DEEP LEARNING FAULT PROTECTION APPLIED TO SPACECRAFT ATTITUDE DETERMINATION AND CONTROL

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Justin R. Mansell

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

August 2020

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Dr. David A. Spencer, Chair

School of Aeronautics and Astronautics

Dr. Daniel DeLaurentis

School of Aeronautics and Astronautics

Dr. Carolin Frueh

School of Aeronautics and Astronautics

Dr. James Garrison

School of Aeronautics and Astronautics

Approved by:

Dr. Gregory Blaisdell

Head of the School Graduate Program

To my parents, Robert and Tina.

To my brother, Trevor, and my dog, Benji.

"Inspiration unlocks the future. Technology will catch up!"

- Giovanni Battista Caproni, *The Wind Rises*

ACKNOWLEDGMENTS

I wish to express my immense gratitude to my advisor, Dr. David Spencer, for his dedicated mentorship over the past three years and whose patience, guidance, and encouragement are a continual source of inspiration. The countless opportunities he entrusted to me have been instrumental to my growth as an engineer. I would also like to thank the members of his research group, Dr. Sylvain Renevey, Rohan Deshmukh, Jannuel Cabrera, Tom Cunningham, Arly Black, Samantha Dickmann, Jay Iuliano, and Dr. Anthony Cofer for their collaboration and camaraderie.

I owe my sincere thanks to Purdue University and the School of Aeronautics and Astronautics for their financial support in the form of multiple teaching and research assistantships during my studies. Among the faculty, I would like to thank my committee members, Dr. Daniel DeLaurentis, Dr. Carolin Frueh, and Dr. James Garrison for their insights and suggestions regarding my work. I would also like to recognize the invaluable assistance provided by my mentors beyond Purdue: Dr. Joseph Bakambu and Nader Abu El Samid at MDA Corporation, Dr. J. Stephen Herring at Idaho National Laboratory, Dr. Michael J. Grant at Sandia National Laboratories, as well as Dr. Andrew Yau, Dr. Christoper Cully, Dr. Phil Langill, Andrew White, and Andrew Howarth at the University of Calgary.

The results of the LightSail 2 mission have formed an important part of this research. I am grateful to the LightSail 2 flight team, Dr. David Spencer, Dr. John Bellardo, Dr. Bruce Betts, Barbara Plante, and Alex Diaz, as well as the members and donors of The Planetary Society, for their collaboration and dedication to the mission.

This work could not have been completed without the unwavering support of my friends and family. Above all, I must express my sincerest gratitude to my parents, Robert and Tina, for embracing my enthusiasm for space exploration and doing ev-

erything in their power to support it. Their unconditional love has been an enduring source of strength. Likewise, it is with deep appreciation that I thank my brother, Trevor, my grandmother, Mabel, my uncle, David, and my aunt, Alice, for their love and commitment.

Finally, I wish to acknowledge the fellowship of the numerous friends who have been by my side throughout my studies. To my girlfriend, Amrita, to the Bender family, to my friends, Geoffrey, Alex, Imad, Katie, Jenna, Michal, and all the others who I do not have enough room to thank personally, your support helps make my studies worthwhile. Thank You.

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

LIST OF FIGURES

SYMBOLS

$f(x)$	Anomaly decision function
k	Kernel similarity measure
L	Total spacecraft angular momentum
$\hat{L}$	Estimated spacecraft angular momentum
$\mathcal{L}$	Arbitrary loss function
q	Attitude quaternion
w	Hyperplane normal
γ	Utility placeholder
μ	Magnetic moment or signal mean
ν	Outlier fraction
ρ	Hyperplane offset
σ	Sigmoid function or signal standard deviation
ϕ	Kernel mapping function
Ω	Reaction wheel rotation speed
ω	Attitude rotation rate

ABBREVIATIONS

ADCS	Attitude determination and control subsystem
AOCS	Attitude and orbit control subsystem
CDH	Command and data handling
6DOF	6 degrees of freedom
DSS	Distributed satellite system
FDIR	Fault detection, isolation, and recovery
FSW	Flight software
IGRF	International geomagnetic reference field
IMU	Inertial measurement unit
LDC	Long duration counter
LSTM	Long short-term memory
MDP	Markov decision process
MSE	Mean squared error
OBCP	On board control procedure
OBSW	On board software
OCSVM	One-class support vector machine
RNN	Recurrent neural network
RPM	Rotations per minute
SMART-FDIR	Sequence-based mapping of anomalies to root causes from telemetry FDIR
TLE	Two-line element
TLM	Telemetry
TRAC	Triangular retractable and collapsible
TTC	Telemetry, tracking, and control
V&V	Verification and validation

GLOSSARY

Anomaly	an unexpected deviation from nominal performance
Collective anomaly	data points that are anomalous when compared to data points from other signals
Contextual anomaly	data points that are anomalous when compared to other data points in the same signal
Failure	the inability to perform an intended function
Fault	the underlying cause of an anomaly or failure
Incipient fault	a fault that emerges gradually
Kernel function	a positive-definite function that defines a notion of similarity between pairs of data points
Transient fault	a fault that is not persistent
Point anomaly	a data point that is anomalous on its own
Utility	a subjective measure of the value of an outcome

ABSTRACT

Mansell, Justin R. Ph.D., Purdue University, August 2020. Deep Learning Fault Protection Applied to Spacecraft Attitude Determination and Control. Major Professor: Dr. David A. Spencer. Associate Professor.

The increasing numbers and complexity of spacecraft is driving a growing need for automated fault detection, isolation, and recovery. Anomalies and failures are common occurrences during space flight operations, yet most spacecraft currently possess limited ability to detect them, diagnose their underlying cause, and enact an appropriate response. This leaves ground operators to interpret extensive telemetry and resolve faults manually, something that is impractical for large constellations of satellites and difficult to do in a timely fashion for missions in deep space. A traditional hurdle for achieving autonomy has been that effective fault detection, isolation, and recovery requires appreciating the wider context of telemetry information. Advances in machine learning are finally allowing computers to succeed at such tasks. This dissertation presents an architecture based on machine learning for detecting, diagnosing, and responding to faults in a spacecraft attitude determination and control system. Unlike previous approaches, the availability of faulty examples is not assumed. In the first level of the system, one-class support vector machines are trained from nominal data to flag anomalies in telemetry. Meanwhile, a spacecraft simulator is used to model the activation of anomaly flags under different fault conditions and train a long short-term memory neural network to convert time-dependent anomaly information into a diagnosis. Decision theory is then used to convert diagnoses into a recovery action. The overall technique is successfully validated on data from the LightSail 2 mission.

1. INTRODUCTION

"To design a spacecraft right takes an infinite amount of effort. This is why it's a good idea to design them to operate when some things are wrong."

- Akin's second law of spacecraft design [1]

Spacecraft are complex machines that operate in a relentlessly hostile environment. Problems are inevitable, and the history of space flight is replete with tales of ruin [2]. The goal of fault management is to ensure satellite safety by detecting anomalies, determining their cause, and ultimately returning the satellite to a fully operational state. The complete cycle is referred to as fault detection, isolation, and recovery (FDIR).

1.1 Motivation

Given the finite response times and attention of human operators, a key objective for spacecraft fault management is to provide autonomy to the FDIR cycle. Most often this is accomplished through on board monitoring of telemetry with techniques such as threshold limit checking [3–5]. If a threshold violation is detected, on board control procedures (OBCP) trigger a predefined response such as switching to a redundant component. If the response fails to rectify the problem, or if the anomaly impacts multiple subsystems or a critical process, the standard procedure is to transition the satellite into safe (or "safe-hold") mode. In safe mode, the spacecraft halts all non-critical operations and transitions to a power-positive attitude that enables communications with the ground [6, 7]. Depending on tracking coverage and staffing schedules, several hours or days may be required for the operations staff to resolve the problem and return the satellite to normal operations. Meanwhile, the satellite is effectively disabled, impacting mission return.

The prevailing approach to FDIR is poorly equipped for the demands of current and future space missions [3–6, 8, 9]. Surveys on the impact of on-orbit failures reveal that operators are unable to fully deal with a large proportion of failures despite apparently adequate training [10, 11]. Emerging mission concepts such as distributed satellite systems (DSS) will exacerbate this issue. These are satellite constellations which allocate functionality over many members rather than a few large spacecraft [12–14]. Concepts for DSS have existed for decades, but it is only recently that advances in small satellites, microelectronics, and inter-satellite communications have elicited the development of constellations of truly enormous size. Table 1.1 presents a summary of mega-constellations that have either been proposed or are under development, with applications ranging from global broadband internet to continuous reconnaissance [14–17].

Table 1.1: Proposed and under development mega-constellations.

Constellation	Satellites	Purpose	Status
SpaceX Starlink	12,000+	Global broadband	482 operational
Amazon Kuiper	3,236	Global broadband	Proposed
OneWeb	648	Global broadband	68 operational
Telesat Canada	512	Global broadband	Proposed
Planet	250	High revisit time imaging	150 operational
Spire Global	175	Maritime, aviation & weather tracking	80 operational
Kepler Communications	140	Inter-satellite links	2 operational

In principle, the unprecedented size of these constellations establishes system reliability through sheer redundancy. Yet, a recent survey of planetary exploration spacecraft found that planetary spacecraft experienced an average of 1–2 safe mode

entries per year [18]. For small satellites in Earth orbit the rate is likely to be higher due to lower radiation tolerance and higher risk posture. Assuming just two safe mode entries per year per spacecraft, a constellation of 1000 satellites can expect an average of 5 spacecraft safing events per day. Given the typical response rates from ground operators, it is clear that major improvements in autonomous FDIR are necessary if operations costs are to remain sustainable.

In addition to DSS, a higher level of FDIR autonomy will benefit satellite operations in hostile environments, where faults may originate from adversary attack. In such environments it is critical to recognize when anomalies are potentially the result of hostile actions and enact immediate countermeasures [19–21]. Spacecraft and missions of all sizes are also becoming increasingly complex. Future missions to the Moon and outer solar system face long signal delays and black-out periods that will require extended periods of autonomous operation [22–25]. An FDIR system that adapts to the priorities of different mission phases could extend fault protection to critical events such as orbit insertion maneuvers where traditional FDIR systems have normally been switched off [7].

1.2 Spacecraft Faults, Failures, and Anomalies

Virtually all spacecraft will experience an anomaly over the course of their mission. An *anomaly* in this case is defined as an unexpected deviation from nominal performance. A *failure* occurs when the system becomes unable to perform an intended function. Finally, a *fault* is the underlying source of an anomaly or failure. Faults may occur individually or in a cascade in which faults lead to other faults. It is the ultimate root cause that is usually of most interest to the satellite operator because this is what the recovery response must address [26]. This section outlines the most common types and sources of anomalies that occur on spacecraft.

1.2.1 Types of Anomalies

Off-nominal deviations in a signal occur in a catalogue of ways. The deviation may appear abruptly or gradually[1]. Furthermore, it may be transient, in which the anomaly appears and disappears sporadically, or it may be persistent. Figure 1.1 illustrates several common ways anomalies can appear in continuous signals [27, 28]. They include offsets, intermittent spikes, repeating values, or otherwise healthy signals that nonetheless give conflicting results. For discrete signals, anomalies can appear in violation of an expected sequence of transitions as illustrated in Figure 1.2, or as a failure to transition in the expected time.

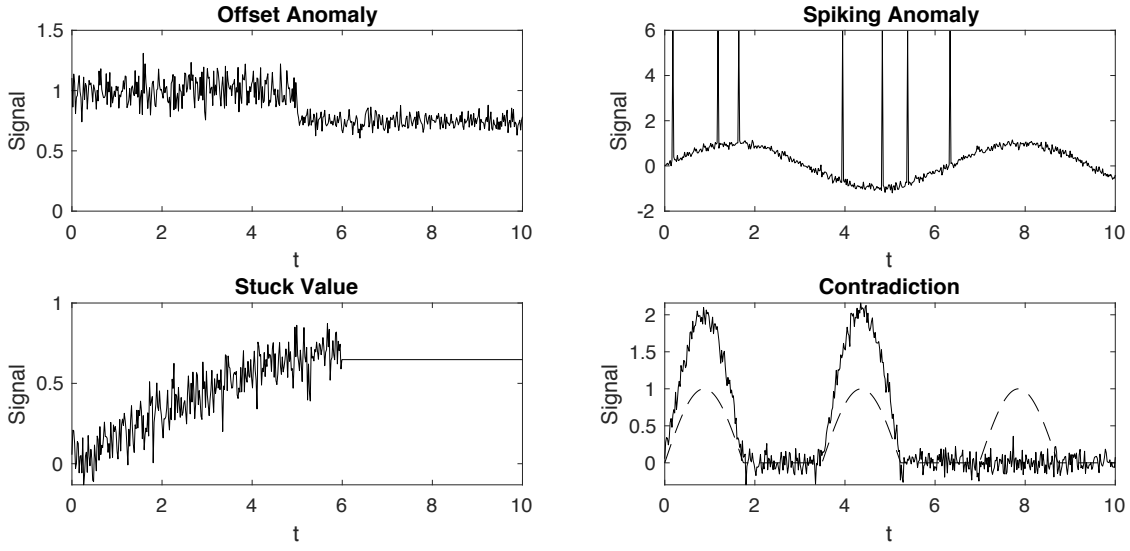


Figure 1.1: Types of anomalies in continuous signals.

The different types of anomalies are often categorized into three classes in FDIR literature: [27]:

- *Point* anomalies: wherein an individual instance of data can be considered anomalous independently of any other data point. For example, the spiking anomaly in Figure 1.1.

[1]Often called incipient

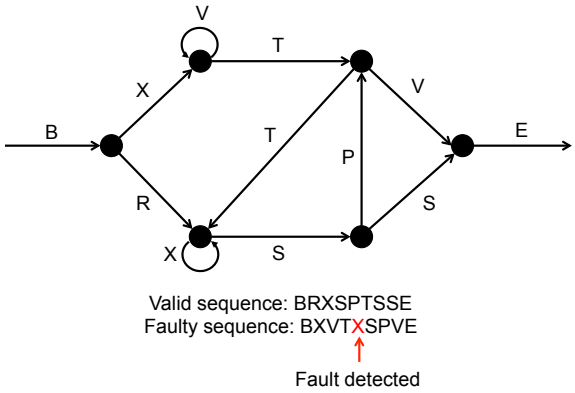


Figure 1.2: Example of an anomaly in a sequence of discrete mode transitions.

- *Contextual* anomalies: where an instance of data is anomalous based on a comparison to other instances in the same signal. Examples are the offset and stuck values in Figure 1.1, or the violation in Figure 1.2. While the individual data points are valid, their relationship to other points in the signal is what indicates the anomaly.

- *Collective* anomalies: where groups of data are only anomalous when considered as part of a larger collection of signals. For example, the contradiction in Figure 1.1.

1.2.2 Sources of Anomalies

There is currently no broadly accessible database of satellite anomalies due to the proprietary nature of the data involved. However, a 2014 survey of satellite anomalies by RAND Corporation outlines the following common root causes of anomalies [29]:

- Total ionizing dosage
- Electrostatic discharge
- Surface/internal charging
- Single event effects
- Faulty hardware or design
- Operator error
- Electromagnetic interference
- Cyberattack

The report states that electromagnetic interference can be intentional or malicious. Additionally, the "faulty hardware/design" category is expansive and includes subtle causes such as corrosion (particularly from atomic oxygen), outgassing, and unexpected environmental conditions that exceed the design specifications of the spacecraft [2]. Faulty hardware and designs are particularly common for CubeSats due to their low redundancy and accelerated development schedule [30].

The consequences of anomalies are as varied as their causes and span the spectrum from inconsequential to total loss of the spacecraft. It is also the case that not all anomalies are due to an underlying fault with the spacecraft. Since anomalies are defined relative to nominal behavior, they can stem from an incorrect expectation of what the nominal behavior is. This is particularly true for spacecraft because, due to the limitations of ground testing, the initial checkout on orbit may be the first time the entire spacecraft has operated in flight-like environment. Being able to recognize when an anomaly may be a false positive is an important skill for operators that depends greatly on intuition, experience, and understanding the wider context in which the anomaly occurs.

Not all spacecraft subsystems contribute equally to the catalogue of faults and anomalies either. A survey of on-orbit failures between 1980 and 2005 reports that 32% were related to attitude and orbit control subsystems (AOCS), the largest of any single category [10]. Other major sources include the electrical power subsystem (particularly solar panels), command and data handling (CDH), and telemetry, tracking, and control (TTC) [2, pp. 177-179] [10,11]. In the European Galileo program, AOCS, CDH, and power account for more than half of all FDIR requirements [8].

The prevalence of AOCS among on-orbit faults is due to the complexity of this subsystem [10]. It encompasses numerous sensors, actuators, and software functions, many of which can be perturbed by environmental effects or faults in other subsystems. Accurate attitude and orbit control is often critical to mission objectives and can even be necessary for spacecraft survival, such as for spacecraft that rely on proper pointing to provide sufficient solar power and ground communication. Fig-

ure 1.3 summarizes the sources and impacts of AOCS failures from the 1980–2005 study [10]. Unlike most other subsystems, the distribution of AOCS failures is spread nearly evenly throughout the mission life rather than being concentrated early after launch. AOCS failures resulted in total loss of the mission in 30% of cases and degraded the ability to meet the mission objectives in 56% of cases [10].

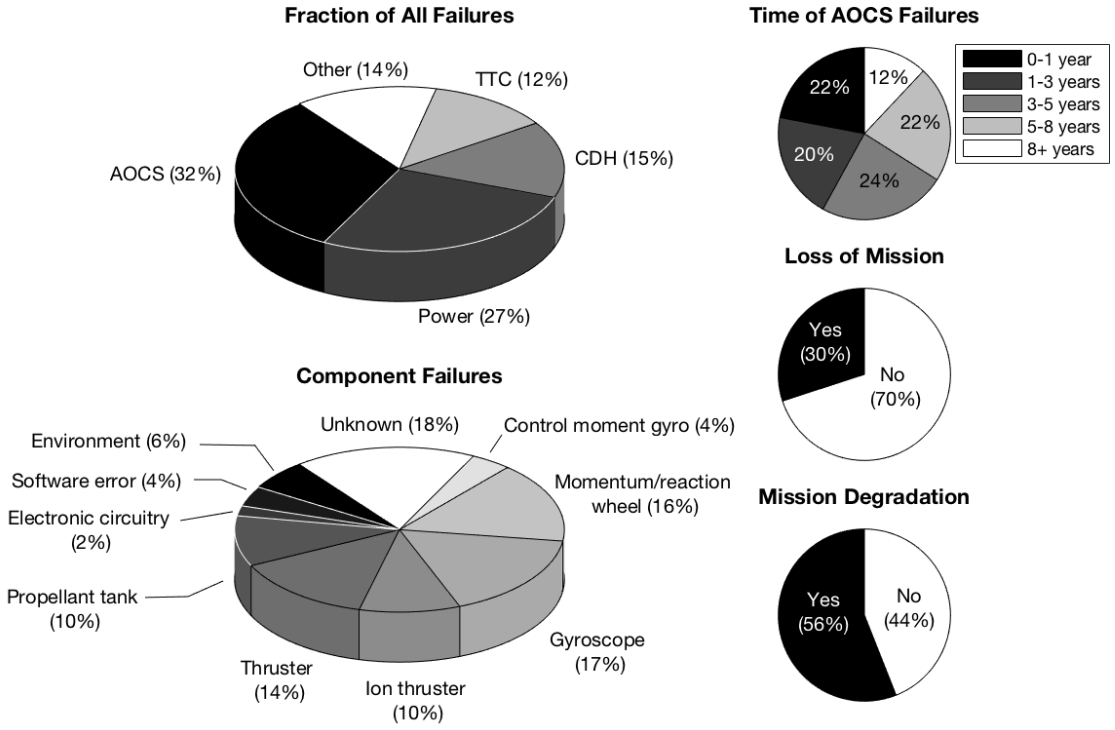


Figure 1.3: Sources and impacts of AOCS failures. Charts reproduced based on data from [10].

More recent studies since the proliferation of the CubeSat standard have found CubeSat failures to be dominated by electrical and "unknown" failures rather than AOCS [31]. However, the distribution is skewed by a large number of amateur satellites lacking active attitude control and many of which end up "dead on arrival" [32, 33]. As more CubeSats are adopted by professional organizations to carry out missions with serious AOCS requirements, the failure distribution will likely become more like that of larger spacecraft. Additionally, the complexity of onboard

software is rising exponentially and the infusion of autonomy is outstripping the capability of traditional V&V practices [8, 34]. The proportion of AOCS failures due to software will likely rise in the future compared to Fig. 1.3.

In summary, AOCS represents a key area to focus innovative FDIR that can impact mission success well after launch. Nevertheless, given the diverse and evolving assortment of anomalies, faults, and failures, techniques that are applicable to all subsystems are desired.

1.3 State of the Art

1.3.1 Methods of Fault Detection

When a fault occurs, anomalies or failures are discovered through analysis of spacecraft telemetry. This is the task of fault detection. In general, fault detection need not be concerned with all of the possible ways a function can go awry, because in order to be detected all that is needed is a deviation from normal functioning [35]. Therefore, in theory, fault detection can be performed with 100% accuracy [26]. The terms fault detection and anomaly detection are often used interchangeably, with anomaly detection representing a slightly broader scope since it encapsulates the detection of all off-nominal deviations rather than only those caused by an underlying fault.

Interest in anomaly detection spans virtually every engineering discipline and beyond. Important applications can be found in fields ranging from finance to cyber security [27, 36–38]. A broad range of techniques have been explored to address anomaly detection problems with various characteristics. Figure 1.4 categorizes the major approaches in the literature. The techniques can focus on anomalies in individual signals or address collective anomalies between multiple signals. The following subsections outline the families of techniques and their applications to spacecraft FDIR.

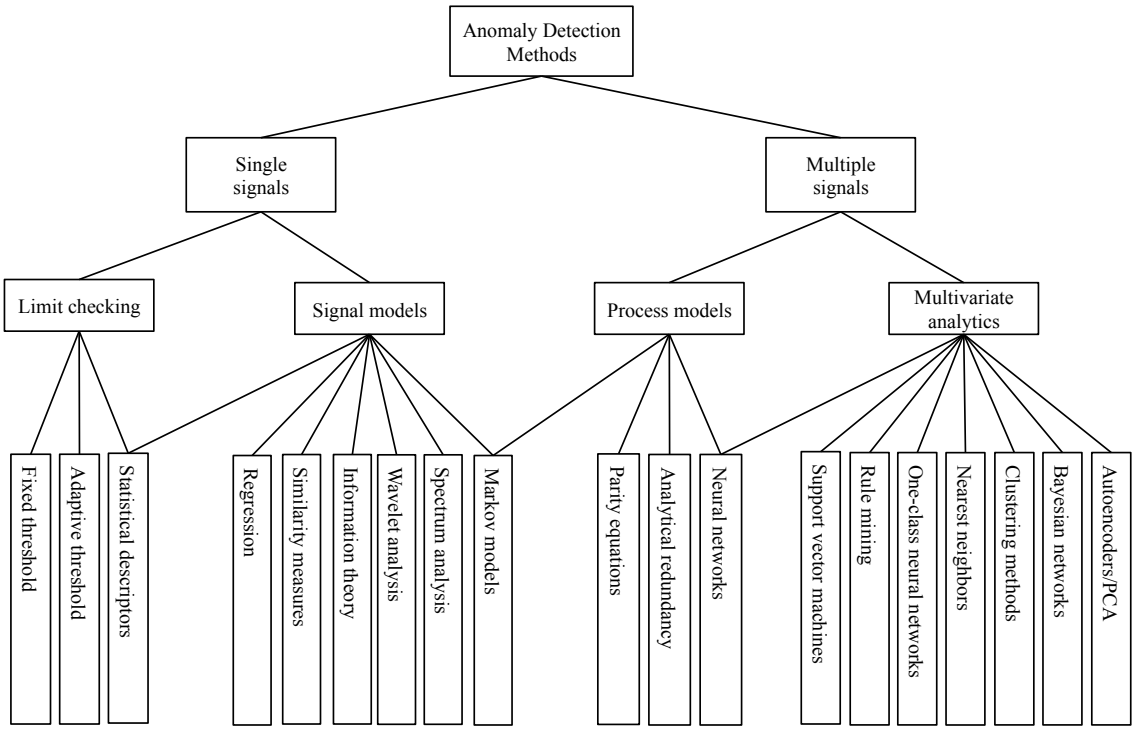


Figure 1.4: Taxonomy of anomaly detection techniques.

Limit Checking

The simplest and most widely applied forms of anomaly detection for individual signals are limit checking methods [3,4,8]. Upper and lower thresholds are set to define a range of acceptable values for a particular signal and violations are flagged as potential anomalies. Sophisticated limit-checking techniques may check for persistent or repeated violations or use "soft" and "hard" limit thresholds [39, 40]. While straightforward to implement, the limitations of this approach motivate much of the FDIR literature. For instance, setting thresholds typically requires detailed component-level knowledge of the signal. Even when the necessary expertise is available, the characteristics of a signal may be unknown before it is monitored in its eventual operating environment. Thresholds are often set conservatively for this reason, causing a large number of false positives and leading to high mission outage times [8]. The solution is often laborious manual tuning of thresholds to balance sensitivity and false positives.

An alternative are adaptive threshold methods that adjust the limits based on past trends or information external to the signal itself [41–44]. Many of these methods are based on process models or data analytics, which are discussed later.

Where it is not possible to define suitable anomaly thresholds for the signal directly, statistical techniques using one or more moving windows can screen data for statistically unlikely variations [27, 39, 45]. Thresholds on bulk window statistics such as mean or variance can then be implemented. Other methods such as weighted cumulative sums can sense anomalies in long term trends [28]. These methods can address both point and contextual anomalies.

Signal Models

The signal modeling approach seeks increasingly rich representations of a signal that can improve sensitivity to contextual anomalies. Anomalies are sensed when the parameters modeling the signal deviate from typical values. The problem of finding arbitrary contextual anomalies is thereby transformed into the much easier problem of identifying point anomalies, to which any point anomaly detection method can be applied. This approach includes moving windows of bulk statistics but also parametric regressions of underlying probability distributions [41] and autoregressive models [27, 36]. If signals contain a uniform number of samples, a similarity measure can be defined to create an abstract notion of distance between signals with which to search for anomalies [27, 36, 37]. Spectral/wavelet analysis is yet another way of representing signals and has been applied to FDIR for spacecraft power subsystems [46, 47].

Methods applicable to symbolic as well as numerical signals include window scoring, compression, and Markov models [37]. Compression finds abnormalities in the information content of the signal while the latter construct finite state automata similar to Fig. 1.2 that flag anomalies when sequences violate the allowed transitions [37, 45, 48, 49].

Process Models

Markov models are not limited to individual signals, but can also represent combinations of signals [48]. The modeling of multiple simultaneous signals, often including their inputs and interactions, forms the process model approach [50–52]. This approach allows for the detection of collective anomalies in addition to point and contextual anomalies. The central technique is analytical redundancy whose core idea is this: given inputs to and/or measurements of the system, predict the behavior of the system using the process model and compare the results with the real system. Whereas physical redundancy compares parallel sensors for signs of a discrepancy, analytical redundancy compares measurements to analytically derived quantities. In the parity equations method, large residuals in the states or outputs of the model signify and anomaly [53]. Transfer functions or state-space representations (including Kalman filters) of the underlying system can be used for modeling [53–58]. If the dynamics of the process model are either unknown or too complicated to implement, neural networks can be used to approximate system behavior [44, 59, 60].

Multivariate Analytics

Multivariate analytics is a collection of techniques for identifying outliers of a generic vector, $\mathbf{X}$. A common assumption is that representative examples of nominal data are readily available or that anomalies are rare in any given collection of data. Thus, the problem is formally stated: given samples of nominal data $\{\mathbf{X_o}, ..., \mathbf{X_n}\}$, is a new observation $\mathbf{X}^*$ nominal? Multivariate methods come in two varieties: multi-class and one-class. In the multi-class setting, an additional vector $\mathbf{Y}$ is available corresponding to each $\mathbf{X}$ denoting the class to which $\mathbf{X}$ belongs. A classification technique is then applied to predict the class to which a new instance $\mathbf{X}^*$ belongs. If the resulting confidence or probability for every class is low, then $\mathbf{X}^*$ is flagged as an anomaly. Classifiers applied to anomaly detection in this way include rule-mining, neural networks, support vector machines, and Bayesian networks [36, 61].

Several of these methods also include variants applicable to the one-class problem where $\mathbf{Y}$ consists entirely of a single class (e.g. "nominal data"). Autoencoder neural networks attempt to transform each $\mathbf{X_i}$ in the nominal set to itself by using a number of input and output neurons equal to the number of features (i.e. components) of $\mathbf{X_i}$ [62,63]. If the number of hidden neurons is fewer, the network is forced to compress the information of the nominal set into an abstract representation from which it then reconstructs an approximation of $\mathbf{X_i}$ at the output. If the reconstruction error of a new input $\mathbf{X}^*$ is high, it is considered an anomaly. Anomaly detection via principal component analysis (PCA) operates on a similar principle [62]. Bayesian networks can extend the multi-class approach by including an anomaly class and using Laplace smoothing to remove the resulting zero prior probability (since the anomaly class has no representatives in the training set) [27]. Lastly, one-class neural networks and one-class support vector machines (OCSVM) use a hyperplane to separate the nominal data from the origin in a higher dimensional space, with the hyperplane representing the boundary to distinguish anomalies [64–67]. OCSVMs are described in more detail in Section 2.1.

The last two multivariate methods in Fig. 1.4 both invoke the assumption that anomalies are confined to low density regions of the feature space of $\mathbf{X}$. In clustering, nominal data is grouped into clusters and new data is evaluated based on its distance from each of the clusters. If the new $\mathbf{X}^*$ is far from any cluster, it is an anomaly. Similarly, nearest neighbours methods compute either the distance of the new data point to its nearest neighbor(s) or the local density to identify anomalies. Both have been applied to anomaly detection in aerospace applications [47, 68–70].

Other methods include information-theoretic and spectral techniques [27, 36, 38]. Though the various multivariate methods are fundamentally point anomaly detectors, their generality means they can be combined with signal modeling to fully address contextual anomalies. Multiple signals can also be combined into a single input vector $\mathbf{X}$ to enable detection of collective anomalies. Yet, their most important advantage

is that they are implemented in a purely data-driven way that avoids the need for an expert designer to set thresholds or develop process models.

1.3.2 Methods of Fault Isolation

Fault isolation is typically more difficult than fault detection because not all of the information needed to uniquely identify the source of an anomaly may be present [26, 35]. Thus, the traditional approach for spacecraft has been to skip onboard fault diagnosis altogether and instead react directly to anomalies [7, 71]. Ground operators must then assume the time-consuming task of formulating hypotheses about anomaly root causes based on their expert knowledge and checking telemetry for clues that confirm or reject these hypotheses [9]. Since the late 1980s, however, advances in the field of artificial intelligence have introduced techniques for autonomously isolating faults given observations about a system. Figure 1.5(a) presents the major methods that have been explored in FDIR literature. The terms fault isolation and fault diagnosis are often used interchangeably.

Propositional Logic

In the logic-based approach to fault diagnosis, expert knowledge about the relationships between faults and symptoms is represented using a catalogue of "If-Then" rules called the knowledge base. Computer reasoning techniques such as Selective Linear Definite (SLD) clause resolution and assumption-based truth maintenance are then used to hypothesize faults that are consistent with the observed symptoms [72, Chapter 5]. The first successful applications of this approach to spacecraft FDIR were expert systems, which were initially investigated in the late 1980s as a way to diagnose satellite anomalies induced by the space environment [73–75]. The inclusion of fuzzy logic later allowed these systems to handle uncertainty [76]. An alternative means of incorporating uncertainty as well as missing or contradictory information is with Dempster-Shafer Evidence Theory [77, 78]. The theory departs from traditional

probability by leaving some probability or "belief" unassigned amongst competing hypothesis, thus representing the level of ignorance. The belief functions evolve according to a combination rule when new evidence is collected. The technique has found modern applications in sensor fusion, fault diagnosis, and satellite tracking [79–82].

While logic-based systems represent a mature and rigorous approach to fault diagnosis, they are limited by the need to assemble expert rules. In subsystems such as AOCS, the relationships between faults and symptoms can be difficult to represent cleanly with logical propositions. Even when this is possible, the resulting knowledge base can exceed more than 200 rules [74]. Maintaining the consistency and completeness of such a large rule base is difficult, especially if changes are necessary late in development or during operations [42]. This motivates continuing interest in model-based methods of fault diagnosis.

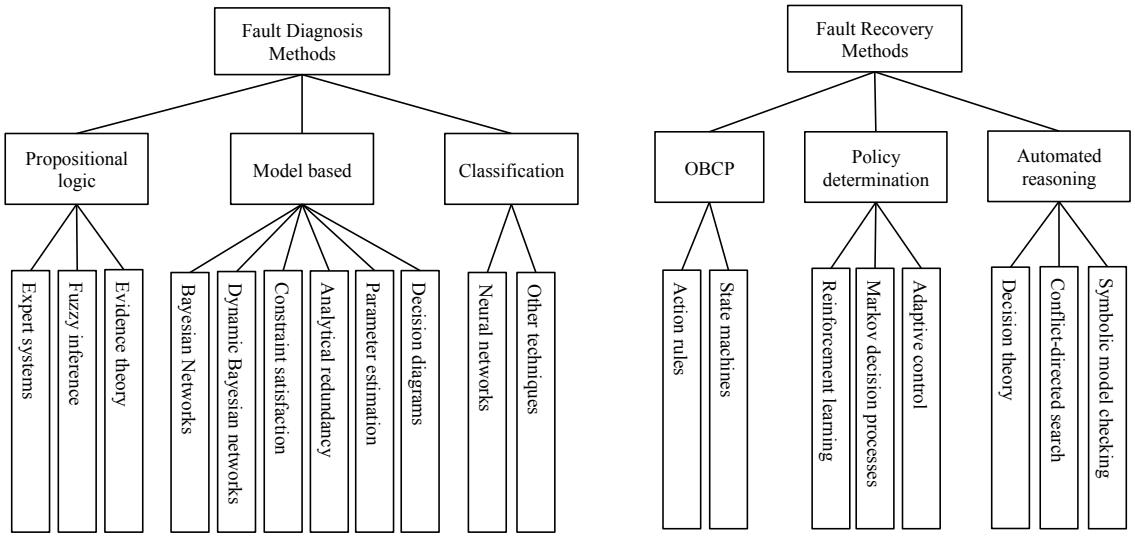


Figure 1.5: Taxonomy of fault diagnosis and recovery methods.

Model-based

Model-based methods are an attractive approach to FDIR because they can be inspected visually and are highly modular [4]. The development of subsystem models is an integral part of spacecraft design and these models represent rich sources of expert knowledge. As such, there is significant interest in reusing these models to aid fault isolation during operations [9].

A pioneering application to space flight was the "Livingstone" diagnosis system, which was incorporated into the remote agent on Deep Space 1 in 1999 and several missions afterwards [83–85]. Livingstone uses a qualitative process model for anomaly detection. To diagnose discrepancies between the model and flight unit, the system leverages the tools of propositional logic by converting fault diagnosis into a constraint satisfaction problem and applying a conflict-directed graph search [72, Chapter 3] [86]. Other qualitative models for fault diagnosis include decision diagrams such as fault trees and state machines [4, 87].

Bayesian networks are a quantitative model-based approach to diagnosis [72, Chapter 8]. Bayesian networks model uncertain knowledge by representing probabilistic causal dependencies as a directed acyclic graph. Nodes in the graph represent the states of components in the system and connections between nodes convey conditional probabilities about how the state of one node influences others. Given a set of observations about the states of some nodes in the network, Bayesian inference is used to compute probabilities for the states of other nodes that are not directly observed. A variation are dynamic Bayesian networks, which copy a static Bayesian network through time to model time-dependent relationships. Bayesian networks have proved popular in diagnosing electrical systems [28, 88], and applications to spacecraft FDIR have included power systems for rovers [89], ground station monitoring systems [90], propulsion and orbit control [42, 91], and display of critical information [92]. The network's structure and parameters can be designed by an expert, learned from data, or generated automatically from fault trees [93–97].

Finally, analytical redundancy approaches can provide limited fault diagnosis by identifying exactly which signals deviate from nominal operating conditions [50–52,98, 99]. However, a human operator or separate system is often still needed to interpret the combination of deviant signals and arrive at a root cause diagnosis [60, 100]. A closely related approach is parameter estimation, wherein sensor measurements are used in conjunction with the model to infer properties (e.g. friction or electrical resistance) of system components that are not directly sensed [101]. If the estimated value of a component differs by an excessive amount, the fault can be localized to that component. This approach allows the severity of faults to be measured, which can influence the appropriate response [98].

Classification

In some applications, data from off-nominal scenarios may be readily available. Fault diagnosis thus becomes a classification problem between a nominal class and one or more fault classes. Numerous techniques have been applied in this domain, including neural networks [98,100], clustering/nearest neighbors [4,47], and traditional machine learning models [51, 102]. Like many methods of anomaly detection, signal processing such as PCA or wavelet analysis is usually needed to reduce complex signals to abstract inputs for classification [98]. However, some advanced types of neural networks can skip this step and operate on signals directly [60,103]. Given the need for samples of faulty data, applications are mostly focused on detection of manufacturing defects, but studies have been performed for spacecraft electrical power systems [47] and rocket engines [104].

1.3.3 Methods of Fault Recovery

Once a fault has been detected and isolated, either by ground operators or by an onboard system, the spacecraft must respond to mitigate current or imminent failures and ultimately return the vehicle to normal operations. This is the process of fault

recovery. A high level of human involvement is typically required because selection of the appropriate recovery action(s) is often highly situational. Generally it involves an assessment of fault severity, failure prognosis, identifying response options, assessing the likely outcomes of the response options, prioritizing responses, developing a recovery procedure, and ultimately executing the recovery. Potential response actions are highly varied, but are generally drawn from four categories [26]:

- Goal change: abandoning or downgrading the current objective
- Failure recovery: such as entering safe mode or enacting a hardware reset
- Failure masking: containing the fault (e.g. by switching to a redundant component)
- Operational avoidance: avoiding conditions that are predicted to lead to failure

The following subsections review the most prominent techniques for automating the fault recovery process. The approaches are summarized in Fig. 1.5(b). The selection of recovery actions in FDIR falls within the larger domain of automated planning and the methods described here are by no means exhaustive. Extensive literature reviews can be found in [8, 50–52, 105].

Onboard Control Procedures

Reconfiguration of the spacecraft in the event of an anomaly is traditionally handled by onboard control procedures (OBCP) [7,8,105]. An illustrative example is the Solar Dynamics Observatory [71]. Response actions are sequences of pre-programmed commands that react to anomalies in isolation. The goal is not necessarily to address the root cause of these anomalies, but rather, "allow any cascading failures to settle into a communicative, power-positive, thermally safe attitude" [71]. OBCPs are triggered in a hierarchical fashion as depicted in Fig. 1.6. Anomalies at the lowest level may invoke no response at all aside from dumping relevant information into an error

log for the ground crew to inspect [7]. Unresolved anomalies propagate upwards to trigger increasingly extensive responses; critical or simultaneous anomalies often lead to spacecraft safe mode. Overall, response actions are limited solely to recognized threats to spacecraft safety and are generally conservative [105].

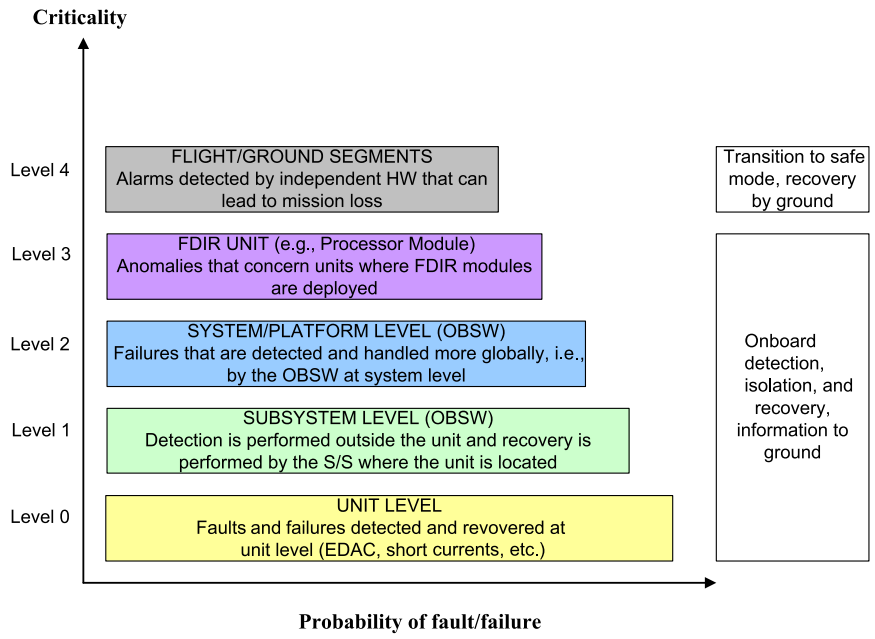


Figure 1.6: Hierarchical architecture for spacecraft fault response from [8].

