.0	1.0

Table E.2: Pair-wise comparison of criteria

E.3. Comparison to classical ranking

Criteria	**Weight**	**Signal processing**	**Parity Space**	**Cognitive Automation**	**Neural Networks**	**Support Vector Machine**	**Dynamic Bayesian Network**
Verification & validation feasibility	5	100	110	90	90	80	80
Fault detection accuracy	4	80	90	110	110	100	100
System complexity	2	110	90	90	100	100	90
Model complexity	1	120	90	110	110	100	80
Computational resources	3	110	90	80	120	90	100
	Results	0.99	0.97	0.95	1.04	0.91	0.91
Killer Req	Thesis	1	1	0	1	1	0
	Software	1	1	1	1	1	1
	Final Score	0.99	0.97	0.00	1.04	0.91	0.00

Table E.3: Trade Off with alternative weighting method (ranking)

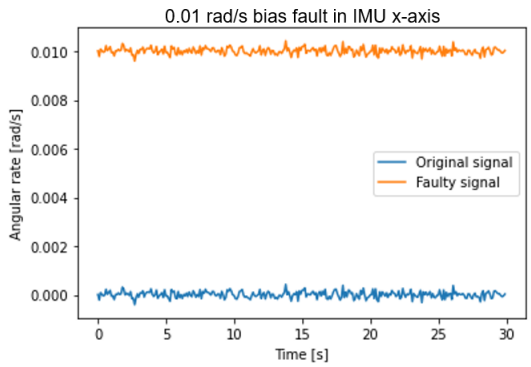
E.4. Comparison to Pugh Matrix Scoring Method

Criteria	Weight	Signal processing	Parity Space	Cognitive Automation	Neural Networks	Support Vector Machine	Dynamic Bayesian Network
Verification & validation feasibility	1.3	1	1	0	1	0	0
Fault detection accuracy	1.387	-1	0	1	1	1	-1
System complexity	0.607	1	0	0	1	0	-1
Model complexity	0.41	1	-1	0	1	0	-1
Computational resources	1	0	-1	0	1	0	0
	+	3	1	1	5	1	0
	0	1	2	4	0	4	2
	-	1	2	0	0	0	3
	Results	0.93	-0.11	1.387	4.704	1.387	-2.404
Killer Req	Thesis	1	1	0	1	1	0
	Software	1	1	1	1	1	1
	Final Score	0.93	-0.11	0	4.704	1.387	0

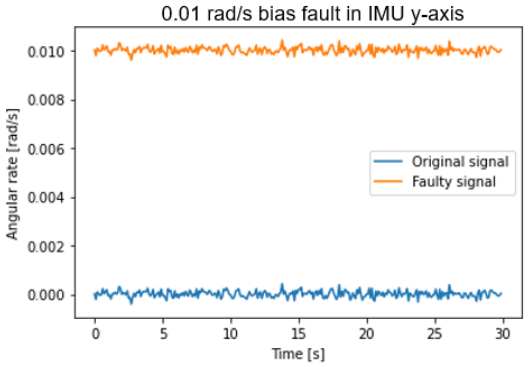
LUMIO and OPS-SAT Faulty Signals

Note: faults 4, 8 and 12 are combinations of the other faults and are not shown in a single telemetry signal here for both LUMIO and OPS-SAT.