OBCPs are typically implemented as a rule base that becomes challenging to manage for any complex mission. Recent work has introduced an alternative which represents OBCPs using finite state machines [4, 5, 106–109]. The state machine approach models a graph of the different phases/modes/conditions the satellite may be in along with the criteria for transitioning between them. This graphical approach is easier for engineers to specify, understand, and conduct verification and validation (V&V) on. Instead of specifying hundreds of global rules to capture every possible situation, designers need only specify the relevant transitions and the rules to trigger them from each state.

Policy Determination

With traditional OBCPs the actions to take in the event of an anomaly are rigidly predefined. Consequently, they struggle to cope with real aspects of the space environment such as uncertainty, contradicting information, dynamic evolution, and partial observability of the spacecraft's health [8, 105]. OBCPs also rarely take into account high-level goals or capabilities, and faults can lead to abandoning objectives that could have been safely continued. This has motivated extensive literature on model-based approaches to fault recovery [50–52, 105]. These approaches use a model of the spacecraft and the expected effects of each action or external event to search for actions that achieve a high-level goal. The sequence of actions is called the policy. State machines are one such approach, but others include Markov Decision Processes (MDP), reinforcement learning, and adaptive control. Since the latter methods do not embed the policy within the model itself, they are grouped under the category of policy determination.

The MDP method extends state machines to include external events and probabilistic transitions between states. By formulating an objective function to represent mission objectives, approximate dynamic programming techniques can be applied to determine the optimal recovery policy to maximize the completion of remaining objectives following a fault [105, 110–112]. For attitude control systems, the recovery procedure can involve switching to control laws that avoid failed components. In adaptive control, the control law is formulated such that it is robust to abrupt changes in system parameters following a fault [52].

The complexity of spacecraft is often such that explicit modeling of the possible states and interactions necessary to formulate an MDP can be prohibitive. Reinforcement learning provides a model-free approach to determining the transition and reward functions from experience [72, Chapter 11]. Reinforcement learning can be used to adapt a control system to new and uncertain system dynamics following a fault [113]. It has also been used to synthesize error recoveries in computers [114]. Ap-

plications to spacecraft FDIR include selecting sources of assistance in human-robot interactions [115, 116].

Automated Reasoning

Logical reasoning techniques used in fault diagnosis can also be adapted to selecting recovery actions [105, 117]. The same conflict-directed search used to isolate faults can be applied to find the least-cost combination of recovery actions to achieve a set of objectives. A similar approach is symbolic model checking [118, 119]. Symbolic model checking functions by representing states symbolically, thus avoiding the exploding number of states often encountered when explicitly modeling complex systems. The method then searches for logical formulas that produce the desired goal state. Both conflict-directed search and symbolic model checking are model-based.

When uncertainty is involved, actions can be selected based on their expected utility. Utilities define the value of different outcomes. By multiplying each utility by the probability of the corresponding outcome, the decision maker can select an action that will provide the highest expected utility. This approach is an elementary application of the field of decision theory. Decision nodes can be embedded directly into Bayesian networks to create influence diagrams [72, Chapter 9]. Decision theoretic approaches have been applied to risk management during spacecraft development [120, 121], and to evaluating recovery options based on uncertain root cause diagnosis with dynamic Bayesian networks [89].

1.3.4 Limitations of the Approaches

One reason that model-based approaches to all stages of FDIR are prevalent in aerospace engineering is because the models themselves are a ubiquitous tool for analysis. FDIR aside, computer models are critical to the design and validation of space systems because realistic conditions often cannot be simulated otherwise. Spacecraft models thus represent a rich source of knowledge about system behavior that can be

leveraged for FDIR. Unfortunately, simulating a system model alongside the actual spacecraft for the purpose of analytical redundancy is often too computationally intensive to perform on board [85]. Even when approximate techniques such as neural networks are used in place of a detailed model, interpreting root causes from combinations of identified anomalies is often still left to the operator [60].

Various formal methods for diagnosing root causes from observed symptoms were therefore reviewed in the previous subsections. In many cases, the techniques require expert knowledge of both the technique and the system in order to implement. This is likely out of reach for many designers and operators of small spacecraft. Indeed, CubeSat teams are often characterized by small teams with limited expertise to spare for extensive design of FDIR [30, 32]. Even for professionally managed spacecraft, the increasing complexity of onboard software is a growing concern [34]. Most of the techniques reviewed in Section 1.3.2 will only contribute to this trend.

Moreover, out of all the fault diagnosis methods reviewed in Section 1.3.2, only dynamic Bayesian networks explicitly incorporate the time-dependent context of anomaly information. Yet, the limitations and complexity of assembling dynamic Bayesian networks is discussed in recent literature [122]. In real situations, data is often missing or contradictory. Multiple faults may be present and faults often affect multiple components. Anomalies can be false positives or related to each other with arbitrarily long time dependencies. Past data could be crucial to diagnosis or completely irrelevant. These characteristics are particularly applicable to spacecraft attitude determination and control subsystems (ADCS), which interact directly and extensively with the time-varying space environment. Fusing anomaly information to arrive at a system-level root cause diagnosis requires being sensitive to temporal context, but the details are difficult to quantify. When should information be forgotten? How persistent or frequent must an anomaly be before it can no longer be considered a false positive? Under what circumstances does one piece of information negate the relevance of others?

In the past, the answers to these questions belonged solely to experienced spacecraft operators. This thesis will show how they can be addressed using deep learning.

1.4 Contributions of This Dissertation

Deep learning is a powerful trend in machine learning whereby data is processed into hierarchical levels of abstraction by the successive layers of a neural network [38]. Different types of neural networks can be stacked together to provide a modular architecture. The depth of these networks allows them to achieve nearly human-level performance on tasks such as image recognition and language processing that require a highly contextual understanding of the data. An important deep learning model is long short-term memory (LSTM), which is capable of selectively remembering and forgetting information from time series to aid in time series classification or prediction [123]. LSTM networks provide the sort of contextual processing needed to convert time-dependent anomaly information into a high-level root cause diagnosis.

Deep learning is limited by a large need for data in order to achieve acceptable performance [38]. This has limited its applicability to spacecraft FDIR because the many samples of different faults needed to train a deep learning classifier are simply not available for most missions. Software and hardware models of the spacecraft can simulate its response to faults, but the results may not be sufficiently realistic to use in training directly [124, 125].

The primary contribution of this dissertation is to enable the application of deep learning to spacecraft FDIR through the use of *transfer learning*. Transfer learning is a technique where learning achieved by a neural network in one domain (e.g. a simulation) is applied to a different but related domain (e.g. real life) [126, 127]. This is accomplished for FDIR by using OCSVMs to convert raw telemetry into an abstract representation that is effectively identical for both a spacecraft simulator and the flight unit. An LSTM network designed to diagnose faults from this common representation can therefore train on data from the simulator and transfer its learning to operate on

data from the real satellite. This solves the problem of lacking representative data from faults on board the flight unit, since an arbitrary number of faulty examples can be generated from the spacecraft simulator. Figure 1.7 presents a schematic of the process.

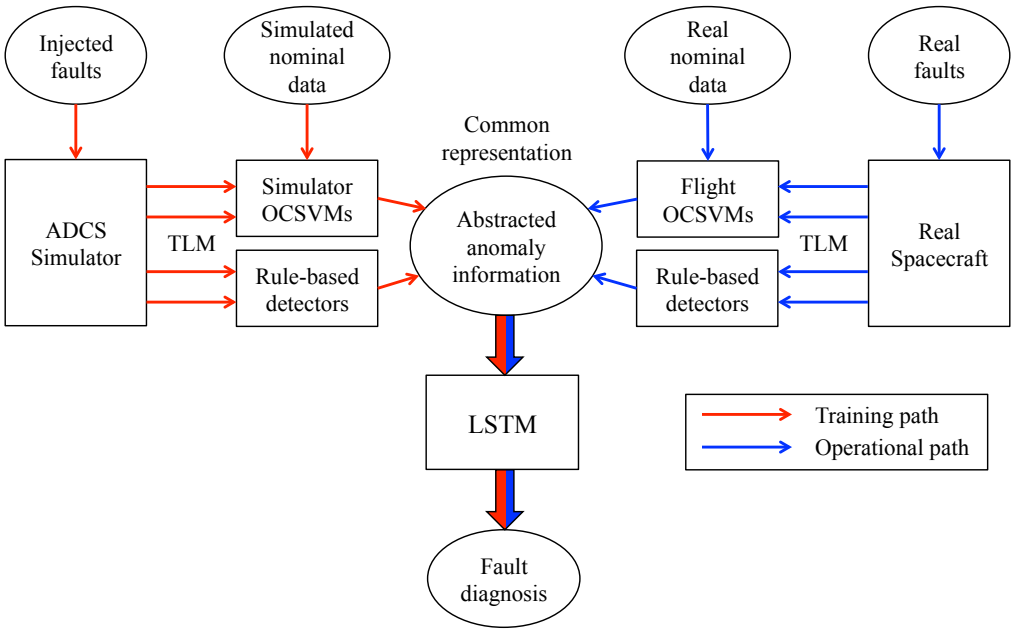


Figure 1.7: Transfer learning architecture for deep learning of spacecraft FDIR.

This approach to FDIR is termed: Sequence-based Mapping of Anomalies to Root causes from Telemetry (SMART-FDIR). The architecture is applied to detect, isolate, and recover from faults in small satellite ADCS. ADCS is chosen due to its complexity and the historically high number of anomalies and failures associated with this subsystem. Specific contributions are outlined below.

1.4.1 Development of an ADCS Fault Simulator

A fault simulator is developed based a high-fidelity model of the complete ADCS for LightSail 2. The simulator is modified to represent either LightSail 2 or a generic

3U CubeSat. A detailed ADCS fault tree is developed and the fault simulator includes the capability to inject up to 35 ADCS-related faults.

1.4.2 Anomaly Detection of ADCS Signals

This dissertation identifies key ADCS signals for informative fault detection. Anomaly detectors based on OCSVMs are designed, including several based on composite signals that provide novel ways of sensing anomalies. Anomaly scores from the various detectors provide a common representation for training and operating an LSTM diagnosis network on either the fault simulator or a real spacecraft. An important advantage is that the anomaly signals can still be interpreted by a human operator as a means of validating the LSTM's diagnosis.

1.4.3 Deep Learning Fault Diagnosis

An LSTM network is designed to isolate ADCS faults based on time series inputs from anomaly detectors. The LSTM network leverages time-dependent context in the inputs and can isolate multiple system-level faults occurring simultaneously. In addition to OCSVMs, the network also accepts input from several rule-based anomaly detectors. Decision theory is applied to suggest recovery actions in the event of a fault and the full FDIR cycle is demonstrated on the fault simulator.

1.4.4 Application to LightSail 2

Transfer learning allows an LSTM diagnosis network to train on simulations from a fault simulator yet also operate on an actual spacecraft. This approach is applied to detect and isolate ADCS faults on the LightSail 2 spacecraft. Both known and previously unknown faults are correctly identified. The results demonstrate the readiness of the technology for further spacecraft and other autonomous vehicle applications.

Dissertation Outline

Chapter 1 has provided the motivation and literature review for current methods of spacecraft FDIR. Chapter 2 details the development of an ADCS fault simulator and reviews the theory of OCSVMs and LSTM networks. Chapter 3 applies these methods along with decision theory to perform the full FDIR cycle in the fault simulator. Chapter 4 extends the approach to detect and isolate anomalies on LightSail 2. Finally, Chapter 5 summarizes significant results and suggests directions for future work.

2. THEORY AND BACKGROUND

2.1 One-Class Support Vector Machines

The FDIR cycle begins with fault detection. OCSVMs are a formal method for detecting when new telemetry is anomalous with respect to past observations. Aside from the identification of past nominal data, no expert knowledge about the signal is assumed. The following subsections review the mathematical basis of OCSVMs and extend the method to yield additional insight into the causes of anomalies.

2.1.1 Derivation

The canonical support vector machine is a supervised approach to binary classification [66]. The basic notion is to find a hyperplane that separates all data points of one class from those of the other class. OCSVMs are a variation of the standard support vector machine method that address the problem of outlier detection. Given a training set of vectors $x_1, x_2, ..., x_\ell \in \mathbb{R}^N$ from a single class (e.g. nominal data), the goal is to find a function $f(x)$ that is positive in a closed region capturing most of the training data and negative elsewhere, with the boundary $f(x) = 0$ discriminating between nominal data and anomalous data. What follows is a procedure for finding $f(x)$ as first laid out by Schölkopf [65].

The first step is to separate the data from the origin in a higher-dimensional vector space using a kernel transformation:

$$k(x_i, x_j) = \phi^\intercal(x_i)\phi(x_j) \tag{2.1}$$

The kernel function is positive-definite and defines a notion of similarity between pairs of vectors. As $k(x_i, x_j) \to 0$, x_i and x_j are increasingly dissimilar. Additional discussion about the mapping $\phi(x)$ is provided in Appendix A.

The anomaly decision function is modeled as a hyperplane with normal vector w and offset ρ which divides the transformed data from the origin as illustrated by Fig. 2.1. This approach can be thought of a binary classifier with the origin representing all members of the anomaly class, since it corresponds to $k(x_i, x) = 0$ (i.e. x is infinitely dissimilar from the training data). The hyperplane decision function is written:

$$f(x) = w^\intercal \phi(x) - \rho \tag{2.2}$$

The hyperplane is found by solving the following quadratic program:

$$\min_{w,\xi,\rho} \; V(w, \xi, \rho) = \frac{1}{2} w^\intercal w + \frac{1}{\nu \ell} \sum_{i=1}^{\ell} \xi_i - \rho \tag{2.3a}$$

Subject to:

$$w^\intercal \phi(x_i) \geq \rho - \xi_i \quad \forall i \tag{2.3b}$$

Where ξ_i are slack variables that have been introduced to allow some error in the solution. This error serves to make the position of the hyperplane less sensitive to outliers in the training data. The parameter $\nu \in (0, 1)$ controls this sensitivity by setting an upper bound for the fraction of outliers. A larger ν corresponds to a tighter decision boundary (i.e. a smaller region of $f(x) > 0$) as more points are permitted to lie to the left of the hyperplane in Fig. 2.1.

Using the Wolfe duality [128], we add Lagrangian multipliers $\alpha_i > 0$ and represent Eq. 2.3 in its dual form:

$$\arg\min_{\alpha,w,\xi,\rho} \; - V(w, \xi, \rho) - \sum_{i=1}^{\ell} \alpha_i \{-w^\intercal \phi(x_i) + \xi_i - \rho\} \tag{2.4a}$$

Subject to:

$$\nabla V(w, \xi, \rho) + \sum_{i=1}^{\ell} \alpha_i \nabla \{-w^\intercal \phi(x_i) + \xi_i - \rho\} = 0 \tag{2.4b}$$

Eq. 2.4b yields:

$$[w^\intercal, (\nu\ell)^{-1}, ..., (\nu\ell)^{-1}, -1] = \sum_{i=1}^{\ell} \alpha_i [\phi^\intercal(x_i), 1, ..., 1, -1] \tag{2.5}$$

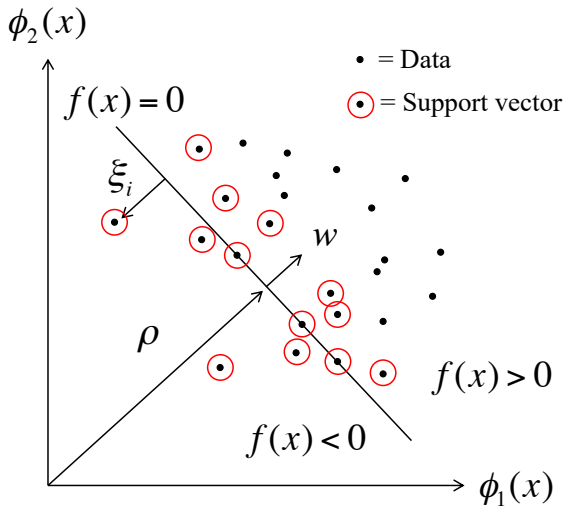


Figure 2.1: Conceptual illustration of a OCSVM hyperplane decision function.

From Eq. 2.5 it is clear that the optimal w solves:

$$w^\intercal = \sum_{i=1}^{\ell} \alpha_i \phi^\intercal(x_i) \tag{2.6}$$

Substituted into Eq. 2.2, this provides the hyperplane decision function:

$$f(x) = \left\{ \sum_{i=1}^{\ell} \alpha_i k(x_i, x) \right\} - \rho \tag{2.7}$$

Eq. 2.5 also provides further constraints on the Lagrange multipliers, revealing:

$$\sum_i \alpha_i = 1 \tag{2.8a}$$

$$\alpha_i \leq \frac{1}{\nu\ell} \tag{2.8b}$$

Since $f(x)$ contains the α_i, it remains to solve for the Lagrange multipliers. Fixing optimal w, ξ_i, and ρ, Eq. 2.4a reduces to:

$$\arg\min_{\alpha} \sum_{i=1}^{\ell} \alpha_i w^\intercal \phi(x_i) \tag{2.9}$$

Which, upon substitution of $w^\intercal$ and multiplication of a constant, is equivalent to:

$$\arg\min_{\alpha} \frac{1}{2} \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} \alpha_i \alpha_j k(x_i, x_j) \tag{2.10a}$$

We also collect the constraints on α_i:

$$0 \leq \alpha_i \leq \frac{1}{\nu\ell}, \quad \sum_i \alpha_i = 1, \tag{2.10b}$$

Training data with non-zero α_i are called the support vectors. These vectors lie exactly on the hyperplane if $0 < \alpha_i < (\nu\ell)^{-1}$. The offset ρ can be recovered from any one of these by noting that any x_i on the decision boundary $f(x) = 0$ solves:

$$\rho = \sum_j \alpha_j k(x_j, x_i) \tag{2.11}$$

Eqs. 2.10 and 2.11 provide everything that is needed to solve for the decision function $f(x)$. Figure 2.2 provides an example for two different values of ν and the Gaussian kernel:

$$k(x, y) = e^{-||x-y||^2} \tag{2.12}$$

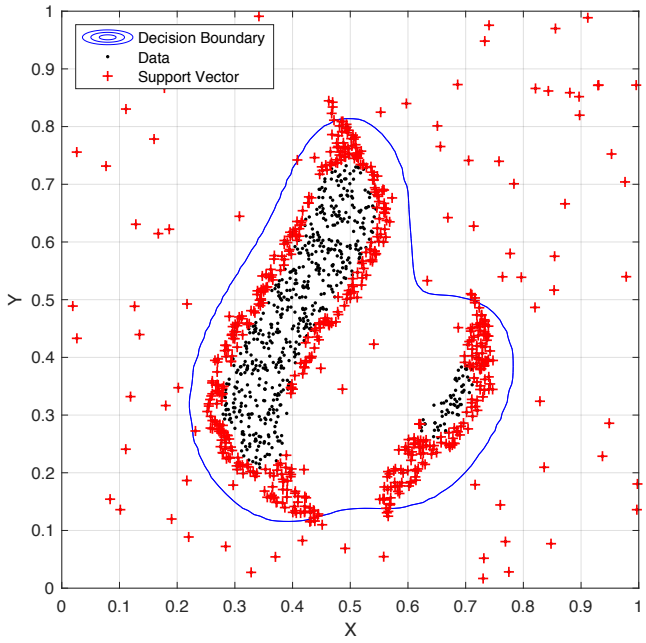
Eq. 2.12 defines $\phi(x)$ implicitly and is a popular choice of kernel which is used throughout the rest of this dissertation. Since it is the unit Gaussian, it is important to normalize the training vectors x_i and any later inputs to be approximately in the range $(-1, 1)$. It is also useful to apply the logit transform to the decision output:

$$F(x) = \frac{\exp\{-f(x)\}}{1 + \exp\{-f(x)\}} \tag{2.13}$$

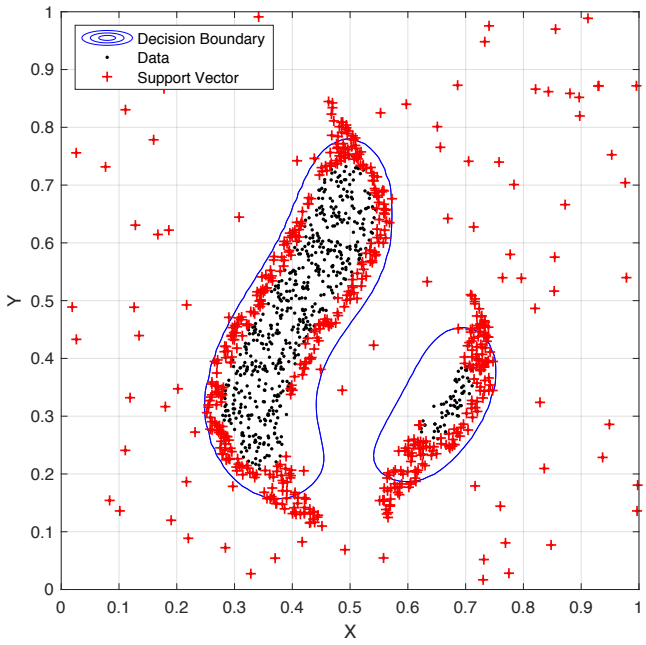
Under this transform, $F(x)$ represents a confidence score that a new observation x is an anomaly. Data for which $F(x) \approx 0$ are confidently nominal, while $F(x) > 0.5$ are anomalies. Such anomaly scores for different ADCS signals will become the inputs to an LSTM network for fault isolation.

2.1.2 Interpreting OCSVM Anomalies

The anomaly decision function, Eq. 2.13, provides little in the way of explanatory power aside from simply identifying anomalies. Particularly in space flight applications, it is also important to understand *why* an instance of data is deemed anomalous



(a) $\nu = 0.075$



(b) $\nu = 0.15$

Figure 2.2: Example decision boundaries determined using a OCSVM with different outlier fractions. An important property of the method is the ability to reject outliers within the training data when forming the decision boundary.

by the algorithm. We can gain additional insight into the OCSVM's anomaly decisions in this regard by calculating the gradient of the anomaly score for a vector $x \in \mathbb{R}^N$:

$$\nabla F(x) = \frac{\partial F}{\partial x} = \frac{\partial F}{\partial f} \circ \frac{\partial f}{\partial x} \tag{2.14}$$

where $\circ$ denotes the Hadamard (element-wise) product. Let us concern ourselves only with the direction of the anomaly gradient. We can then restrict our analysis to the second term in Eq. 2.14 by noting that directions of decreasing f correspond to directions of increasing anomaly scores. Recalling Eq. 2.7:

$$\frac{\partial f}{\partial x} = \sum_{i=1}^{\ell} \alpha_i \frac{\partial k(x_i, x)}{\partial x} \tag{2.15}$$

Assuming the Guassian kernel function (Eq. 2.12):

$$\frac{\partial k(x_i, x)}{\partial x} = -2(x - x_i) \exp(-||x - x_i||^2) \tag{2.16}$$

$$\Rightarrow \nabla f(x) = \frac{\partial f}{\partial x} = -2 \sum_{i=1}^{\ell} \alpha_i (x - x_i) k(x_i, x) \tag{2.17}$$

which is an $N \times 1$ vector of derivatives.

Normalizing and taking the negative of Eq. 2.17 gives the direction in which the anomaly score of x is increasing fastest. We will refer to this as the "anomaly gradient," and it can be interpreted as showing the relative contribution of the different components of x to the anomaly score. If the gradient lies heavily along one axis of x, it is an indication that the anomaly is associated with that component. Figure 2.3(a) illustrates the anomaly gradient for the decision boundary in Fig. 2.2(a).

Far field Approximation

Unfortunately, Eq. 2.17 does not provide a foolproof way of understanding where an anomalous vector x lies relative to the set of nominal data. Since $f(x)$ does not necessarily decrease monotonically away from the decision boundary, the anomaly gradient can become reversed relative to the boundary as shown in the top right of

Fig. 2.3(a). A further complication is that, in practice, anomalies are often so far removed from the set of nominal data that $k(x_i, x) \ll 1$, thus causing a numerical underflow that results in $\nabla f(x) = \vec{0}$.

In order to compute a sensible anomaly gradient in this regime, we introduce the following "far field" approximation:

$$||x - x_i||^2 \approx ||x - \mu||^2 \tag{2.18}$$

$$\Rightarrow k(x_i, x) \approx k(\mu, x) \tag{2.19}$$

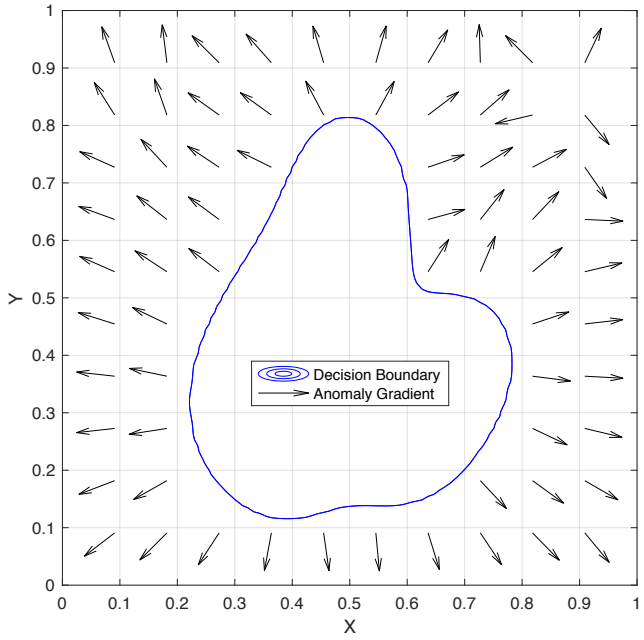
where μ is the centroid of the nominal data, x_i. Invoking this approximation allows us to factor $k(x_i, x)$ out of the sum in Eq. 2.17 and then remove it through subsequent normalization. This gives the approximate "far field" anomaly gradient:

$$-\frac{\nabla f}{||\nabla f||} \approx \frac{\sum_{i=1}^{\ell} \alpha_i (x - x_i)}{||\sum_{i=1}^{\ell} \alpha_i (x - x_i)||} \tag{2.20}$$

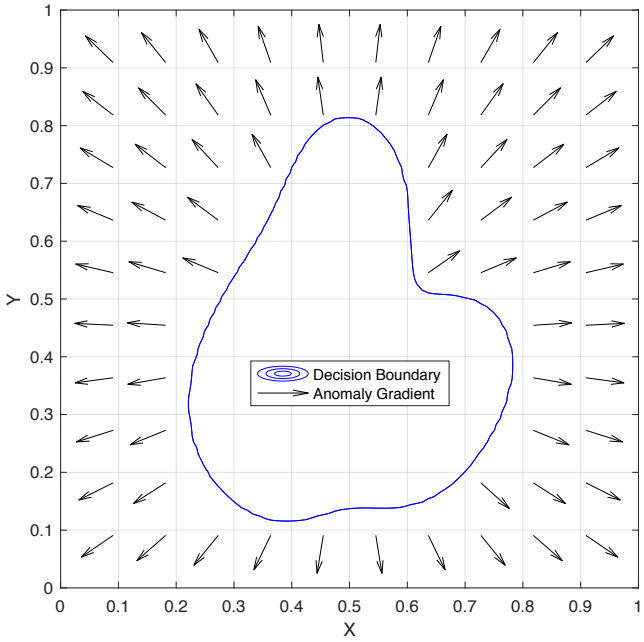
The approximation Eq. 2.18 is justified when the distance of x to each x_i is much larger than the distances between the x_i themselves. In other words, when x is far from any cluster of nominal data. However, it is useful even near the decision boundary, where it can mitigate troublesome gradient reversals relative to the decision boundary. Figure 2.3(b) compares the anomaly gradient determined with far field approximation with Fig. 2.3(a). The far field anomaly gradient will be useful for isolating anomalies to specific sensors and actuators in Section 3.2.2.

2.1.3 Advantages of OCSVMs

Numerous other anomaly detection techniques exist as discussed in Section 1.3.1, but OCSVMs possess a number of advantages that make them well suited to fault detection in ADCS signals. For one, it is not necessary to specify the number of clusters in the training dataset as is the case for some cluster-based methods. This is exemplified in Fig. 2.2(b) where the decision boundary bifurcates to independently surround the separate clusters. The bifurcation is controlled by the parameter ν,



(a) Accurate anomaly score gradients



(b) Approximate far field gradients

Figure 2.3: OCSVM anomaly score gradients can provide insight into the anomaly decisions of the algorithm by revealing the relative contribution of each component of a vector to the anomaly score. The far field approximation simplifies the calculation and ensures the gradient can still be computed when anomalies are large.

which is intuitive to tune. It controls how tightly the boundary should enclose the nominal data and sets an upper-bound for the false positive rate of the anomaly detector.

OCSVMs are also more computationally efficient to evaluate than methods such as nearest neighbours. Since $\alpha_i \neq 0$ only for the support vectors, the kernel function in Eq. 2.7 need only be evaluated for the support vectors. This efficiency is an important consideration for anomaly detection performed on board resource-constrained flight hardware.

Lastly, OCSVMs operate over any number of dimensions and are straightforward to apply to anomaly detection in a diverse selection of ADCS signals (see Sections 3.1 and 4.2). Using a single method for numerous anomaly detectors simplifies the overall FDIR design. This is a priority in FDIR research.

2.2 Long Short-Term Memory

Artificial neural networks are abstract models of animal nervous systems that can approximate any function from input-output pairs from the function. They are typically used in applications where the function of interest is not explicitly known or is otherwise difficult to evaluate directly [129]. For FDIR, neural networks will be used to model the complex relationships between anomalies and underlying faults in spacecraft ADCS. The network will learn these relationships implicitly from examples generated by a fault simulator rather than by explicit programming. The particular type of network to be used is called long short-term memory. This section describes its architecture and implementation for fault isolation.

2.2.1 Architecture

The most elementary artificial neural network is a multi-layer perceptron (MLP), commonly known as a feed-forward neural network. MLPs arrange neurons such that

the output of each layer is a linear combination of the inputs passed through some non-linear activation function φ:

$$y = \varphi(Wx + b) \tag{2.21}$$

Where $x_{n\times 1}$ is a vector of inputs, $y_{m\times 1}$ is a vector of outputs, $W_{m\times n}$ contains the weights of the neuron connections, and $b_{m\times 1}$ are bias terms. This is sufficient to model a static input-output relationship.

For cases where y depends dynamically on x, recurrent neural networks (RNN) introduce a feedback loop to one or more layers. This gives them an evolving internal state h_t that can influence the output from one time step to the next. Figure 2.4(a) compares the architecture with an MLP. The update and output of an RNN is described by:

$$h_t = \varphi_1(Wx_t + Uh_{t-1} + b_1) \tag{2.22a}$$

$$y_t = \varphi_2(Vh_t + b_2) \tag{2.22b}$$

Unfortunately, both RNNs and MLPs with many layers can be difficult to train due to the vanishing/exploding gradients problem [130]. LSTM overcomes this problem by introducing memory cells to the standard RNN [123]. The memory cells can selectively save values, forget them, and influence the network's hidden state and hence, the network's output. Figure 2.4(b) provides a schematic of the basic LSTM memory cell. The following equations control the different gates inside the cell:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{2.23a}$$

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{2.23b}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{2.23c}$$

Note that σ represents the sigmoid activation function:

$$\sigma(z) = \frac{e^z}{1 + e^z} \tag{2.24}$$

The cell state and hidden state are modified by the activations of the gates according to the following equations:

$$c_t = f_t \circ c_{t-1} + i_t \circ \tanh\left(W_c x_t + U_c h_{t-1} + b_c\right) \tag{2.25a}$$

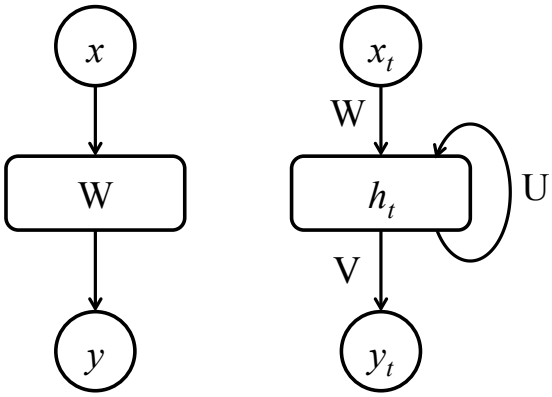
$$h_t = o_t \circ \tanh\left(c_t\right) \tag{2.25b}$$

In summary, the inputs and hidden state of the LSTM network actuate gates that can selectively overwrite or read the cell's memory. When the cell memory is read, it modifies the hidden state. A copy of the hidden state is passed to subsequent layers of the neural network where it is interpreted to give the final output of the network at that time step.

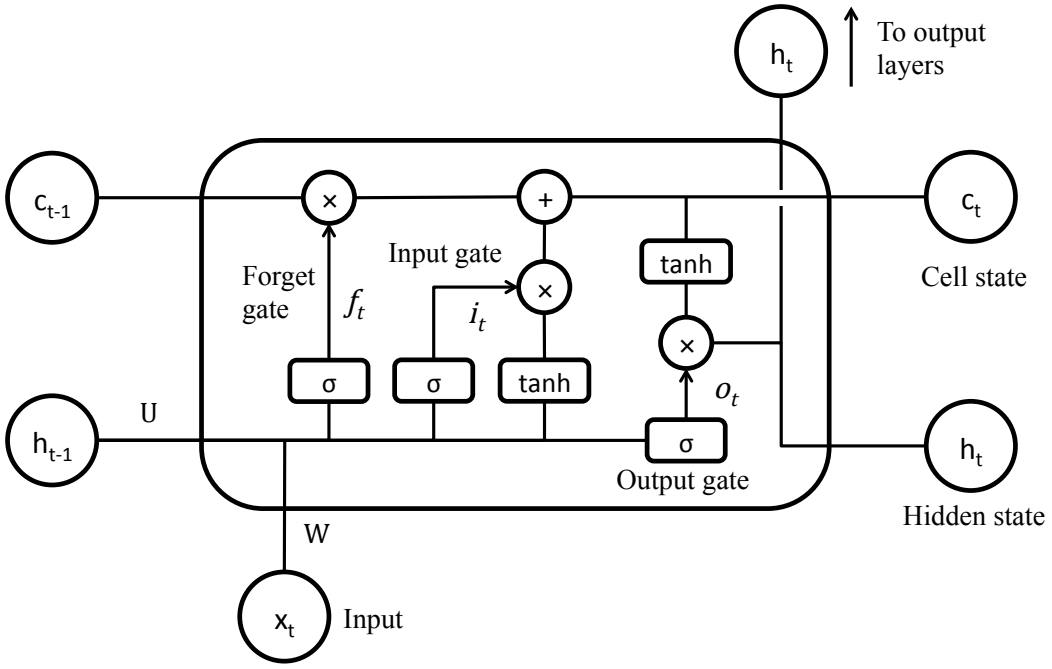
2.2.2 Fault Isolation Using LSTM

During fault isolation, LSTM will be applied to perform sequence-to-sequence regression. Given a stream of anomaly signals as input, the goal is to compute an instantaneous confidence value for each possible underlying fault. This is different from the sequence classification approach. In LSTM sequence classification, the class is typically predicted only once the entire time series of inputs has been processed, but in fault isolation the sequence of inputs streams continuously. The fault case can also change with time as faults appear or disappear. Treating the problem as a regression allows the LSTM network to provide a running prediction of the fault case. Furthermore, by avoiding the softmax activation function used in most classification architectures, the constraint that the confidence scores must sum to unity is relaxed. This allows the network to diagnose multiple faults occurring simultaneously or leave confidence unassigned in a manner similar to the notion of ignorance in Dempster-Shafer evidence theory [78].

To interpret the hidden state h_t of the LSTM memory cells into non-exclusive confidence outputs in the range (0,1), we use a fully connected (a.k.a. "dense") layer computing Eq. 2.22b where $\varphi_2 = \sigma$. One-hot encoding is used for the fault confidence scores, meaning each element of the output vector y_t contains a score representing



(a) Architectures of a MLP and RNN



(b) LSTM memory cell

Figure 2.4: Comparison of neural network architectures. In addition to the hidden state feedback of a traditional RNN, LSTM includes memory cells that store selected information.

the network's confidence that the corresponding fault scenario is occurring at that moment. The target outputs during training are binary flags indicating whether each fault is actually present in the system. Figure 2.5 depicts the entire architecture.

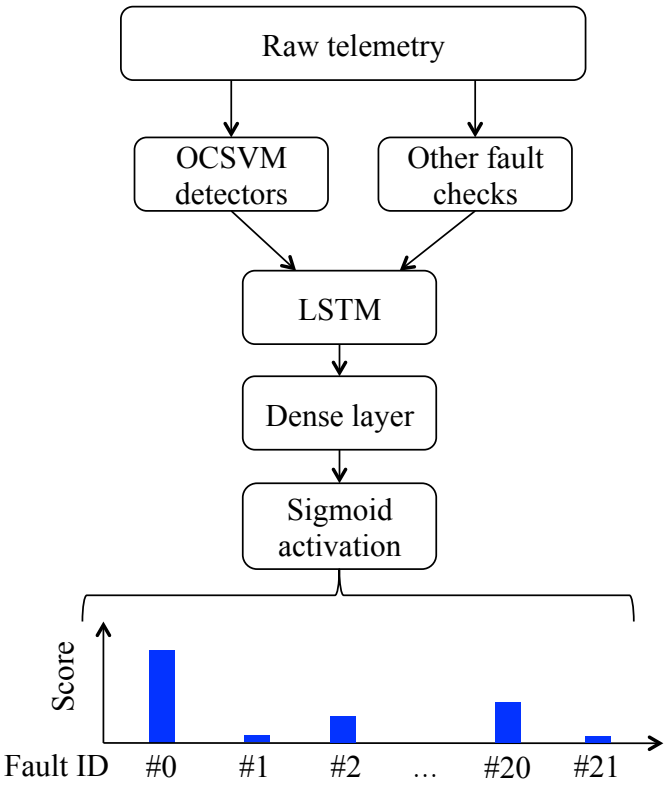


Figure 2.5: Hierarchy of fault isolation layers.

2.2.3 Network Training

Training a neural network involves finding the optimal set of weights and biases to minimize some loss function $\mathcal{L}$ assessing the accuracy of the network's predictions. The loss function is usually calculated in reference to target outputs y^T that represent the true output to input x based on input-output training pairs. For sequence-to-

sequence regression, the loss function is normally half of the mean squared error (MSE) of the outputs at each time step:

$$\mathcal{L} = \frac{1}{2s} \sum_{t=1}^{s} \sum_{j=1}^{m} (y_{jt} - y_{jt}^{T})^2 \tag{2.26}$$

Where $t = 1, ..., s$ is the sequence of $n \times 1$ inputs and $j = 1, ..., m$ are the possible fault scenarios. This is the loss function used to train the LSTM network's fault confidence scores. Thus, $\mathcal{L}$ is minimized when the LSTM's confidence for each fault at each time matches the actual fault flag as close as possible.

The weights and biases of the neural network are initially randomized. Each iteration of the training cycle then proceeds in three steps:

1. Forward pass: Input vectors $x_1, ...x_s$ for each training sequence are provided to the LSTM network and its outputs $y_1, ..., y_s$ are computed using Eqs. 2.23, 2.25, and 2.22b.
2. Loss computation: $\mathcal{L}$ and $\dfrac{\partial \mathcal{L}}{\partial y_{jt}}$ are computed using Eq. 2.26.
3. Backward pass: $\dfrac{\partial \mathcal{L}}{\partial \{W, U, V, b\}}$ are found from $\dfrac{\partial \mathcal{L}}{\partial y_{jt}}$ and W, U, V, and b are updated via gradient descent.

The process of deriving the gradient descent parameter updates from the loss during the backward pass is provided in Appendix B. Numerous 'tricks' can enhance the convergence of W, U, V, and b. The gradients in step 3 are often multiplied by a constant called the learning rate (typically $\ll 1$) to encourage smaller updates. This can prevent gradient descent from "stepping over" good solutions or diverging altogether. In a similar vein, gradient clipping re-normalizes gradients if they exceed a threshold value. Techniques such as weight regularization and drop-out, which randomly removes neurons during training, can also reduce overfitting by encouraging the network to learn multiple representations of an input-output relationship [131]. Training multiple networks from randomly initialized parameters can also allow multiple local optima for W, U, V, and b to be found, and the best among them selected.

2.2.4 Understanding LSTM's Decisions

Once the LSTM network is trained, insight can be gained into its fault isolation decisions by computing the gradient of the network's output with respect to its inputs. This reveals exactly which inputs are having the most influence on the network's confidence scores and provides a way of validating that the network has learned sensible relationships between anomaly symptoms and faults. Another advantage is efficient fault reporting, since the gradient can reveal the subset of telemetry signals relevant to a fault as well as localize the time when the fault first appeared.

The following equations derive the gradient of the LSTM confidence scores with respect to the inputs at time t. This is accomplished via the chain rule of Calculus. The current and previous states of the network and all intermediate variables (i.e. the gates) are assumed to be known. Letting $a_t = \tanh\left(W_c x_t + U_c h_{t-1} + b_c\right)$ in Eq. 2.25a, Fig. 2.6 traces all of the paths through the LSTM memory cell by which x_t can influence y_t.

Summing all parallel paths from y_t to x_t, we can derive the following equation for the gradient of interest:

$$\frac{\partial y_t}{\partial x_t} = \frac{\partial y_t}{\partial h_t}\left[\frac{\partial h_t}{\partial o_t} \circ \frac{\partial o_t}{\partial x_t} + \frac{\partial h_t}{\partial c_t} \circ \left(\frac{\partial c_t}{\partial f_t} \circ \frac{\partial f_t}{\partial x_t} + \frac{\partial c_t}{\partial i_t} \circ \frac{\partial i_t}{\partial x_t} + \frac{\partial c_t}{\partial a_t} \circ \frac{\partial a_t}{\partial x_t}\right)\right] \tag{2.27}$$

The left-most term is determined from Eq. 2.22b and by noting that the derivative of the sigmoid function is $\sigma' = (1-\sigma)\sigma$:

$$\frac{\partial y_t}{\partial h_t} = \frac{\partial y_t}{\partial \sigma}\frac{\partial \sigma}{\partial h_t} = (1 - y_t) \circ y_t \circ V \tag{2.28}$$

Moving from left to right through Fig. 2.6, the derivatives branching from h_t follow from Eq. 2.25b:

$$\frac{\partial h_t}{\partial o_t} = \tanh\left(c_t\right) \tag{2.29}$$

$$\frac{\partial h_t}{\partial c_t} = o_t \circ \{1 - \tanh^2(c_t)\} \tag{2.30}$$

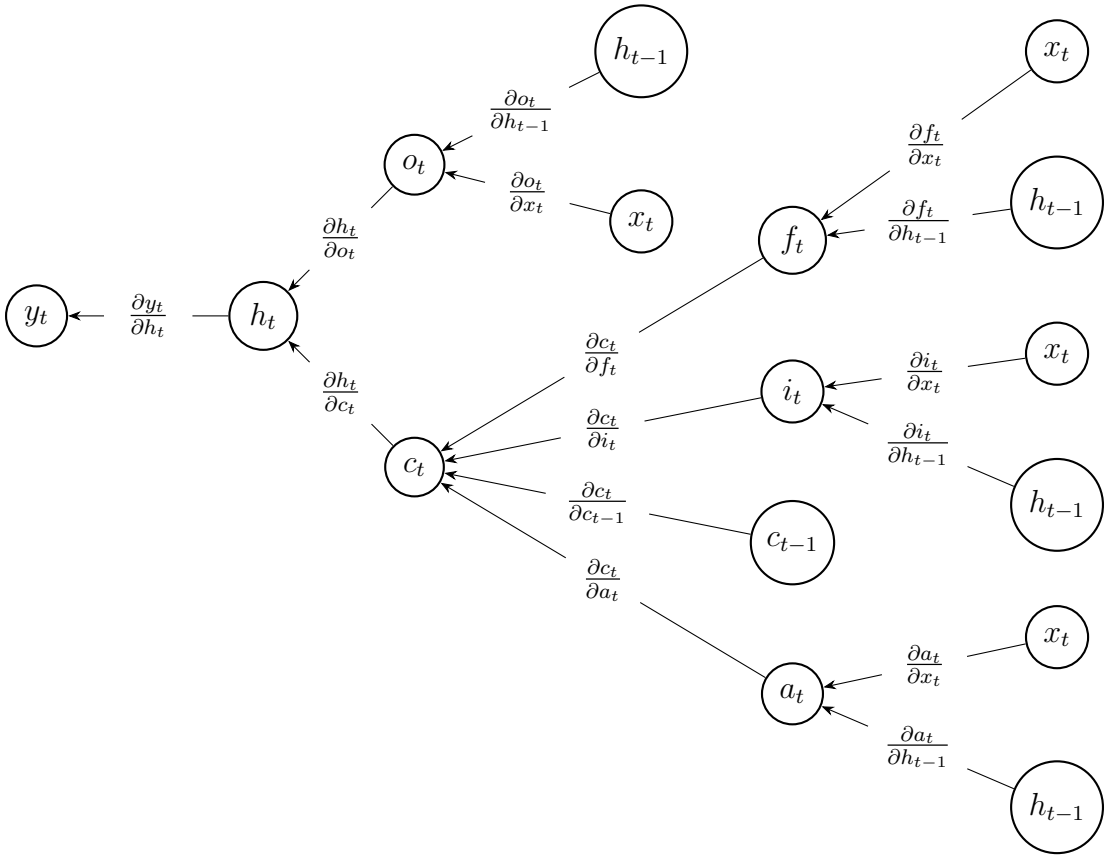


Figure 2.6: Propagation path for differentiating the output of the LSTM network with respect to the current input.

The derivatives of each of the gates are determined similar to Eq. 2.28:

$$\frac{\partial o_t}{\partial x_t} = o_t \circ (1 - o_t) \circ W_o \tag{2.31}$$

$$\frac{\partial i_t}{\partial x_t} = i_t \circ (1 - i_t) \circ W_i \tag{2.32}$$

$$\frac{\partial f_t}{\partial x_t} = f_t \circ (1 - f_t) \circ W_f \tag{2.33}$$

Finally, from the cell state update Eq. 2.25a we have:

$$\frac{\partial c_t}{\partial f_t} = c_{t-1} \tag{2.34}$$

$$\frac{\partial c_t}{\partial a_t} = (1 - a_t^2) \circ W_c \tag{2.35}$$

This covers all of the terms necessary to evaluate Eq. 2.27. It is also possible to determine the gradient with respect to prior time steps $\left\{ \frac{\partial y_t}{\partial x_{t-p}} : p > 0 \right\}$. The usefulness and practicality of this is limited, however. Equation 2.27 already involves computing an $m \times n$ tensor for every time step. To propagate the gradient back p time steps for every one of T steps would involve computing $p \times T$ such tensors − a significant computational and memory expense. As will be evident in Chapters 3 and 4, even just $\frac{\partial y_t}{\partial x_t}$ can provide valuable insight into the LSTM network's fault decisions.

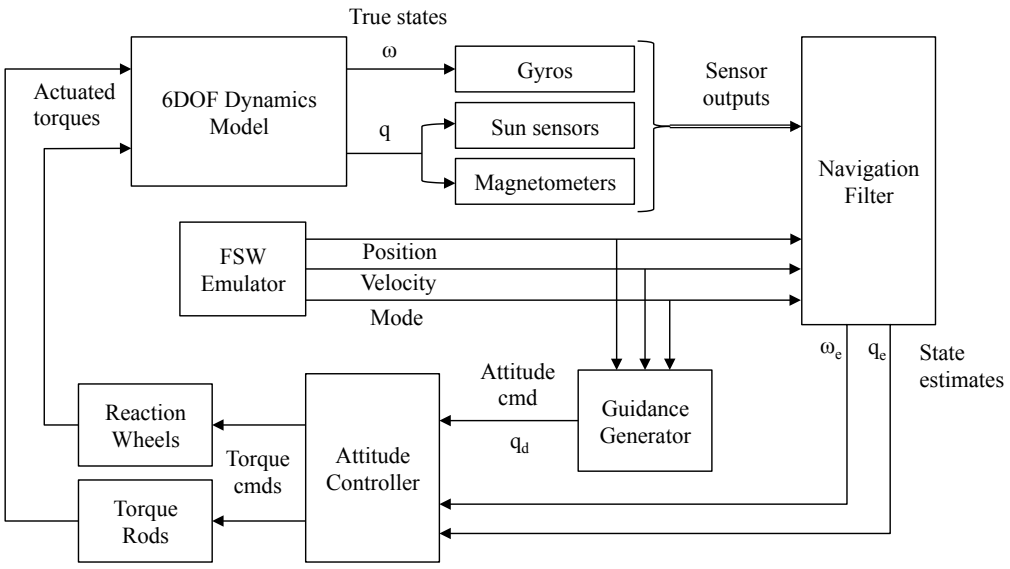
2.3 ADCS Fault Simulator

The purpose of the fault simulator is to provide a reasonably complete and accurate model of how a spacecraft's ADCS operates. By interfering with or spoofing certain signals within the model, the fault simulator can give a picture of how the ADCS will behave when affected by different faults.

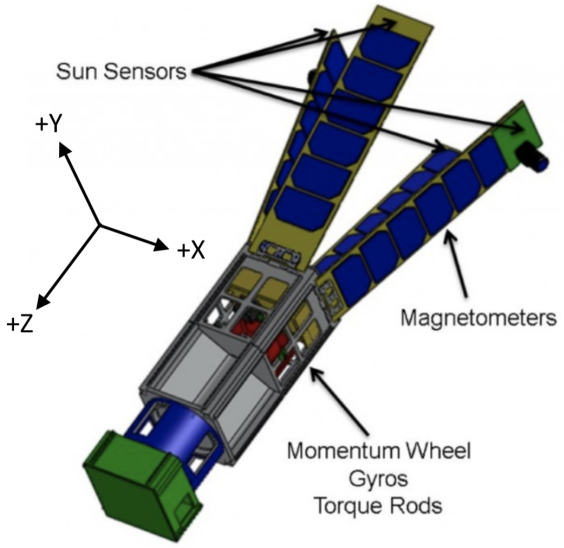
The 6DOF spacecraft ADCS model used in this research is an adaptation of the model inherited from the LightSail 2 solar sail mission [132–135]. The model is implemented in Simulink$^{\text{TM}}$ and includes sensors, actuators, flight software (FSW) attitude determination and control algorithms as well as a high-fidelity rigid body dynamics propagator for the attitude and orbit of the satellite. To make the model more representative of a typical 3U (30 cm × 10 cm × 10 cm) CubeSat, features specific to LightSail 2 such as the solar sail have been disabled and the model spacecraft includes a reaction wheel for each axis. Figure 2.7 provides an overview of the ADCS model alongside the structural configuration of the spacecraft. Attitude quaternions are represented by q and rates are represented by ω. The following subsections outline the key components and capabilities of the simulation.

2.3.1 Attitude Sensors and Determination

A suite of four magnetometers, five sun sensors, and three gyros provide attitude knowledge to the spacecraft. The magnetometers each measure the magnetic field



(a) ADCS block diagram



(b) Satellite configuration

Figure 2.7: Overview of the CubeSat attitude determination and control system simulated in the fault simulator.

along 3 axes from a location halfway along each solar panel. The magnetometer outputs are represented in the simulation by determining the local magnetic field vector from the 2010 International Geomagnetic Reference Field (IGRF) and transforming it into the sensor-fixed frame of each sensor. Gaussian measurement noise, saturation, and digitization are then applied based on the specifications of actual hardware. The Honeywell HMC6343 magnetometer is used for this purpose [136]. The sun sensors represent coarse sun sensors made by ELMOS [137]. Four of the sensor fields of view (FOV) face outwards along each deployed solar panel while the final sensor is located on the –Z panel. Note that the panels are nominally opened 165 degrees from the spacecraft body. Finally, each gyro measures the rotation rate about one of the geometric axes of the spacecraft with measurement noise and digitization representing those of the Analog Devices ADIS16135 model [138].

A voting mechanism helps ensure the integrity of sensor measurements before they are used for attitude determination. Each 3-axis magnetometer measurement is transformed into the spacecraft body frame and used to compute an average magnetic field vector. If any sensor's measurement differs from the average by more than the known 3σ noise of the sensor, the sensor is marked as invalid for the current sample period (1 second). If more than half of the sensors differ in this way, all are rejected. A similar scheme is applied to the sun sensors with the additional consideration that all sun sensors are ignored if the spacecraft is believed to be in eclipse based on an estimate of its current position. In the simulation, sun sensors that do not have the sun in their FOV give random outputs. Since the gyros each measure a different axis, no voting is performed.

Accepted sensor measurements are transformed into the spacecraft body frame using the appropriate rotation matrices and incorporated into an extended Kalman filter (EKF) to estimate the spacecraft attitude quaternion in the Earth Centered Inertial (ECI) frame [139]. The IGRF model is used to provide the reference magnetic field vector in this frame in order to interpret the magnetometer readings. A basic solar position model from the U.S. Naval Observatory Astronomical Almanac fulfils

the same purpose for any sun sensors that are accepted [140]. Querying the IGRF model and checking for eclipse requires the spacecraft to know the universal time and its position in the ECI frame. For small satellites in Earth orbit this is generally provided in the form of an orbit two-line element (TLE) uplinked to the satellite and integrated to the current time of the spacecraft clock by an onboard propagator. For the purpose of the simulation, the spacecraft's simulated position and time are used, but these can be subjected to errors to represent an incorrect time or TLE. With this combination of sensors and algorithms, attitude determination is typically accomplished to within 15 degrees error.

2.3.2 Attitude Actuators

Attitude control is actuated by a reaction wheel and a magnetic torque rod about each axis. The torque limits, saturation RPM, moment of inertia, and torquing errors of the reaction wheels are based on the specifications of the RW-0.060-28 model manufactured by Sinclair Interplanetary [141]. The magnetic torque rods are simulated by including their magnetic dipoles in the calculation of the spacecraft's magnetic torques. The dipoles range between ± 1 Am^2 depending on the command and are derived from specifications for torque rods produced by StrasSpace [142]. The currents needed to produce the commanded dipole are also simulated and measurement noise is added.

2.3.3 Guidance and Control

The 6DOF simulator incorporates the attitude control modes listed in Table 2.1.

In Sun, nadir, or velocity pointing modes, the spacecraft's ECI position and velocity (velocity pointing) or solar position (Sun pointing) are used to derive a desired

Table 2.1: Simulated attitude control modes.

No.	Mode	Description
0	Detumble	Valid magnetometer readings are averaged in the spacecraft body frame and basic B-dot control [143] is used to generate torque rod commands to arrest the spacecraft's rotation. Reaction wheels are not used in this mode.
1	Magnetic alignment	Similar to detumble except that the Z-axis torque rod is set to constant maximum power. This aligns the spacecraft Z-axis (with some precession) to the local magnetic field vector.
2	Sun pointing	All actuators are used to point the spacecraft's –Z panel towards the Sun.
3	No torques	All actuators are disabled.
4	Nadir pointing	All actuators are used to maintain nadir pointing of the Z-axis.
5	Velocity pointing	All actuators are used to align the spacecraft Z-axis with the velocity vector.

attitude quaternion q_d in the ECI frame. This is combined with the spacecraft's estimated quaternion q_e from attitude determination to derive an error quaternion:

$$q_\varepsilon = q_d^{-1} q_e \tag{2.36}$$

The error quaternion along with the measured angular rates from the gyros are used in a state feedback controller to compute control torques for each axis. These torques are sent simultaneously to both the torque rods and reaction wheels. The resulting pointing error with respect to the commanded quaternion is generally less than 10 degrees.

2.3.4 Dynamics

Propagation of the spacecraft's rotational and orbit dynamics occurs in Simulink using a fourth order Runge-Kutta integrator. The following disturbances are modeled:

- J2-J6 gravity harmonics
- Aerodynamic drag based on the NRLMSISE-00 atmospheric model
- Solar radiation pressure
- Third body lunar gravity
- Gravity gradient
- Magnetic torque for the residual magnetic fields of the reaction wheels

The true state of the spacecraft serves as input to the various sensor models to simulate sensor outputs. The mass of the spacecraft is 5 kilograms.

2.3.5 Fault Injection

A key feature of the 6DOF simulator is the ability to inject faults in various parts of the model. A collection of ADCS fault trees have been developed and provided in Appendix C. The fault trees were assembled based on personal experience, extensive brainstorming, and examples from literature [2].

3. DESIGN AND DEMONSTRATION OF FDIR

In this chapter, the Smart-FDIR approach will be developed for the ADCS Fault Simulator. The generic 3U CubeSat detailed in Section 2.3 is considered. Section 1 details the application of OCSVMs to monitoring relevant ADCS signals for anomalies alongside a handful of simple rule-base checks. Section 2 describes the process of training the LSTM fault isolator and provides assessments and demonstrations of its ability to isolate faults from anomalies. Section 3 applies decision theory to convert uncertain fault diagnoses into an appropriate response action given the risk posture of the mission.

3.1 Anomaly Detector Development

3.1.1 Relevant Signals and Signal Processing

Table 3.1 catalogs numerous signals relevant to monitoring for anomalies in a CubeSat ADCS. Each signal is sampled at a rate of 1 Hz, leading to a continuously streaming timeseries of data. These signals represent the "raw" telemetry in Fig. 1.7 and provide the basis for further processing into intermediate signals and anomaly scores.

Table 3.1.: Relevant ADCS signals for fault detection.

ADCS Signal	Description & Purpose
Gyro angular rates	Spacecraft rotation rates measured by gyros. Useful for identifying excessive rotation rates or anomalous gyro readings.

Quaternion estimate	Attitude quaternion estimate from the Kalman filter. Provides the estimated transform between the spacecraft body and inertial frames.
Torque rod current error	The expected current is estimated from the commanded dipole and compared to the measured current. Useful for identifying electrical anomalies in the torque rods.
Reaction wheel torquing error	The commanded torque is compared to the measured torque from the wheel in order to identify performance anomalies.
Reaction wheel RPM	The rotation speed of each reaction wheel can reveal momentum saturation, an unexpected build-up of angular momentum, or a lack of response when commanded.
Attitude pointing error	The angle between the estimated spacecraft attitude and the commanded attitude. Useful for identifying under-performing attitude control.
Sun sensor voting	Logs of which sensors were accepted/rejected during voting. Useful for identifying malfunctioning sensors or incorrect measurement processing.
Magnetometer voting	Logs of which sensors were accepted/rejected during voting. Useful for identifying malfunctioning sensors or incorrect measurement processing.
Long duration count	A timer that increments for every minute of uninterrupted ADCS operation. Resets to zero whenever a reboot occurs, allowing anomalous resets to be identified.
Average B-vector	The average magnitude and direction (in the spacecraft body frame) of the magnetic field across all magnetometers can be used to identify an unexpected magnetic field.

Average Sun vector	The mean direction of the Sun vector (in the spacecraft body frame) across all sun sensors can reveal discrepancies compared to other sources of attitude knowledge.
Solar power	The instantaneous total power from all solar panels. This can reveal discrepancies in orbit knowledge if the power is non-zero during a predicted eclipse.
Position and time	The propagated position of the spacecraft and the UNIX time. Used to query the position of the Sun and the Earth's magnetic field in the inertial frame. Useful for identifying errors in orbit knowledge when compared to other information.

3.1.2 One-class Support Vector Machine Detectors

The following subsections describe the intermediate signals and methods of processing necessary to develop a collection of OCSVM anomaly detectors. To generate nominal data, at least 20 runs of the fault simulator were performed under a "no fault" case and with parameters randomized according to Table 3.2. In every case, the inputs X to a given OCSVM are normalized by the mean μ_X and standard deviation σ_X of the OCSVM's training data by:

$$\bar{X} = \frac{X - \mu_X}{\sigma_X} \tag{3.1}$$

Gyro Variance Anomalies

One of the most straightforward OCSVM detectors monitors the measurement variance of the three gyros. Using a 1-minute sliding variance window, the variance of the last 60 samples of each gyro is extracted and the results are compiled into a 3×1 vector. Since the nominal simulations are comprised of 20 runs of 2 orbits

Table 3.2: Simulation parameters for generating nominal datasets.

Parameter	Values
Date	Randomized between January 1 − December 31, 2020
Orbit	Random 700 km circular orbits at 60 deg inclination
Mode	Cycled through 0 to 5
Number of orbits	2
Initial quaternion	Random (-1,1) for each component, then normalized
Initial angular rate	Random (-1,1) deg/sec for each axis

(approximately 5940 seconds) each, this results in 3960 vectors for training. Figure 3.1 shows the resulting OCSVM decision boundary with an outlier fraction of $\nu = 0.001$. The OCSVM includes 1983 support vectors.

This anomaly detector is useful for sensing changes in the measurement noise of any one of the gyros. It is sensitive to increases in the noise but also decreases that may occur if one of the gyros gives a stuck reading (i.e. the variance drops to zero).

Actuator Torquing Errors

Anomalous differences between the commanded and measured torques from the spacecraft's actuators can reveal problems with the actuators themselves. For magnetic torque rods, the generated magnetic dipole $\vec{\mu}$ is related to the current I flowing through the solenoid, the number of turns n, and the cross-sectional area A of the solenoid:

$$||\vec{\mu}|| \propto nIA \tag{3.2}$$

Since most magnetorquers include a ferrite core in order to boost the dipole for a given amount of current, the exact relation is determined by the magnetic susceptibility χ measured during ground testing of the torquer. A typical value for $(1+\chi)nA$ is 10 m^2

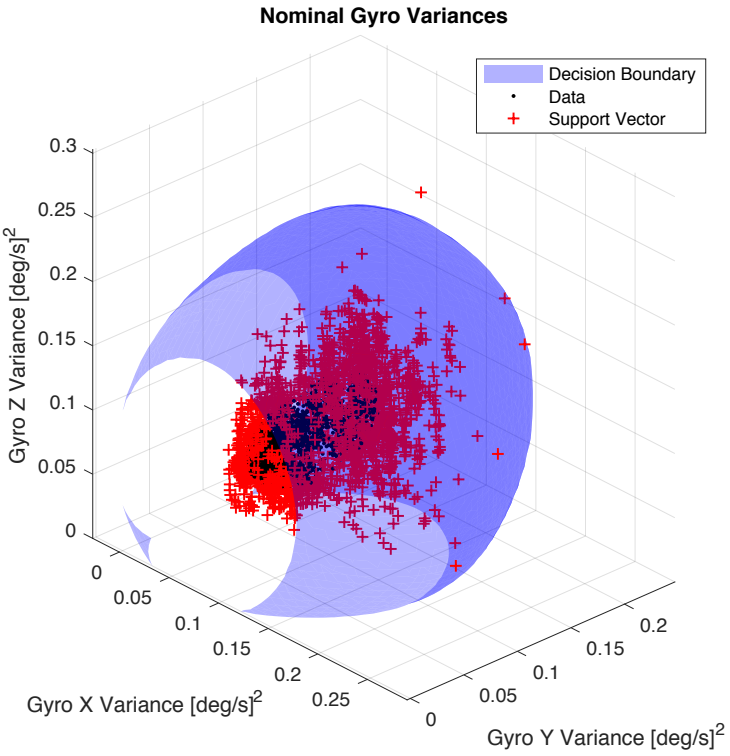


Figure 3.1: OCSVM for detecting anomalies in the 60 sample (1-minute) variances of the 3 gyros.

and this is the value used by the fault simulator [142]. Thus, the expected current $\hat{I}$ for a given dipole command $||\vec{\mu_c}||$ is:

$$\hat{I} = \frac{||\vec{\mu_c}||}{(1+\chi)nA} \tag{3.3}$$

The fault simulator uses Eq. 3.3 to also compute the measured current, but adds mean-zero Gaussian errors with $\sigma = 10$ mA and is subject to torque rod faults such as short-circuits. Dipole commands are capped at 1 Am^2 by a saturation filter in the attitude control system. The torque rod current error is simply:

$$\varepsilon_{TR} = \hat{I} - I \tag{3.4}$$

In a similar vein, the commanded torque τ_c for each reaction wheel can be compared to the measured torque τ_m to give the torquing error:

$$\varepsilon_{RW} = \tau_c - \tau_m \tag{3.5}$$

where τ_m is provided directly by the wheel software [141]. Alternatively, it can be inferred from changes in the wheel speed Ω if the moment of inertia J_{RW} for the wheel is known:

$$\tau_m = J_{RW}\frac{\Delta\Omega}{\Delta t} \tag{3.6}$$

Eqs. 3.4 and 3.5 provide a time series of errors for each actuator. Similar to the gyro variance OCSVM, we can use a 60-sample sliding window to extract the means and variances of the error signals for a set of actuators[1] and compile them into a 6×1 vector of the form $(\mu_1, \sigma_1^2, \mu_2, \sigma_2^2, \mu_3, \sigma_3^2)^\intercal$. This allows us to create OCSVMs for detecting anomalies in the torques of the reaction wheels or currents of the torque rods.

Sensor Voting

The ADCS fault simulator maintains logs of which magnetometers and sun sensors were accepted during sensor voting. In the log, each sensor is labeled with either a '1' if accepted or '-1' if rejected. The results of voting apply only to the current time step and rejected sensors are eligible for acceptance at future time steps. This is useful because even nominally operating sensors are occasionally rejected due to random noise in excess of the 3-σ voting threshold. In contrast, if one or more sensors are repeatedly rejected during voting, this could indicate an anomaly.

To create a OCSVM for magnetometer voting, a 4×1 vector is formed where each component contains the number of times the corresponding magnetometer was rejected during the preceding minute. For example, $X = (1, 0, 0, 3)^\intercal$ would indicate that magnetometer #1 was rejected once, while magnetometer #4 was rejected three times.

[1]Either the set of 3 reaction wheels or 3 torque rods

A similar approach using a 5×1 vector can be followed to create a OCSVM for sun sensor voting. An important nuance must be observed, however. Whereas the magnetometers can make valid measurements at any time, the sun sensors only obtain valid readings when the Sun is in their FOV. When the Sun is not in their FOV, they tend to track the next brightest object, resulting in contradictory and unreliable readings. Thus, given an arbitrary attitude, it is often impossible to distinguish between a true anomaly in sensor voting and a sensor that is simply not facing towards the Sun. Only Sun pointing mode (ADCS mode 2) reliably maintains the Sun in view of all of the sensors simultaneously. Therefore, we restrict all training and operation of the sun sensor voting OCSVM to mode 2.

Both the sensor voting OCSVMs and the actuator OCSVMs are difficult to visualize due to their high (> 3 dimensionality), but their successful operation will be demonstrated in Section 3.2.2.

Angular Momentum Accumulation

Perturbation torques to the spacecraft attitude tend to impart angular momentum to the spacecraft body. To keep the attitude aligned with the target direction, this momentum is absorbed by the reaction wheels, and their RPM will increase over time as more momentum is absorbed. Eventually, the wheels will reach their maximum RPM and become unable to stabilize the attitude against further torques. This state is referred to as momentum saturation.

When momentum saturation occurs, the wheel speeds are reset to a lower (or zero) RPM, causing their stored angular momentum to be transferred back into the spacecraft body. The resulting rotation can then be damped out using the torque rods in detumble mode. Since the torque rods actuate external rather than internal torques, angular momentum is removed from the spacecraft. This is called momentum dumping.

The accumulation of angular momentum is a nominal part of spacecraft operations, but certain faults can cause the rate of accumulation to become excessive and limit the time the spacecraft is able to operate between momentum dumps. To create a OCSVM to monitor for anomalies in momentum accumulation, we can estimate the total angular momentum of the spacecraft at time t as:

$$\mathbf{L} = \begin{pmatrix} J_{RW} & 0 & 0 \\ 0 & J_{RW} & 0 \\ 0 & 0 & J_{RW} \end{pmatrix} \begin{pmatrix} \Omega_1(t) \\ \Omega_2(t) \\ \Omega_3(t) \end{pmatrix} + \mathbf{J_{SC}}\boldsymbol{\omega} \tag{3.7}$$

where the Ω are the rotation rates of the three reaction wheels, $\mathbf{J_{SC}}$ is the moment of inertia tensor of the spacecraft in the body-axis frame, and $\boldsymbol{\omega}$ is the vector of rotation rates as measured by the 3 gyros. Then, letting $\hat{L}(t)$ be the average of $||\mathbf{L}||$ over the preceding s samples, we can form a vector:

$$\mathbf{X} = \begin{pmatrix} \hat{L}(t) - \hat{L}(t-s) \\ \hat{L}(t-s) - \hat{L}(t-2s) \\ \vdots \\ \hat{L}(t-rs+s) - \hat{L}(t-rs) \end{pmatrix} \tag{3.8}$$

where r is the number of sample windows over which we wish to monitor the change in momentum.

Choosing $s = 120$ and $r = 3$, Fig. 3.2 shows the resulting OCSVM. As with the other OCSVMs, an outlier fraction of $\nu = 0.001$ is used. There are 3820 total data points and 1914 support vectors.

B-field Magnitude Discrepancy

Next, we develop a OCSVM for sensing anomalies in the overall magnitude of the magnetic field. This can be useful for identifying electromagnetic interference or an unexpected location within the Earth's magnetic field. Based on the spacecraft clock and estimated position (from propagating the orbital TLE), the IGRF model can be queried for the expected magnitude of the local magnetic field, B_{IGRF}. This can then

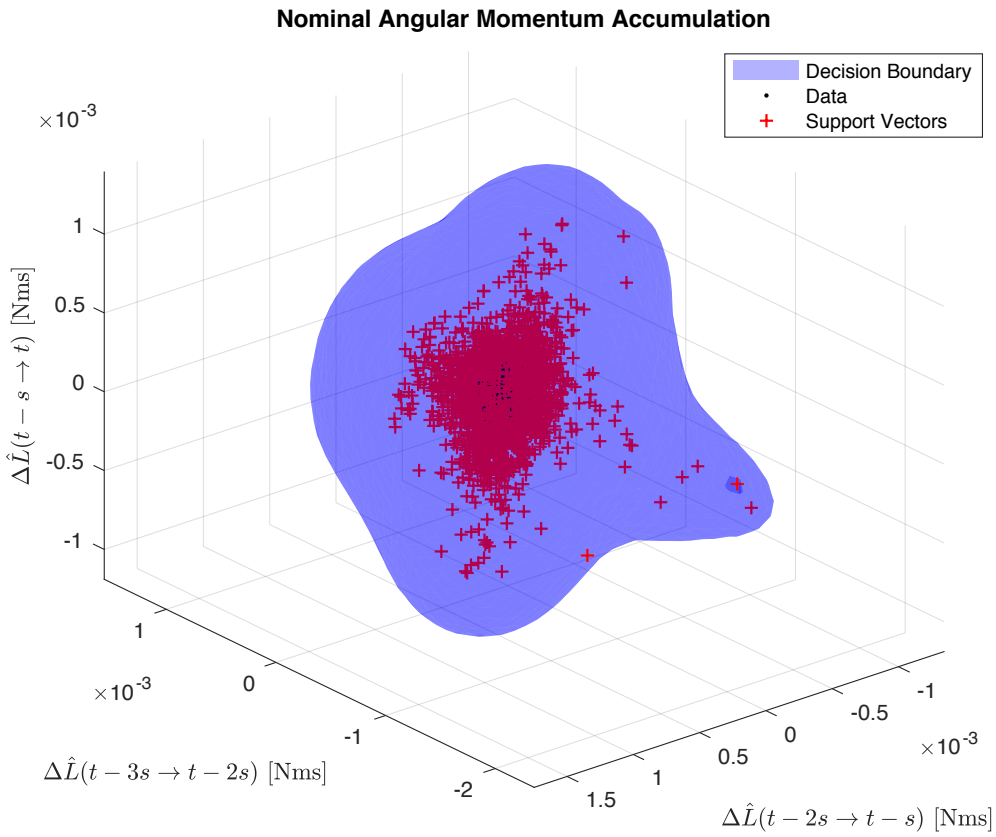


Figure 3.2: OCSVM for detecting anomalies in the accumulation of angular momentum.

be compared to the average magnitude $\overline{B_{meas}}$ measured by magnetometers accepted during voting by calculating the fractional error:

$$f_B = \frac{\overline{B_{meas}} - B_{IGRF}}{B_{IGRF}} \tag{3.9}$$

Extracting the running mean and variance of f_B using a 120-sample sliding window allows the creation of the OCSVM shown in Fig. 3.3. Working with 2-dimensional vectors such as with this OCSVM enables a straightforward visualization of the decision boundary that can be used to tweak the parameter ν. In this case, $\nu = 0.005$ has been used. The OCSVM uses 3940 data points and 1972 support vectors.

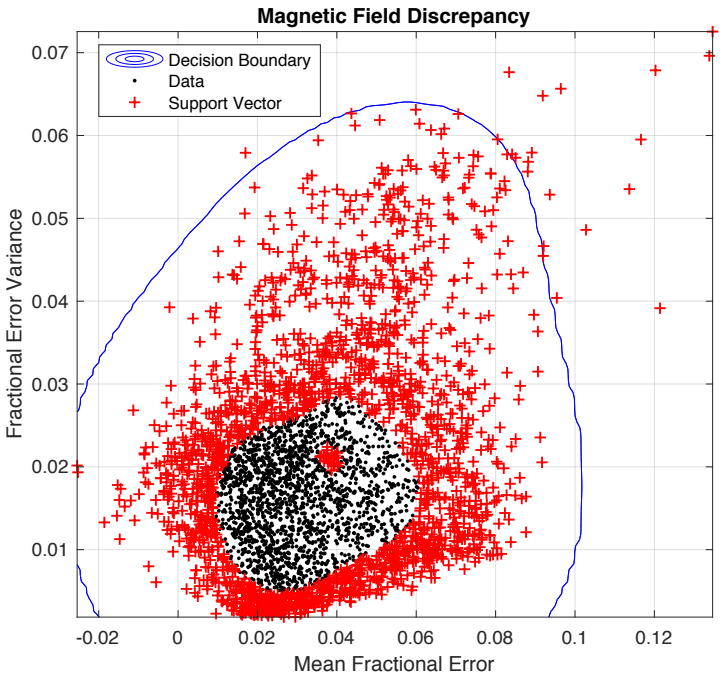


Figure 3.3: OCSVM for detecting anomalies in the overall magnitude of the magnetic field.

Quaternion-B-field Discrepancy

Many ADCS faults affect the accuracy of the spacecraft's attitude estimate. In addition to the attitude quaternion, the Kalman filter also provides an estimate of the error, but this does not take into account the integrity of the measurements flowing into the filter. To verify the consistency of the attitude estimate with lower level information, discrepancies between the measured direction of the magnetic field and the predicted direction based on the IGRF model can be evaluated. The process is depicted in Fig. 3.4(a) and proceeds as follows:

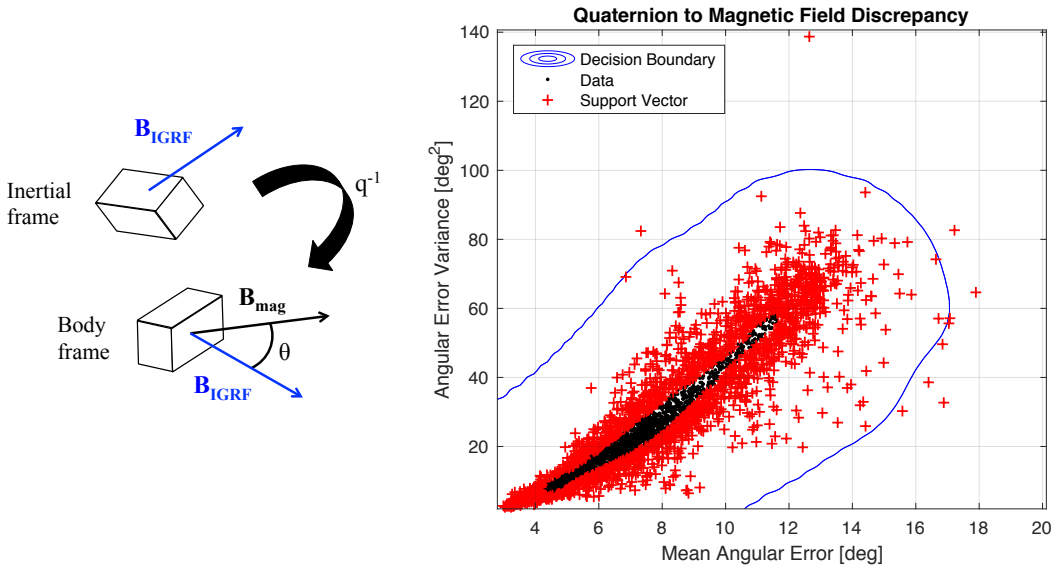
1. Compute the local magnetic field $\mathbf{B_{IGRF}}$ in the inertial frame based on the spacecraft's position and IGRF model

2. Compute the average magnetic field vector $\mathbf{B_{mag}}$ in the spacecraft body frame using magnetometers accepted by voting

3. Transform $\mathbf{B_{IGRF}}$ into the spacecraft body frame using the rotation matrix derived from the inverse of the estimated attitude quaternion q

4. Find the angular discrepancy between $\mathbf{B_{mag}}$ and $\mathbf{B_{IGRF}}$

Anomalously high error angles provide an independent indication that the attitude estimate is not reliable. The mean and variance of the discrepancy angle θ are extracted from nominal data using a 120-sample sliding window and used to create the OCSVM in Fig. 3.4(b). The OCSVM uses $\nu = 0.005$, 3940 data points, and 1976 support vectors.



(a) Quaternion discrepancy check (b) Nominal quaternion discrepancies

Figure 3.4: OCSVM for sensing anomalies in the quaternion to B-field discrepancy.

To demonstrate how the OCSVM method operates in practice, Fig. 3.5 shows an example timeseries of quaternion-B discrepancies obtained from the fault simulator. Just over 80 minutes into the simulation, additional noise is added to the gyro measurements, resulting in a significant degradation of attitude knowledge. The stream of quaternion-B discrepancies is processed into (μ, σ^2) by a 2-minute sliding mean/variance window. Each (μ, σ^2) is used to compute an anomaly score using the

scaled OCSVM decision function (Eqs. 2.7 and 2.13). When the degradation of attitude knowledge causes (μ, σ^2) to fall outside of the OCSVM decision boundary, the anomaly scores exceed 0.5.

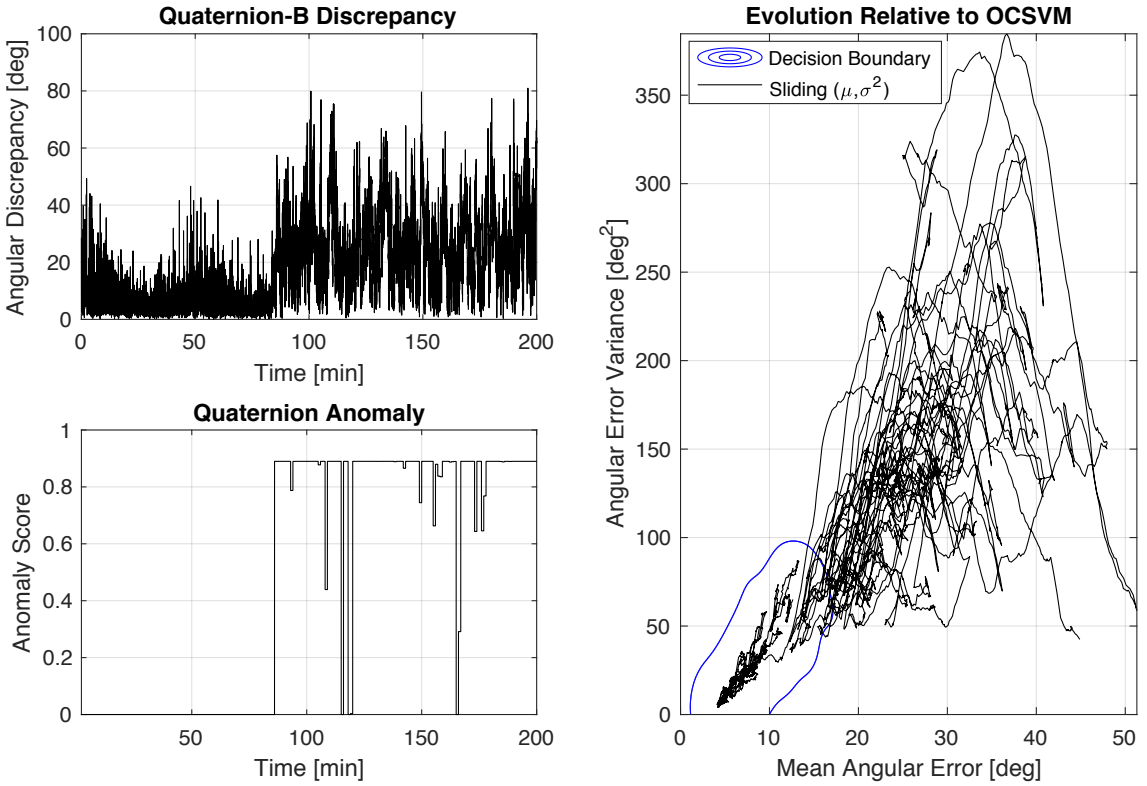


Figure 3.5: Demonstration of a OCSVM detecting a disruption in attitude knowledge.

3.1.3 Rule-based Detectors

The OCSVMs developed in Section 3.1.2 are a powerful tool for sensing anomalies when it is difficult to specify exactly what constitutes an anomaly. In some cases, however, the nature of an anomaly is straightforward to define with simple If-then rules. It therefore makes sense to leverage the rule-based approach where it can provide additional anomaly information with minimal complexity. The following subsections

describe signals and If-then rules that supplement the anomaly information provided by the OCSVM detectors.

Angular Rate Checks

An immediate indication of a severe ADCS anomaly is if the spacecraft begins to spin with an excessive angular rate. For most spacecraft, rotation rates of more than a few degrees per second will never be encountered during nominal operations. A simple anomaly check is thus used to implement an angular rate threshold which, if violated by any measurement from the spacecraft's gyros, raises an anomaly flag:

Anomaly Rule 1: Violations of a rotation rate threshold

RateThreshold (ω);
Input : Vector of measured rotation rates from gyros
Output: Anomaly flag indicating an excessive rotation rate
if any $\omega_i > 2$ deg/sec **then**
    return ExcessiveRate = 1;
else
    return ExcessiveRate = 0;
end

Opposite to an excessive angular rate is if the angular rate is exactly 0 deg/sec. Gyro rate measurements are typically discretized, allowing small rotation rates to become rounded down to 0 deg/sec. In practice, measurement noise and the finite pointing stability of a CubeSat's ADCS makes it exceptionally rare for a zero rate to appear for more than a single measurement. Where one does persist, it is likely an indication of a failed gyro. To capture such anomalies, we can create Rule 2.