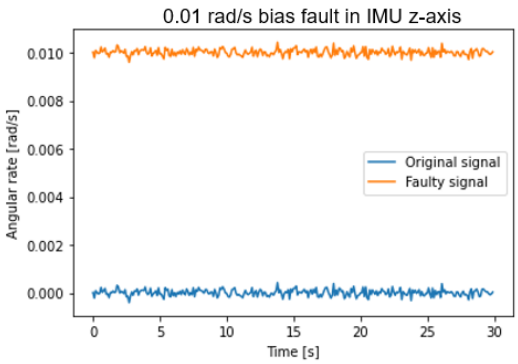
F.1. LUMIO IMU Faults



(a) Fault 1: signal bias in IMU x-axis measurement

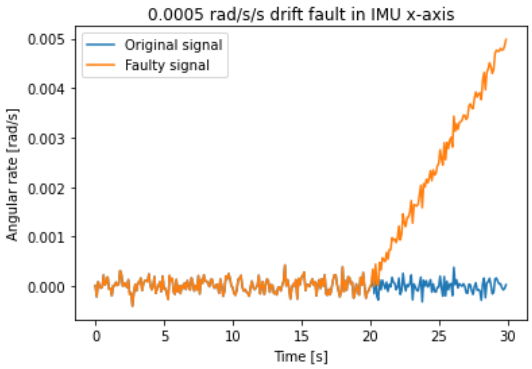


(b) Fault 2: signal bias in IMU y-axis measurement

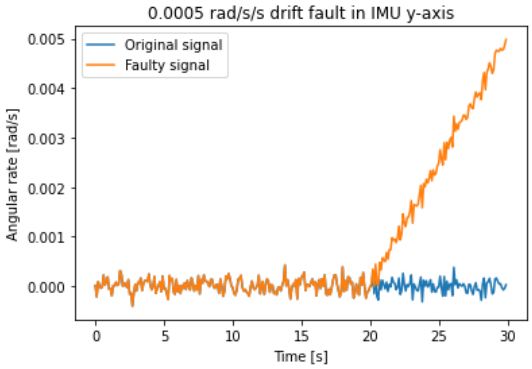


(c) Fault 3: signal bias in IMU z-axis measurement

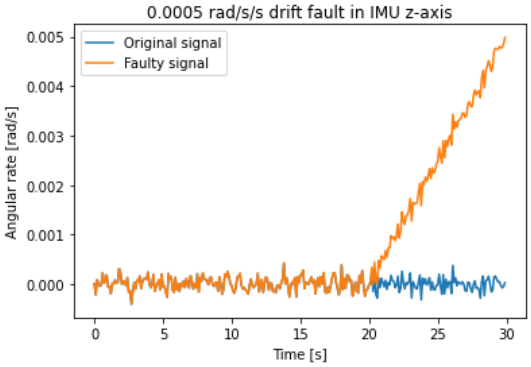
Figure F.1: LUMIO signals bias faults



(a) Fault 5: signal bias in IMU x-axis measurement

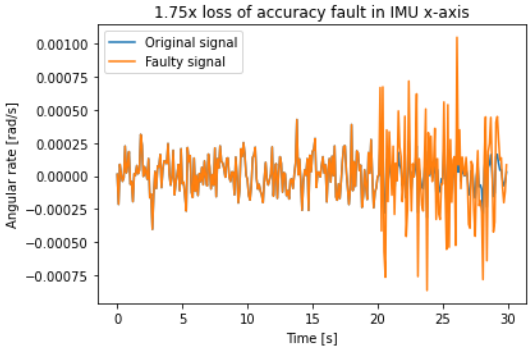


(b) Fault 6: signal bias in IMU y-axis measurement

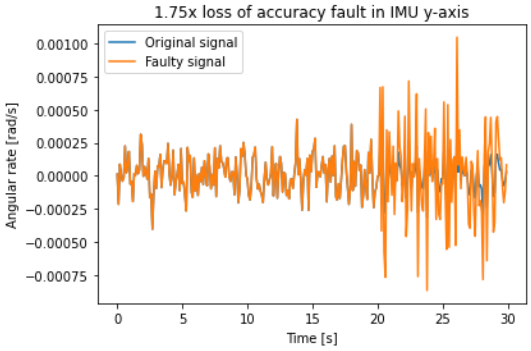


(c) Fault 7: signal bias in IMU z-axis measurement

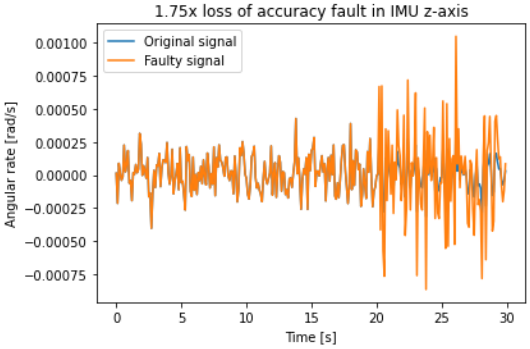
Figure F.2: LUMIO signals drift faults



(a) Fault 9: signal bias in IMU x-axis measurement



(b) Fault 10: signal bias in IMU y-axis measurement



(c) Fault 11: signal bias in IMU z-axis measurement

Figure F.3: LUMIO signals loss of accuracy (calibration) faults