Note that this rule does not capture a persistent zero rate and will lead to many false positives as gyros occasionally report zero degrees/sec. This is actually not a problem. As we will see in Section 3.2.2, the LSTM network learns to reject false

Anomaly Rule 2: Gyro rates exactly zero

ZeroRate (ω);

Input : Vector of measured rotation rates from gyros

Output: Anomaly flag indicating a zero rotation rate

if any $\omega_i = 0$ **then**

    return RateIsZero = 1;

else

    return RateIsZero = 0;

end

positives and search for a persistent RateIsZero flag itself, thus simplifying the rules that we need to develop.

Eclipse-Power Discrepancy

While not directly related to ADCS, the amount of solar polar generated by the electrical system can reveal a contradiction between where the spacecraft thinks it is and where it actually is. Put simply: solar power should not be generated in eclipse. If it is, this is an anomaly. This is captured in Rule 3.

Anomaly Rule 3: Power and eclipse discrepancy

EclipsePower (Power, InEclipse);

Input : Amount of solar power, eclipse flag from propagating TLE

Output: Discrepancy flag

if Power $>$ 0.1 Watts **and** InEclipse **then**

    return EclipsePowerDiscrep = 1;

else

    return EclipsePowerDiscrep = 0;

end

The inputs to Rule 3 are the total instantaneous solar power from all solar panels and the eclipse flag determined from propagating the spacecraft's onboard TLE. If the power exceeds a small finite value (necessary to reject measurement errors) when the eclipse flag is active, an anomaly flag is raised. Note that this flag will not be active continuously even in the presence of a fault. It can only be active when the propagated TLE is in eclipse.

Reaction Wheel Response and Saturation

Two other simple tests are checking for responsiveness and saturation of the reaction wheels. The wheel saturation check raises an anomaly flag when any of the reaction wheel speeds are close to to their saturation limit. For the wheels used in the fault simulator this is 6500 RPM [141]. Setting the threshold for the saturation flag slightly below this (e.g. 5850 RPM) can make the anomaly flag more persistent and provide an early warning before the wheel actually reaches saturation. This is accomplished by Rule 4:

Anomaly Rule 4: Reaction wheel saturation threshold

WheelSaturated (Ω);

Input : Vector of rotation speeds for the reaction wheels

Output: Wheel saturation flag

if any $\Omega_i > 5850$ RPM **then**

    return WheelSat = 1;

else

    return WheelSat = 0;

end

To test that each wheel is responding to torque commands, Rule 5 checks that the speed of each wheel changes when subjected to a non-zero torque command. If all torque commands are zero (such as when in ADCS modes that do not operate the

reaction wheels), the responsiveness is unknown and the rule returns "NaN". This rule is designed to detect a seized reaction wheel and help distinguish the resulting torquing anomalies from other reaction wheel faults.

Anomaly Rule 5: Reaction wheel responsiveness test

WheelResponding (τ, $\Delta\Omega$);

Input : Vector of wheel torque commands at the previous time step, vector of changes in wheel rotation speeds between this time step and the previous time step

Output: Wheel responding flag

if all $\tau_i = 0$ **then**
    return Responding = NaN;
else if any ($\tau_i \neq 0$ **and** $\Delta\Omega_i = 0$) **then**
    return Responding = 0;
else
    return Responding = 1;
end

Pointing Accuracy

A common rule-based anomaly flag for ADCS is to check if the pointing error with respect to the commanded attitude is above some acceptable threshold [71]. We include this check in the form of Rule 6. The error angle is easily determined from Eq. 2.36. In contrast to traditional implementations, Rule 6 does not include a persistency parameter, nor does it account for the time required to settle into the new attitude when a new ADCS mode is commanded. The LSTM network will learn both of these implicitly, thus simplifying rule design. Note that Rule 6 can only be evaluated when in a controlled attitude mode (ADCS modes 2, 4, and 5).

Anomaly Rule 6: Pointing error anomaly threshold

PointingAnomaly (ErrorAngle);

Input : Pointing error angle relative to commanded attitude

Output: Excessive attitude excursion flag

if ErrorAngle $>$ 20 deg **then**

  return AttitudeExcursion = 1;

else

  return AttitudeExcursion = 0;

end

Average Sensor Agreement

The sensor voting OCSVMs will score an anomaly when either a single sensor is rejected an anomalous number of times, or all of the sensors are rejected an anomalous number of times. Since the OCSVM only outputs the final anomaly score, additional information is needed to distinguish between these cases. A straightforward approach is to compute the average sensor agreement over the last 60 samples, $\overline{f_{agree}}$. If an anomaly affects all sensors, then $\overline{f_{agree}} \sim 0$, whereas if just a single sensor is affected $\overline{f_{agree}} \sim 0.75$. This provides additional information to supplement the sensor voting OCSVMs. It applies to both the magnetometers and sun sensors. But note that, like the sun sensor OCSVM, it is only meaningful as a detector of sun sensor anomalies when in Sun pointing mode.

Sun Sensor Discrepancy

The quaternion-B field OCSVM compares the measured direction of the magnetic field in the spacecraft body frame with the anticipated direction derived from the IGRF model and attitude quaternion. A similar discrepancy check can be performed between the quaternion and Sun direction. Analogous to Fig. 3.4(a), using a solar

position model to query the Sun vector $\vec{S}$ in the inertial frame at the current time, $\vec{S}$ is rotated into the spacecraft body frame using q^{-1}. The discrepancy angle θ_{Sun} relative to the average of all (accepted) sun sensor measurements is then computed. Though the process is theoretically equivalent as a test of quaternion accuracy, in practice the sun sensors provide a smaller contribution to overall attitude knowledge. This is because the Sun must be within the FOV of a majority of the sensors before they have a chance of being accepted into the Kalman filter. In contrast, magnetometer voting is not influenced by the spacecraft's attitude or the presence of eclipse.

Rather than a gauge of overall quaternion accuracy, the sun sensor discrepancy angle is more useful as a cross-check between the attitude knowledge provided by the sun sensors and the attitude knowledge provided by the magnetometers. If θ_{Sun} is high despite a lack of quaternion-B anomaly, it can indicate that the sun sensors are not providing reliable measurements. We will use θ_{Sun} averaged over a 60 second sliding window to supplement the rest of the anomaly information. If no sun sensors are accepted during voting, θ_{Sun} is defined as NaN.

Long Duration Counter

Lastly, none of the previously mentioned anomaly detectors or signals provide information about ADCS reboots. Reboots interrupt the ADCS process and force it to re-initialize, meaning that attitude knowledge is temporarily lost until the Kalman filter can re-converge. The Long Duration Counter (LDC) tracks the number of minutes of ADCS operation since the last ADCS reset. Including it alongside other anomaly information can help identify when ADCS is being disrupted by a reset.

3.2 LSTM Fault Isolator Development

The anomaly detectors in Sections 3.1.2 and 3.1.3 provide 18 anomaly scores and supporting signals to use as inputs for fault isolation. Summarizing them in the order used by the LSTM network, they are:

1. Long duration counter
2. Magnetometer agreement
3. Magnetometer voting anomaly
4. Sun sensor agreement
5. Sun sensor voting anomaly
6. Sun sensor discrepancy
7. B-field magnitude error anomaly
8. Eclipse-power discrepancy
9. Quaternion-B anomaly
10. Excessive rotation rate
11. Zero rotation rate
12. Attitude rate variance anomaly
13. Attitude error
14. Angular momentum anomaly
15. Torque rod current anomaly
16. Reaction wheel torque anomaly
17. Reaction wheels responding
18. Reaction wheels saturated

The combination of anomaly scores, flags, and other signals across multiple time steps provides a pattern that can be used to identify the underlying fault. This is the purpose of the LSTM network. In addition to the nominal (no fault) case, we consider a total of 21 unique ADCS faults for the LSTM network to isolate. These faults are described in Table 3.3. The LSTM network will thus have 22 outputs with each output providing a measure of confidence $\in (0, 1)$ that the corresponding fault scenario is present in the system.

This section describes development of the fault isolation LSTM network and assesses its performance. Section 2.2.3 covers the process of generating fault scenario data using the fault simulator and training the LSTM network. Section 3.2.2 provides several examples that illustrate the core strengths of the LSTM network when applied to fault isolation. Section 3.2.3 follows with a more complete assessment of the network's performance.

In this Chapter, it is assumed that the faults only occur one at a time. In Chapter 4, the ability of the method to diagnose multiple faults occurring simultaneously will be demonstrated.

3.2.1 Generating Fault Training Data

In order to train the LSTM network, it is necessary to provide a dataset of known inputs and outputs (called the targets). This can be obtained from the ADCS fault simulator by simulating how the 18 anomaly input signals behave under different fault conditions. The target outputs for each simulation are simple binary flags indicating which fault was injected in the simulation at each time. Ideally, we want the LSTM's confidence outputs to match these flags.

Table 3.3 describes the 22 fault conditions simulated to create a dataset for LSTM training. Some fault scenarios comprise multiple "subfaults" that are effectively identical given the inputs defined Section 3.1. In every case, relevant parameters pertaining to the fault to be injected – including the time of injection, affected sensors, signal characteristics, etc. – are randomized. Also randomized are the spacecraft's orbit, initial attitude and attitude rate, and initial epoch according to Table 3.2. The ADCS control mode is selected randomly from applicable modes listed in the rightmost column of Table 3.3. The fault scenarios themselves are cycled through in order to ensure equal representation in the final dataset. The total dataset consists of 1200 total simulations, of which 700 are used to train the LSTM network. The remainder are used to validate its performance.

Table 3.3.: Simulated ADCS faults for LSTM training. Each fault scenario corresponds to one of the output channels of the LSTM network.

ID	**Fault Scenario**	**Applicable Modes**
0	**Nominal**: No faults.	All
1	**Magnetometer interference**: Magnetometer noise means shifted by up to 2×10^{-5} T and variance increased by up to 4×10^{-10} T^2.	All

2	**Failed magnetometer**: a magnetometer outputs a constant "stuck" value following the failure.	All
3	**Failed sun sensor**: a sun sensor gives random readings even when the Sun is within its FOV.	2
4	**False sun**: sun sensors register a bright object that is not the Sun if the Sun is not in their FOV. The non-solar source is a constant vector specified in the spacecraft body frame.	0,1,3,4,5
5	**Orbit ephemeris or clock error**: the spacecraft's position is off by $36 - 216$ degrees along its orbit or its clock is incorrect by between $10^4 - 10^8$ seconds.	All
6	**Magnetometer phasing or rotation matrix error**: two random sensors are cross-wired or incorrect rotation matrices are used to transform measurements into the spacecraft body frame. The incorrect matrices are simply the correct ones transposed.	All
7	**Sun sensor phasing or rotation matrix error**: similar to above, but applied to the sun sensors.	2
8	**Panel deployment fault**: a random solar panel deploys to an anomalous angle between $0 - 120$ degrees rather than the nominal 165 degrees, invalidating rotation matrices for sensors on that panel.	2
9	**Failed gyro**: a gyro ceases to register rotation.	All
10	**Gyro calibration error**: gyro rate measurements are biased by a random constant value of up to 1 deg/sec.	All
11	**Gyro phasing error**: two gyros are cross-wired.	All
12	**Reaction wheel phasing error**: two reaction wheels are cross-wired.	2,4,5
13	**Unresponsive reaction wheel**: a reaction wheel ceases to actuate when commanded.	2,4,5

14	**Reaction wheel torque error**: measured torques do not match the commanded torques. Either noise is added to wheel torques, torques are reduced by a constant multiplicative factor, or wheel friction coefficients are increased.	2,4,5
15	**Reaction wheel saturated**: one or more reaction wheels fail to actuate due to a saturated RPM.	2,4,5
16	**Failed torque rod or current error**: measured currents do not match the dipole command due to a damaged coil or electrical problem. Either the actuated dipole of one of the rods is set to 0 or noise is added to the currents of affected torque rods.	All
17	**Torque rod polarity error**: one or more torque rods actuate in the wrong direction due to a sign error.	0
18	**Excessive magnetic dipole**: a constant residual magnetic dipole from the torque rods or other source leads to unexpected magnetic torques. To represent this, a random dipole vector in the body frame with a magnitude of at least 0.85 Am^2 is applied to the spacecraft.	2,4,5
19	**Unexpected torque**: the spacecraft experiences a constant anomalous torque of magnitude 10^{-5} to 2.5×10^{-5} Nm in the spacecraft body frame.	1,2,3,4,5
20	**Attitude rate jitter**: mechanical vibrations result in up to $\sigma = 5$ deg/sec of additional noise applied to the gyros.	All
21	**Reboot cycle**: The ADCS process enters a continuous reboot cycle with resets occurring every $1 - 50$ minutes.	All

Numerical Conditioning

Several pre-processing steps are necessary to improve the training and operability of the neural network. The first is the length of the training sequences. Even though

the LSTM network can operate on a continuously streaming sequence of inputs, training sequences with more than several hundred time steps can result in long training times and vanishing gradients. It is important to keep the length of the training sequences bounded to $\lesssim 400$ time steps while also ensuring that the sample time captures the general time scale on which ADCS faults manifest. A sample period of 1 minute suits these criteria. All training and testing data from the fault simulator and anomaly checks are thus downsampled to 1/60 Hz. This is the frequency at which the LSTM network will process new time steps.

Second, many of the anomaly signals can be indeterminate in certain situations. The reaction wheels do not operate in ADCS modes 0,1, or 3, for instance, and thus cannot be tested for responsiveness or torque anomalies. Other OCSVMs can only check for anomalies after having collected sufficient data. In any of these cases the input signal can be set to a code to indicate missing data and the LSTM network will learn the treat it appropriately. A value of -1 is used throughout this thesis.

Lastly, input signals should be normalized to the range of approximately $(-1, 1)$. As with the OCSVMs, this is accomplished by subtracting the mean and dividing by the standard deviation of each input. A special case is needed for the LDC input. Since this input is unbounded (it can grow indefinitely as long as the ADCS subsystem does not reboot), it is capped at 180 minutes before normalization.

Ensuring the Integrity of Training Data

Since the inputs of the LSTM network are primarily anomaly scores, the values of the various parameters used to simulate faults in Table 3.3 are somewhat arbitrary. In terms of the anomaly scores and other input signals that result, a panel deployment fault with the solar panel deployed 0 degrees will be essentially identical to one deployed to 90 degrees. Both will lead to an anomalous number of magnetometer and sun sensor rejections during voting. There is one sense in which the parameters are not arbitrary, however. They must be large enough to produce detectable anomalies

in the OCSVMs and other checks that form the inputs to the LSTM. If this is not the case, then the inputs will not be useful for determining the fault scenario. The LSTM will end up learning bogus correlations that are present in the training dataset but provide no predictive value when applied to new data. Thus, it is crucial to ensure that the injected faults manifest anomalies in the inputs to the LSTM network so that the network's predictions can be evaluated fairly.

3.2.2 Detailed Examples

To train the LSTM fault isolator, one selects the number of memory cells to include in the model and solves the optimization problem described in Section 2.2.3. Selecting the number of memory cells to be slightly larger than the number of fault outputs generally provides the best performance. This gives the network the opportunity to learn multiple parallel representations of each anomaly-fault relationship without overfitting. In other words, a single input can propagate through independent paths in the network that ultimately reinforce each other at the output. If one of the paths is compromised[2], the output is relatively unaffected. On the other hand, if the number of cells is less than the number of outputs, the network may lose important information as it is forced to compress the information in the inputs into an overly compact representation.

To provide specific examples of fault isolation performance, this section and the next are based on a 35-cell LSTM network that was trained in less than 10 minutes using Matlab's Deep Learning Toolbox®. The following examples demonstrate the key advantages of the LSTM approach to fault isolation.

Nominal Nadir Alignment

Figure 3.6 presents the inputs, evolving cell states, and outputs of the LSTM network when applied to monitor 2 orbits of nadir pointing mode. No faults are

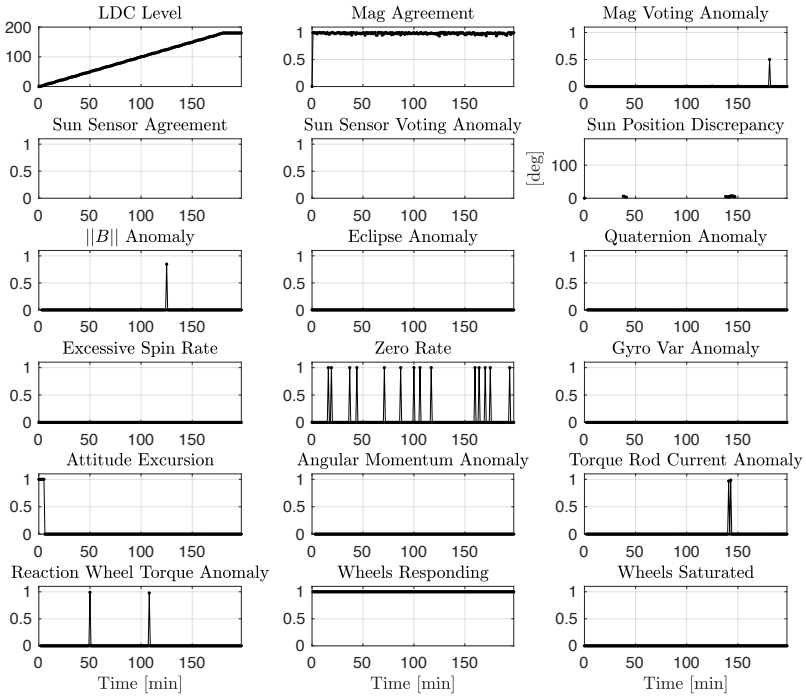
[2]For example, by a competing input that cancels out the signal of one path

present. Several anomaly false positives are still present, however. In Fig. 3.6(a), the OCSVMs for magnetometer voting, magnetic field magnitude, torque rod currents, and reaction wheel torques all report sporadic anomalies. Yet, rather than declaring a fault, the LSTM network is able to appreciate these false positives for what they are. Though small momentary perturbations to the "No fault" confidence around these anomalies are visible in Fig. 3.6(c), the network ultimately converges to a high confidence in nominal performance.

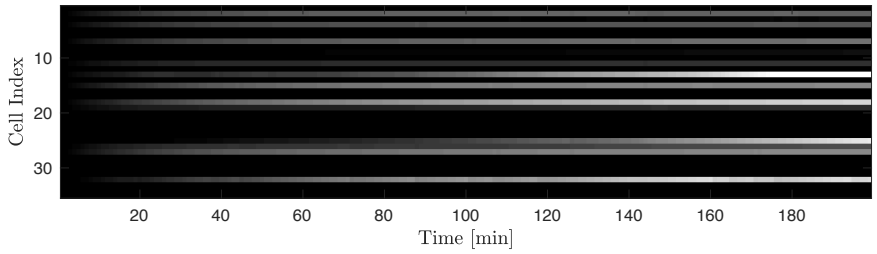
Figure 3.6(b) shows the activation of each of the 35 memory cells on a $[0 - 255]$ brightness scale. The increasing brightness of certain cells shows how cells associated with a nominal fault diagnosis receive higher levels of activation as evidence supporting this diagnosis accumulates. Meanwhile, cells associated with other diagnoses receive very low activation and remain dark. In later examples, the cell states will show how the LSTM network accumulates and sometimes strategically dumps evidence in order to evaluate competing diagnoses.

Sun Sensor Failure

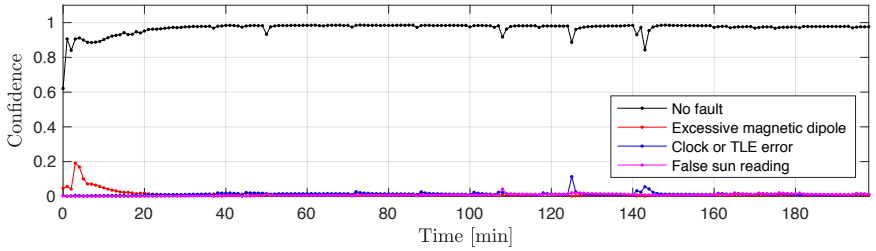
Next we explore a sun sensor failure. This fault is triggered in the -X sensor 5000 seconds (83 minutes) into a simulation of Sun pointing mode. Anomaly inputs and the LSTM's response are shown in Fig. 3.7. The failed sensor immediately leads to a drop in the average sun sensor agreement and triggers an anomaly from the sun sensor voting OCSVM. Confidence in nominal performance plummets in 3.7(c) as a result. The network subsequently allocates confidence to two fault hypotheses: a failed sun sensor and a panel deployment failure. Both faults will cause an increased number of sun sensor rejections, but the latter case should also lead to a similar result in the magnetometers. When a drop in magnetometer agreement fails to materialize, confidence allocated to the panel deployment failure declines while the sun sensor failure comes to dominate. The growing activation of neurons associated with the sun sensor fault can be see in Fig. 3.7(b).



(a) Anomaly inputs



(b) LSTM cell state evolution



(c) Fault confidence outputs

Figure 3.6: Fault isolation applied to a nominal case in nadir pointing mode. The LSTM network successfully rejects false positives in the anomaly inputs and converges to a correct diagnosis.

This example demonstrates two important characteristics of the LSTM's fault isolation performance: entertaining multiple fault hypotheses and iterative refinement. The network succeeds at distributing confidence between two reasonable explanations for the observed anomalies. Confidence shifts between the two hypotheses as more information becomes available, allowing the network to slowly converge to a stable result. As with Fig. 3.6, the diagnosis is robust to false positives from the other anomaly detectors.

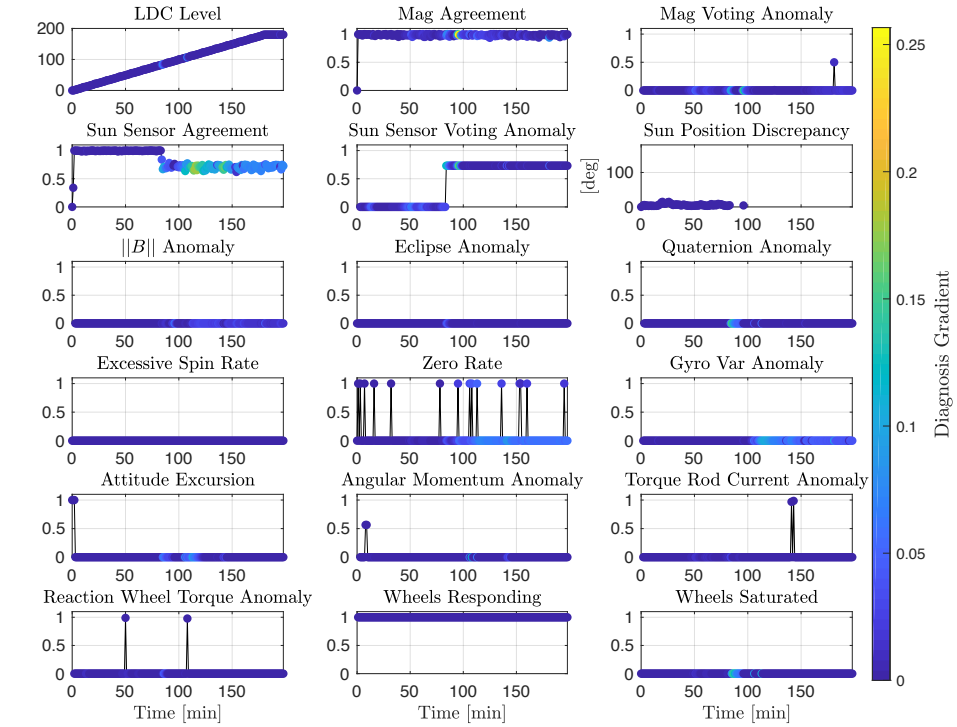
To help understand the LSTM network's confidence allocation, Fig. 3.7(a) colors each input based on the gradient with respect to the sun sensor failure (fault ID = 4): $\frac{\partial y_t^{(\mathrm{ID})}}{\partial x_t}$ computed according to Eq. 2.27. An additional negative sign is introduced for the LDC, sun sensor agreement, magnetometer agreement, and wheel responding flag, since it is decreases in these inputs that are anomalous rather than increases.

The relative gradients show which inputs and which data points are most significant to the LSTM's "failed sun sensor" diagnosis. In order, the highest gradients belong to the inputs:

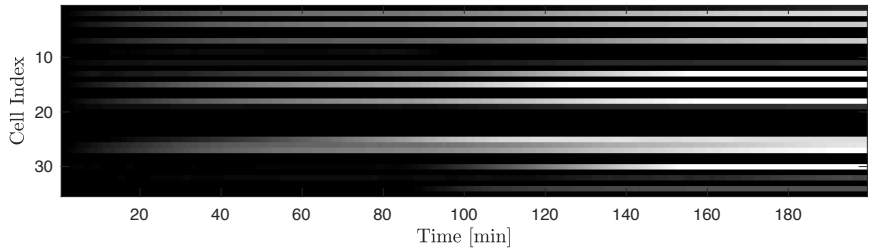
1. Magnetometer agreement
2. Sun sensor agreement
3. Sun sensor voting anomaly
4. Magnetometer voting anomaly

These are exactly the inputs that a human operator would focus on when isolating the fault. Not only does the neural network highlight the anomalous signals, but it also identifies inputs that provide a crucial source of additional information even though those inputs are themselves nominal. This allows us to verify that the network has learned a sensible relation between the anomaly inputs and the failed sun sensor fault.

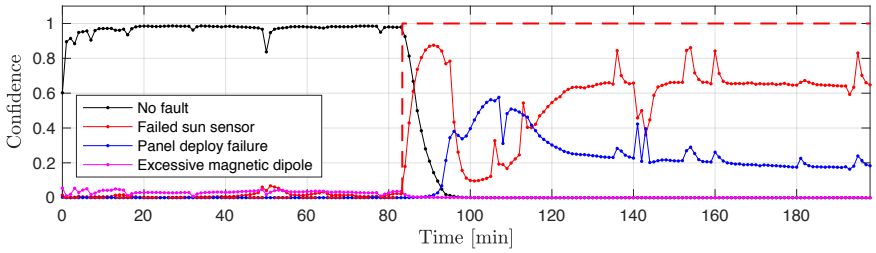
Beyond knowing that a sun sensor failure has occurred, we may be interested in isolating exactly which sensor is problematic. This can be accomplished using the sun sensor voting OCSVM and the far field anomaly gradient developed in Section 2.1.2,



(a) Anomaly inputs



(b) LSTM cell state evolution



(c) Fault confidence outputs

Figure 3.7: Diagnosis of a failed sun sensor. The LSTM network identifies two possibles causes for the anomalous drop in sun sensor agreement and converges towards the correct diagnosis. The diagnosis gradient shows which inputs are most influential to the network's ultimate decision.

Eq. 2.18. Under nominal conditions, each sensor should have a relatively equal contribution to the gradient $-\frac{\nabla f}{||\nabla f||}$. When a sensor fails and triggers an anomalous number of rejections, we expect the anomaly gradient to be positive in the direction corresponding to the failed sensor, since it is this sensor that is causing the anomaly.

Figure 3.8 plots each component of $-\frac{\nabla f}{||\nabla f||}$ throughout the failed sun sensor example. Soon after the failed sun sensor is injected, the -X component departs from the other sensors and becomes positive. This clearly and correctly identifies the -X sun sensor as the failed sensor. Note that in practice we should ignore the anomaly gradient prior to the appearance of an anomaly, since it only provides useful information when an anomaly is present.

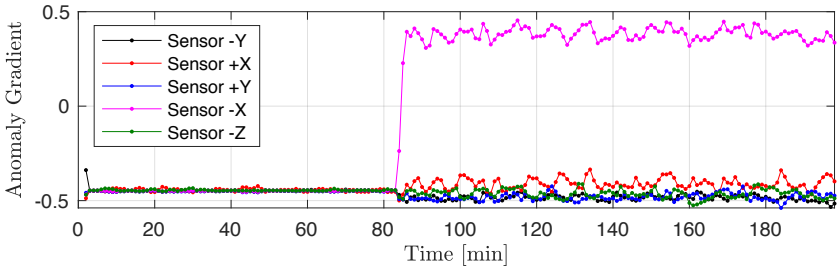


Figure 3.8: The anomaly gradient of the sun sensor voting OCSVM can be used to isolate the failed sensor. A positive anomaly gradient indicates that the -X sun sensor is the source of the voting anomaly in the failed sun sensor example.

3.2.3 Assessing Overall Performance

Appendix D provides additional examples of isolating faults on the 6DOF fault simulator to strengthen those provided in Section 3.2.2. To provide a wider picture of the LSTM network's overall performance, Fig. 3.9 presents a confusion matrix that summarizes the network's diagnosis decisions for all 500 validation examples. The chart shows the network's final diagnosis on the left axis against the actual injected fault on the bottom. Each cell contains the number of times the LSTM network made the corresponding fault diagnosis in the corresponding fault scenario. Since

the LSTM fault confidences can vary over each time series, the final diagnosis by the network is defined to be the output with the highest average confidence over the final 30 minutes of each simulation.

The average fault diagnosis accuracy over all scenarios is 96%. For individual fault scenarios, the accuracy ranges between 59% and 100%, with the unexpected torque fault being a particularly difficult case to diagnose. This fault is often confused with the torque rod polarity error (ID 17) and excessive dipole faults (ID 20), both of which also involve an anomalous build up of angular momentum. This indicates that additional information is needed to more clearly distinguish these faults. However, even when the network's ultimate diagnosis is incorrect, the confusion plot shows that it is still able to select reasonable alternatives. In addition to the unexpected torque fault, this is also visible in column 3 of the matrix, where 4 sun sensor failures were mis-classified as sun sensor phasing or rotation matrix errors.

Another notable result of Fig. 3.9 is the reliability with which nominal cases are diagnosed. Out of 22 nominal cases, all were correctly identified by the LSTM network as being devoid of faults. Of the 24 simulations diagnosed as nominal by the network, only 2 actually possessed a fault. Thus, the LSTM method provides both a high sensitivity (detecting a fault when one is present) and a high reliability (faults being present when one is detected) when applied to a diverse assortment of ADCS faults.

3.3 Decision Theory for ADCS Fault Recovery

In this section, the LSTM fault isolator is combined with decision theory to demonstrate the selection of an appropriate ADCS mode in response to a fault. This capability is referred to as the decision system. Section 3.3.1 describes a method for selecting actions under uncertainty given tuneable mission priorities called utilities. Section 3.3.2 develops these utilities for the fault simulator and Section 3.3.3 provides an in-depth example. Finally, Section 3.3.4 assesses the overall performance of the method.

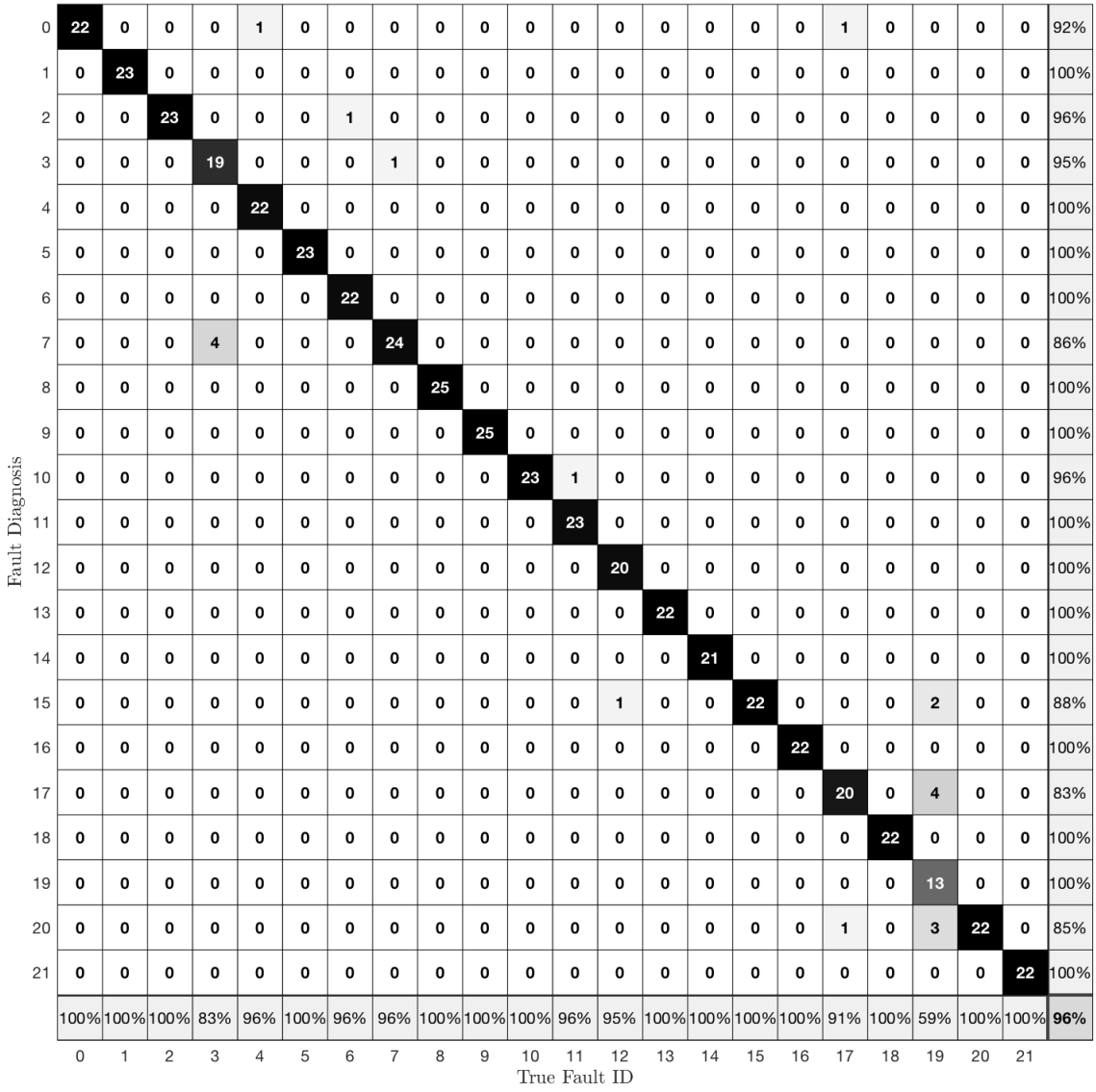


Figure 3.9: Confusion matrix summarizing LSTM fault isolation performance on all 500 validation examples using the fault simulator.

3.3.1 Introduction to Decision Theory

Decision theory is a framework for selecting actions that maximize some notion of benefit under uncertainty [72, Chapter 9]. The benefit of different outcomes is measured by quantities called utilities. Each utility is an abstract but quantitative measure of the value of a particular outcome if it were to manifest. By multiplying the utilities by the probability of each outcome, we can obtain the expected utility of taking a given action. The optimal action to take is thus the one that maximizes the expected utility.

Decision theoretic approaches have been applied to risk management during spacecraft development [120, 121], and have also been applied to evaluate recovery options based on uncertain root cause diagnoses with Bayesian networks [89]. A clear advantage is the ability for operators to tune response logic by updating the utilities to reflect new priorities, goals, or risk posture.

3.3.2 Developing the Utility Matrix

The fault simulator ADCS is controlled at a high level by selecting one of 6 ADCS modes (Table 2.1). These modes are the primary way of commanding ADCS behavior and represent the possible actions one might take in response to a fault. Since faults impact each mode differently, we can assemble our utility values into a 22×6 matrix $\mathbf{U}$ with rows denoting possible fault scenarios (Table 3.3) and columns containing the available ADCS modes. Each entry $u_{i,j}$ contains the utility for commanding mode j when fault i is present. For simplicity, we will standardize the $u_{i,j}$ into the 5 categories described in Table 3.4.

Several of the utility categories in Table 3.4 are defined relative to the current mission objective. To see why this is necessary, consider a "no fault" scenario diagnosed with 100% confidence. If the utilities for each mode under the no fault case are all equal, there is no reason to choose one mode over any other. In reality, the spacecraft operator will have a "desired mode" that aligns with the current tasks the

Table 3.4: Standard utility value categories.

Category	Utility	Description
Dangerous	-5	The fault has the potential to become unrecoverable (resulting in loss of mission) if the spacecraft enacts or persists in the given mode
Disruption	-2	The fault disrupts the proper operation of the given mode, potentially resulting in reduced performance or loss of data
No benefit	0	The mode is unaffected by the given fault but it is not desirable to remain in the anomalous state
Recovery	2	Enacting the mode takes an active step towards recovering from a fault
Objective completion	5	The mode leads to nominal performance, allowing mission objectives to be achieved

spacecraft is performing. It is important that our utility values provide a preference to this mode so that the spacecraft does not abandon it without an adequate reason.

With this in mind, Table 3.5 presents a generalized utility matrix $\mathbf{U}$. The variable γ is a placeholder that denotes either "No benefit" or "Objective completion" depending on the desired mode of the operator:

$$\gamma_{i,j} = \begin{cases} 0 & \text{if } j \neq \text{desired mode} \\ 5 & \text{if } j = \text{desired mode} \end{cases} \tag{3.10}$$

The choices of utility values are generally derived from the operational criteria outlined in Table 3.6. No torques mode has no requirements, but is not a suitable response to an unexpected torque since this could lead to an accelerating spin rate if left unopposed. Accelerating spin rates can also result from gyro faults or reaction wheel phasing errors while in modes with active attitude control, or from torque

Table 3.5: Generalized utility matrix for ADCS mode selection.

Fault Scenario	Detumble	B-align	Sun point	No torques	Nadir point	Velocity point
No fault	γ	γ	γ	γ	γ	γ
Magnetometer interference	-2	-2	-2	γ	-2	-2
Failed magnetometer	γ	γ	γ	γ	γ	γ
Failed sun sensor	γ	γ	γ	γ	γ	γ
False sun	γ	γ	γ	γ	γ	γ
Orbit ephemeris or clock error	γ	2	-2	γ	-2	-2
Magnetometer phasing/rotation matrix error	-5	-5	-2	γ	-2	-2
Sun sensor phasing/rotation matrix error	γ	γ	γ	γ	γ	γ
Panel deployment fault	-2	-2	-2	γ	-2	-2
Failed gyro	γ	γ	-5	γ	-5	-5
Gyro calibration error	γ	γ	-5	γ	-5	-5
Gyro phasing error	γ	γ	-5	γ	-5	-5
Reaction wheel phasing error	γ	γ	-5	γ	-5	-5
Unresponsive reaction wheel	γ	γ	γ	γ	-2	-2
Reaction wheel torque error	γ	γ	-2	γ	-2	-2
Reaction wheel saturated	2	γ	-2	γ	-2	-2
Failed torque rod or current error	0	0	γ	γ	γ	γ
Torque rod polarity error	-5	-5	γ	γ	γ	γ
Excessive magnetic dipole	2	0	-2	γ	-2	-2
Unexpected torque	2	0	-2	-5	-2	-2
Attitude rate jitter	γ	γ	-2	γ	-2	-2
Reboot cycle	γ	γ	-2	γ	-2	-2

rod polarity errors or severe mis-interpretations of the B-field when the detumble controller is active. In any of these cases, the utility value is -5 (dangerous).

For many of the faults considered, the root cause cannot be corrected autonomously as it will necessarily involve further investigations and software updates by ground operators. Two exceptions are faults resulting in a large momentum build-up and the clock/TLE error. The former can be recovered from by entering detumble mode to desaturate the reaction wheels while the latter can be addressed by transitioning to B-alignment mode. This mode is ideal for downlink/uplink with the ground and provides the best opportunity for the spacecraft to receive a TLE or clock update. In any large constellation, it is likely that these updates would be handled automatically. Each of these cases is assigned a utility value of 2 (recovery).

Table 3.6: Operational requirements for each ADCS mode.

Modes	Requires
Detumble	– Accurate B-field determination
B-align	– Functioning torque rods
Sun point	– Accurate attitude knowledge
Nadir align	– Functioning reaction wheels
Velocity align	

Mode Selection

Given a 22×1 vector of instantaneous fault confidences y_t from the LSTM network, the expected utility of each ADCS mode is given by a vector $\hat{U}$, computed:

$$\hat{U_t} = \mathbf{U}^\intercal y_t \tag{3.11}$$

The optimal mode selection at any time is then the one corresponding to the largest component of $\hat{U_t}$.

Additional Considerations

Several considerations are necessary to ensure stable mode selection. Since the LSTM network was only trained with a fixed ADCS mode, it is prudent to reset the network's internal states (both h_t and c_t) after each mode change. Failing to do so can lead to incorrect confidence outputs that prompt further mode changes. Second, the LSTM network requires a certain amount of time to converge to a robust fault diagnosis. For the first $10 - 20$ time steps the fault confidences are often low and highly sensitive to the input signals. By instituting a mandatory waiting period of 20 time steps (20 minutes) between mode changes, it can be ensured that the LSTM network has sufficient input to make a confident fault diagnosis. This prevents the initial state of the LSTM network from triggering mode changes on the first few time steps after each reset.

Finally, some faults − such as those involving reaction wheels − are only detectable in certain ADCS modes. Enacting a mode change in response to such a fault along with an LSTM reset can cause the fault to disappear from the LSTM's diagnosis. This can lead the decision system to switch back into the original mode only to rediscover the fault and switch out again, thus becoming trapped in an oscillation between modes where the fault is visible and modes where it is not. To prevent such oscillations, it is necessary to retain the justification for mode changes across each mode change. A simple approach is this: whenever a mode change is commanded, update the desired mode to the mode being switched to. This strategy alters the interpretation of γ in Table 3.5 in a way that favors the new mode, at least until another fault is detected.