F.2. OPS-SAT IMU Faults

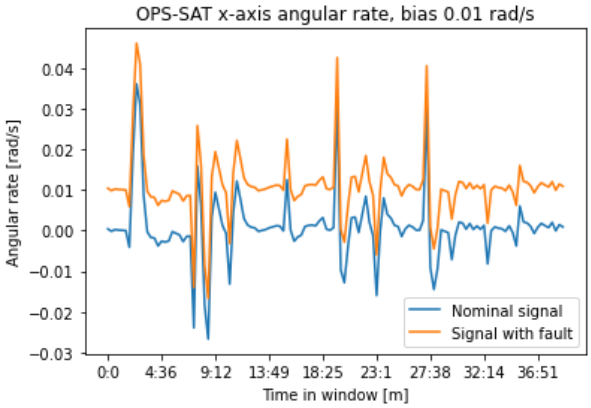


Figure F.4: Fault 1: signal bias in IMU x-axis measurement

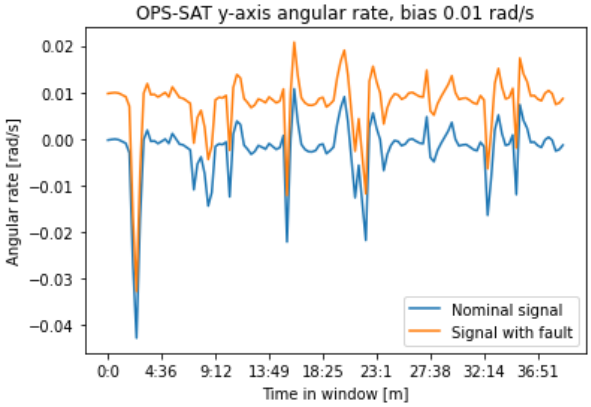


Figure F.5: Fault 2: signal bias in IMU y-axis measurement

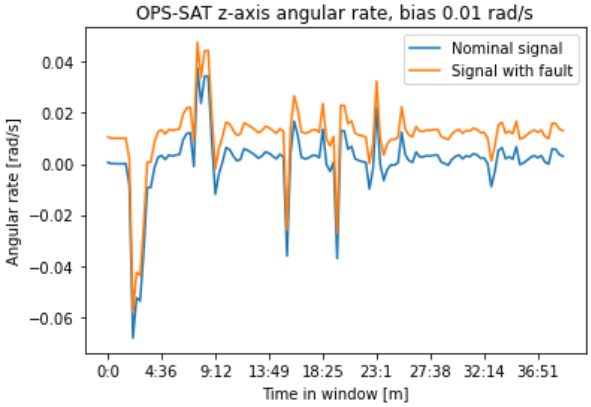


Figure F.6: Fault 3: signal bias in IMU z-axis measurement

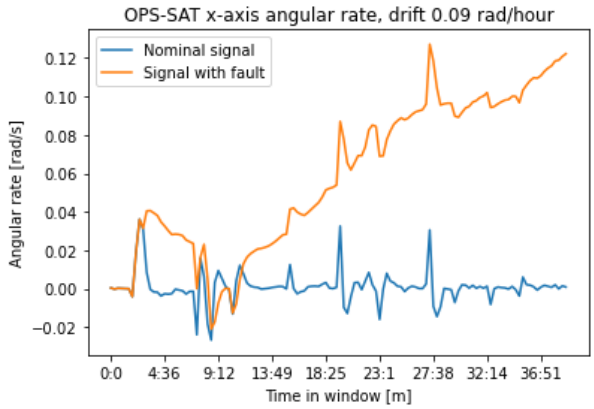


Figure F.7: Fault 5: signal drift in IMU x-axis measurement

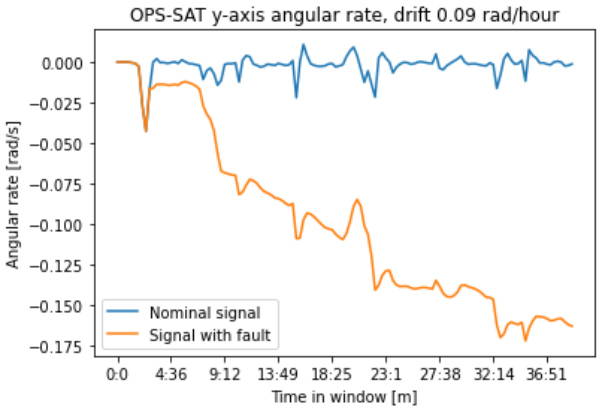