3.3.3 Detailed Examples

Priority Change

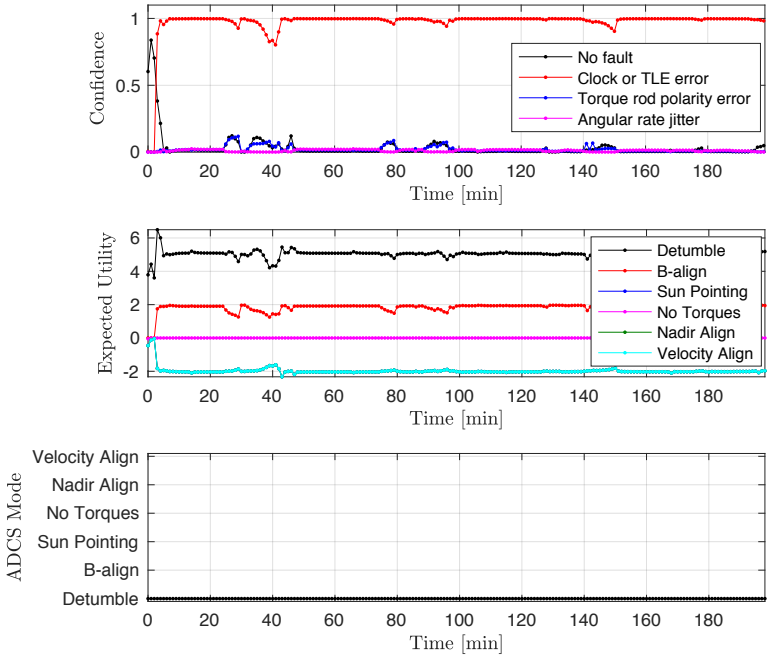
The utility values assigned in Table 3.5 are intentionally subjective. They represent the relative preferences of the spacecraft operator for different modes under different faults. For instance, the recovery utility ($u = 2$) is less than the utility for objective completion ($u = 5$) in order to prioritize an objective in progress, so long as that objective is not jeopardized by whatever fault is detected.

Figure 3.10 provides a concrete example for a spacecraft in detumble mode. In Fig. 3.10(a) the spacecraft clock is advanced by nearly 5.5 days, causing a major disruption to attitude knowledge that is correctly diagnosed by the LSTM network. This fault does not impact detumble performance, however, and since detumble is the desired mode, the expected utility of this mode remains high (~ 5). The spacecraft stays firmly in detumble mode to complete the momentum dump.

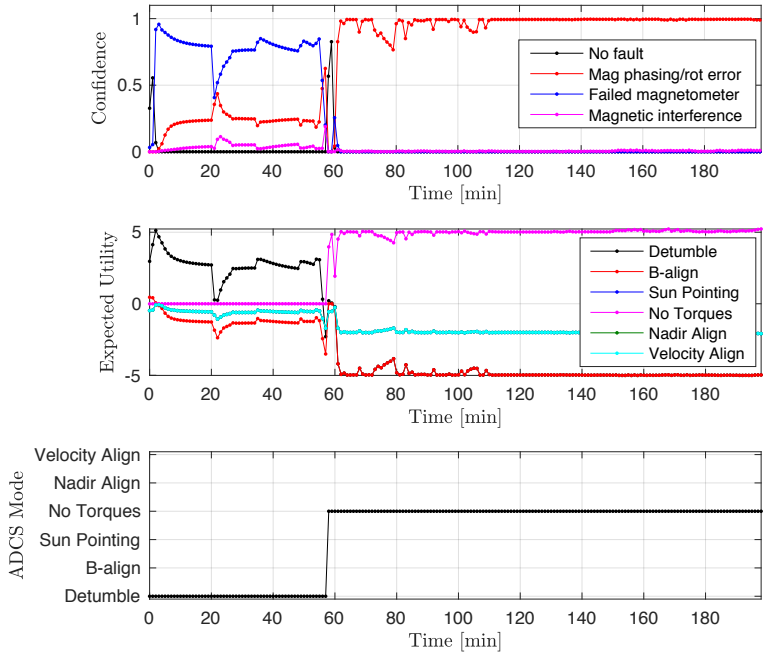
Contrast this with Fig. 3.10(b), in which the fault disrupts the objective and the initial mode is rightfully abandoned. Figure 3.10(b) presents the same simulation as Fig. 3.10(a) but with the magnetometer rotation matrices transposed in place of the clock offset. For the first 55 minutes the LSTM network attributes the various anomalies to a failed magnetometer. This would not normally warrant a mode change[3] based on Table 3.5, and so the spacecraft remains in detumble mode. Nevertheless, around 56 minutes the LSTM network redistributes its confidence in the failed magnetometer to the (correct) magnetometer rotation matrix error. Unlike the failed magnetometer or the clock offset, this fault can seriously degrade the effectiveness of mode 0. The expected utility of detumble mode drops to negative while "no torques" becomes the favored mode and the spacecraft changes modes accordingly.

The balance of utilities can be modified to suit the operator's evolving priorities. Suppose that it is considered imperative to update the spacecraft clock rather than

[3]The bad magnetometer will usually be voted out, preventing it from corrupting the magnetic field measurement



(a) Clock offset error



(b) Magnetometer rotation matrix error

Figure 3.10: Comparison of fault responses for a clock offset and a magnetometer rotation matrix error. The clock offset does not degrade detumble performance and the decision system takes no action. The rotation matrix error does affect this mode, causing a transition to no torques.

finish a momentum dump in progress[4]. To reflect this change in priorities, the utilities for recovery and objective completion in Table 3.4 can be swapped. Figure 3.11 repeats the simulation of Fig. 3.10(a) to show the altered response. Instead of remaining in detumble mode, the spacecraft switches to B-alignment mode at the first opportunity (after the 20 minute wait period) in order to establish contact with the ground. Hence, the fault response logic is easily modified via the utility values to reflect the mission's priorities and risk tolerance.

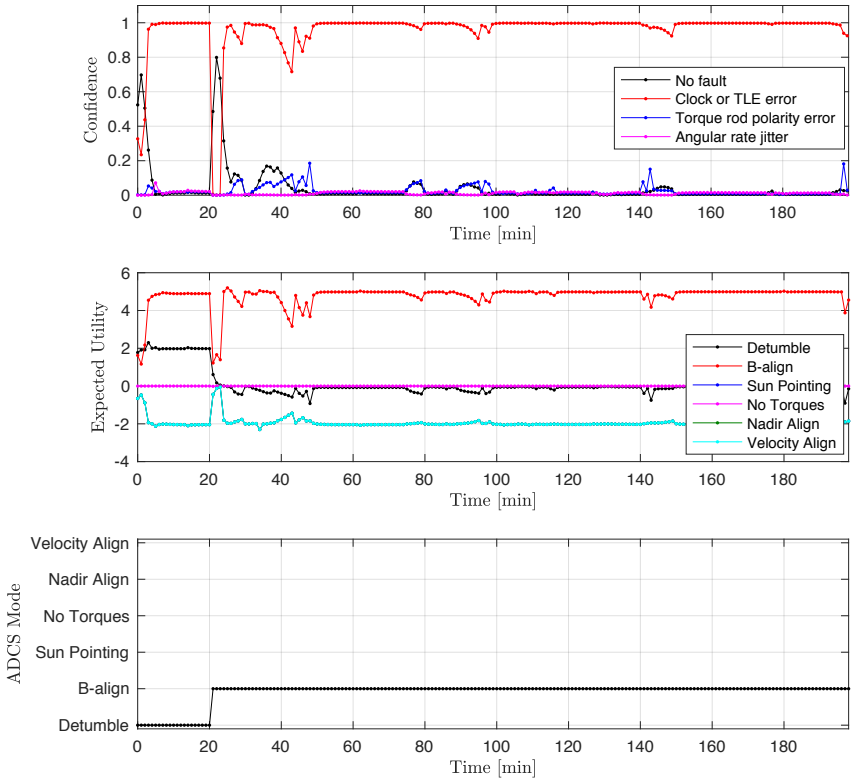


Figure 3.11: Alternative response to a clock error based on new priorities. Compared to Fig. 3.10(a), the utility values for B-alignment mode and detumble are reversed, thus motivating the system to prioritize addressing the fault over the current objective.

[4]A reasonable priority given that a clock error may affect subsequent parts of the mission timeline.

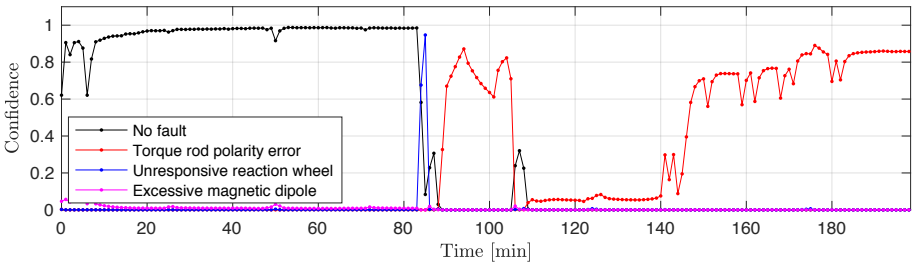
Combination of Faults

The next example considers a particularly insidious combination of two separate faults to demonstrate how decision theory can enact a more sophisticated fault response. Consider a spacecraft in velocity pointing mode. The first fault is a torque rod polarity error on both the X and Y torque rods. This fault can easily lurk undetected in modes with active attitude control because the torquer commands generally average to zero once the reaction wheels have aligned the spacecraft with the commanded attitude. A reaction wheel failure will be injected near the end of the first orbit, disrupting attitude control and causing the decision system to enact a mode change to detumble mode. Yet, due to the torque rod polarity error, this could easily be a fatal mistake. The FDIR system must act quickly to diagnose the polarity error and transition the spacecraft to a more survivable mode. Figure 3.12 shows its response.

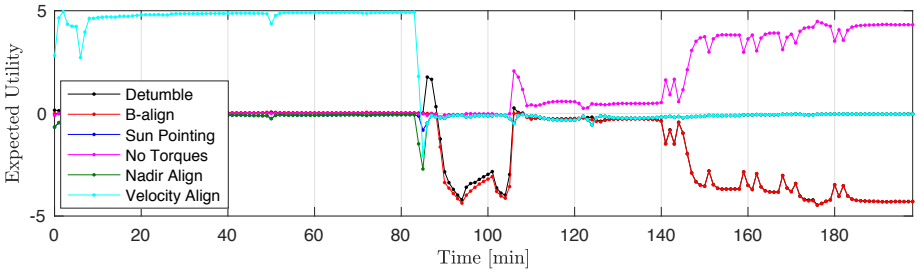
The reaction wheel failure is quickly detected and diagnosed with high confidence immediately after the fault is injected at $t = 83$ minutes in Fig. 3.12(a). The spacecraft switches into detumble mode in response. As the spin rate begins to rise, the LSTM network correctly diagnoses the torque rod polarity error with $60 - 80\%$ confidence. No torques becomes the favored mode to maximize the expected utility and the spacecraft switches as soon as the minimum wait time is reached. Even though the LSTM network is subsequently reset, the excess angular momentum acquired during the brief period in detumble allows the network to recover its prior diagnosis, thus ensuring it remains settled in mode 3.

3.3.4 Overall Performance

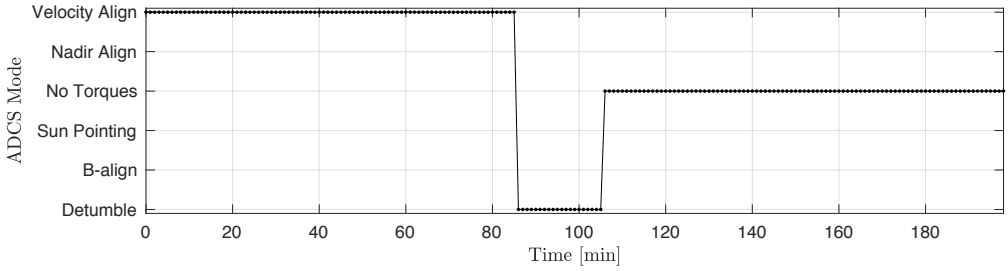
The combination of 22 faults and 6 ADCS modes leads to 132 unique scenarios for selecting a fault response. To gauge the effectiveness of the decision system over the full range of scenarios, Fig. 3.13 summarizes simulations for every one. The chart plots the injected fault against the initial ADCS mode with each cell containing the final



(a) Subset of fault confidences



(b) Expected utilities of ADCS modes



(c) Resulting mode selection

Figure 3.12: Sequence of ADCS mode updates to respond to two faults. The first is a reaction wheel failure that prompts the decision system to abandon velocity pointing mode for detumble mode. This allows the torque rod polarity error to manifest. Smart-FDIR diagnoses the fault and enacts a safe recovery to no torques mode.

mode that the spacecraft settled in to at the end of 2 orbits. The appropriateness of the final mode is measured in terms of utility regret, defined as the difference between the highest utility that could have been obtained for the given fault and the utility of the actual outcome. Where the regret is non-zero, the system could have obtained a higher utility by selecting a different ADCS mode.

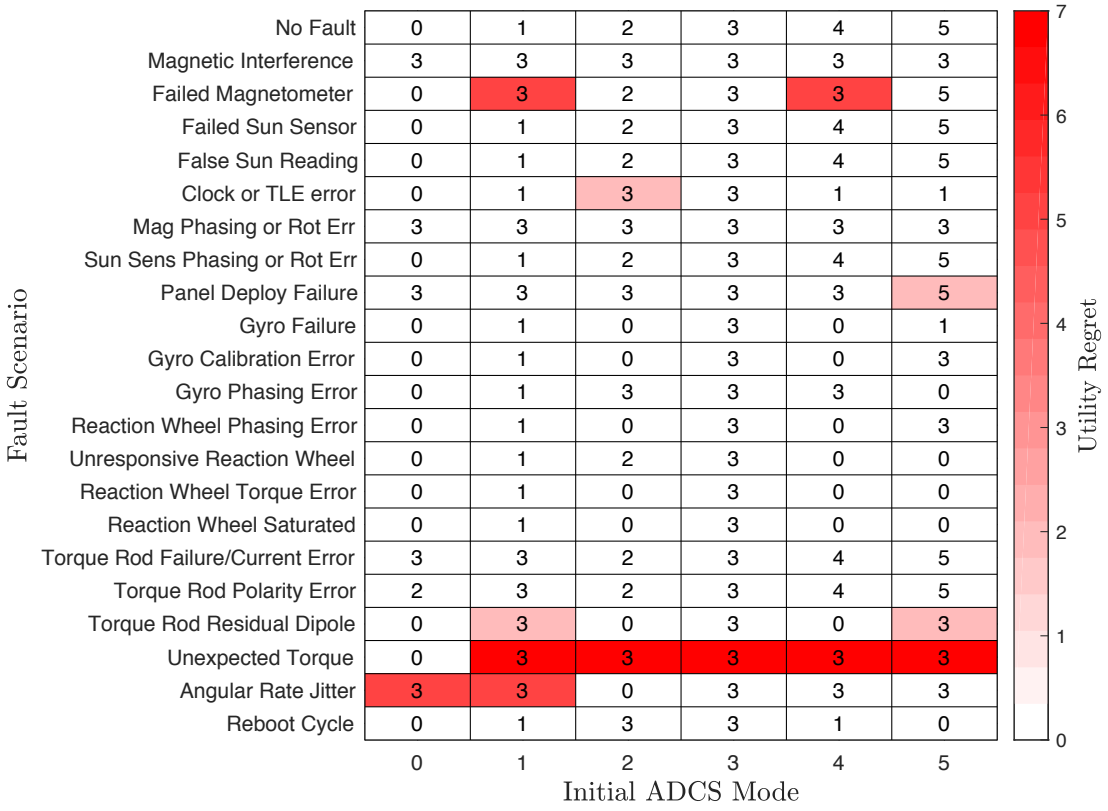


Figure 3.13: Effectiveness of response actions over all fault–mode combinations. Each cell contains the final mode selected to respond to the corresponding fault. The regret measures the appropriateness of each final mode.

In 119 (90%) of the scenarios, the FDIR system selected a mode that yielded the maximum utility. Note from Table 3.5 and Eq. 3.10 that several modes may possess identical utilities for a given fault, meaning that the optimal fault response is often not unique. An important case is the first row (No Fault), since it shows that the system adhered to the objective mode under nominal conditions. The initial mode is

also maintained in numerous cases where the fault does not impact operations (e.g. failed sun sensor, gyro calibration error in modes 0, 1, and 3).

In the 13 cases with regret, the decision system selected a suboptimal action. Suboptimal fault responses result primarily from one of two things:

1. The LSTM network makes an incorrect fault diagnosis
2. The utility values allow an outcome with low confidence to overpower more confident predictions

The second case occurs when the utility values exist on very different orders of magnitude. It can be tempting to assign extremely large negative values to catastrophic outcomes when comparing them to the benefit of a single objective. This is a mistake because it allows the expected utilities of outcomes with low confidence to outweigh more confident ones. Decisions will thus be dominated by diagnoses with virtually zero confidence. We must also note that the confidence outputs of the LSTM network are not equivalent to a rigorous notion of probability. This is especially true when considering very low and very high confidences from the network. For these reasons, the utility values in Table 3.4 are defined within the same order of magnitude.

Instead, the 13 incorrect cases in Fig. 3.13 are due to incorrect fault diagnoses by the LSTM network. Previously (Fig. 3.9), the LSTM's accuracy was scored based on the prevailing diagnosis in the final 30 time steps of each scenario. Yet, as can be seen in the various examples, the LSTM network often trades off several competing hypotheses before it converges to its ultimate diagnosis. When coupled with the decision system, an incorrect diagnosis at even a single time step (after the 20 minute wait) can result in the expected utilities triggering a mode change. This is the case with the failed magnetometer in Fig. 3.13, where a brief allocation of confidence to the competing "Magnetometer phasing/rotation error" fault led to modes 1 and 4 being abandoned unnecessarily. The most troublesome case, however, is the unexpected torque fault. This is not surprising given the low diagnosis accuracy seen in Fig. 3.9.

4. FAULT MONITORING FOR LIGHTSAIL 2

The FDIR technique presented in the prior chapters has thus far been developed and tested exclusively using the 6DOF ADCS fault simulator. To demonstrate that the method is useful in practice, this chapter applies the technique to detect and diagnose faults on the LightSail 2 solar sail satellite, which in 2019 demonstrated the first controlled solar sailing in Earth orbit. The author has been involved with the mission as the flight mechanics engineer from late testing through the first year of operations and completion of the primary mission. Section 1 provides a historical and technical overview of the mission while Section 2 develops the anomaly detectors. Section 3 applies the anomaly detectors along with an LSTM network to perform fault isolation, discovering both known and unknown faults on the spacecraft.

4.1 Mission Background

4.1.1 History of the LightSail Program

The Planetary Society (TPS) is a non-profit and member-supported organization dedicated to stimulating public interest in the peaceful exploration of outer space. Since the early 2000's, the Society has sought to achieve this by advancing the technology of solar sailing. Solar sails are based on the fact that radiation pressure from the Sun exerts a force that can be used to propel a spacecraft without propellant. The concept can be traced back to Johannes Kepler and has appeared in science fiction stories from Jules Verne to Arthur C. Clarke [144]. The possibilities of solar sailing were also popularized by the The Planetary Society's founders and have been investigated in an extensive body of scientific literature [145]. Yet, it was not until the 21st century that the technology would be realized.

In 2005, TPS attempted to fly Cosmos 1, a 100 kg spacecraft with a 600 m^2 solar sail. Regrettably, the satellite was destroyed when the launch vehicle failed to reach orbit [144]. A NASA CubeSat solar sail, NanoSail D, was similarly lost in a launch failure in 2008 [146]. The first solar sail to fly in space was the 306 kg JAXA spacecraft, IKAROS, which demonstrated acceleration and limited attitude control[1] using a 196 m^2 sail while enroute to Venus in 2010 [147]. NASA's 4 kg NanoSail D2 deployed a 10 m^2 solar sail in Earth orbit the following year but did not attempt controlled propulsion [148].

The low mass and miniaturization of CubeSats makes them ideal for solar sailing. Following the loss of Cosmos 1, TPS reorganized their solar sailing objectives and created the LightSail program. The new program began in 2010 and raised $7.5 million USD from 50,000 members and donors worldwide to develop a pair of 3U CubeSats to meet the following objectives:

- Demonstrate controlled solar sail propulsion using a CubeSat platform
- Raise the public and technical profile of solar sailing
- Excite and engage the public
- Share the program results with future missions, the technical community, and the public

Each CubeSat has a mass of 5 kg and carries a 32 m^2 Mylar solar sail. Since the characteristic acceleration of a solar sail depends on the ratio of the mass to sail area, the LightSail satellites have the highest solar acceleration of any solar sail flown to date.

LightSail 1 flew as a test mission in 2015 and was dedicated to performing a check-out of CubeSat platform and validating deployment of the solar sail in space [149]. These were accomplished with successful sail deployment on June 7th, 2015. Due to

[1]The sail orientation was changed a fraction of a degree by varying the reflectance of 80 liquid crystal panels

the low orbit altitude (356 km $\times$ 705 km) however, the spacecraft was quickly pulled from orbit by atmospheric drag and reentered the atmosphere on June 14th, 2015.

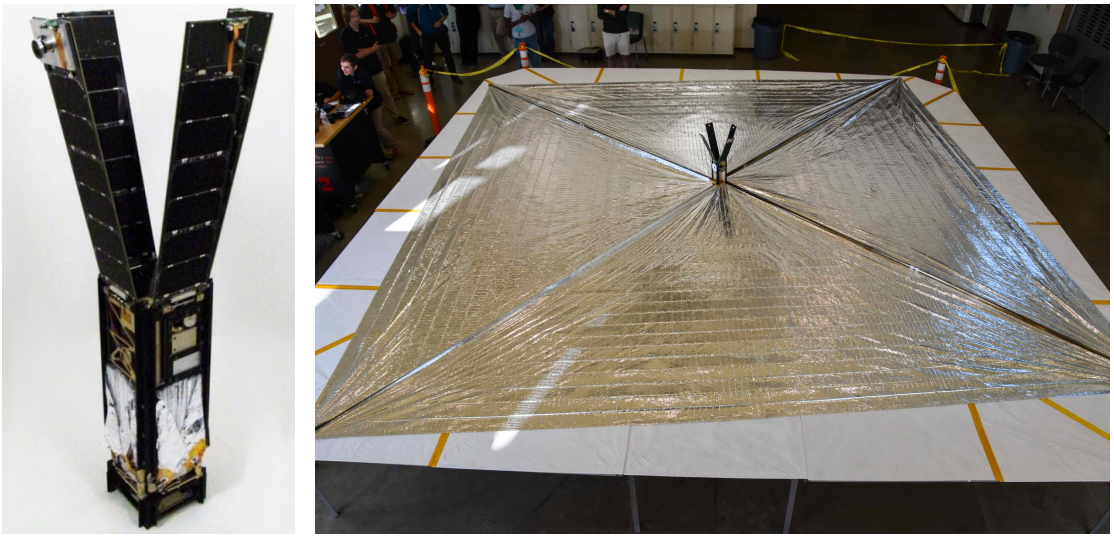
The experience gained from LightSail 1 resulted in a number of hardware and software changes for LightSail 2. Whereas LightSail 1 was uncontrolled, LightSail 2 incorporates a full ADCS that enables it to control its orbit via changing the orientation of the sail relative to the Sun. To better observe the effects of solar pressure, the spacecraft was also carried to a higher orbit (709 km $\times$ 726 km, 24 deg inclination) as part of the SpaceX STP-2 mission launched on June 25th, 2019. Initially contained within another small spacecraft (Prox 1), it was deployed to fly on its own July 3rd, 2019 [135, 150].

4.1.2 Mission Overview

Spacecraft Bus

Figure 4.1 shows the LightSail 2 spacecraft during ground testing in 2016. The spacecraft bus is similar to the one modeled by the 6DOF fault simulator as described in Section 2.3. Both are 3U CubeSats with nearly identical solar panel configurations, definitions of the spacecraft body frame (see Fig. 2.7(b)), as well as mass and inertial properties (when the solar sail is stowed). LightSail 2 does not have a solar panel on the +Z face – it was removed to install a cluster of retro-reflectors for precision orbit determination. The sensors and actuators of the fault simulator are based on the hardware used by LightSail 2, but several differences are important to note. In contrast to the fault simulator, LightSail 2 has only a single reaction wheel actuating about the +Y axis. Additionally, only 2 (+X and +Y) of the original 4 magnetometers were found to be functional prior to launch. LightSail 2 also possesses two sets of gyros. The primary (PIB) gyros are the ADIS16135 model and measure calibrated angular rates about each spacecraft body axis for attitude knowledge and control [138]. To conserve power, the PIB gyros are only operated in conjunction with the reaction

wheel and not in the detumble, B-align, or no torques ADCS modes. In these modes, a secondary set of gyros built in to the spacecraft's Intrepid motherboard is used.



(a) Spacecraft bus (b) Solar sail deployment testing

Figure 4.1: Pre-launch photos of LightSail 2 during 2016. Both show the spacecraft with its solar panels deployed.

LightSail 2's -X and +X solar panels are equipped with custom-made cameras built by the Aerospace Corporation for imaging. The cameras face outwards once the solar panels are deployed and take 2-megapixel color images using a 185 deg fish-eye lens. This FOV is enough to capture the deployed sail along with parts of the Earth and space. The solar sail itself consists of four triangular sheets of Aluminized Mylar®. Each segment is 4.6 microns thick and 5.6 m on a side, giving a total deployed area of 32 m^2. The sail is deployed by four 4 m Triangular Retractable And Collapsible (TRAC) booms wound around a common spindle behind the +Z face of the spacecraft. An electric motor rotates the spindle to extend the booms during sail deployment. Deploying the sail drastically increases the moments of inertia of LightSail 2. Since the spacecraft is equipped with only a single reaction wheel, the

large moments of inertia make attitude control far less precise compared the generic 3U CubeSat considered in Chapters 2 and 3.

LightSail 2's ADCS software operates very similar to the fault simulator described in Section 2.3 and shares many of the same control modes. LightSail 2's ADCS modes are summarized in Table 4.1. Note that mode 5 (velocity pointing) has not been used at the time of writing while mode 4 (Sun pointing) has only been used for a limited number of tests. The most important and most-used mode is mode 2 (solar sailing). The function of this mode is summarized in Fig. 4.2 and described in the following subsection.

Solar Sailing Concept of Operations

The central goal of LightSail 2 is to demonstrate controlled solar sail propulsion. This is complicated by the fact that a solar sail generates thrust any time it is in sunlight, meaning attitude control is not strictly required to modify the orbit. To discern the effects of sail control, LightSail 2's solar sailing mode enacts a cycle of 90 degree "On-Off" slews that switch the sail orientation between edge-on and thrusting attitudes twice per orbit. This control strategy is illustrated in Fig. 4.2. When the spacecraft is moving away from the Sun, LightSail 2 aligns its +Z axis to maximize the solar radiation pressure on the sail and ensure that the projection of the thrust onto the orbital velocity is positive. On the other half of the orbit, the "Off" or edge-on attitude is adopted to minimize thrust from the sail and prevent it from removing energy from the orbit. The "On-Off" control scheme thus allows the solar pressure to contribute an increase in the orbital energy that can oppose losses due to atmospheric drag.

Table 4.1: LightSail 2 attitude control modes.

No.	Mode	Description
0	Detumble	Valid magnetometer readings are averaged in the spacecraft body frame and basic B-dot control [143] is used to generate torque rod commands to arrest the spacecraft's rotation. The reaction wheel is not used in this mode.
1	Magnetic alignment	Similar to detumble except that the Z-axis torque rod is set to constant maximum power. This aligns the spacecraft Z-axis (with some precession) to the local magnetic field vector.
2	Solar sailing	The PIB gyros and reaction wheel are powered on and LightSail 2 slews between thrusting and edge-on attitudes relative to the Sun as described in Fig. 4.2.
3	No torques	All actuators are disabled.
4	Sun pointing	All actuators are used to point the spacecraft's –Z panel towards the Sun.
5	Velocity pointing	All actuators are used to align the spacecraft Z-axis with the velocity vector.

4.1.3 Mission Events

Launch and Early Operations

Lift-off of the STP-2 mission occurred at 2:30 am EDT from Launch Complex 39A at NASA's Kennedy Space Center, Florida. The mission was the first night launch of SpaceX's Falcon Heavy rocket and delivered a payload of 24 satellites to various orbits using four separate upper stage burns [151]. The launch was attended by the

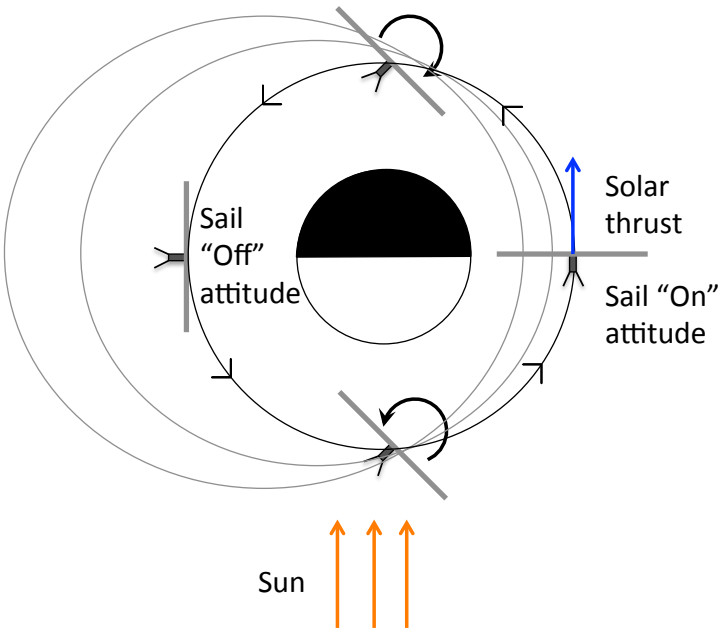
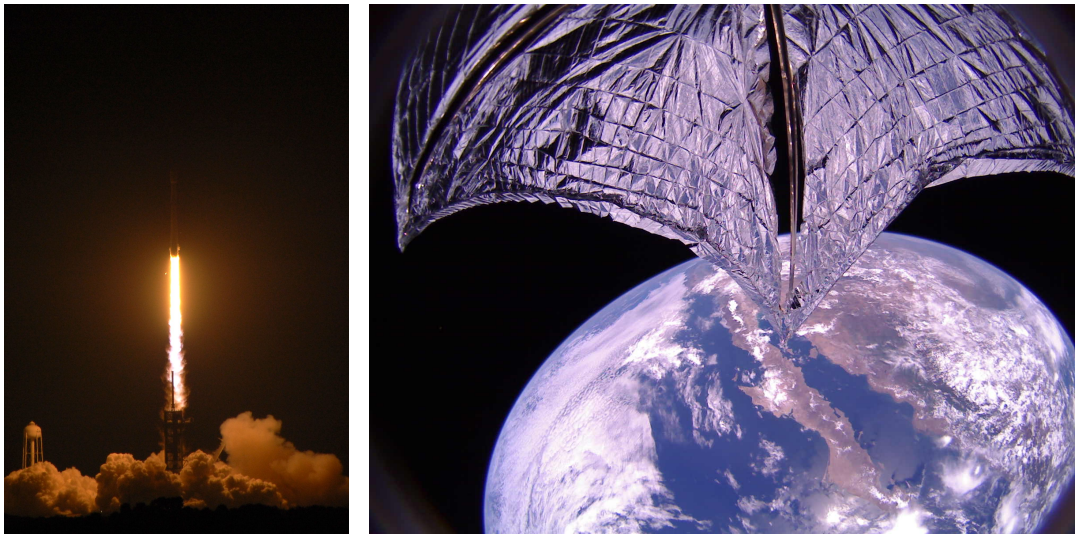


Figure 4.2: Solar sailing strategy for LightSail 2. When moving away from the Sun the spacecraft presents the maximum sail area to the Sun to deliver an increase in the orbital energy. On the opposite arc, LightSail 2 transitions to an edge-on attitude to minimize solar pressure and prevent the sail from removing energy.

author and a personal photograph is included in Fig. 4.3(a). LightSail 2 was stored within a P-POD on Prox 1 and the pair was deployed into a 709 km × 726 km orbit at 3:49 am EDT. Table 4.2 summarizes the timeline of major events for LightSail 2 that followed in the weeks after the launch.

Prox 1 autonomously deployed LightSail 2 exactly 7 days after its separation from the Falcon upper stage. Following a 45 minute charging period, the spacecraft's Morse code identifier was received exactly on time at 4:34 am, July 2nd. The first full telemetry beacons were decoded later that day and the influx of detailed subsystem information marked beginning of an extensive checkout period.

ADCS testing began in earnest on July 6 following the deployment of LightSail 2's solar panels. This exposed the sensors to the space environment and revealed that the two magnetometers were providing inconsistent measurements. Using mode 1



(a) Launch, 25 June 2019 (b) Sail deployment, 23 July 2019

Figure 4.3: Launch and on-orbit deployment of LightSail 2's solar sail. Figure (a) was taken by the author on the night of the launch.

(B-alignment) to provide a known direction of the local magnetic field relative to the spacecraft (approximately aligned with the +Z axis), the +Y magnetometer was deemed to be faulty since it showed the field in the -Z direction. Note that the spacecraft had not detumbled at this point and the Intrepid gyros showed rates between 2−3 deg/s. A number of software and parameter updates pertaining to the torque rods and B-dot controller were uplinked, and on July 13 the detumble mode successfully reduced the rotation rate below 1 deg/s.

Pre-deployment testing of mode 2 (solar sailing) began on July 14 and was initially unsuccessful. Errors with sun sensor measurement processing, sun sensor voting, eclipse checking, and onboard orbit propagation were corrected between July 14 and 17. Gain tuning and software updates to the reaction wheel finally resulted in a successful demonstration of mode 2 on July 22. The sail was deployed the following day from a B-aligned attitude over Central America at 2:47 pm EDT. Fig. 4.3(b) shows the sail towards the end of deployment. The spacecraft transitioned to solar

Table 4.2: LightSail 2 major mission events.

Date	ADCS Events/Testing
June 25	Launch on STP-2 Falcon Heavy
July 2	Deployment from Prox 1
July 6	Momentum wheel test and solar panel deployment
July 8	+Y magnetometer taken offline, successful mode 1 alignment
July 9–13	Multiple parameter and software updates
July 13	Successful detumble
July 14–17	Sensor software updates
July 14–22	Mode 2 testing, control parameter tuning and software updates
July 22	Successful mode 2 test
July 23	Solar sail deployment
July 29	Orbit changes confirmed; mission success announced

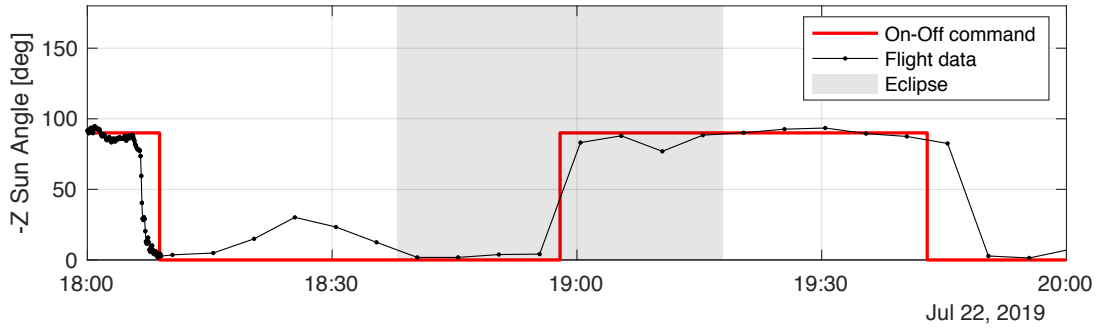
sailing mode immediately after deployment. Changes in apogee and perigee on the order of several hundred meters per day along with the first recognizable On-Off slews became visible in downlinked telemetry in the days afterwards. Mission success was announced on July 29, 2019.

Solar Sailing Performance

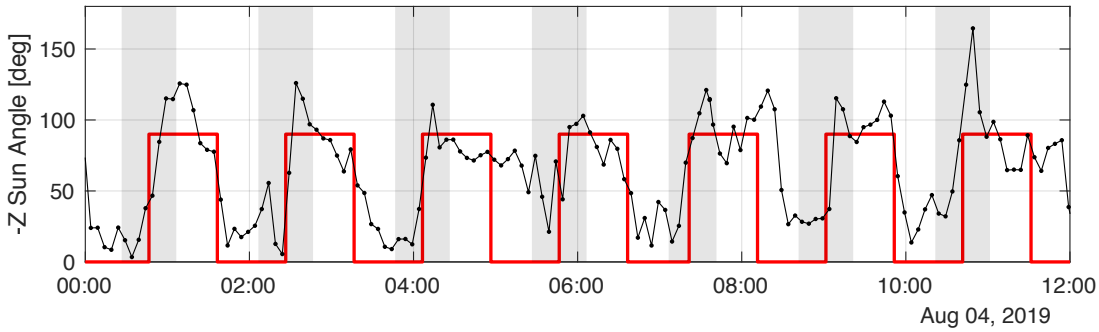
During the final mode 2 test prior to sail deployment, LightSail 2's attitude control accuracy was consistent with the performance seen in the 6DOF fault simulator. The spacecraft maneuvered quickly between On and Off attitudes and was able to maintain the commanded attitude generally to within 15 degrees. A plot of the -Z to Sun angle during the test is shown in Fig. 4.4(a).

The On-Off cycle continued after sail deployment but proved difficult to maintain. The reaction wheel saturated within a matter of hours and several desaturation strate-

gies were tested to find a balance between solar sailing and momentum dumping that kept the wheel RPM within an operable range. Numerous control gain adjustments were also iterated on, as it was found that gains tuned using the ADCS Simulink model did not necessarily provide comparable performance on-orbit (possibly due to the flexibility of the real sail). To further enhance spacecraft control and improve the effectiveness of momentum dumping, mode 1 (B-align) was effectively eliminated on August 1st. This allowed the power of the torque rods to be increased to allow dipoles of up to 1.2 Am^2. By August 4th, LightSail 2's On-Off performance had been refined to the point where it was consistent with the simulation model. Fig. 4.4(b) shows an example of the spacecraft's On-Off performance with the sail deployed.



(a) The final pre-deployment solar sailing test



(b) Sailing using the On-Off strategy with the sail deployed

Figure 4.4: Examples of LightSail 2's solar sailing performance.

With the exception of a handful of dates in which LightSail 2's orbital energy increased, the spacecraft's semi-major axis has decreased throughout the mission

due to atmospheric drag. However, an analysis of the orbit decay rate between late July and November 2019 showed that the rate was demonstrably reduced by at least 10 m/day when solar sailing with regular momentum dumping [135].

4.1.4 On-orbit Anomalies

Various anomalies were encountered by LightSail 2 both during the checkout phase as well as later operations. Many of these either originated in or otherwise affected the spacecraft's ADCS. The following subsections describe the most significant anomalies that have been investigated so far during the mission. These anomalies will form the basis for demonstrating and assessing autonomous fault detection and isolation in Section 4.3.

+Y Panel Deployment

Among the first anomalies identified during the mission were the implausible measurement readings of the +Y magnetometer. The sensor was placed in passive mode to prevent it from corrupting the spacecraft's attitude knowledge. In September, an analysis of sun sensor voting statistics revealed that the +Y sun sensor was being rejected nearly twice as often as its sister sensors. This led to speculation that the anomalies of the +Y sun sensor and magnetometer were related. These suspicions were confirmed in January when an image was returned showing the shadows of the solar panels on the sail. Pictured in Fig. 4.5, the shadow of the +Y panel reveals that this panel is deployed approximately orthogonal to the spacecraft bus instead of the nominal 155−165 degrees of the other panels (see also Fig. 4.1(a)). Thus, the root cause of the +Y sensor anomalies was not that these sensors were faulty, but that the coordinate transforms used to interpret their measurements did not reflect the actual orientation of the panel on which they were mounted. Detailed investigation determined the +Y panel to be deployed 92±6 degrees from the spacecraft bus.

New rotation matrices were derived and the +Y sun sensor and magnetometer were restored to active mode.

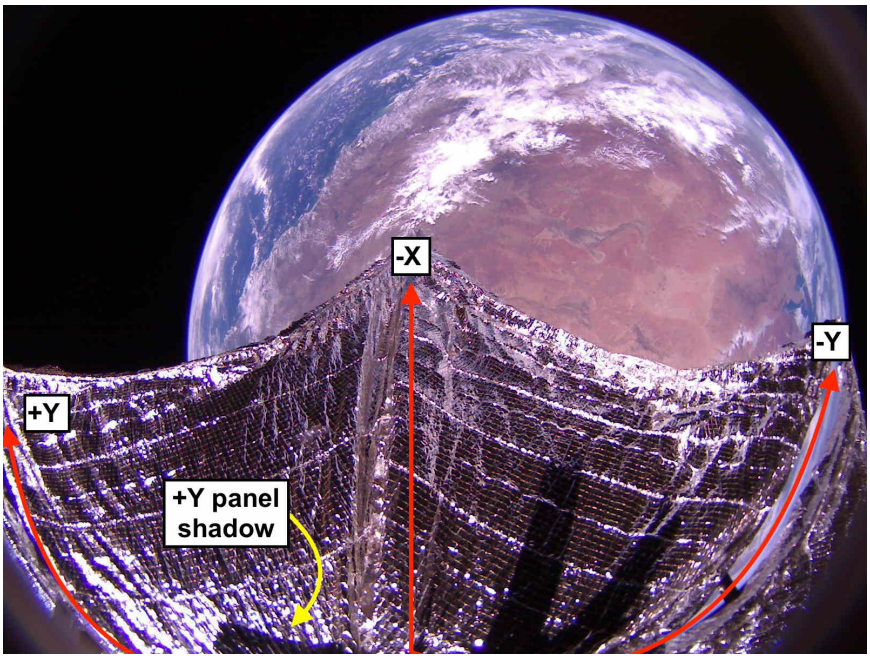


Figure 4.5: In this image taken 15 January 2020, the shadows of the deployed solar panels revealed that the +Y panel was only partially deployed.