Figure F.8: Fault 6: signal drift in IMU y-axis measurement

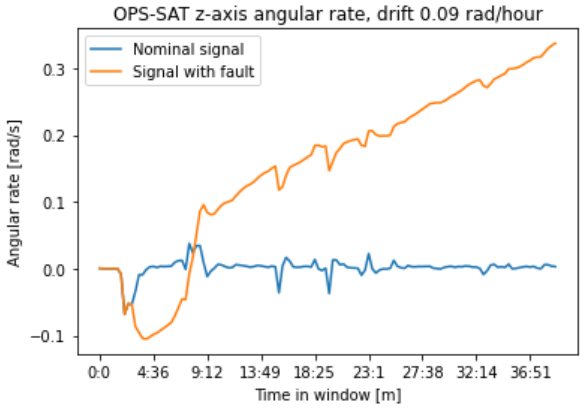


Figure F.9: Fault 7: signal drift in IMU z-axis measurement

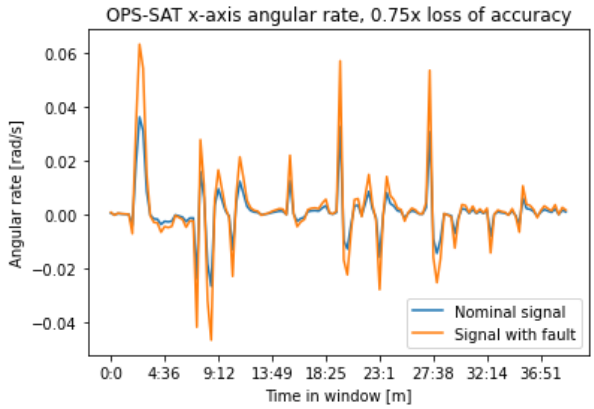


Figure F.10: Fault 9: signal loss of accuracy in IMU x-axis measurement

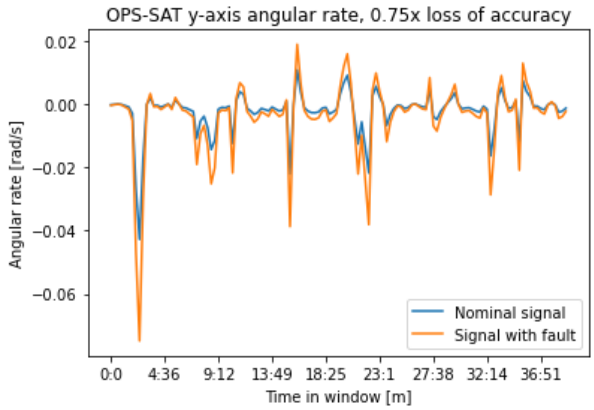


Figure F.11: Fault 10: signal loss of accuracy in IMU y-axis measurement

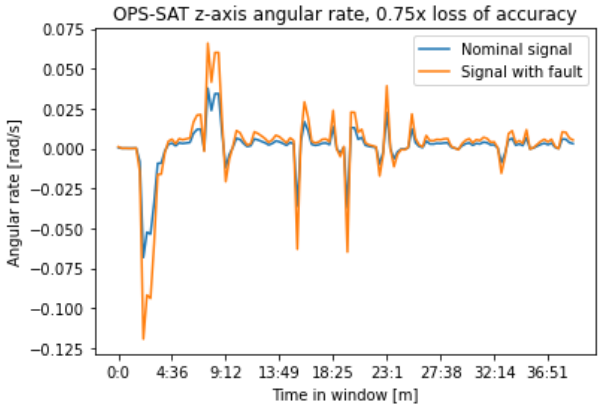


Figure F.12: Fault 11: signal loss of accuracy in IMU z-axis measurement

G

OPS-SAT Telemetry

G.1. Quaternion Data

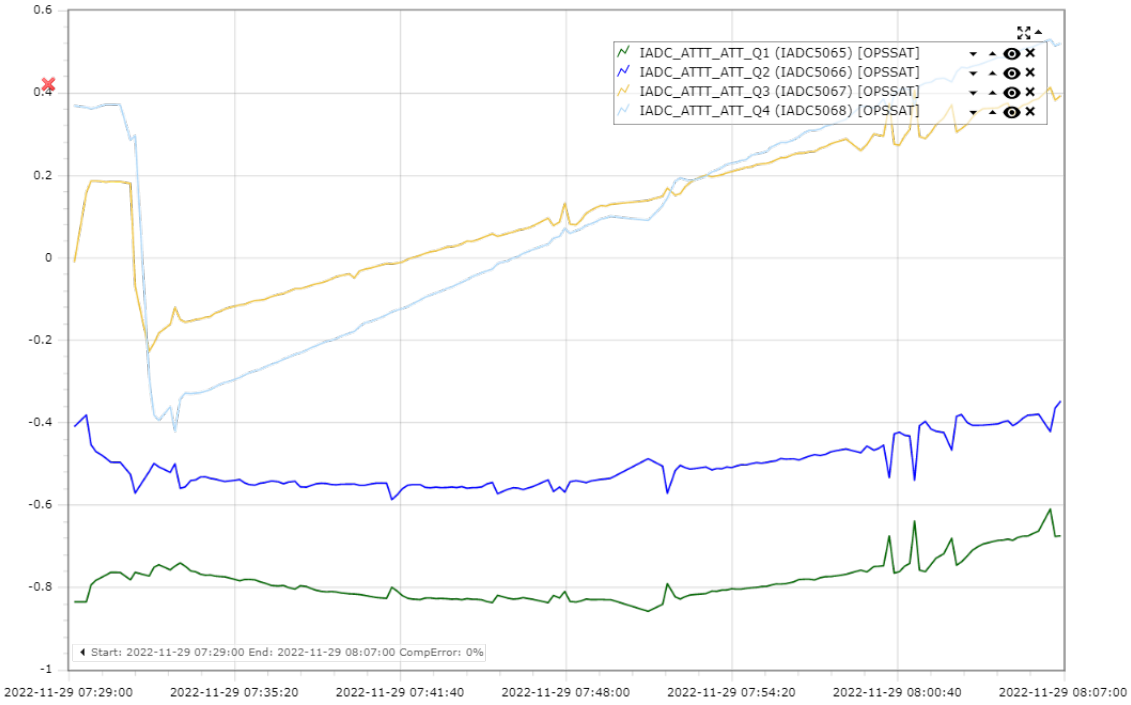


Figure G.1: Window 2 29/11/22 7:29 - 8:07 Quaternion Data

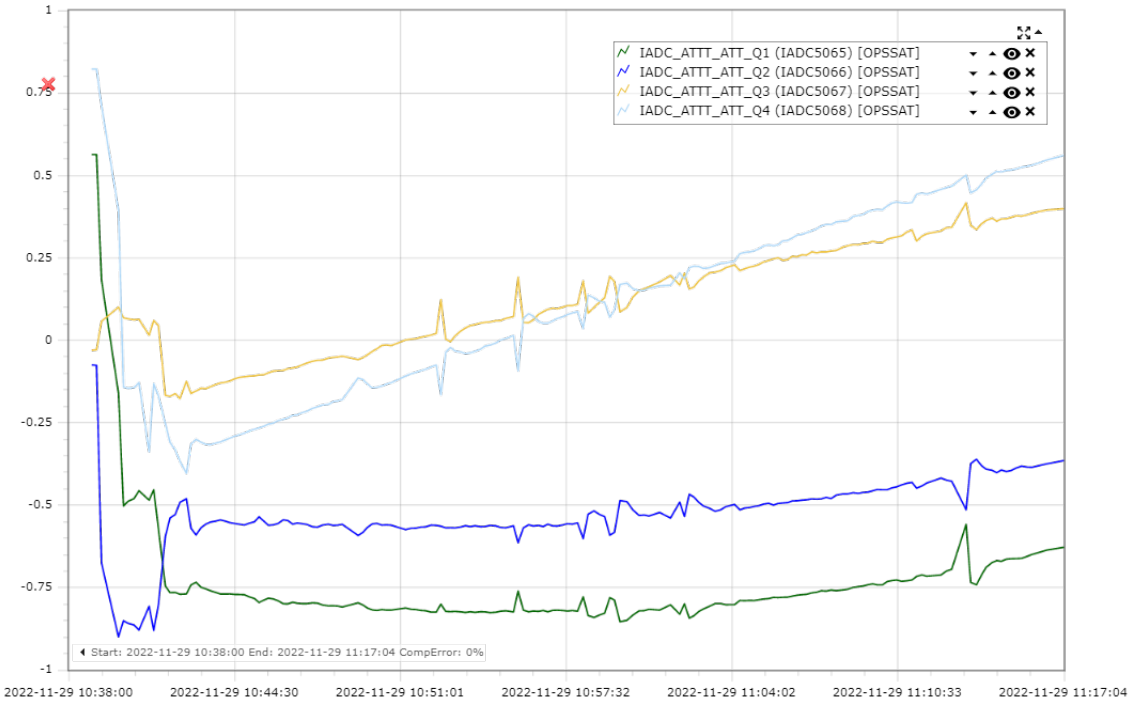


Figure G.2: Window 3 29/11/22 10:38 - 11:17 Quaternion Data

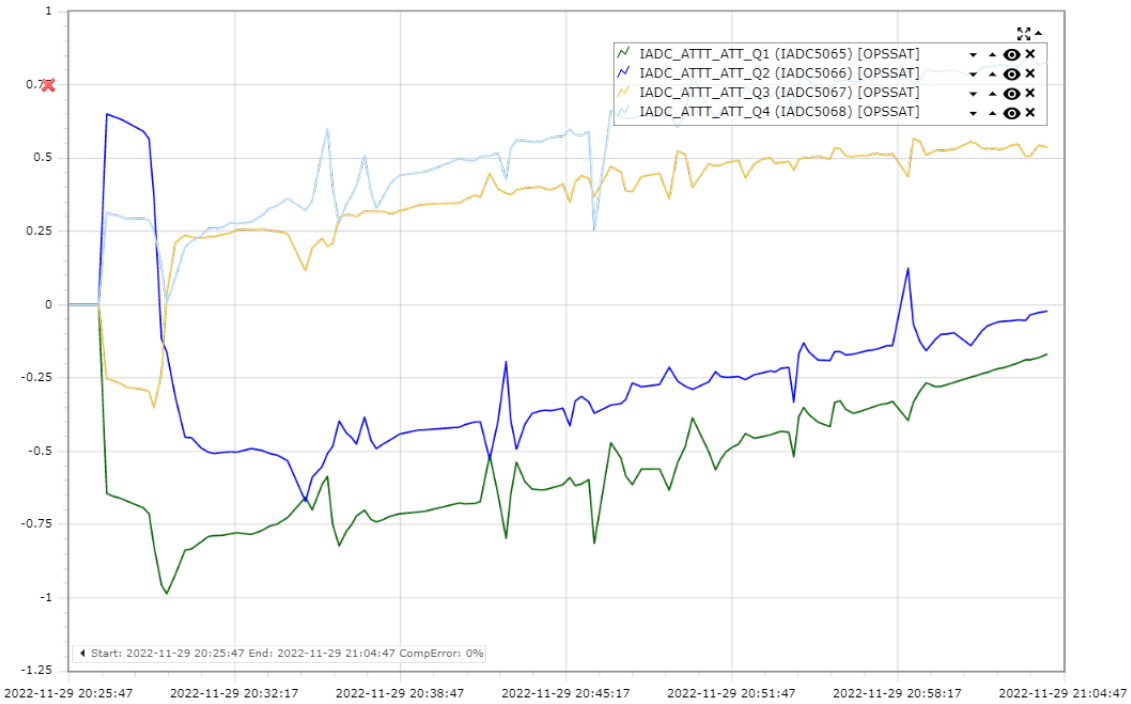


Figure G.3: Window 4 29/11/22 20:25 - 21:04 Quaternion Data

G.2. Reaction Wheel Data

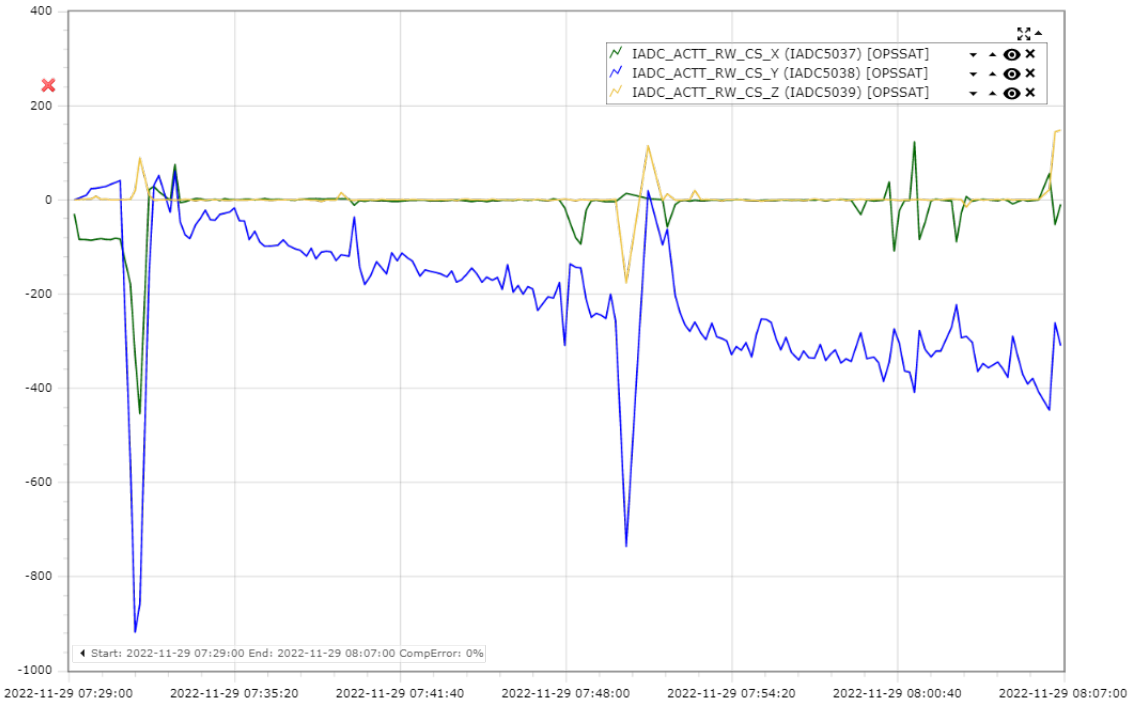