Intrepid Gyro Calibration

Another anomaly identified early on was a miscalibration of the Intrepid board gyros. This was detected by applying the quaternion-B discrepancy check described in Fig. 3.4(a) to validate the spacecraft's attitude knowledge. When in solar sailing mode with the PIB gyros active, the quaternion-B discrepancy averaged around 15 degrees. In other modes, however, attitude knowledge was severely disrupted. Figure 4.6 shows the discrepancy angle across several mode changes. A closer inspection of the readings from the Intrepid gyros showed that the Z-axis measurement was implausibly large, being in the range of $2-6$ deg/sec. Attempts were made to cross-calibrate the Intrepid gyros against their PIB counterparts but this was unsuccessful because

the offset was found to vary significantly over time. Fortunately, the impact of the unreliable Intrepid gyros on the mission has been minimal due to the fact the PIB gyros are used in modes were explicit attitude knowledge is required.

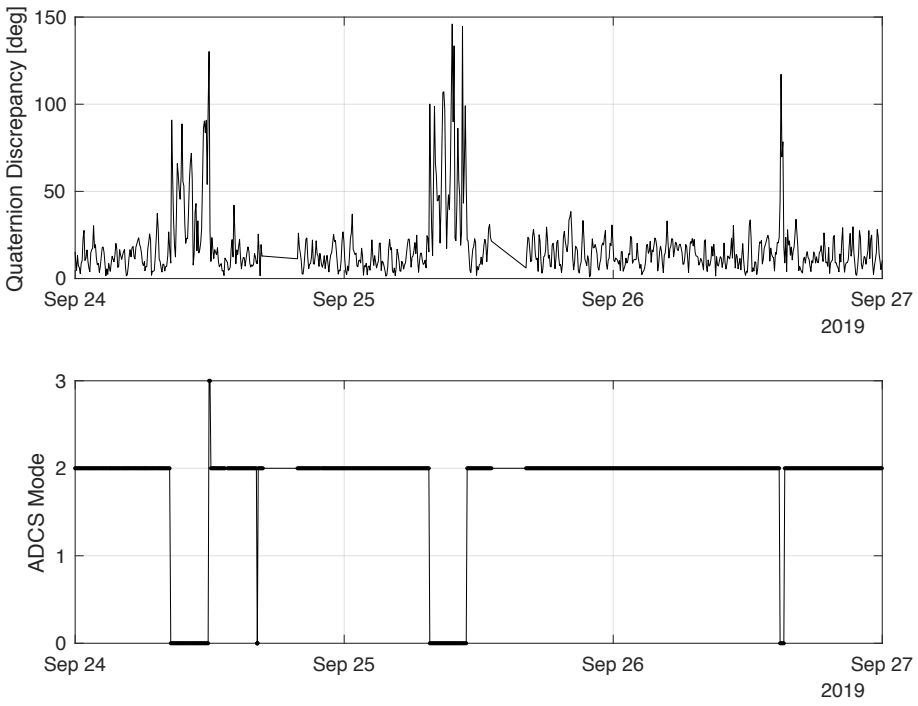


Figure 4.6: LightSail 2 quaternion-B discrepancy angle across several mode changes. While attitude knowledge is accurate in modes that utilize the PIB gyros, the poorly calibrated Intrepid gyros disrupt attitude determination in others.

Reaction Wheel Anomalies

The initial mode 2 tests prior to solar sail deployment failed due to a lack of responsiveness from the reaction wheel. Despite non-zero torque commands, the wheel's RPM remained at 0 except for sporadic pulses as shown in Fig. 4.7. Given the high amount of chatter in the commands, it was theorized that the signal was simply too incoherent for the wheel's motor to follow. A moving average filter was instituted to smooth the commands over a 10-second window, but the wheel proved

even less responsive in the subsequent test. This helped reveal that the magnitudes of the commands were too small to engage the wheel from 0 RPM. Attitude control gains were made more aggressive for the next test so that the commanded torques during slews were $> 4 \times 10^{-4}$ Nm. This succeeded in engaging the wheel for sustained periods and revealed a final software error that was preventing the wheel from reversing direction. After correcting this error, the final pre-deployment solar sailing test was successful (Fig. 4.4(a)).

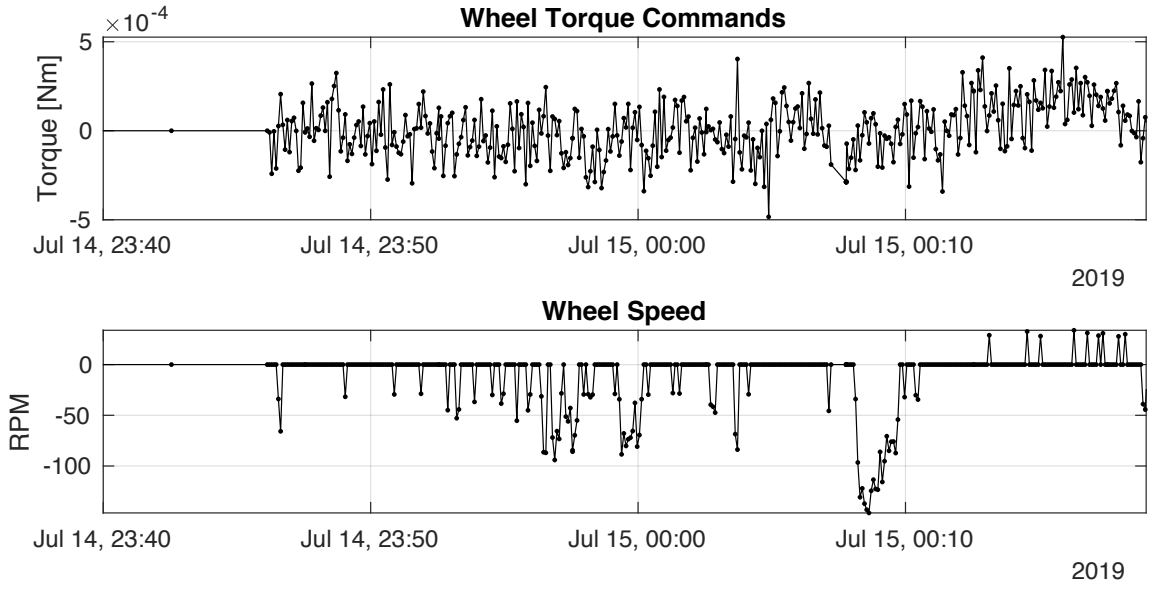


Figure 4.7: Reaction wheel torque commands and RPM response during initial mode 2 testing. Weak commands and a software error made engaging the wheel difficult.

Shadowing and ADCS Resets

A frequent problem that appeared beginning in mid-August were cyclic flight computer resets that impacted ADCS performance. The resets tended to follow one of two patterns:

- A "sawtooth" pattern with resets occurring regularly every 215 minutes

- Resets occurring at a rapid but irregular interval

Examples of each are shown in Fig. 4.8. The cause of the sawtooth-type was traced to corrupted databases on the spacecraft. Data corruption often occurs following a hard reset, such as those caused by a loss of power. The corruption is detected automatically on board and attempts are made every hour to repair it. If unsuccessful, the repairs can be attempted at most three times before a watchdog timer reboots the spacecraft. This is why the reboots in the top plot of Fig. 4.8 occur at regular intervals. When this pattern of resets manifests, cycling the databases via a ground command is usually able to correct the issue.

The second pattern of resets correlates with low battery voltages and is likely the result of electrical brownouts. These resets most often arise during extended periods of momentum dumping. This fact was initially perplexing because the detumble mode draws significantly less power than mode 2. Nevertheless, the uncontrolled attitude induced during momentum dumping leads to regular shadowing of the solar panels by the sail. If the attitude remains uncontrolled for many hours, the spacecraft's rotation tends to stabilize about its Z-axis. Once this occurs, it is essentially a 50/50 draw as to whether the solar panels are on the sunlit side of the sail.[2]

This is precisely what happened in late October after a memory overload crashed the ADCS software and forced the spacecraft into a tumbling attitude for several days until memory could be freed. The panel-shadowing attitude into which LightSail 2 stabilized was indicated by the hot (> 70 deg C) temperatures on the +Z panel and the cold (< -30 deg C) temperatures on the -Z panel. The resulting onslaught of hard reboots triggered two secondary anomalies that took over 2 months to fully resolve:

- The loss of power caused the spacecraft's clock to reset to its date of manufacture in 2010.

[2]Recall that there are no solar cells on the +Z face of the spacecraft.

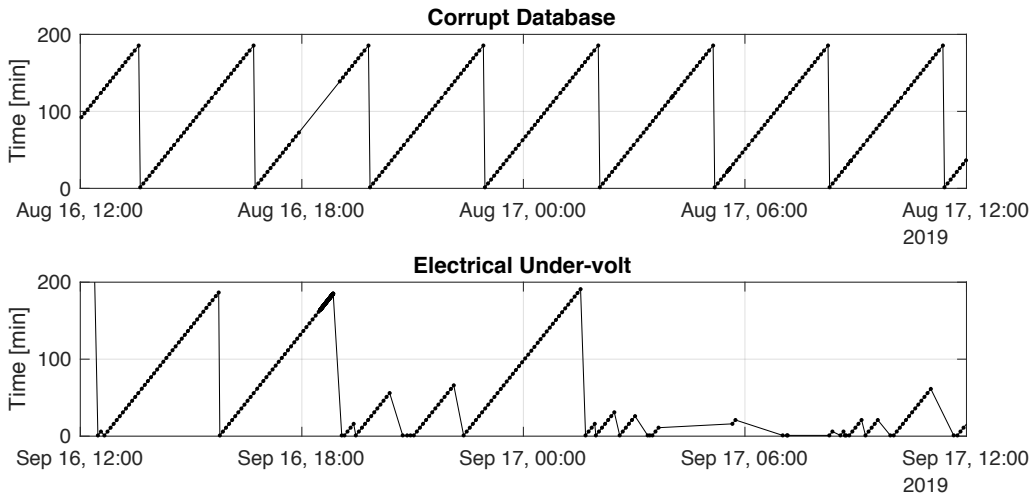


Figure 4.8: Patterns of flight computer resets from two common causes. The time is the time since the last reboot.

- While re-uplinking ADCS configuration files, the active sensor flags were set to their pre-launch configuration. Since the partial +Y panel deployment had yet to be identified, this caused misinterpreted measurements from the +Y magnetometer and sun sensor to be ingested into the attitude filter. Attitude knowledge was thus degraded until the incorrect flags were found and corrected.

Fortunately, LightSail 2 was eventually able to accept a transition into mode 2 and slew to a safe attitude to recharge its batteries. Various updates have since been made to the spacecraft to improve the power budget and reduce the chance of the spacecraft becoming stuck in an uncontrolled attitude for extended periods of time.

Magnetometer Glitches

Though the initial anomalies with the +Y magnetometer were ultimately due to the partially deployed +Y solar panel, there have also been anomalies with the sensors themselves. These have taken the form of glitches during which one of the magnetometers outputs a constant "stuck" measurement for all three of its axes.

Figure 4.9 provides an example from the x-axis channel of the +Y magnetometer. Other examples have affected the +X magnetometer. In every case so far the anomaly is cleared by a reboot. The glitches are extremely rare and were in fact unknown prior to their diagnosis by Smart-FDIR in Section 4.3. Their underlying cause remains unknown and may be radiation related.

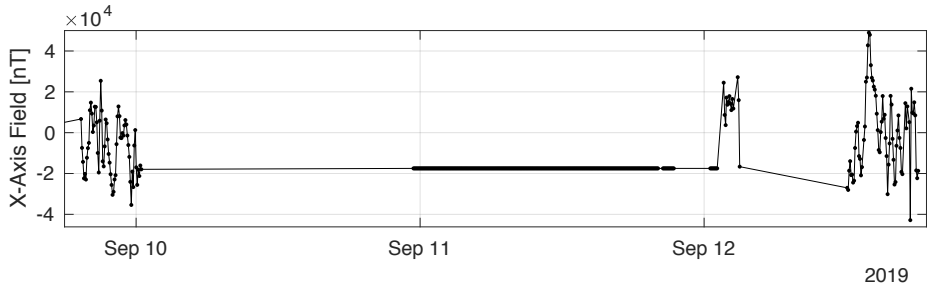


Figure 4.9: Stuck values in the +Y magnetometer. Glitches like this occurred occasionally during the mission and went undetected prior to discovery by Smart-FDIR.

Other Anomalies

Numerous other anomalies and faults that had an indirect effect on ADCS were investigated during the LightSail 2 mission. One such anomaly was with the sail deployment itself. About halfway along the -Y axis in Fig. 4.5 is metallic artifact. This is thought to be a TRAC boom that buckled during sail deployment. The resulting asymmetry of the sail may explain why the reaction wheel has saturated almost exclusively in the negative RPM direction throughout the mission.

A more serious fault mentioned briefly in the previous subsections involved LightSail 2's memory. The spacecraft stores telemetry files and images on its onboard computer and these files need to be deleted periodically to keep the computer's memory free. Three separate times during the mission, communication with the spacecraft was disrupted long enough for the memory to fill up completely. When this happens, ADCS is unable to initialize a critical configuration file and crashes. A complex set of

commands needs to be uplinked each time to clear memory and restore the files necessary for ADCS to operate. This carries the risk of operator errors (see Shadowing and ADCS Resets subsection).

Finally, LightSail 2's sun sensors do not record the intensity of the light source whose direction they are measuring. This makes them vulnerable to detecting non-solar objects such as the Moon or limb of the Earth. While this is partially mitigated by sensor voting, analysis of the sun sensor discrepancy (Section 3.1.3) suggested that the sun sensors were locking on to a non-solar object almost half of the time. Though the overall effect on LightSail's attitude knowledge was minimal (as indicated by the low quaternion-B discrepancy), it was eventually decided to place all sun sensors in passive mode and rely on the two magnetometers.

4.2 LightSail 2 Anomaly Detection

4.2.1 LigthSail 2 Telemetry Dataset

There are two sources of telemetry from LightSail 2: beacons and stored telemetry. Beacons are broadcast every 7 seconds and can be received when the spacecraft is being tracked by one of the three stations: Cal Poly in San Luis Obispo, Purdue, or Georgia Tech. Each beacon packet contains essential subsystem data sampled at the time the beacon was broadcast. Thus, the resulting telemetry covers occasional 10–15 minute periods when the spacecraft was passing over the continental United States, something that only about a third of the orbits do.

Telemetry from all of LightSail 2's orbits is stored in onboard logs. These files are regularly downlinked alongside the beacon packets and contain the full catalog of over 300 telemetry signals and flags. The data is nominally sampled at a rate of 1/300 Hz, but this is sometimes increased to the high rate of 1/5 Hz. Each file contains 10–20 samples of each signal at the low rate or up to 350 samples at the high rate. To date, more than 1800 telemetry files (1.5 GB) have been downlinked.

Using the Data

Some pre-processing is needed to make LightSail 2's stored telemetry fit for anomaly detection. Duplicate samples, missing data, and conversion of raw data into meaningful units must all be dealt with appropriately. Individual signals are not always sampled at consistent times and the sample rate can vary between the high and low rates even within the same file. The files also do not span the entirety of the mission from beginning to present. Even from early in the mission there remain gaps in the dataset where the corresponding files were not downlinked due to the finite data rate and need to downlink other mission products (e.g. images). Since each file begins where the previous one left off, it is useful to merge the time series extracted from successive files so that each contains the longest period of uninterrupted data between gaps in the files.

The details of this conditioning are not worthy of discussion here except to note the characteristics of the final processed dataset:

1. All telemetry is resampled at the low rate (1/300 Hz), interpolating where necessary using a zero order hold

2. Telemetry segments span periods that are uninterrupted by either missing telemetry files or ADCS mode changes

Both the resampling and splitting of time series about ADCS mode changes are driven by the needs of the LSTM network described in Section 4.3.

LightSail 2's telemetry files generally do not provide intermediate data internal to ADCS. An example is the results of sun sensor voting. However, the intermediate signals can often be reconstructed a-posteriori from the raw data that is available in the file. In the case of sun sensor voting, the raw measurements can be converted into the spacecraft body frame using the sensor rotation matrices and compared, ignoring sensors known to be invalid. Because the reconstruction depends on external information like the sensor valid flags, this introduces the possibility of representing more faults than actually occurred in flight. For example, using the rotation matrix

for the partially deployed +Y panel to process the magnetometers while assuming a fully deployed panel for the sun sensors can effectively simulate a sun sensor failure, even though such a fault never occurred. This trick will be used in Section 4.3 to create additional examples for testing the LSTM network's ability to diagnose sun sensor and magnetometer faults.

4.2.2 Detectors

The following subsections catalogue the suite of OCSVM anomaly detectors that have been developed from the LightSail 2 telemetry data. The availability of certain signals precludes an identical set of anomaly detectors to the fault simulator version in Section 3.1.2. For one, the telemetry files do not contain the torque rod dipole commands. The torque rod currents are also unreliable and often read zero even when the torque rod is actuating. This is due to the torque rods operating on a 700-ms on/300-ms off duty cycle to avoid interfering with the magnetometer readings. If the sample happens to fall within the 300-ms portion, the currents will read zero even if the rod actuated during the 700-ms portion. Because of this, it is not possible to create an effective anomaly detector for the torque rods. On the other hand, LightSail 2 regularly encountered solar panel shadowing due to the spacecraft's large sail and this fault was not applicable to the more generic CubeSat described in Section 2.3. Detecting shadowing anomalies requires an entirely new type of anomaly detector.

Detectors Similar to Fault Simulator

Several of the OCSVMs developed in Section 3.1.2 are still relevant to LightSail 2. Figure 4.10 presents four detectors analogous to those developed for the fault simulator. Since there is only a single reaction wheel on LightSail 2, the torquing error is visualized in 2-dimensions. The nominal data for each detector was selected from several dates between August and mid-October during which solar sailing was most successful. Figure 4.10(b) also includes data through to December, since the mea-

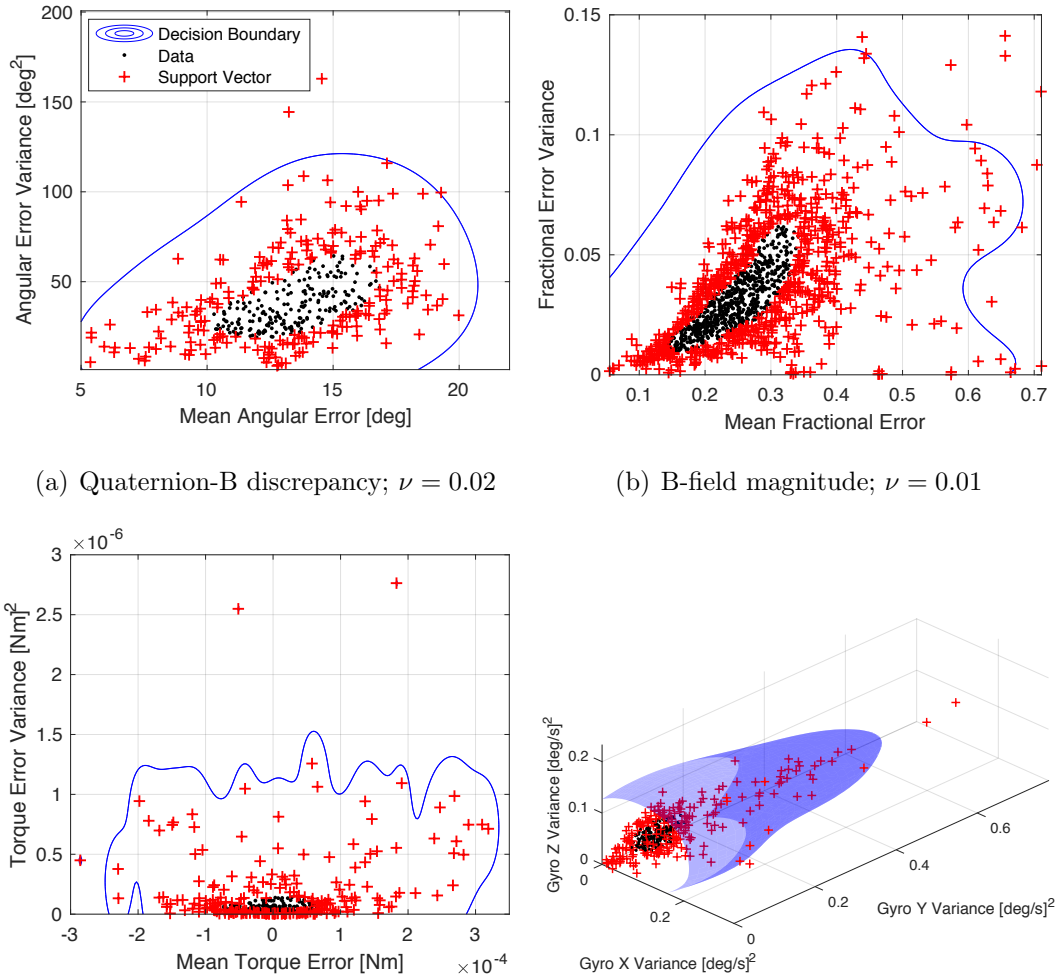
sured B-field magnitude was unaffected by the faults occurring at that time. This illustrates an important nuance: nominal data can be obtained on a signal-by-signal basis. It is not necessary for all signals to be nominal at the same time to acquire useful data for training OCSVMs.

The sliding windows used to extract the means and variances are each 20 time steps (i.e. 100 minutes $\approx$ 1 orbit) wide and were slid along the signals with steps of 10 time steps. Figure 4.10(b) contains 1365 total data points (689 support vectors) while Fig. 4.10(a)-4.10(c) each have 470 data points (240 support vectors). Outlier fractions ν were selected to give reasonable decision boundaries. The OCSVM technique of allowing a certain fraction of outliers when solving for the anomaly boundary reveals itself as a feature here. Consider the two support vectors near the top of the plot in Fig. 4.10(c). When working with real data, it can be difficult to guarantee that all of the training data is in fact nominal. If we were to insist that the decision boundary enclose all the data, anomalies such as those in Fig. 4.10(c) would dramatically skew the boundary and degrade sensitivity to detecting anomalies. Thus, the method is robust to anomalies that may be accidentally ingested into the nominal training set.

In addition to the four OCSVMs in Fig. 4.10, Section 3.1.3 provides anomaly checks that are also relevant to LightSail 2. Anomaly rules 2–5 can be applied without any modifications to LightSail 2. Rule 1 is less applicable because the large inertia of LightSail 2's sail means that the spacecraft is unlikely to exceed a 2 deg/sec rotation rate unless left uncontrolled for several weeks. Rule 6 is also less relevant due to the spacecraft's low pointing accuracy, even when operating nominally.

Alternate Sun Sensor Voting Anomaly Detector

The sun sensor anomaly detector developed in Section 3.1.2 is difficult to apply to LightSail 2. In its original form, the OCSVM assumes a Sun pointing attitude. Even though LightSail 2 has a Sun pointing ADCS mode (mode 4), this mode was only introduced in mid-December 2019 and has only seen very limited use since then.



(a) Quaternion-B discrepancy; $\nu = 0.02$

(b) B-field magnitude; $\nu = 0.01$

(c) Wheel torque errors; $\nu = 0.001$

(d) Gyro variances; $\nu = 0.02$

Figure 4.10: Anomaly detectors similar to the fault simulator version. The OCSVMs are based on nominal telemetry from LightSail 2. The elongated variances of (d) along the Y axis are due to this axis being the slew axis.

To detect sun sensor anomalies, it is necessary to reformulate the detector so that it functions in detumble, no torques, or solar sailing mode.

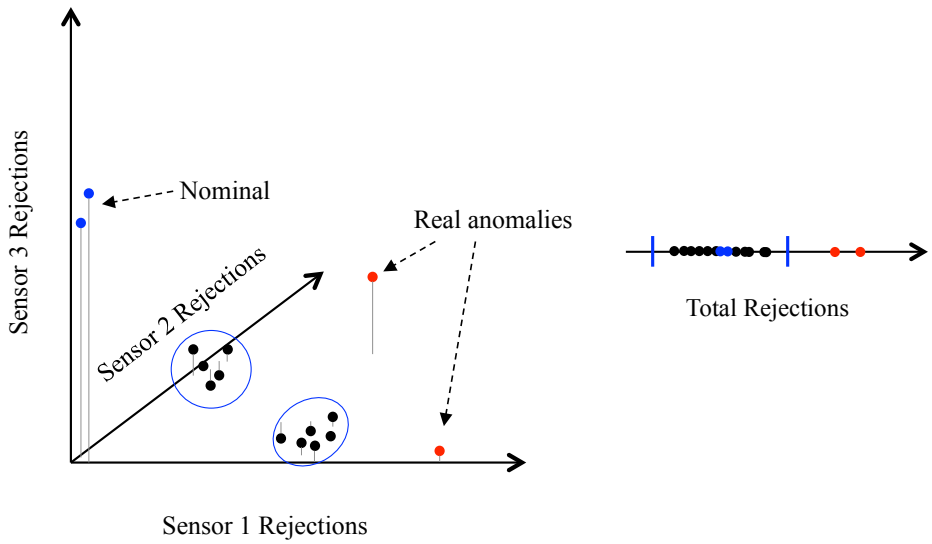
Instead of formulating the OCSVM input vector as a 5×1 vector containing the number of times each individual was rejected during the sliding window, we will form a 3×1 vector with components:

$$X = \begin{bmatrix} \#\text{ of times multiple sensors rejected} \\ \#\text{ of times one sensor rejected} \\ \#\text{ of times no sensors rejected} \end{bmatrix}$$

Formulating the input this way reduces the number of dimensions at the cost that we can no longer narrow down anomalies to specific sensors. The lower number of dimensions works to concentrate the nominal training data and reduce overfitting of the anomaly boundary. Figure 4.11 provides an illustration. Instead of representing individual sensors as in (a), focusing on the total number of rejections in (b) allows the decision boundary to more completely enclose the nominal data. The downside is that anomalies can no longer be traced to a specific sensor without further investigation.

This alternative sun sensor voting OCSVM has another advantage: we can ignore or include specific sensors in the voting without having to retrain the OCSVM. Whereas ignoring a sensor would previously reduce the input vector from 5×1 to 4×1, the new 3×1 representation allows us to easily toggle the valid flags of one or more sun sensors. This is useful because it allows us to train the OCSVM on voting data that ignores the anomalous +Y sun sensor but then apply it to data that includes the +Y sensor if we want to attempt to detect the anomaly.

Figure 4.12 shows the OCSVM trained from voting data extracted with a sliding window 40 time steps long. The axes show the number of instances that one or more sensors were rejected during the last two orbits (approximately 40 samples). The window steps are 10 steps long as before. The longer window helps average out the occasional bad orbit where a single sensor may never have the Sun in its FOV. Nominal data is drawn from both mode 0 and mode 2 during dates when solar sailing was most successful. The +Y sensor is ignored when reconstructing the nominal



(a) (b)

Figure 4.11: Given a limited amount of training data, reducing the number of dimensions can help concentrate the data and reduce overfitting of the anomaly boundary.

voting data. In total there are 546 data points and 287 support vectors. The large number of "multiple rejected" cases is due to a combination of eclipse periods and times when none of the sensors were facing the Sun. Times where only a single sensor was rejected are comparatively rare.

Magnetometer Discrepancy

Unlike the CubeSat in Section 2.3, LightSail 2 does not have enough functional magnetometers to perform voting. In place of voting, we construct an intermediate signal which is the angle between the two magnetometer B-vectors in the spacecraft body frame. We call this the magnetometer discrepancy. Similar to other signals, a 20-sample sliding window is used to create a OCSVM for the mean and variance of the signal. Nominal data was reconstructed by using the correct (partially deployed)

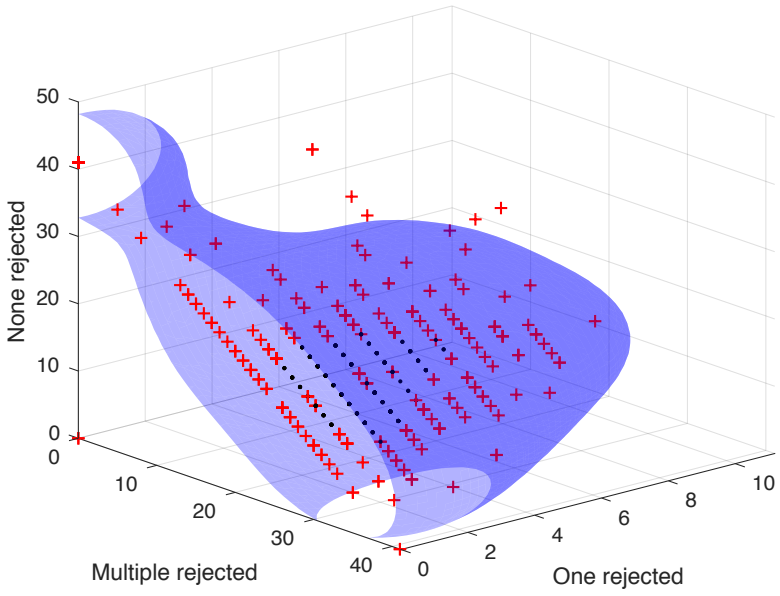


Figure 4.12: Anomaly detector for sun sensor voting on LightSail 2 with $\nu = 0.02$.

rotation matrix for the +Y magnetometer. The result is shown in Fig. 4.13. There are 452 total data points and 229 support vectors.

This anomaly detector serves a similar purpose to the voting anomaly detectors developed thus far. It is sensitive to disagreements between the two magnetometers that may be due to a failed sensor, interference, or a measurement processing error.

Solar Power Anomaly Detector

The final OCSVM for LightSail 2 considers solar power generation. While this would normally fall under the domain of the electrical subsystem rather than ADCS, it is relevant as an indication of whether the solar sail may be shadowing the solar panels from the Sun. The solar power can be found by multiplying the voltages and currents from each solar cell and summing the results. Rather than the mean and variance of this signal, we extract the peak power and the integrated power (measured in Watt-hours) from each 40-sample (10-sample shifts) sliding window. Figure 4.14 plots the data and associated OCSVM. Nominal data comes from carefully screened

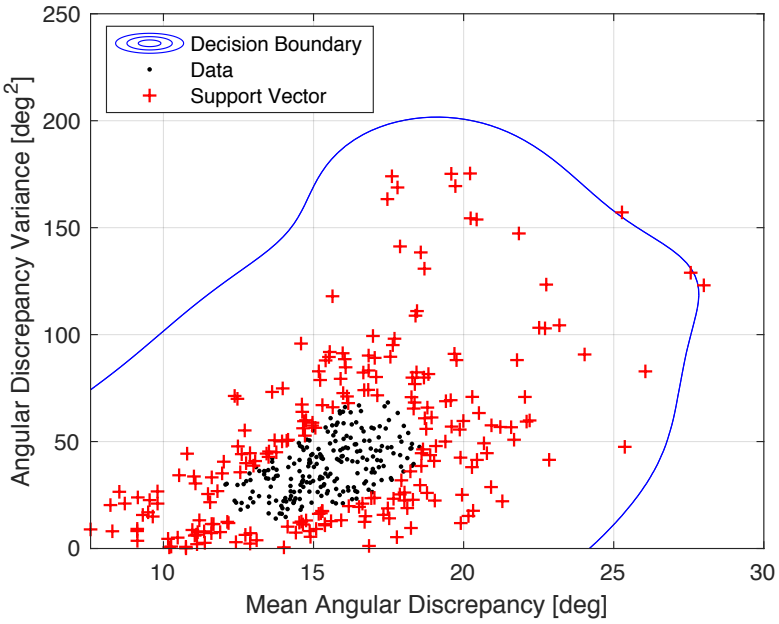


Figure 4.13: Anomaly detector for the directional discrepancy between LightSail 2's magnetometers; $\nu = 0.02$.

periods of both mode 0 and mode 2 around dates of successful solar sailing. There are 239 total data points and 123 support vectors.

4.3 LightSail 2 Fault Isolation

4.3.1 Fault Simulations

Given the retinue of anomaly detectors trained on nominal LightSail 2 telemetry, the task of fault isolation can now be addressed. A version of the fault simulator was created to represent LightSail 2 instead of the generic CubeSat used in Chapter 3, the main changes being the solar sail, reduced number of magnetometers and reaction wheels, and the ADCS modes. Twins of the anomaly detectors described in Section 4.2.2 were also trained from nominal data generated by the LightSail fault simulator. The parameter ranges for generating the nominal data are given in Ta-

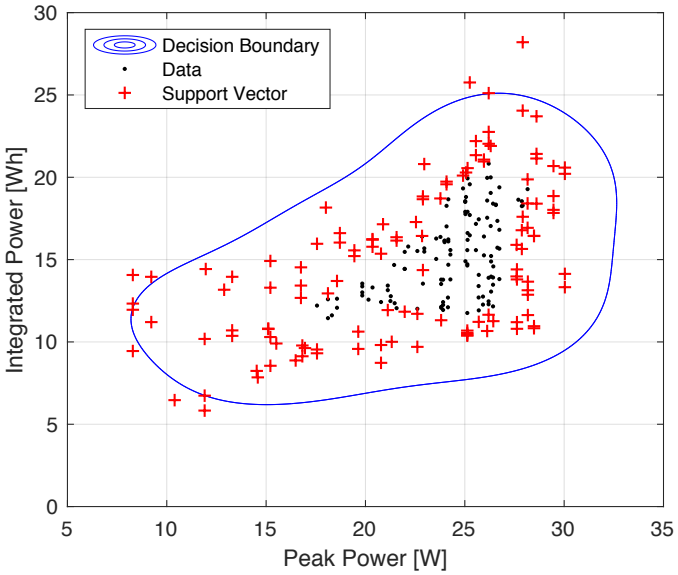


Figure 4.14: Anomaly detector for the total and peak solar power generated by LightSail 2 over 2 orbits; $\nu = 0.05$.

ble 4.3. The fault simulator anomaly detectors allow high-level anomaly patterns to be abstracted from the simulated fault cases to follow.

Table 4.3: Simulation parameters for generating fault isolation datasets for LightSail 2.

Parameter	Values
Date	Randomized between Jan. 1, 2019 – Dec. 31, 2020
Orbit	Random circular orbits between 600 – 720 km altitude at 24 deg inclination
Mode	Cycled through 0,2,3
Number of orbits	4 – 6
Initial quaternion	Random (-1,1) for each component, then normalized
Initial angular rate	Random (-0.1,0.1) deg/sec for each axis

Table 4.4 describes the fault scenarios simulated for LightSail 2. Faults are injected between 0 and 5000 seconds into each simulation and the ADCS mode (either 0, 2, or 3)[3] is randomized among those applicable to the given fault. Other parameters defining the simulation are randomized from Table 4.3.

Not all of the faults applied to the generic CubeSat in Chapter 3 (Table 3.3) are applicable to LightSail 2. The reduced number of magnetometers makes it difficult to distinguish a failed magnetometer from erroneous rotation matrices. The lack of Sun pointing mode and a simplified sun sensor voting OCSVM has similar implications for the sun sensors. Thus, the phasing and rotation matrix errors have been removed. The lack of an anomaly detector for torque rod telemetry also necessitates the removal of torque rod faults. The false sun fault was removed after preliminary analysis showed that the sun sensors were locking on to a non-solar object almost 50% of the time, resulting in a lack of nominal data for validating fault isolation performance. Lastly, LightSail 2's large moment of inertia, noisy gyros, and single reaction wheel makes it challenging to assess the overall angular momentum of the spacecraft. Faults such as the unexpected torque, excessive dipole, and torque rod polarity error have therefore been removed.

On the other hand, several new faults have been introduced. The reboot cycle has been split into the corrupt database fault and the electrical brownout fault in order to distinguish the two. The solar panel shadowing fault has also been introduced by initializing the satellite in an attitude that points the +Z axis towards the Sun. To represent the fact that this often leads to a brownout, the electrical brownout fault is also triggered some time into the simulation.

Table 4.4.: Simulated ADCS faults for LightSail 2 LSTM training.

ID	**Fault Scenario**	**Applicable Modes**
0	**Nominal**: No faults.	0,2,3

[3]Recall that mode 1 was effectively eliminated shortly after sail deployment

1	**Magnetometer interference**: Magnetometer noise means shifted by up to 5×10^{-5} T and variance increased by up to 1×10^{-8} T^2.	0,2,3
2	**Failed magnetometer**: As in Table 3.3.	0,2,3
3	**Failed sun sensor**: As in Table 3.3.	0,2
4	**Orbit ephemeris or clock error**: As in Table 3.3.	0,2,3
5	**Failed gyro**: As in Table 3.3.	0,2,3
6	**Gyro calibration error**: gyro rate measurements are biased by a random constant value of up to 2 deg/sec.	0,2,3
7	**Attitude rate jitter**: As in Table 3.3.	0,2,3
8	**Solar panel shadowing**: The initial attitude aligns the +Z axis with the Sun vector such that the sail shadows all solar panels. An additional 0.2 to 1.2 deg/sec rotation is added to the Z-axis rotation to maintain this orientation. The electrical brownout fault (below) is also triggered after $0 - 3$ orbits.	0,3
9	**Electrical brownout**: ADCS resets are triggered randomly with a probability of 0.2% each second.	0,2,3
10	**Corrupted database**: ADCS resets are triggered every 43 counts of the LDC.	0,2,3
11	**Unresponsive reaction wheel**: As in Table 3.3.	2
12	**Reaction wheel torque error**: As in Table 3.3.	2
13	**Reaction wheel saturated**: The reaction wheel's initial speed is set at the saturation limit.	2
14	**Panel deployment fault**: As in Table 3.3.	0,2

4.3.2 LSTM Training

The LightSail 2 fault simulator was used to run 500 simulations of the fault scenarios in Table 4.4 (approximately 33 for each fault). All 500 simulations were used to train a 20-cell LSTM network with the following inputs:

1. Long duration counter
2. Sun sensor anomaly
3. B-field magnitude error anomaly
4. Eclipse-power discrepancy
5. Quaternion error anomaly
6. Magnetometer discrepancy anomaly
7. Zero rate
8. Gyro variance anomaly
9. Power anomaly
10. Reaction wheel torque anomaly
11. Reaction wheels responding
12. Reaction wheels saturated

The output channels of the network are the 15 possible fault scenarios. Note that the LDC is capped at 100. The LSTM network operates with time steps of 300 seconds to be consistent with LightSail 2's telemetry sampling frequency. Since each training simulation utilizes a constant ADCS mode, it is important that the LSTM network is not propagated across mode changes when used on flight data. This is why the telemetry segments have been split around mode changes as described in Section 4.2.1.

The target confidence outputs are normally set based on the time and identity of the fault injected. As in Chapter 3, it is crucial to ensure that each fault manifests as an anomaly in the LSTM inputs in order to fairly judge the accuracy of the network's fault confidence. Prior to training, the target fault flags in the training set were adjusted based on whether the injected fault was detectable. These adjustments are described below:

- The reaction wheel saturation target confidence is set to 1 only if the saturation anomaly flag was raised for at least 20% of the previous orbit. Since LightSail 2

requires large RPM changes to maneuver, the wheel speed often crosses in and out of the saturation threshold at a high frequency. We are mainly interested in periods where the wheel speed persists above the saturation threshold.

- If a sun sensor failure is injected, the corresponding confidence target is elevated only where the sun sensor voting anomaly score exceeds 0.5. This ensures the LSTM network is only judged on its ability to isolate the fault when there is a sun sensor anomaly. However, note that a sun sensor voting anomaly alone does not indicate a sun sensor fault. It could be a false positive or due to another fault.

- If a TLE/clock error is injected, the corresponding confidence target is elevated only after the first time the eclipse-power discrepancy flag is raised. Like the above, this helps ensure the LSTM's confidence assignments are grounded in evidence.

- If a panel deployment fault is injected but the sun sensor voting anomaly never exceeds 0.5, the example is removed from the training set. It is important to remove such cases because they are often indistinguishable from a magnetometer fault.

With the corrected training set in hand, the LSTM network was allotted 20 memory cells and was trained until the MSE loss function plateaued. This required approximately 1 minute on a typical laptop computer. The subsequent section applies the trained network in combination with the anomaly detectors developed in Section 4.2.2 to assess its performance at diagnosing faults in flight data.