Figure G.4: Window 2 29/11/22 7:29 - 8:07 Reaction Wheel Data

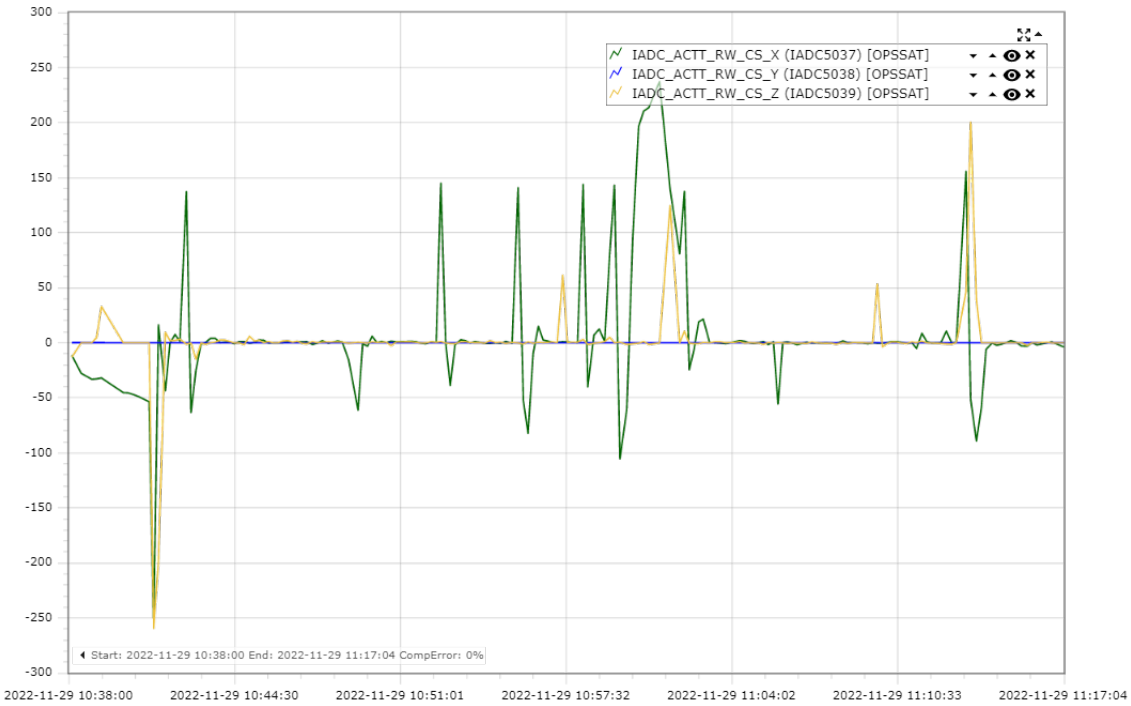


Figure G.5: Window 3 29/11/22 10:38 - 11:17 Reaction Wheel Data

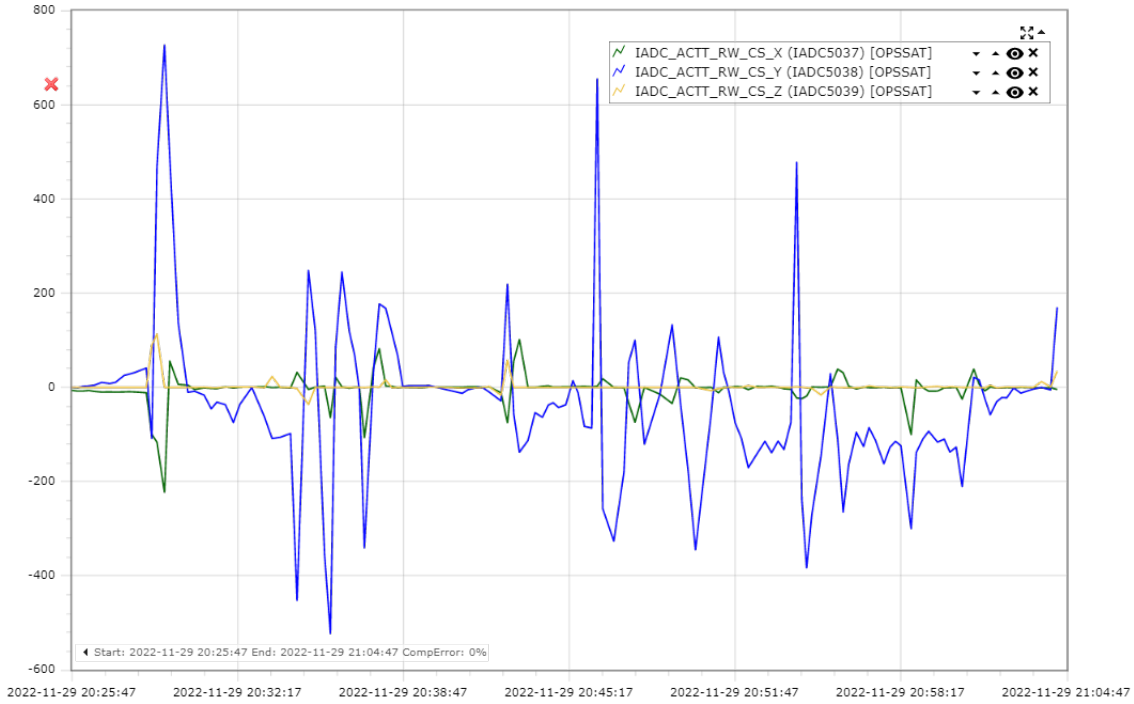


Figure G.6: Window 4 29/11/22 20:25 - 21:04 Reaction Wheel Data

G.3. IMU Data

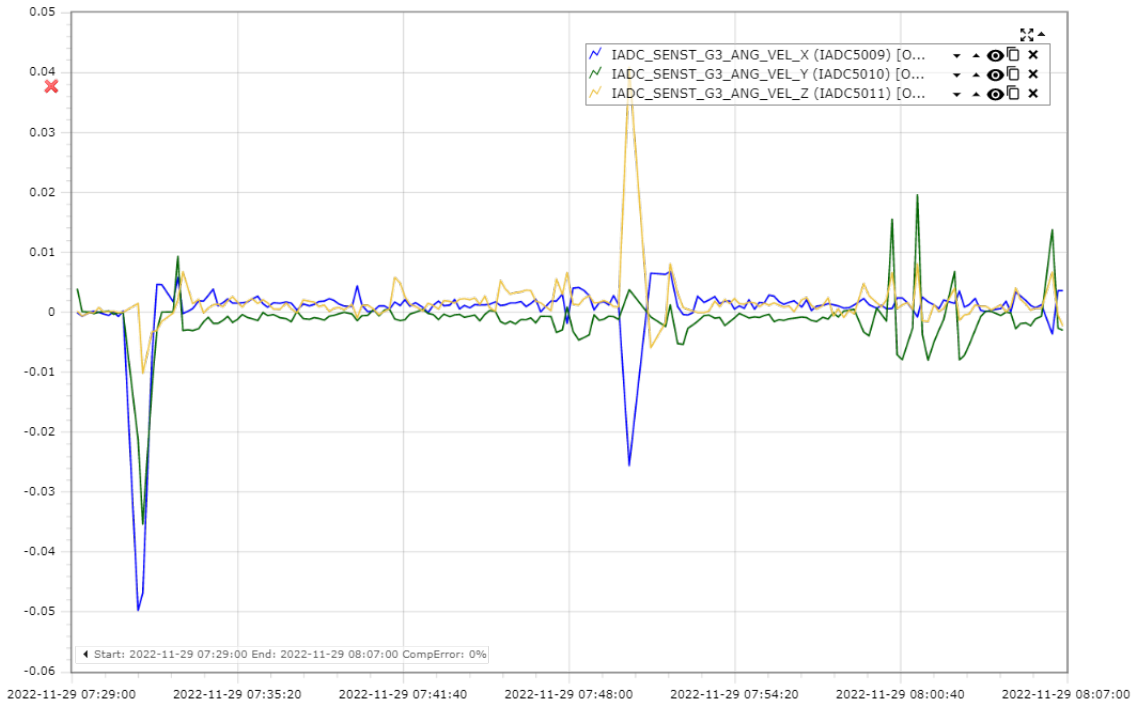


Figure G.7: Window 2 29/11/22 7:29 - 8:07 IMU Data

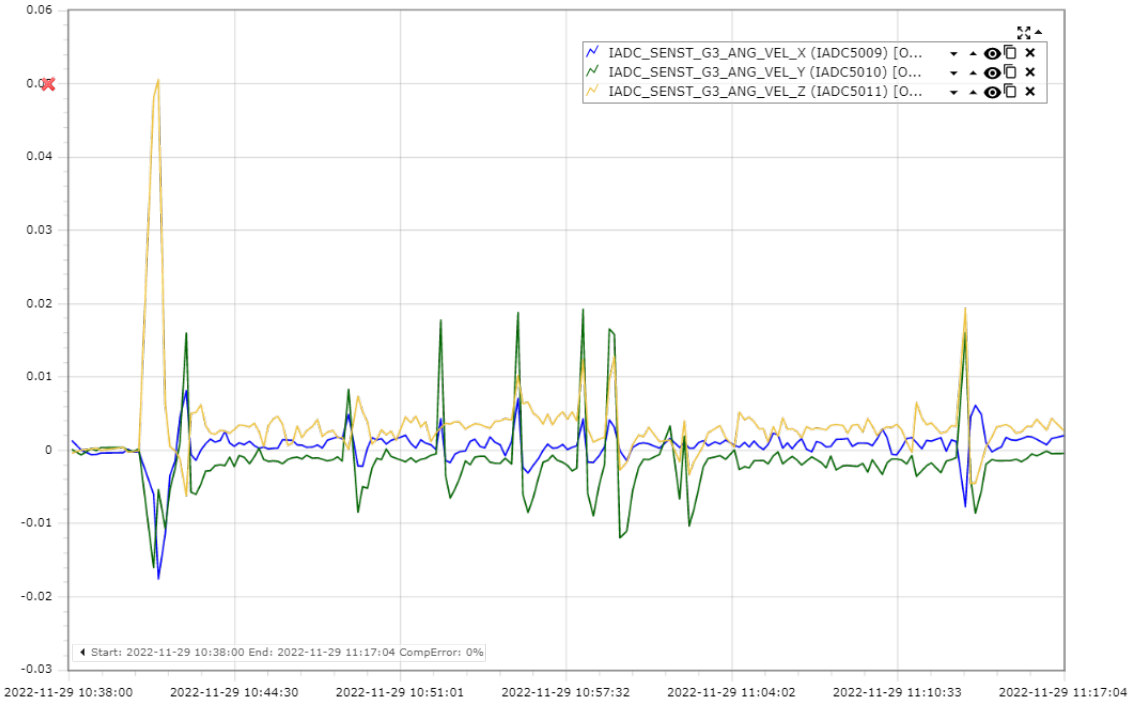