4.3.3 Examples

Magnetometer Glitch

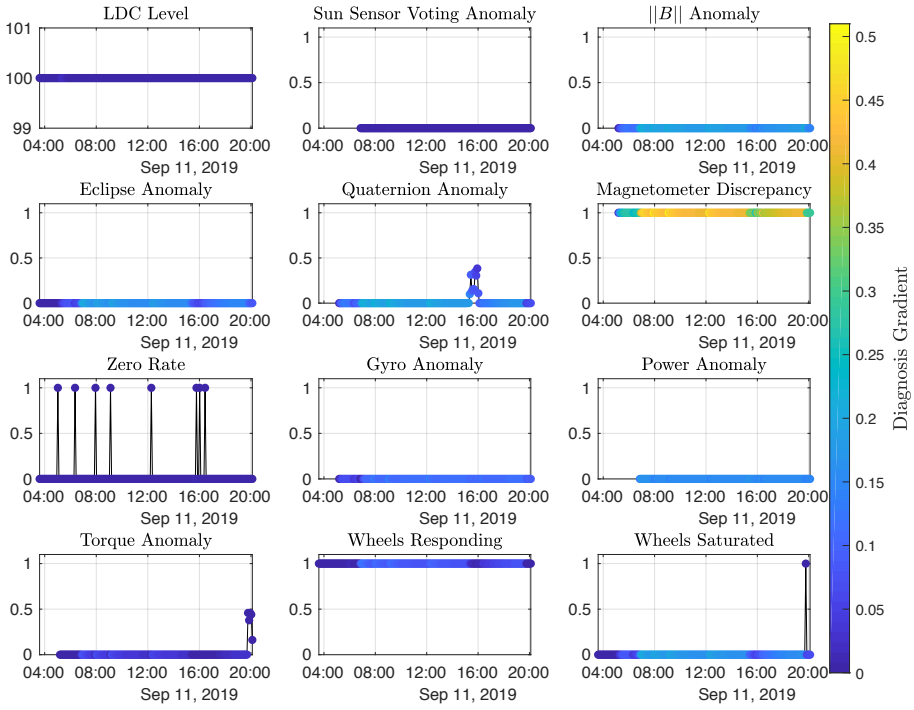
An important demonstration of the value of Smart-FDIR is its ability to reveal previously unknown glitches in the magnetometers. Showcased in Fig. 4.9, the glitches

are characterized by stuck values in the magnetometer measurements. Though temporary, they have the potential to disrupt the accuracy of the spacecraft's attitude knowledge.

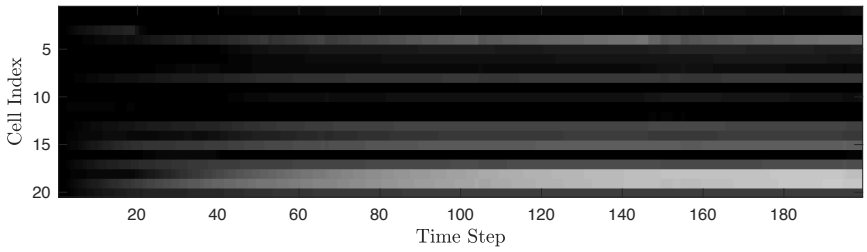
Figure 4.15 applies the anomaly detectors and LSTM network to the portion of telemetry surrounding the glitch in Fig. 4.9. Note that the correct rotation matrix for the +Y magnetometer is used. The final outputs of the LSTM network correctly isolate the magnetometer fault with 80% confidence. To help justify this diagnosis, the gradient with respect to the magnetometer fault sensibly highlights the anomaly raised by the OCSVM monitoring the discrepancy between the two magnetometers in Fig. 4.15(a). Before converging to the magnetometer fault, however, the LSTM network elevates the possibility of a panel deployment fault to around 50% confidence. This is a reasonable deduction because prior to around 07:00 the sun sensor voting OCSVM has not accumulated enough data to produce an anomaly score. Once the scores become available and a sun sensor anomaly fails to materialize, the panel deployment fault is rejected in favor of the correct diagnosis.

Orbit Propagation Error and Solar Panel Shadowing

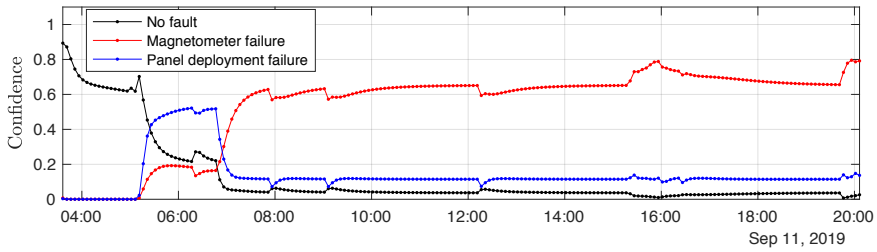
The next examples shows how Smart-FDIR may have prevented the most perilous cascade of faults that occurred during the LightSail 2 mission. Figure 4.16 captures the moment following the memory overload described in Section 4.1.4 when LightSail 2 entered into an attitude in which the sail shadowed the solar panels during an extended period in detumble mode. Though telemetry was still being recorded by ADCS, the onboard orbit propagator had already crashed at this point. Evidence of the incorrectly predicted spacecraft position emerges just prior to 18:00 in Fig. 4.16(a) when the eclipse flag contradicts the small amount of solar power being generated. This small amount of solar power triggers the power anomaly discrepancy, which feeds through the LSTM network to the corresponding shadowing fault. To add another fault to the backdrop, attitude knowledge in detumble mode relies on the



(a) Anomaly inputs



(b) LSTM cell state evolution



(c) Fault confidence outputs

Figure 4.15: Anomalies and associated fault confidences when isolating a previously unknown magnetometer glitch. Highlighting the magnetometer discrepancy and distributing confidence between reasonable hypotheses demonstrates key strengths of Smart-FDIR.

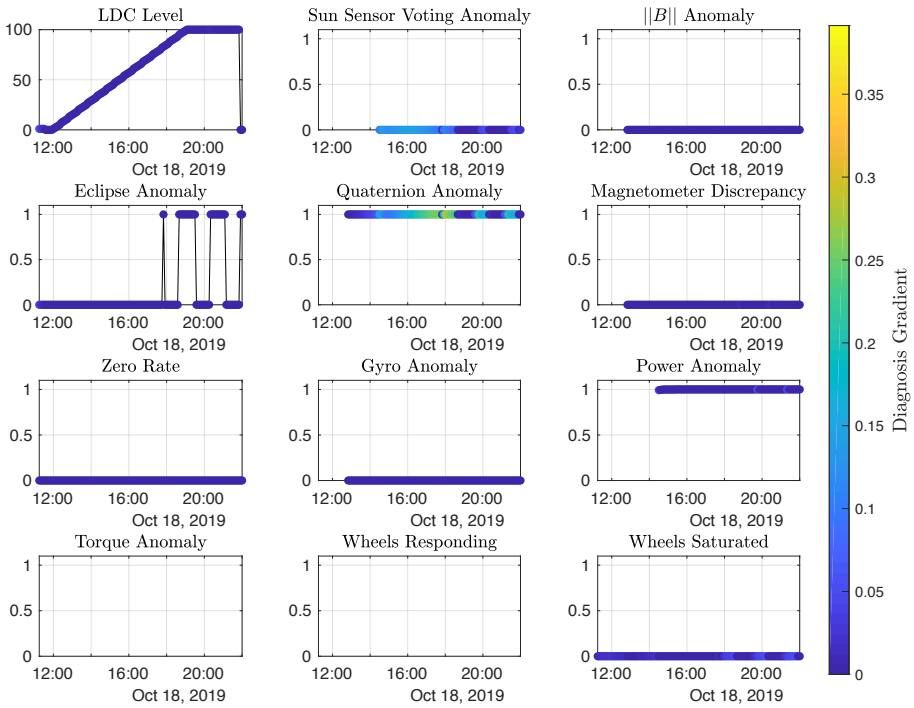
mis-calibrated Intrepid gyros. The resulting impact on the quaternion-B discrepancy is visible in Fig. 4.16(a).

Fig. 4.16(c) provides the LSTM network's fault confidences for the nominal case and the 3 most confident fault hypotheses. Nominal confidence drops and the shadowing fault quickly rises to full confidence as soon as the as the solar power anomaly scores become available around 14:00. Also visible in Fig. 4.16(c) is the rising confidence in the gyro calibration error. The inputs in Fig. 4.16(a) are highlighted according to the gradient of this fault and show that the rising confidence is driven by the persistent quaternion-B anomaly that appears to have no other cause (such as a magnetometer discrepancy or gyro variance anomaly). Whereas the LSTM network was trained to isolate the solar panel shadowing simultaneously with other faults, the fault training examples never combined simultaneous gyro calibration and TLE/Clock errors. Thus, once evidence arises for the TLE error around 18:00, the two faults tend to compete with each other for confidence. Nonetheless, the LSTM network is successful at correctly identifying no less than 3 faults that occurred simultaneously on LightSail 2.

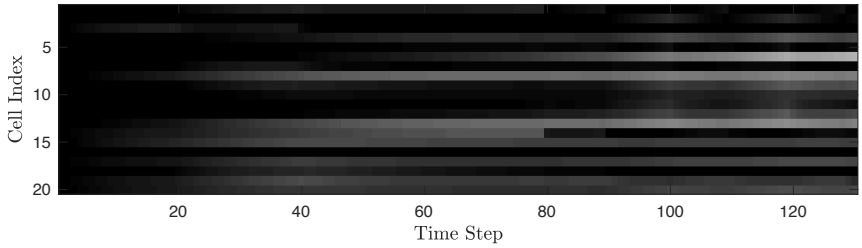
A reset in the LDC is visible towards the end of the telemetry in Fig. 4.16(a). This was the first in a sequence of rapid reboots due to electrical brownouts caused by the shadowed solar panels. Had Smart-FDIR been available to alert the flight team to the impending danger, it may have accelerated efforts to recover the satellite before the onset of the brownouts.

Partial Solar Panel Deployment

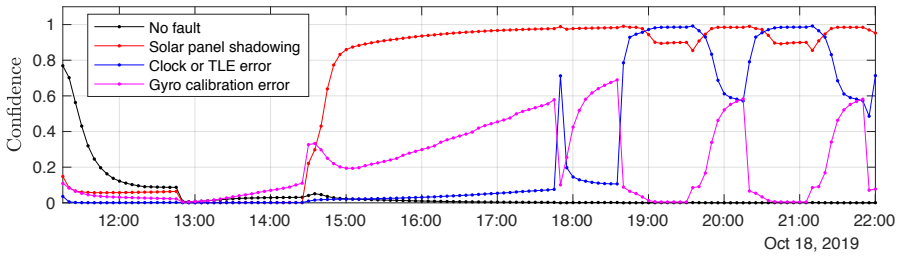
Next we investigate the partially deployed +Y solar panel. Figure 4.17 presents an example from late November when both the +Y magnetometer and +Y sun sensor were being erroneously ingested into the attitude determination filter. Unlike the previous two examples, the original (incorrect) rotation matrices for the +Y sensors are used and the +Y sun sensor is incorporated in voting. This reflects LightSail 2's



(a) Anomaly inputs



(b) LSTM cell state evolution



(c) Fault confidence outputs

Figure 4.16: Anomalies and fault confidences when LightSail 2 became stuck in an attitude with low power input. The LSTM network succeeds in diagnosing the three separate faults affecting the spacecraft.

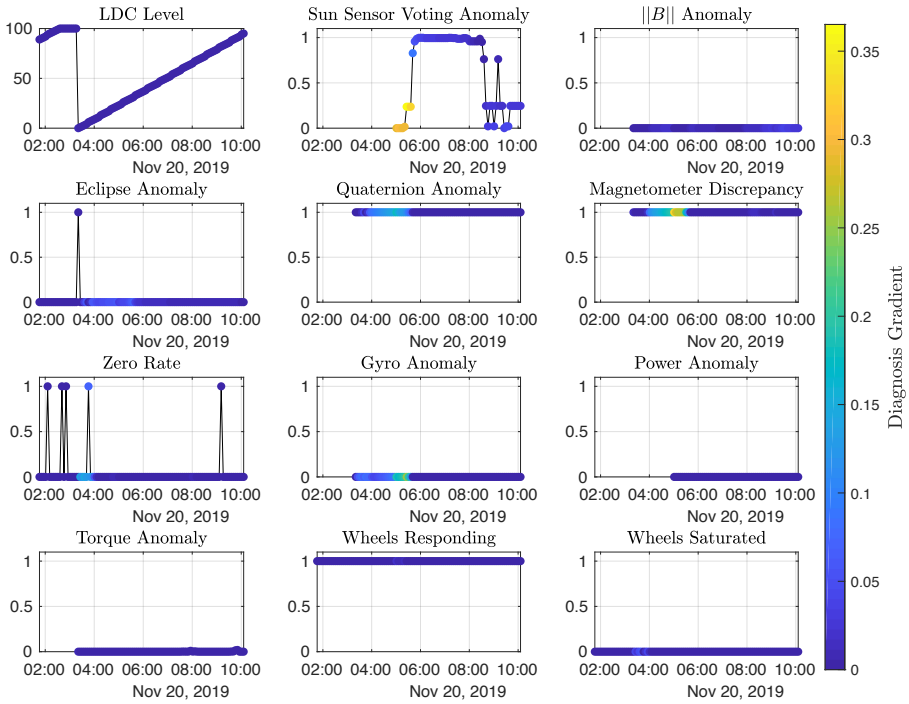
ADCS during the fault. The result was that attitude knowledge averaged around 40 degrees in error rather than the more typical 20 degrees. This discrepancy triggers the quaternion-B anomaly detector in Fig 4.17(a). A magnetometer discrepancy anomaly also arises from its OCSVM. When the sun sensor voting anomaly score becomes available at 05:00 the scores are initially nominal and confidence in the panel deployment fault plateaus at around 35%. The sun sensor anomaly score soon rises followed by a corresponding leap in confidence in the panel deployment fault. The gradient reveals the sun sensor voting, magnetometer discrepancy, gyro variance, and quaternion anomalies to be the most significant inputs. Indeed, all are logical signals to focus on to understand the diagnosis.

Even though the sun sensor voting anomaly begins to disappear towards the end of the data, the LSTM network retains confidence in the panel deployment fault. A similar case is evident in Fig. 4.16(c) where confidence in the TLE error remains above 50% (despite some interference from the gyro calibration fault) between periods where the eclipse anomaly is triggered. This "memory" of anomalies that may no longer be present when computing confidence scores is one of the features of LSTM networks and not something that non-recurrent neural networks can do.

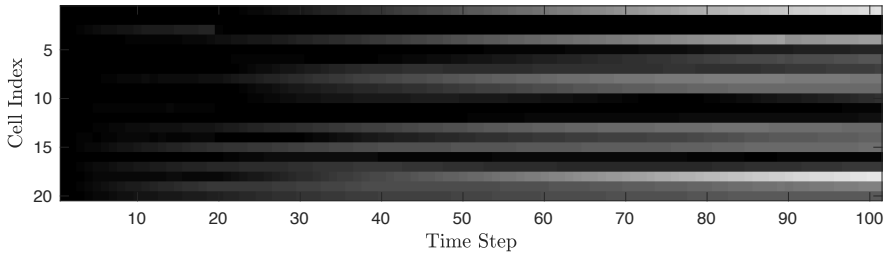
The next most confident fault to occur in Fig. 4.17(c) is the electrical brownout. This diagnosis is driven by the reset occurring around 03:00 in the telemetry. Confidence rises to around 65% and decreases afterwards as the LDC increments without further resets. It is likely that this one-off reboot was due not to a brownout but a radiation-induced upset of the flight computer. Since radiation-induced faults were not modeled by the LSTM network, the electrical brownout diagnosis represents a sensible best guess.

4.3.4 Overall Performance

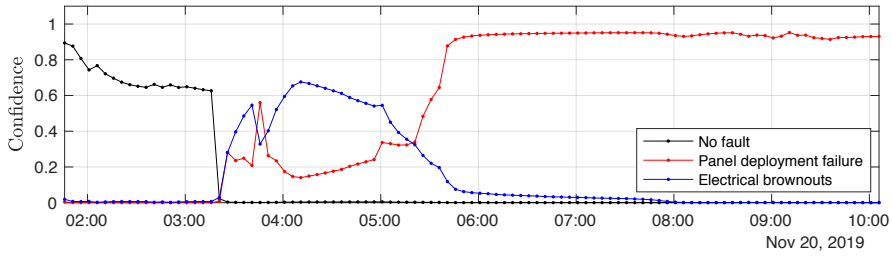
The previous section showed some of the most interesting LightSail 2 fault diagnoses achieved with Smart-FDIR. To assess its performance more widely, we will



(a) Anomaly inputs



(b) LSTM cell state evolution



(c) Fault confidence outputs

Figure 4.17: Diagnosing the partially deployed +Y panel on LightSail 2. The LSTM network retains the correct diagnosis even once the sun sensor anomaly subsides.

now apply the method to all of the LightSail 2 telemetry from spacecraft checkout through to the end of 2019. This period represents the most complete and continuous portion of the downlinked data. In 2020, the COVID-19 pandemic and a lightning strike on the Purdue tracking station impacted operations and significantly reduced the amount of data being downlinked.

To perform the assessment, all the files from July 9, 2019 to December 31, 2019 have been processed into individual telemetry segments according to technique outlined in Section 4.2.1. Segments with less than 40 times steps (200 minutes) have been removed because these are shorter than the time needed for several of the OCSVMs to accumulate enough data to calculate an anomaly score.

Since some faults (e.g. sun sensor failures) depend on how intermediate data is reconstructed from the downlinked files, it is necessary to decide which rotation matrices and sensor valid flags to use when reconstructing sun sensor voting and magnetometer discrepancies. For periods when the +Y magnetometer was being incorporated into the attitude filter, it is important to use the original (165 deg deployment angle) rotation matrix for this sensor since this was ultimately the cause of the degraded attitude knowledge affecting these periods. For segments when the +Y magnetometer was passive, we can use the correct (92 deg deployment angle) matrix despite the fact that this correction was not yet onboard the spacecraft. If it had been, there would be no effect on other ADCS signals since the +Y measurements were not being used. This allows us to represent nominal data from these periods. Alternatively, we can also "simulate" a failed magnetometer or panel deployment failure by using an incorrect rotation matrix. In a similar vein, each sun sensor is averaged with the four other sun sensors and thus has only a limited impact on the wider ADCS. We can incorporate or ignore the +Y sun sensor in voting and/or use the correct rotation matrix depending on whether we want to represent nominal data, a failed sun sensor, or a panel deployment failure.

Enumerating all of the fault scenarios that can represented from the LightSail 2 telemetry, Table 4.5 summarizes the telemetry segments for validating fault isolation.

The segments vary widely in length, so the rightmost column shows the total amount of time spanned by the segments. Each telemetry segment was inspected manually to independently assess which faults were present in the segment. Some segments possess multiple faults while others have been rejected due a lack of relevant anomalies, such as a sun sensor voting anomaly failing to appear during a sun sensor failure segment.

Table 4.5: Catalog of faults in LightSail 2 telemetry segments.

Fault scenario	Telemetry segments	Total time
Nominal	17	215 hr
Gyro calibration error	6	26 hr
Sun sensor failure	7	141 hr
Corrupt database	4	87 hr
Electrical brownouts	11	137 hr
Solar panel shadowing	13	203 hr
Saturated reaction wheel	11	143 hr
Unresponsive reaction wheel	1	3 hr
Clock or TLE error	12	94 hr
Magnetometer failure	18	287 hr
Panel deployment	13	213 hr
Total	113	1549 hr

The results of applying Smart-FDIR to the segments in Table 4.5 are summarized by the confusion plot in Fig. 4.18. The LSTM's prevailing diagnosis for each segment has been determined from a combination of manual inspection and simple rules customized to the nature of the fault. Generally, the diagnosis is deemed successful if the confidence corresponding to the correct fault persists above 50% for at least a full orbit. Out of 113 telemetry segments, the LSTM network achieves 91% overall accuracy. This is only slightly lower than what was achieved with the fault simulator in Section 3.2.3.

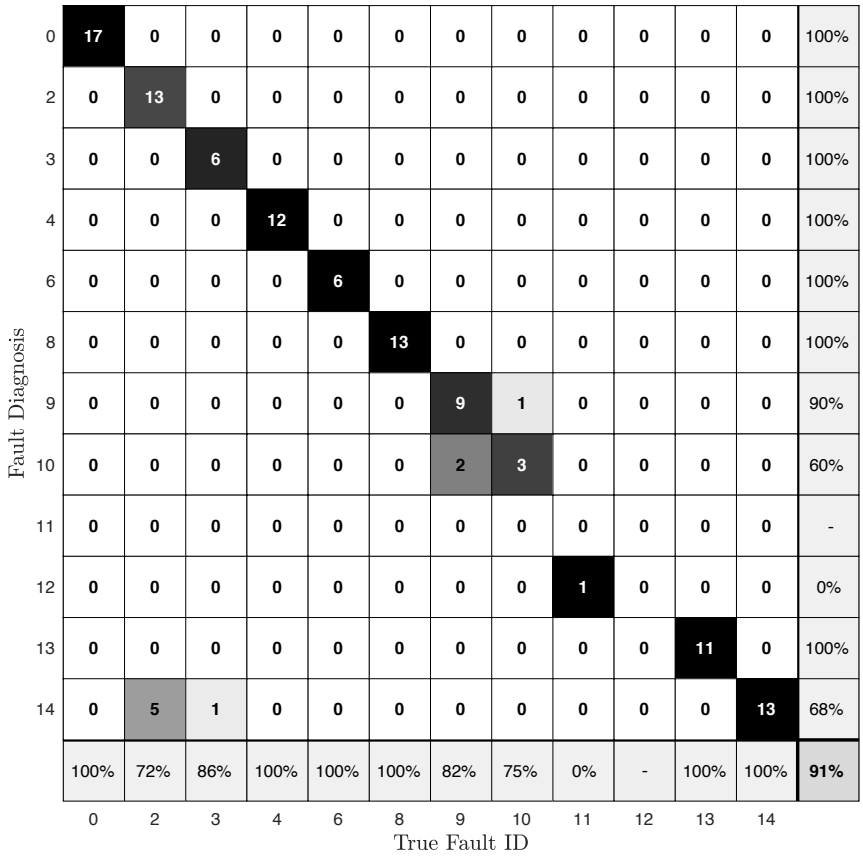


Figure 4.18: Prevailing diagnoses when applying anomaly detection and fault isolation to LightSail 2 telemetry.

Like the fault simulator, the accuracy varies between fault scenarios. The nominal case is very accurately diagnosed just as it is in the fault simulator. The most difficult cases are the magnetometer failures, corrupt databases, and the reaction wheel difficulties encountered during the initial testing of mode 2 (similar to Fig. 4.7). Whether the latter fault counts as an unresponsive wheel or simply a torquing error (the LSTM's diagnosis) is somewhat semantic. The inputs and diagnosis for this telemetry segment are presented in Appendix D.2. Meanwhile, the corrupt database fault can be difficult to distinguish from electrical brownouts since its anomaly pattern is confined almost exclusively to the LDC. Finally, the significant fraction of magnetometer failures that are classified as panel deployment faults suggests that the

sun sensor anomaly detector is not fully adequate. Thus, the LSTM network learns to assign confidence to this fault even when only magnetometer (and not sun sensor) anomalies are present. This is what is causing the errors in column 2 of the confusion matrix.

5. CONCLUSIONS

5.1 Summary

Traditional fault management technologies have plateaued just as current trends within the space flight industry are demanding greater autonomy for fault detection, isolation, and recovery. This dissertation has developed a data-driven architecture that uses transfer learning to identify anomaly patterns from a spacecraft simulator and apply this knowledge to isolate and recover from faults on the real spacecraft. The method was demonstrated on the full FDIR cycle for ADCS faults in a generic CubeSat simulation and used to diagnose both known and previously unknown faults on the LightSail 2 solar sail spacecraft.

The core idea is to make both simulated and actual faults look identical from the perspective of fault isolation. A set of OCSVMs and rule-based checks are used to process raw telemetry into a more abstract collection of anomaly scores. The OCSVMs are trained exclusively from nominal data, can be applied to a variety of ADCS signals, are efficient to evaluate, and intuitive to tune. Since the anomaly criteria are learned implicitly from nominal data, they do not require detailed subsystem knowledge to implement. An ADCS fault simulator is initially used to train the OCSVMs and simulate their anomaly scores under different fault scenarios. The patterns contain interdependent and time-dependent clues that characterize the fault scenario.

To ensure that the temporal context of anomalies is utilized in fault isolation, an LSTM network is trained to assign confidence scores to each possible fault based on the signals of the anomaly detectors and the faults injected in the ADCS simulator. The LSTM network assigns its confidence incrementally and does so despite contradicting signals, multiple faults, missing data, and false positives. Unlike tradi-

tional fault isolation techniques, it is able to do all this without explicit programming. Because the LSTM network makes its diagnosis based on abstract anomaly scores instead of raw telemetry, it is able to train on data from the fault simulator and apply its knowledge to an actual spacecraft once the OCSVMs are retrained on nominal flight data. The method attained an overall fault isolation accuracy of 96% on the ADCS simulator and 91% when applied to LightSail 2.

The diagnosis decisions of the LSTM network can be understood using the gradient of the network's confidence with respect to the anomaly inputs. This reveals the subset of signals that are most important to the network's decision. An approximation to the OCSVM anomaly score gradient was derived to extend this explanatory power to the level of individual anomalies and was used to isolate faults to a specific sensor or actuator.

Decision theory can be applied to transform the uncertain fault confidences of the LSTM network into recovery action. Recovery actions are controlled by utility values that can be set and adjusted to reflect evolving mission priorities and risk posture. The method was applied to select control modes in the ADCS CubeSat simulator. Examples were provided demonstrating a successful recovery from multiple faults and also illustrating how the response behavior can be configured by changing utility values.

Other contributions of the investigation include an extensive fault tree for the ADCS of Earth-orbiting satellites and diagnosis of faults that go beyond the simple hardware failures that are typically the focus of FDIR literature.

In summary, this research has provided a new approach to FDIR that provides value by reducing the need for expert knowledge during both design and operations. The approach leverages a spacecraft simulator to learn the complex temporal patterns between faults and anomalies implicitly while samples of nominal telemetry are used to adapt anomaly criteria to the specific flight environment during operations. When a fault occurs, the recovery action is based not on fixed action paths, but on operator priorities that can be adjusted as the mission evolves. The architecture is more

flexible, more capable, and better attuned to the needs of future space missions than current approaches.

5.2 Directions for Future Work

In addition to the main contributions, an important outcome of this investigation has been to reveal promising avenues for further research. Several limitations of the investigated method along with suggestions for how they might be overcome are described below.

Improving Confidence Assignment

The single largest difficulty encountered in training LSTM networks for fault isolation was ensuring that the faults injected into the fault simulator were in fact detectable. If faults fail to manifest as meaningful anomalies in the LSTM inputs, the network may learn questionable relationships in an effort to minimize the MSE of its confidences on the training set. This can be a problem even if anomalies do manifest but do so late in the training time series. In essence, it is crucial that the LSTM's fault confidences are scored fairly when compared to the target fault flags.

In Chapter 4 this was achieved by revising the injected fault flags so that they were elevated only after one or more anomalies appeared in a given time series. However, this required an expectation of what anomalies would be triggered by certain faults. Several alternative approaches that may prove more general are:

- New loss functions: under MSE loss the LSTM network is penalized as much for failing to assign confidence to an injected fault as it is for assigning confidence to a fault that is not present. This kind of loss function does not adequately capture the need to sometimes withhold confidence until sufficient evidence is amassed to come to a firm conclusion. Loss functions that penalize confidence assigned to absent faults more heavily or weight errors towards the end of the

time series may encourage the LSTM network to increment confidence more gradually and with more stability than MSE.

- Skewing the distribution of faults: this investigation gave equal representation to each fault scenario in the LSTM training set. In reality, some scenarios (e.g. nominal) are much more likely than others. Skewing the training set to reflect the higher likelihood of certain faults could prevent the LSTM network from allotting confidence to faults until anomaly evidence suggests otherwise.

- Pruning insignificant neuron connections: even if there remain cases where target fault flags are elevated prior to anomalies, spurious anomaly patterns learned by the LSTM are likely to be represented by weak neuron connections. Pruning these connections with methods such as "optimal brain damage" may reduce questionable confidence assignment and improve generalization [152].

Concurrent Training

Central to the ability of the LSTM network to learn fault patterns from the ADCS simulator are the OCSVM anomaly detectors. Training these detectors was done separately from the LSTM network and involved two copies of each detector: one trained from the simulation and another from flight data. Embedding the detectors directly into the LSTM architecture could simplify development by allowing the anomaly detectors to be trained concurrently with the LSTM network. Instead of anomaly scores, the combined network would receive raw telemetry from the simulation and learn its own parameters and signal processing for the detectors. One-class neural networks will likely be better suited to this than OCSVMs [64].

Such an architecture may be able to exploit transfer learning more fully. An initial fault detection and isolation network could be developed from simulation and refined with further training during operations. In contrast, Smart-FDIR adapts only the OCSVM anomaly detectors during operations and keeps the LSTM fault isolation

network fixed. Learning achieved from flight telemetry could also flow back to the simulation to refine models of the spacecraft.

Robust Recovery Decisions

The decision theory approach to fault recovery developed in Section 3.3 selects an ADCS mode based on the relative balance of the fault confidences. Hence, the decision becomes very sensitive when two or more faults are allotted similar levels of confidence. Modifying the method to allow the LSTM network to resolve competing fault hypotheses before taking action could enhance the stability and robustness of recovery decisions.

It is also recognized that many spacecraft operators will be initially reluctant to allow an external system such as Smart-FDIR to enact fault recovery commands on their spacecraft. A more palatable approach may be to instead use Smart-FDIR to recommend candidate recovery actions to ground operators. In future work, reinforcement learning may supersede the simplistic decision theory used in this thesis to allow Smart-FDIR to improve its fault recovery actions based on operator feedback about the quality of its recommendations. This would serve the dual purpose of building operator confidence while improving performance.

Embedded Applications

Though Smart-FDIR demonstrated its value as ground software in Chapter 4, it has significant potential to provide onboard FDIR as part of the flight software. Neural networks are inherently parallel, making them efficient to evaluate once trained, and the LSTMs considered in Chapters 3 and 4 occupy low memory footprints (< 50 kB). OCSVMs also occupy small footprints and are efficient to evaluate. This makes Smart-FDIR suitable for operating on flight processors. Nevertheless, extending the system to an embedded environment will require addressing several practical considerations, such as the need to yield priority to other processes and operate in-

crementally whenever processing time is available to support FDIR. This may mean that the LSTM network must operate with time steps of non-equal length. Additional requirements may be required to guarantee compatibility with a wide variety of flight software cores.

REFERENCES

[1] D. L. Akin, "Akin's laws of spacecraft design," Dave Akin's Web Site, https://spacecraft.ssl.umd.edu/akins_laws.html, (Accessed 19 February 2020).

[2] D. M. Harland and R. D. Lorenz, *Space Systems Failures*. Berlin, Germany: Springer, 2005.

[3] A. Wander and R. Förstner, "Innovative fault detection, isolation, and recovery strategies on-board spacecraft: State of the art and research challenges," in *Deutscher Luft- und Raumfahrtkongress*, Berlin, Germany, September 2012.

[4] P. Z. Schulte, "A state machine architecture for aerospace vehicle fault protection," Ph.D. dissertation, Georgia Institute of Technology, 2018.

[5] P. Z. Schulte and D. A. Spencer, "State machine fault protection architecture for aerospace vehicle guidance, navigation, and control," *Journal of Aerospace Information Systems*, pp. 1–16, 2019.

[6] National Aeronautics and Space Administration, "Fault management handbook," Tech. Rep. NASA-HDBK-1002, 2012.

[7] T. Uhlig, F. Sellmaier, and M. Schmidhuber, *Spacecraft Operations*. Wien, Austria: Springer, 2015, pp. 240–242.

[8] M. Tipaldi and B. Bruenjes, "Survey on fault detection, isolation, and recovery strategies in the space domain," *Journal of Aerospace Information Systems*, vol. 12, no. 2, pp. 235–256, 2015.

[9] M. Tipaldi, L. Feruglio, P. Denis, and G. D'Angelo, "On applying AI-driven flight data analysis for operational spacecraft model-based diagnostics," *Annual Reviews in Control*, vol. 49, pp. 197 – 211, 2020.

[10] M. Tafazoli, "A study of on-orbit spacecraft failures," *Acta Astronautica*, vol. 64, no. 2, pp. 195 – 205, 2009.

[11] N. Sultan and P. Groepper, "Analyzing real cost of past orbital failures for satellite test effectiveness and insurance," in *18th International Communications Satellite Systems Conference and Exhibit*, no. AIAA 2000-1227, Oakland, CA, 10–14 April 2000.

[12] C. Castel, J.-F. Gabard, C. Tessier, B. Laborde, and R. Soumagne, "FDIR strategies for autonomous satellite formations-a preliminary report," in *AAAI Fall Symposium: Spacecraft Autonomy*, 2006, pp. 35–42.

[13] D. Selva, A. Golkar, O. Korobova, I. L. i Cruz, P. Collopy, and O. L. de Weck, "Distributed Earth satellite systems: What is needed to move forward?" *Journal of Aerospace Information Systems*, vol. 14, no. 8, pp. 412–438, 2017.

[14] C. Araguz, E. Bou-Balust, and E. Alarcon, "Applying autonomy to distributed satellite systems: Trends, challenges, and future prospects," *Systems Engineering*, vol. 21, pp. 401–416, 2018.

[15] D. Messier, "SpaceX wants to launch 12,000 satellites," Parabolic Arc, http://www.parabolicarc.com/2017/03/03/spacex-launch-12000-satellites/, November 2017, (Accessed 01 October 2019).

[16] H. J. Kramer, "Lemur-2 nanosatellite constellation of spire global," eoPortal Directory, https://directory.eoportal.org/web/eoportal/satellite-missions/l/lemur, (Accessed 15 June 2020).

[17] M. Antoniou, A. G. Stove, A. Sayin, G. Atkinson, M. Cherniakov, H. Ma, H. Kuschel, D. Cristallini, P. Wojaczek, C. I. Underwood, A. Moccia, A. Renga, and G. Fasano, "Passive SAR satellite constellation for near-persistent earth observation: Prospects and issues," *IEEE Aerospace and Electronic Systems Magazine*, vol. 33, no. 12, pp. 4–15, Dec 2018.

[18] T. Imken, T. Randolph, M. DiNicola, and A. Nicholas, "Modeling spacecraft safe mode events," in *IEEE Aerospace Conference*, Big Sky, MT, 3-10 March 2018.

[19] N. Cohen, W. A. Wheeler, R. Ewart, and J. Betser, "Spacecraft embedded cyber defense- prototypes & experimentation," in *AIAA SPACE 2016*, no. AIAA 2016-5231, Long Beach, CA, 13–16 September 2016.

[20] W. A. Wheeler, N. Cohen, J. Betser, C. Meyers, W. Snavely, S. Chaki, M. Riley, and B. Runyon, "Cyber resilient flight software for spacecraft," in *2018 AIAA SPACE and Astronautics Forum and Exposition*, no. AIAA 2018-5220, Orlando, FL, 17–19 September 2018.

[21] B. Bailey, R. J. Speelman, P. A. Doshi, N. C. Cohen, and W. A. Wheeler, "Defending spacecraft in the cyber domain," Aerospace Corporation, Tech. Rep. OTR202000016, November 2019.

[22] D. Sternberg, J. Essmiller, C. Colley, A. Klesh, and J. Krajewski, "Attitude control system for the Mars Cube One spacecraft," in *2019 IEEE Aerospace Conference*, Big Sky, MT, 2–9 March 2019, pp. 1–10.

[23] Canadian Space Agency, "Canadarm, Canadarm2, and Canadarm3 – a comparative table," Canadian Space Agency, https://www.asc-csa.gc.ca/eng/iss/canadarm2/canadarm-canadarm2-canadarm3-comparative-table.asp, May 2019, (Accessed 18 February 2020).

[24] R. D. Lorenz, E. P. Turtle, J. W. Barnes, M. G. Trainer, D. S. Adams, K. E. Hibbard, C. Z. Sheldon, K. Zacny, P. N. Peplowski, D. J. Lawrence *et al.*, "Dragonfly: A rotorcraft lander concept for scientific exploration at Titan," *Johns Hopkins APL Technical Digest*, vol. 34, no. 3, p. 14, 2018.

[25] J. D. Frank, "Artificial intelligence: Powering human exploration of the Moon and Mars," *arXiv preprint arXiv:1910.03014*, 2019.

[26] S. B. Johnson, *The Theory of System Health Management*. John Wiley & Sons, Ltd, 2011, ch. 1, pp. 3–27.

[27] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly Detection: A Survey," University of Minnesota, Tech. Rep. TR 07-017, August 2007.

[28] B. Ricks and O. J. Mengshoel, "Methods for probabilistic fault diagnosis: An electrical power system case study," in *Proc. 1st Annu. Conf. Prognostics Health Manag Soc.*, San Diego, CA, 27 September - 1 October 2009.

[29] D. A. Galvan, B. Hemenway, W. Welser IV, D. Baiocchi *et al.*, "Satellite anomalies: Benefits of a centralized anomaly database and methods for securely sharing information among satellite operators," RAND Corporation, Santa Monica, CA, Tech. Rep. RR-560-DARPA, 2014.

[30] C. C. Venturini, "Improving mission success of cubesats," Aerospace Corporation, Tech. Rep. TOR-2017-01689, June 2017.

[31] M. Langer and J. Bouwmeester, "Reliability of cubesats – statistical data, developers' beliefs and the way forward," in *30th Annual AIAA/USU Conference on Small Satellites*, no. SSC16-X-2, Logan, UT, 6–11 August 2016, pp. 1–13.

[32] M. Swartwout, "Secondary spacecraft in 2016: Why some succeed (and too many do not)," in *2016 IEEE Aerospace Conference*, Big Sky, MT, 5–12 March 2016, pp. 1–13.

[33] M. A. Swartwout, "Cubesat database," Saint Louis University, https://sites.google.com/a/slu.edu/swartwout/home/cubesat-database#refs, 2019, (Accessed 20 February 2020).

[34] A. West, "NASA study on flight software complexity," NASA, Tech. Rep., March 2009.

[35] S. B. Johnson, "Introduction to systems health management in aerospace," in *1st Integrated Systems Health Engineering and Management Forum*, Napa, CA, 7–10 November 2005.

[36] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 1–58, 2009.

[37] ——, "Anomaly detection for discrete sequences: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 5, pp. 823–839, 2012.

[38] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," *CoRR*, vol. abs/1901.03407, 2019. [Online]. Available: http://arxiv.org/abs/1901.03407

[39] A. K. Carlton, "Fault detection algorithms for spacecraft monitoring and environmental sensing," Master's thesis, Massachusetts Institute of Technology, 2016.

[40] A. Carlton, R. Morgan, W. Lohmeyer, and K. Cahoy, "Telemetry fault-detection algorithms: Applications for spacecraft monitoring and space environment sensing," *Journal of Aerospace Information Systems*, vol. 15, no. 5, pp. 239–252, 2018.

[41] T. Yairi, M. Nakatsugawa, K. Hori, S. Nakasuka, K. Machida, and N. Ishihama, "Adaptive limit checking for spacecraft telemetry data using regression tree learning," in *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583)*, vol. 6, Hague, Netherlands, 10–13 October 2004, pp. 5130–5135.

[42] T. Yairi, Y. Kawahara, R. Fujimaki, Y. Sato, and K. Machida, "Telemetry-mining: a machine learning approach to anomaly detection and fault diagnosis for space systems," in *2nd IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT'06)*, Pasadena, CA, 17–20 July 2006.

[43] A. Alkaya and I. Eker, "Variance sensitive adaptive threshold-based pca method for fault detection with experimental application," *ISA Transactions*, vol. 50, no. 2, pp. 287 – 302, 2011.

[44] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Söderström, "Detecting spacecraft anomalies using LSTMs and nonparametric dynamic thresholding," *CoRR*, vol. abs/1802.04431, 2018. [Online]. Available: http://arxiv.org/abs/1802.04431

[45] C. N. Hadjicostis, "Probabilistic fault detection in finite-state machines based on state occupancy measurements," in *Proceedings of the 41st IEEE Conference on Decision and Control*, vol. 4. Las Vegas, NV: IEEE, 10–13 December 2002, pp. 3994–3999.

[46] J. A. Momoh and R. Button, "Design and analysis of aerospace DC arcing faults using fast Fourier transformation and artificial neural network," in *2003 IEEE Power Engineering Society General Meeting (IEEE Cat. No.03CH37491)*, vol. 2, Toronto, Canada, 13–17 July 2003, pp. 788–793.

[47] G. Biswas, H. Khorasgani, G. Stanje, A. Dubey, S. Deb, and S. Ghoshal, "An approach to mode and anomaly detection with spacecraft telemetry data," *International Journal of Prognostics and Health Management*, vol. 7, pp. 1–18, 2016.

[48] S. Sarkar, S. Sarkar, N. Virani, A. Ray, and M. Yasar, "Sensor fusion for fault detection and classification in distributed physical processes," *Frontiers in Robotics and AI*, vol. 1, no. 16, pp. 1–9, 2014.

[49] O. Neuner, "Automatic learning of state machines for fault detection systems in discrete event based distributed systems," Master's thesis, KTH Royal Institute of Technology, 2014.