Figure G.8: Window 3 29/11/22 10:38 - 11:17 IMU Data

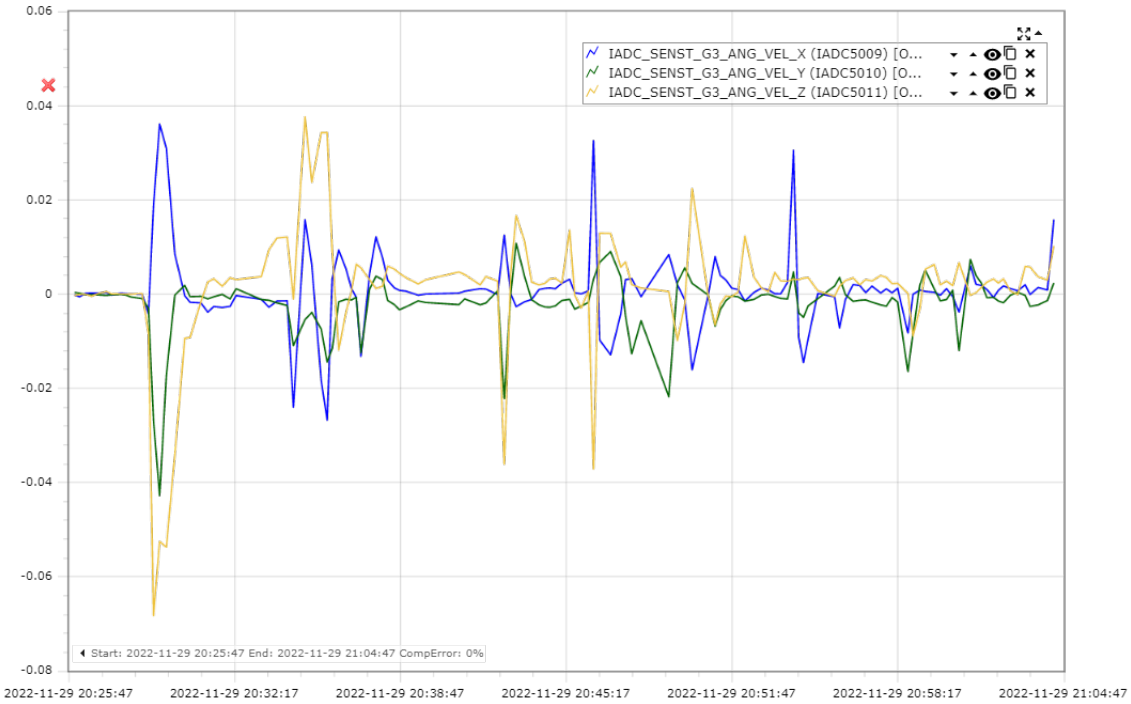


Figure G.9: Window 4 29/11/22 20:25 - 21:04 IMU Data

G.4. Sun Angle Telemetry

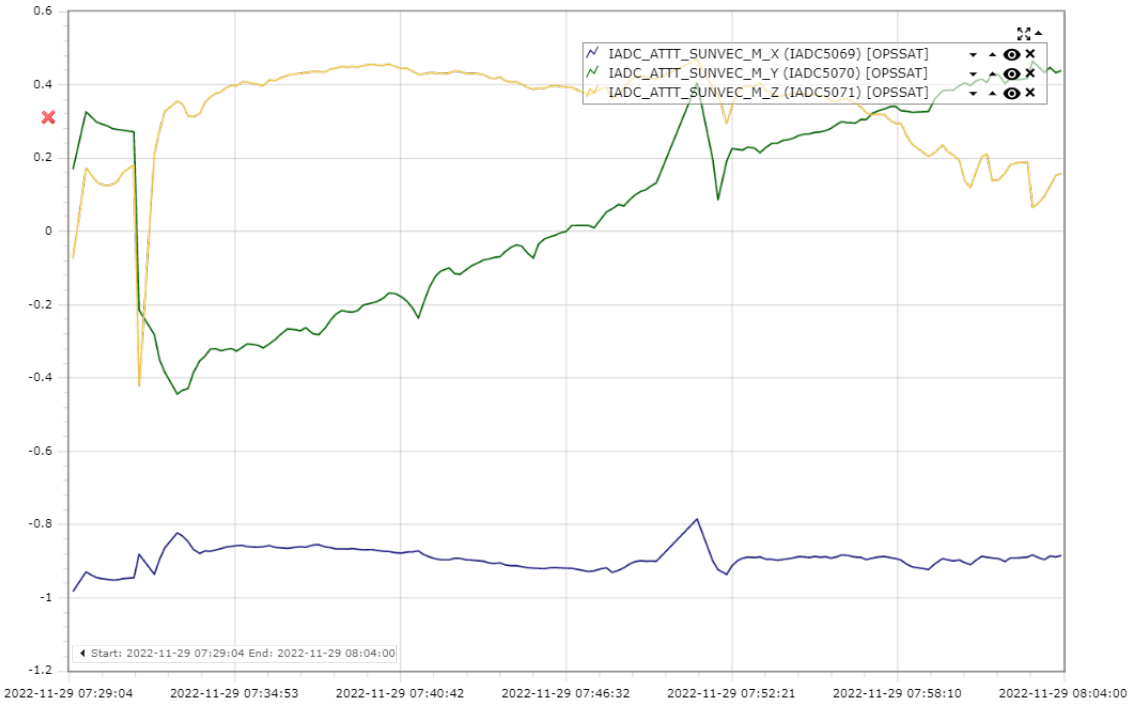


Figure G.10: Window 2 29/11/22 7:29 - 8:07 Sun Angle Data

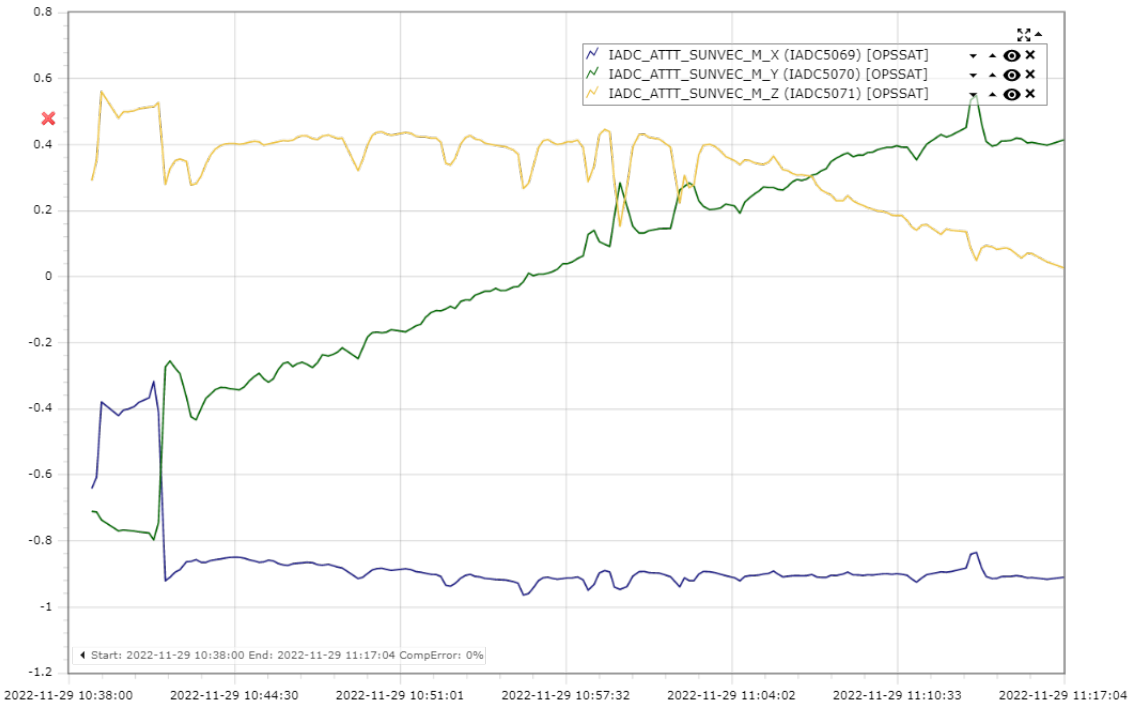


Figure G.11: Window 3 29/11/22 10:38 - 11:17 Sun Angle Data

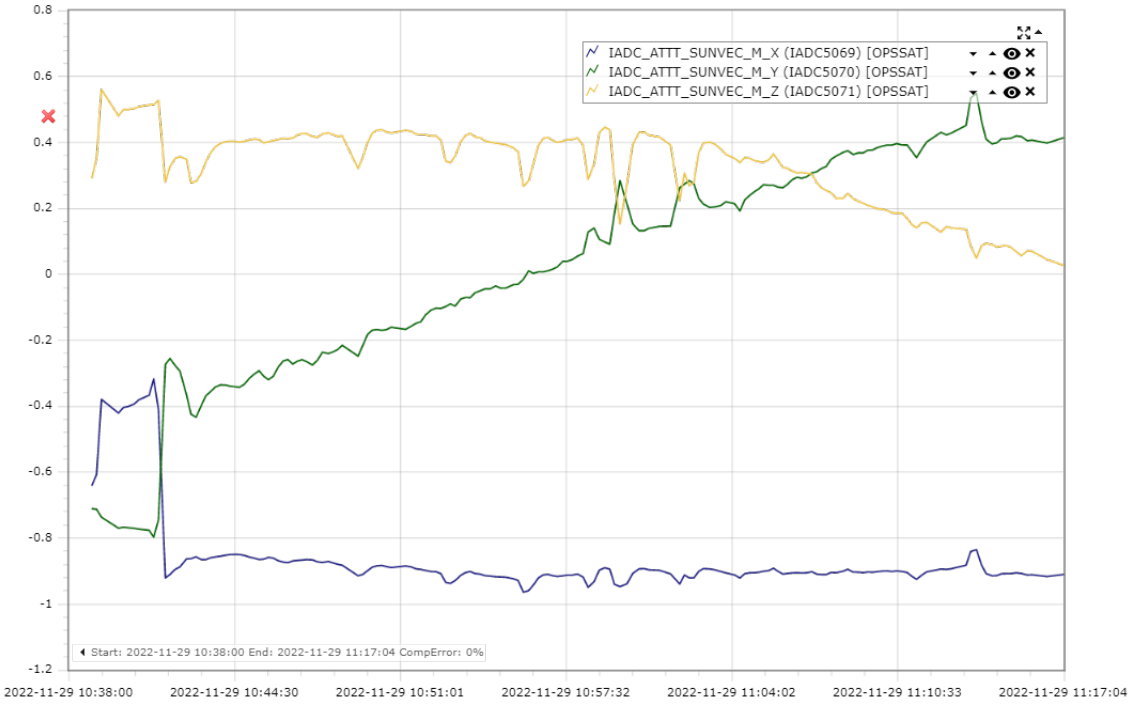


Figure G.12: Window 4 29/11/22 20:25 - 21:04 Sun Angle Data