[50] A. Zolghadri, "Advanced model-based FDIR techniques for aerospace systems: Today challenges and opportunities," *Progress in Aerospace Sciences*, vol. 53, pp. 18 – 29, 2012.

[51] J. Marzat, H. Piet-Lahanier, F. Damongeot, and E. Walter, "Model-based fault diagnosis for aerospace systems: a survey," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of aerospace engineering*, vol. 226, no. 10, pp. 1329–1360, 2012.

[52] I. Hwang, S. Kim, Y. Kim, and C. E. Seah, "A survey of fault detection, isolation, and reconfiguration methods," *IEEE transactions on control systems technology*, vol. 18, no. 3, pp. 636–653, 2009.

[53] I. Fagarasan and S. S. Iliescu, "Parity equations for fault detection and isolation," in *2008 IEEE International Conference on Automation, Quality and Testing, Robotics*, vol. 1, Cluj-Napoca, Romania, 22–25 May 2008, pp. 99–103.

[54] N. Tudoroiu and K. Khorasani, "Satellite fault diagnosis using a bank of interacting Kalman filters," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 43, no. 4, pp. 1334–1350, 2007.

[55] F. Pirmoradi, F. Sassani, and C. de Silva, "Fault detection and diagnosis in a spacecraft attitude determination system," *Acta Astronautica*, vol. 65, no. 5, pp. 710 – 729, 2009.

[56] M. N. Pontuschka and I. M. da Fonseca, "FDIR for the IMU component of AOCS systems," *Mathematical Problems in Engineering*, 2014.

[57] C. Pittet, A. Falcoz, and D. Henry, "A model-based diagnosis method for transient and multiple faults of AOCS thrusters," *IFAC-PapersOnLine*, vol. 49, no. 17, pp. 82–87, 2016, 20th IFAC Symposium on Automatic Control in Aerospace.

[58] W. R. Williamson, J. L. Speyer, V. T. Dang, and J. Sharp, "Fault detection and isolation for deep space satellites," *Journal of guidance, control, and dynamics*, vol. 32, no. 5, pp. 1570–1584, 2009.

[59] S. Tariq, S. Lee, Y. Shin, M. S. Lee, O. Jung, D. Chung, and S. S. Woo, "Detecting anomalies in space using multivariate convolutional LSTM with mixtures of probabilistic PCA," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. Anchorage, AK: ACM, July 2019, pp. 2123–2133.

[60] W. Sun, A. R. C. Paiva, P. Xu, A. Sundaram, and R. D. Braatz, "Fault Detection and Identification using Bayesian Recurrent Neural Networks," *arXiv e-prints*, p. arXiv:1911.04386, Nov 2019.

[61] T. Yairi, Y. Kato, and K. Hori, "Fault detection by mining association rules from housekeeping data," in *International Symposium on Artificial Intelligence, Robotics, and Automation in Space*, Montreal, Canada, 18–22 June 2001.

[62] M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AIChE journal*, vol. 37, no. 2, pp. 233–243, 1991.

[63] S. Hawkins, H. He, G. Williams, and R. Baxter, "Outlier detection using replicator neural networks," in *International Conference on Data Warehousing and Knowledge Discovery*. Aix-en-Provence, France: Springer, 4–6 September 2002, pp. 170–180.

[64] R. Chalapathy, A. K. Menon, and S. Chawla, "Anomaly detection using one-class neural networks," *arXiv preprint arXiv:1802.06360*, 2018.

[65] B. Schölkopf, R. Williamson, A. Smola, J. Shawe-Taylor, and J. C. Platt, "Support vector method for novelty detection," in *Advances in Neural Information Processing Systems 12*, S. Solla, T. Leen, and K. Müller, Eds. MIT Press, 2000, pp. 582–588.

[66] K. Müller, S. Mika, G. Ratsch, K. Tsuda, and B. Scholkopf, "An introduction to kernel-based learning algorithms," *IEEE Transactions on Neural Networks*, vol. 12, no. 2, pp. 181–201, March 2001.

[67] S. Fuertes, G. Picart, J.-Y. Tourneret, L. Chaari, A. Ferrari, and C. Richard, "Improving spacecraft health monitoring with automatic anomaly detection techniques," in *14th International Conference on Space Operations*, Daejeon, South Korea, 16–20 May 2016, p. 2430.

[68] D. L. Iverson, "System health monitoring for space mission operations," in *2008 IEEE Aerospace Conference*, Big Sky, MT, March 2008, pp. 1–8.

[69] K. Li, Y. Wu, S. Song, Y. Sun, J. Wang, and Y. Li, "A novel method for spacecraft electrical fault detection based on FCM clustering and WPSVM classification with PCA feature extraction," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 231, no. 1, pp. 98–108, 2017.

[70] H. Izakian and W. Pedrycz, "Anomaly detection in time series data using a fuzzy c-means clustering," in *2013 Joint IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS)*. Edmonton, Canada: IEEE, 24–28 June 2013, pp. 1513–1518.

[71] S. R. Starin, M. F. Vess, T. M. Kenney, M. D. Maldondao, and W. D. Morgenstern, "Fault detection and correction for the Solar Dynamics Observatory attitude control system," in *AAS 2008 Guidance and Control (GN&C) Conference*, Breckenridge, CO, 1–6 February 2008.

[72] D. Poole and A. Mackworth, *Artificial Intelligence*, 2nd ed. Cambridge University Press, 2017.

[73] H. C. Koons and D. J. Gorney, "Spacecraft environmental anomalies expert system," Aerospace Corporation, El Segundo, CA, Tech. Rep. ATR-88(9562)-1, 1988.

[74] M. Rolincik, M. Lauriente, H. C. Koons, and D. Gorney, "An expert system for diagnosing environmentally induced spacecraft anomalies," in *Proc. of 5th Annual Space Operations and Applications Research Symposium*, Houston, TX, 9–11 July 1991, pp. 36–44.

[75] T. P. O'Brien, "SEAES-GEO: A spacecraft environmental anomalies expert system for geosynchronous orbit," *Space Weather*, vol. 7, no. S09003, 2009.

[76] D. Cayrac, D. Dubois, and H. Prade, "Handling uncertainty with possibility theory and fuzzy sets in a satellite fault diagnosis application," *IEEE Transactions on Fuzzy Systems*, vol. 4, no. 3, pp. 251–269, 1996.

[77] A. P. Dempster, "Upper and lower probabilities induced by a multivalued mapping," *The Annals of Mathematical Statistics*, vol. 38, no. 2, pp. 325–339, 1967.

[78] G. Shafer, *A mathematical theory of evidence*. Princeton university press, 1976, vol. 42.

[79] X. Fan and M. J. Zuo, "Fault diagnosis of machines based on D–S evidence theory. part 1: D–s evidence theory and its improvement," *Pattern Recognition Letters*, vol. 27, no. 5, pp. 366–376, 2006.

[80] ——, "Fault diagnosis of machines based on d–s evidence theory. part 2: Application of the improved d–s evidence theory in gearbox fault diagnosis," *Pattern Recognition Letters*, vol. 27, no. 5, pp. 377–385, 2006.

[81] M. Song and W. Jiang, "Engine fault diagnosis based on sensor data fusion using evidence theory," *Advances in Mechanical Engineering*, vol. 8, no. 10, p. 1687814016673291, 2016.

[82] D. A. Marsillach, S. Virani, M. J. Holzinger, M. W. Chan, and P. P. Shenoy, "Real-time telescope tasking for custody and anomaly resolution using judicial evidential reasoning," in *29th Space Flight Mechanics Meeting*, no. 19-534, Maui, HI, January 2019.

[83] K. Rajan, D. Bernard, G. Dorais, E. Gamble, B. Kanefsky, J. Kurien, W. Millar, N. Muscettola, P. Nayak, N. Rouquette *et al.*, "Remote agent: An autonomous control system for the new millennium," in *Proceedings of the 14th European Conference on Artificial Intelligence*. Berlin, Germany: IOS Press, 20–25 August 2000, pp. 726–730.

[84] P. Robinson, M. Shirley, D. P. Fletcher, R. Alena, D. Duncavage, C. Lee, and N. D. Ames, "Applying model-based reasoning to the FDIR of the command and data handling subsystem of the International Space Station," in *7th International Symposium on Artificial Intelligence, Robotics, and Automation in Space*, Nara, Japan, 19–23 May 2003.

[85] S. C. Hayden, A. J. Sweet, and S. E. Christa, "Livingstone model-based diagnosis of Earth Observing One," in *Proceedings of the AIAA 1st Intelligent Systems Conference*, Chicago, IL, 20–22 September 2004.

[86] B. C. Williams and R. J. Ragno, "Conflict-directed A* and its role in model-based embedded systems," *Discrete Appl. Math.*, vol. 155, no. 12, p. 1562–1595, 2007.

[87] A. Barua, P. Sinha, K. Khorasani, and S. Tafazoli, "A novel fault-tree approach for identifying potential causes of satellite reaction wheel failure," in *Proceedings of 2005 IEEE Conference on Control Applications*. Toronto, Canada: IEEE, 28–31 August 2005, pp. 1467–1472.

[88] O. J. Mengshoel, M. Chavira, K. Cascio, S. Poll, A. Darwiche, and S. Uckun, "Probabilistic model-based diagnosis: An electrical power system case study," *IEEE Transactions on Systems, Man, and Cybernetics: Systems and Humans*, vol. 40, no. 5, 2010.

[89] D. Codetta-Raiteri and L. Portinale, "Dynamic Bayesian networks for fault detection, identification, and recovery in autonomous spacecraft," *IEEE Transactions on Systems, Man., and Cybernetics: Systems*, vol. 45, no. 1, pp. 13–24, 2015.

[90] S. Bottone, D. Lee, M. O'Sullivan, and M. Spivack, "Failure prediction and diagnosis for satellite monitoring systems using Bayesian networks," in *2008 IEEE Military Communications Conference*, San Diego, CA, 16–19 November 2008, pp. 1–7.

[91] J. Schumann, O. J. Mengshoel, and T. Mbaya, "Integrated software and sensor health management for small spacecraft," in *Fourth IEEE Conference on Space Mission Challenges for Information Technology*, Palo Alto, CA, 2–4 August 2011.

[92] E. Horvitz and M. Barry, "Display of information for time-critical decision making," in *Proceedings of the 11th conference on uncertainty in artificial intelligence*, Montréal, Canada, 18–20 August 1995.

[93] J. P. Matsuura and T. Yoneyama, "Learning Bayesian networks for fault detection," in *IEEE Workshop on Machine Learning for Signal Processing*, Sao Luis, Brazil, 29 Septmeber – 1 October 2004.

[94] Z. Ghahramani, *Learning dynamic Bayesian networks*. Berlin, Heidlberg: Springer, 1998.

[95] R. Daly, Q. Shen, and S. Aitken, "Learning Bayesian networks: approaches and issues," *The Knowledge Engineering Review*, vol. 26, no. 2, pp. 99–157, 2011.

[96] S. Montani, L. Portinale, and D. Codetta-Raiteri, "RADYBAN: A tool for reliability analysis of dynamic fault trees through conversion into dynamic Bayesian networks," *Reliability Engineering and System Safety*, vol. 93, pp. 922–932, 2008.

[97] M. Medkour, L. Khochmane, A. Bouzaouit, and O. Bennis, "Transformation of fault trees into Bayesian networks methodology for fault diagnosis," *Mechanika*, vol. 23, no. 6, pp. 891–899, 2017.

[98] A. A. A. M. Amiruddin, H. Zabiri, S. A. A. Taqvi, and L. D. Tufa, "Neural network applications in fault diagnosis and detection: an overview of implementations in engineering-related systems," *Neural Computing and Applications*, pp. 1–26, 2018.

[99] H. A. Talebi, K. Khorasani, and S. Tafazoli, "A recurrent neural-network-based sensor and actuator fault detection and isolation for nonlinear systems with application to the satellite's attitude control subsystem," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 45–60, Jan 2009.

[100] N. Gugulothu, T. Vishnu, P. Gupta, P. Malhotra, L. Vig, P. Agarwal, and G. Shroff, "On practical aspects of using RNNs for fault detection in sparsely-labeled multi-sensor time series," in *Proceedings of the Annual Conference of the PHM Society*, vol. 10, no. 1, Philadelphia, PA, 24–27 September 2018.

[101] T. Jiang, K. Khorasani, and S. Tafazoli, "Parameter estimation-based fault detection, isolation and recovery for nonlinear satellite models," *IEEE Transactions on control systems technology*, vol. 16, no. 4, pp. 799–808, 2008.

[102] H. A. Nozari, P. Castaldi, H. D. Banadaki, and S. Simani, "Novel non-model-based fault detection and isolation of satellite reaction wheels based on a mixed-learning fusion framework," *IFAC-PapersOnLine*, vol. 52, no. 12, pp. 194–199, 2019.

[103] A. Verner, "LSTM networks for detection and classification of anomalies in raw sensor data," Ph.D. dissertation, Nova Southeastern University, April 2019.

[104] Z. Feng and T. Xu, "Comparison of SOM and PCA-SOM in fault diagnosis of ground-testing bed," *Procedia Engineering*, vol. 15, pp. 1271–1276, 2011.

[105] M. Tipaldi and L. Glielmo, "A survey on model-based mission planning and execution for autonomous spacecraft," *IEEE Systems Journal*, vol. 12, no. 4, pp. 3893–3905, 2018.

[106] P. Z. Schulte, D. A. Spencer, and M. Goggin, "Mars sample return terminal rendezvous state-based fault protection," *Journal of Spacecraft and Rockets*, pp. 1–11, 2019.

[107] P. Z. Schulte and D. A. Spencer, "State machine fault protection for autonomous proximity operations," in *68th International Astronautical Congress*, Adelaide, Australia, 25–29 September 2017.

[108] M. Aguilar, "Fault management using model based system engineering (MBSE) tools and techniques," NASA Spacecraft Fault Management Workshop, September 2011.

[109] M. D. Ingham, R. D. Rasmussen, M. B. Bennett, and A. C. Moncada, "Engineering complex embedded systems with state analysis and the mission data system," *Journal of Aerospace Computing, Information, and Communication*, vol. 2, no. 12, pp. 507–536, 2005.

[110] A. Nasir, E. Atkins, and I. Kolmanovsky, "A mission based fault reconfiguration framework for spacecraft applications," in *Infotech@ Aerospace 2012*, Garden Grove, CA, 19–21 June 2012, p. 2403.

[111] A. Nasir, E. M. Atkins, and I. V. Kolmanovsky, "Mission-based fault reconfiguration for spacecraft applications," *Journal of Aerospace Information Systems*, vol. 10, no. 11, pp. 513–516, 2013.

[112] S. Müller, A. Gerndt, and T. Noll, "Synthesizing failure detection, isolation, and recovery strategies from nondeterministic dynamic fault trees," *Journal of Aerospace Information Systems*, vol. 16, no. 2, pp. 52–60, 2019.

[113] F. Fei, Z. Tu, Y. Yang, X. Zhang, D. Xu, and X. Deng, "Learn to recover: Reinforcement learning-assisted fault tolerant control for quadrotor UAVs," Purdue University, Tech. Rep., 2018.

[114] Q. Zhu and C. Yuan, "A reinforcement learning approach to automatic error recovery," in *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07)*, Edinburgh, UK, 25–28 June 2007, pp. 729–738.

[115] S. McGuire, P. M. Furlong, C. Heckman, S. Julier, D. Szafir, and N. Ahmed, "Failure is not an option: Policy learning for adaptive recovery in space operations," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1639–1646, 2018.

[116] S. McGuire, P. M. Furlong, T. Fong, C. Heckman, D. J. Szafir, S. Julier, and N. Ahmed, "Everybody needs somebody sometimes: Validation of adaptive recovery in robotic space operations," *IEEE Robotics and Automation Letters*, pp. 1–1, 2019.

[117] B. C. Williams, M. D. Ingham, S. H. Chung, and P. H. Elliott, "Model-based programming of intelligent embedded systems and robotic space explorers," *Proceedings of the IEEE*, vol. 91, no. 1, pp. 212–237, 2003.

[118] F. Giunchiglia and P. Traverso, "Planning as model checking," in *5th European Conference on Planning: Recent Advances in AI Planning*. Springer, September 1999, pp. 1–20.

[119] M. Bozzano, A. Cimatti, A. Guiotto, A. Martelli, M. Roveri, A. Tchaltsev, and Y. Yushtein, "On-board autonomy via symbolic model-based reasoning," in *10th ESA Workshop on Advanced Space Technologies for Robotics and Automation (ASTRA'2008)*, 11–13 November 2008, pp. 11–13.

[120] K. B. Gamble and E. G. Lightsey, "Decision analysis tool for small satellite risk management," *Journal of Spacecraft and Rockets*, pp. 420–432, 2016.

[121] D. A. Spencer and R. Tolson, "Aerobraking cost and risk decisions," *Journal of Spacecraft and Rockets*, vol. 44, no. 6, pp. 1285–1293, 2007.

[122] D. Codetta-Raiteri and L. Portinale, "Generalized continuous time Bayesian networks as a modelling and analysis formalism for dependable systems," *Reliability Engineering & System Safety*, vol. 167, pp. 639–651, 2017.

[123] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[124] S. A. Johnson, "Introduction to complex fault protection software testing," in *Proceedings of the 1998 American Control Conference. ACC (IEEE Cat. No. 98CH36207)*, vol. 2. Philadelphia, PA: IEEE, 24–26 June 1998, pp. 909–911.

[125] C. L. G. Batista, E. Martins, and M. de Fátima Mattiello-Francisco, "On the use of a failure emulator mechanism at nanosatellite subsystems integration tests," in *2018 IEEE 19th Latin-American Test Symposium (LATS)*, Sao Paulo, Brazil, 12–16 March 2018, pp. 1–6.

[126] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[127] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big data*, vol. 3, no. 1, p. 9, 2016.

[128] P. Wolfe, "A duality theorem for non-linear programming," *Quarterly of applied mathematics*, vol. 19, no. 3, pp. 239–244, 1961.

[129] J. R. Mansell, S. Dickmann, and D. A. Spencer, "Swarm optimization of lunar transfers from earth orbit with operational constraints," *The Journal of the Astronautical Sciences*, pp. 1–22, 2019.

[130] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[131] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[132] R. W. Ridenoure, R. Munakata, S. D. Wong, A. Diaz, D. A. Spencer, D. A. Stetson, B. Betts, B. A. Plante, J. D. Foley, and J. M. Bellardo, "Testing the LightSail program: advancing solar sailing technology using a CubeSat platform," *Journal of Small Satellites*, vol. 5, no. 2, pp. 531–550, 2016.

[133] B. Plante, D. A. Spencer, D. B. Betts, S. Chait, J. M. Bellardo, A. Diaz, and I. Pham, "LightSail 2 ADCS : From simulation to mission readiness," in *International Solar Sailing Symposium*, Kyoto, Japan, 17–20 January 2017.

[134] B. Betts, D. Spencer, J. Bellardo, B. Nye, A. Diaz, B. Plante, J. Mansell, M. Fernandez, C. Gillespie, and D. Garber, "LightSail 2: Controlled solar sailing propulsion using a CubeSat," in *70th International Astronautical Congress*, no. IAC-19.C4.8-B4.5A.2.x51593, Washington, D.C., 21–25 October 2019.

[135] J. R. Mansell, D. A. Spencer, B. A. Plante, M. A. Fernandez, C. T. Gillespie, J. M. Bellardo, A. Diaz, B. Betts, and B. Nye, "Orbit and attitude performance of the LightSail 2 solar sail spacecraft," in *AIAA Scitech 2020 Forum*, no. AIAA 2020-2177, Orlando, FL, 6–10 January 2020.

[136] "HMC6343 three-axis compass with algorithms," Manual available online: https://aerospace.honeywell.com/en/~/media/aerospace/files/datasheet/3-axiscompasswithalgorithimshmc6343_ds.pdf, Honeywell International Inc., May 2014, (Accessed 01 October 2019).

[137] "E910.86 integrated solar angle sensor," Manual available online: http://www.mouser.com/ds/2/594/910_86-224506.pdf, Elmos Inc., December 2010, (Accessed 01 October 2019).

[138] "ADIS16135 precision angular rate sensor," Manual available online: https://www.analog.com/en/products/adis16135.html#product-overview, Analog Devices Inc., 2014, rev. F. (Accessed 01 October 2019).

[139] E. Leffens, F. Markley, and M. Shuster, "Kalman filtering for spacecraft attitude estimation," *Journal of Guidance, Control, and Dynamics*, vol. 5, no. 5, pp. 417–429, 1982.

[140] U.S. Naval Observatory, "Approximate solar coordinates," https://aa.usno.navy.mil/faq/docs/SunApprox.php, November 2012, (Accessed 01 October 2019).

[141] "RW-0.060-28 microsatellite reaction wheels," Manual available online: http://www.sinclairinterplanetary.com/reactionwheels, Sinclair Interplanetary Inc., 2019, (Accessed 01 October 2019).

[142] "Torque rods model MTR-4-1," Stras Space, January 2019, rev. 2. (Personal communication).

[143] M. Lovera, "Magnetic satellite detumbling: The b-dot algorithm revisited," in *2015 American Control Conference (ACC)*, Chicago, IL, 1–3 July 2015, pp. 1867–1872.

[144] L. D. Friedman, "The story of LightSail," The Planetary Society: https://www.planetary.org/explore/projects/lightsail-solar-sailing/story-of-lightsail-part-1.html, 2015, (Accessed 24 May 2020).

[145] D. A. Spencer, L. Johnson, and A. C. Long, "Solar sailing technology challenges," *Aerospace Science and Technology*, vol. 93, p. 105276, 2019.

[146] L. Johnson, M. Whorton, A. Heaton, R. Pinson, G. Laue, and C. Adams, "NanoSail-D: A solar sail demonstration mission," *Acta Astronautica*, vol. 68, no. 5, pp. 571 – 575, 2011.

[147] Y. Tsuda, O. Mori, R. Funase, H. Sawada, T. Yamamoto, T. Saiki, T. Endo, K. Yonekura, H. Hoshino, and J. Kawaguchi, "Achievement of IKAROS — Japanese deep space solar sail demonstration mission," *Acta Astronautica*, vol. 82, no. 2, pp. 183 – 188, 2013.

[148] G. Vulpetti, L. Johnson, and G. L. Matloff, "The NanoSail-D2 NASA mission," in *Solar Sails: A Novel Approach to Interplanetary Travel*. New York, NY: Springer New York, 2015, pp. 173–178.

[149] B. Betts, B. Nye, J. Vaughn, E. Greeson, R. Chute, D. A. Spencer, R. W. Ridenoure, R. Munakata, S. D. Wong, A. Diaz *et al.*, "Lightsail 1 mission results and public outreach strategies," in *The 4th International Symposium on Solar Sailing*, Kyoto Research Park, Kyoto, Japan, Jan 17–20 2017.

[150] D. A. Spencer, B. Betts, J. M. Bellardo, A. Diaz, B. Plante, and J. R. Mansell, "The LightSail 2 solar sailing technology demonstration," 2020, accepted 22 June 2020 for publication in *Advances in Space Research*.

[151] SpaceX, "STP-2 Mission Press Kit," https://www.spacex.com/sites/spacex/files/stp-2_press_kit.pdf, June 2019, (Accessed 2 May 2020).

[152] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Advances in neural information processing systems*, 1990, pp. 598–605.

A. NOTES ON OCSVM KERNEL FUNCTIONS

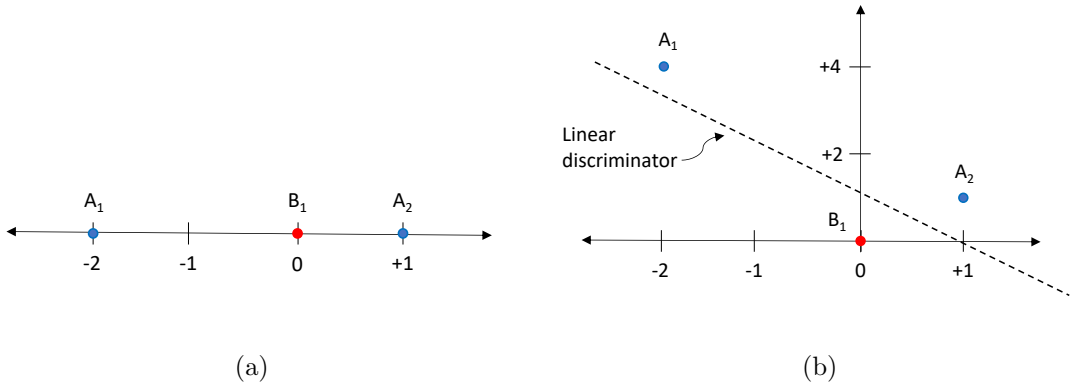
The use of a kernel function $k(x, y)$ is central to the derivation of the anomaly score function $f(x)$ for a one-class support vector machine. It provides a notion of similarity between pairs of vectors x and y, and in Eq. 2.7 its representation as an inner product $\phi^\intercal(x)\phi(y)$ allows $f(x)$ to be expressed as a linear function of k. The purpose of this appendix is to provide additional information on the mapping function, $\phi(x)$.

A.1 The Need for a Kernel Transform

First, some elaboration on why $\phi(x)$ is necessary. Support vector machine methods use a hyperplane to divide members of one class from the other. Yet, in many problems, a linear boundary between the two classes cannot be drawn. Consider the minimal example in Fig. A.1(a). Class A contains points $x = -2$ and $x = +1$ while class B contains the origin. Clearly there is no single boundary that can separate $x = -2$ and $x = +1$ from $x = 0$.

Let us therefore project these points into a higher dimensional space using the transform: $\psi(x) = (x, x^2)$. Class A now contains $(-2, 4)$ and $(1, 1)$ while class B remains at the origin. As depicted in Fig. A.1(b), the two classes are now separable with a linear boundary.

This illustrates the purpose of $\phi(x)$ in Eq. 2.2; by projecting the nominal data $x_1, ..., x_\ell$ into a higher dimensional "kernel space", it becomes possible to divide the data from the origin with a linear boundary. For an arbitrary $\phi(x)$, the origin has no special meaning. However, if we define a similarity measure $k(x, x_i) = \phi^\intercal(x)\phi(x_i)$, then $\phi(x) = 0 \Rightarrow k(x, x_i) = 0$. In other words, $\phi(x) = 0$ corresponds to a point x that bears zero similarity to any point x_i from the nominal set. Such a point is clearly an extreme anomaly, and it makes sense to locate the decision hyperplane $f(x)$ (Eq. 2.2)



(a) (b)

Figure A.1: Separation of data classes using a kernel transform.

as far away as possible from it. This is why the offset term ρ appears with a negative sign in the minimization objective in Eq. 2.3.

A.2 Determining $\phi(x)$ for a Gaussian Kernel

A commonly used similarity measure is the Gaussian kernel given in Eq. 2.12. This kernel is popular because it is based on the Euclidean distance between two points of interest as measured in their original vector space. Thus, points that are further apart are intuitively less similar. As a comparative example, another common kernel is the polynomial kernel, which has parameters θ and d [66]:

$$P(x, y) = (x^{\intercal} y - \theta)^d \tag{A.1}$$

To use the Gaussian kernel $k(x, y)$ for a OCSVM, it is necessary to prove that it can be expressed as the inner product: $\phi^{\intercal}(x)\phi(y)$. A derivation of the explicit form of $\phi(x)$ follows.

Express $k(x, y)$ in terms of the vector components of N-dimensional x and y:

$$\begin{aligned} k(x,y) &= \exp(-||x-y||^2) \\ &= \exp\left(-\sum_{j=0}^{N}(x_j - y_j)^2\right) \\ &= \exp\left(\sum_{j=0}^{N}\{-x_j^2 - y_j^2 + 2x_j y_j\}\right) \\ &= \exp\left(-\sum_{j=0}^{N} x_j^2\right)\exp\left(-\sum_{j=0}^{N} y_j^2\right)\exp\left(\sum_{j=0}^{N} 2x_j y_j\right) \\ &= \prod_{j=0}^{N} e^{-x_j^2} e^{-y_j^2} e^{2x_j y_j} \end{aligned} \tag{A.2}$$

The cross term $e^{2x_j y_j}$ is problematic for separating $k(x, y)$ into an inner product, but it can be expanded using a Taylor series:

$$e^{2x_j y_j} = \sum_{r=0}^{\infty} \frac{(2x_j y_j)^r}{r!} \tag{A.3}$$

This sum can equivalently be expressed as an inner product:

$$e^{2x_j y_j} =< 1, \sqrt{2}x_j, \frac{2}{\sqrt{2}}x_j^2, ... > \bullet < 1, \sqrt{2}y_j, \frac{2}{\sqrt{2}}y_j^2, ... > \tag{A.4}$$

Hence:

$$k(x,y) = \prod_{j=0}^{N} < e^{-x_j^2}, \sqrt{2}x_j e^{-x_j^2}, \frac{2}{\sqrt{2}}x_j^2 e^{-x_j^2}, ... > \bullet < e^{-y_j^2}, \sqrt{2}y_j e^{-y_j^2}, \frac{2}{\sqrt{2}}y_j^2 e^{-y_j^2}, ... > \tag{A.5}$$

Consider the case where $N = 1$. Eq. A.5 reduces to a single inner product:

$$\begin{aligned} k(x,y) =& < e^{-x^2}, \sqrt{2}xe^{-x^2}, \frac{2}{\sqrt{2}}x^2 e^{-x^2}, ... > \bullet < e^{-y^2}, \sqrt{2}ye^{-y^2}, \frac{2}{\sqrt{2}}y^2 e^{-y^2}, ... > \\ =& \phi^{\intercal}(x)\phi(y) \end{aligned} \tag{A.6}$$

Thus, $\phi(x)$ is explicitly determined and two things are revealed: (i) $\phi(x)$ has infinite dimensions and (ii) the interpretation of the components of $\phi(x)$ is unclear. Nonetheless, it is noteworthy that the odd components of $\phi(x)$ are each strictly greater

than zero $\forall x$. This is important because it ensures there is at least one dimension of the kernel space for which all data transformed from $x \to \phi(x)$ are projected to one side of the origin. Therefore, it is possible to divide the transformed data points from the origin with a hyperplane as depicted in Fig. 2.1.

More generally, when $N > 1$, Eq. A.5 shows that $k(x, y)$ is a product of inner products. This can be reduced to a single inner product according to the following proof.

Proof Let $k_1(x, y) = a^\intercal(x)a(y)$ and $k_2(x, y) = b^\intercal(x)b(y)$ be two inner product kernels. Then:

$$\begin{aligned} k_1(x,y)k_2(x,y) &= \left(\sum_p a_p(x)a_p(y)\right)\left(\sum_q b_q(x)b_q(y)\right) \\ &= \sum_p \sum_q \{(a_p(x)b_q(x))(a_q(y)b_p(y))\} \\ &= \sum_p \sum_q c_{pq}(x)c_{pq}(y) \\ &= c^\intercal(x)c(y) \end{aligned} \tag{A.7}$$

■

In the proof, $c(x)$ is a vector containing all the combinations of $a_p(x)$ multiplied by $b_q(x)$. Thus, it is proved that $\forall N$, Eq. A.5 reduces to the form:

$$k(x, y) = \phi^\intercal(x)\phi(y) \tag{A.8}$$

It also follows from Eq. A.7 that the first component of $\phi(x)$ will be:

$$\phi_1(x) = \exp\left(-\sum_{j=0}^{N} x_j^2\right) > 0 \tag{A.9}$$

Hence, all data transformed by ϕ will fall to the positive side of $\phi_1 = 0$. This guarantees that the nominal training data can be divided from the origin in the kernel space.

Explicit expressions for $\phi(x)$ are no less inscrutable or difficult to compute for higher N than for the $N = 1$ case shown in Eq. A.6. Fortunately, there is no need

to evaluate $\phi(x)$ explicitly. In the derivation of the OCSVM score function in Section 3.1.2, the solution of the hyperplane normal (Eq. 2.6) allows $\phi(x)$ to be replaced with $k(x_i, x)$ upon substitution into the score function $f(x)$ (Eq. 2.7). Therefore, anomaly scores can be computed based on the much more intuitive, much easier to compute kernel function k. This is the elegance of kernel methods in traditional machine learning.

B. LSTM BACK-PROPAGATION DERIVATION

This appendix provides a derivation of the weight and bias gradients for an LSTM network with respect to a given loss function, $\mathcal{L}$. Assume that there are N memory cells, n inputs, m outputs, and s time steps in the training sequence. We will also assume that all intermediate variables (e.g. h_t, o_t, etc.) are available for each time step from a forward pass of the network through the training sequence. The goal is to start with the output of the network and work backwards to obtain derivatives of the total loss function with respect to the various network parameters.

B.1 Output Layer

We begin at the output layer, which is described at time t by:

$$y_t = \sigma(Vh_t + b_2) = \sigma(\hat{y}_t) \tag{B.1}$$

We note that V is $m \times N$ and h_t is $N \times 1$. Using the chain rule, the derivative of the loss at time t is:

$$\frac{\partial \mathcal{L}_t}{\partial V} = \frac{\partial \mathcal{L}_t}{\partial y_t} \cdot \frac{\partial y_t}{\partial \sigma} \cdot \frac{\partial \sigma}{\partial \hat{y}_t} \cdot \frac{\partial \hat{y}_t}{\partial V} \tag{B.2}$$

Note for the sigmoid function:

$$\begin{aligned} \frac{\partial \sigma}{\partial \hat{y}_t} =& \{1 - \sigma(\hat{y}_t)\} \circ \sigma(\hat{y}_t) \\ =& (1 - y_t) \circ y_t \end{aligned} \tag{B.3}$$

Which is $m \times 1$ overall. Thus:

$$\boxed{\frac{\partial \mathcal{L}_t}{\partial V} = \frac{\partial \mathcal{L}_t}{\partial y_t}(1 - y_t) \circ y_t \otimes h_t} \tag{B.4}$$

Where $\otimes$ is the outer product between column vectors such that the left hand side is dimension $m \times N$. Similarly for the bias:

$$\frac{\partial \mathcal{L}_t}{\partial b_2} = \frac{\partial \mathcal{L}_t}{\partial y_t} \cdot \frac{\partial y_t}{\partial \sigma} \cdot \frac{\partial \sigma}{\partial \hat{y}_t} \cdot \frac{\partial \hat{y}_t}{\partial b_2} \tag{B.5}$$

$$\Rightarrow \boxed{\frac{\partial \mathcal{L}_t}{\partial b_2} = \frac{\partial \mathcal{L}_t}{\partial y_t}(1 - y_t) \circ y_t} \tag{B.6}$$

To aggregate the derivative over all time steps in the sequence we compute:

$$\frac{\partial \mathcal{L}}{\partial V} = \sum_{t=1}^{s} \frac{\partial \mathcal{L}_t}{\partial V} \tag{B.7a}$$

$$\frac{\partial \mathcal{L}}{\partial b_2} = \sum_{t=1}^{s} \frac{\partial \mathcal{L}_t}{\partial b_2} \tag{B.7b}$$

Finally, note that the loss for a single time step is the sum of the losses for the individual outputs:

$$\frac{\partial \mathcal{L}_t}{\partial y_t} = \sum_{j=1}^{m} \frac{\partial \mathcal{L}_t}{\partial y_{it}} \tag{B.8}$$

Therefore, given the LSTM memory state h_t along with the loss $\dfrac{\partial \mathcal{L}_t}{\partial y_t}$ at each time step from the forward pass, we can compute the parameter updates for the output layer using Eqs. B.7.

B.2 Memory Cell

Derivation of the parameter updates for the memory cells proceeds in similar fashion. First, it is necessary to know how the hidden state h_t affects the loss through the output layer:

$$\begin{aligned} \frac{\partial \mathcal{L}_t}{\partial h_t} &= \frac{\partial \mathcal{L}_t}{\partial y_t} \cdot \frac{\partial y_t}{\partial \sigma} \cdot \frac{\partial \sigma}{\partial \hat{y}_t} \cdot \frac{\partial \hat{y}_t}{\partial h_t} \\ \Rightarrow \frac{\partial \mathcal{L}_t}{\partial h_t} &= \frac{\partial \mathcal{L}_t}{\partial y_t} V^\intercal (1 - y_t) \circ y_t \end{aligned} \tag{B.9}$$

This is the contribution of h_t to the loss at the current time step $\mathcal{L}_t$. But since h_t is also an input to the memory cell at time $t + 1$, it will further contribute indirectly to the loss at all future steps (denoted $t+$). Thus, the total contribution to the loss from h_t is:

$$\frac{\partial \mathcal{L}}{\partial h_t} = \frac{\partial \mathcal{L}_t}{\partial h_t} + \frac{\partial \mathcal{L}_{t+}}{\partial h_t} \tag{B.10}$$

Since the second term is initially unknown, let us begin at the final time step $t = s$ of the training sequence so that only the first term is required. Hence,

$$\frac{\partial \mathcal{L}}{\partial h_s} = \frac{\partial \mathcal{L}_s}{\partial h_s} \tag{B.11}$$

Which can be calculated from Eq. B.9. This is enough to complete an initial pass through the equations to follow. For the sake of generality, we will assume $\frac{\partial \mathcal{L}}{\partial h_t}$ is known for an arbitrary time step then show how to obtain $\frac{\partial \mathcal{L}}{\partial h_{t-1}}$ for earlier time steps.

B.2.1 Output Gate

Starting at h_t of Fig. 2.4(b) and tracing the output path in reverse, we reach the output gate described by:

$$h_t = o_t \circ \tanh(c_t) \tag{B.12a}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) = \sigma(\hat{o_t}) \tag{B.12b}$$

We compute the partial derivatives of the loss for each of the new variables encountered. The output activation:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \hat{o_t}} =& \frac{\partial \mathcal{L}}{\partial h_t} \cdot \frac{\partial h_t}{\partial o_t} \cdot \frac{\partial o_t}{\partial \hat{o_t}} \\ =& \frac{\partial \mathcal{L}}{\partial h_t} \circ \tanh(c_t) \circ (1 - o_t) \circ o_t \end{aligned} \tag{B.13}$$

And cell state:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial c_t} =& \left(\frac{\partial \mathcal{L}}{\partial c_t}\right)_{h_t} + \left(\frac{\partial \mathcal{L}}{\partial c_t}\right)_{c_{t+}} \\ =& \frac{\partial \mathcal{L}}{\partial h_t} \cdot \frac{\partial h_t}{\partial c_t} + \left(\frac{\partial \mathcal{L}}{\partial c_t}\right)_{c_{t+}} \\ =& \frac{\partial \mathcal{L}}{\partial h_t} \circ o_t \circ \{1 - \tanh^2(c_t)\} + \left(\frac{\partial \mathcal{L}}{\partial c_t}\right)_{c_{t+}} \end{aligned} \tag{B.14}$$

The first term represents the loss contribution of c_t through its effect on h_t in Eq. B.12a. The second term arises for reasons similar to the second term in Eq. B.10 (i.e. because c_t is an input to c_{t+1}). As with Eq. B.10, we can neglect this term on our initial pass through the equations at $t = s$.

We now have everything we need to compute the gradients of the weights and biases associated with the output gate at time t:

$$\left(\frac{\partial \mathcal{L}}{\partial W_o}\right)_t = \frac{\partial \mathcal{L}}{\partial \hat{o_t}} \cdot \frac{\partial \hat{o_t}}{\partial W_o}$$

$$\Rightarrow \boxed{\left(\frac{\partial \mathcal{L}}{\partial W_o}\right)_t = \frac{\partial \mathcal{L}}{\partial \hat{o}_t} \otimes x_t} \tag{B.15}$$

The $(\cdot)_t$ notation indicates that this is the gradient of the total loss[1] $\mathcal{L}$ given the intermediate variables at time t. Similarly, for the recurrent weights:

$$\left(\frac{\partial \mathcal{L}}{\partial U_o}\right)_t = \frac{\partial \mathcal{L}}{\partial \hat{o}_t} \cdot \frac{\partial \hat{o}_t}{\partial U_o}$$

$$\Rightarrow \boxed{\left(\frac{\partial \mathcal{L}}{\partial U_o}\right)_t = \frac{\partial \mathcal{L}}{\partial \hat{o}_t} \otimes h_{t-1}} \tag{B.16}$$

And the bias:

$$\left(\frac{\partial \mathcal{L}}{\partial b_o}\right)_t = \frac{\partial \mathcal{L}}{\partial \hat{o}_t} \cdot \frac{\partial \hat{o}_t}{\partial b_o}$$

$$\Rightarrow \boxed{\left(\frac{\partial \mathcal{L}}{\partial b_o}\right)_t = \frac{\partial \mathcal{L}}{\partial \hat{o}_t}} \tag{B.17}$$

B.2.2 Forget Gate

The forget gate controls the retention of the cell state from one time step to the next. For the forward pass:

$$c_t = f_t \circ c_{t-1} + i_t \circ \tanh\left(W_c x_t + U_c h_t + b_c\right) \tag{B.18a}$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) = \sigma(\hat{f}_t) \tag{B.18b}$$

Thus:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \hat{f}_t} =& \frac{\partial \mathcal{L}}{\partial c_t} \cdot \frac{\partial c_t}{\partial f_t} \cdot \frac{\partial f_t}{\partial \hat{f}_t} \\ =& \frac{\partial \mathcal{L}}{\partial c_t} \circ c_{t-1} \circ (1 - f_t) \circ f_t \end{aligned} \tag{B.19}$$

Giving us:

$$\left(\frac{\partial \mathcal{L}}{\partial W_f}\right)_t = \frac{\partial \mathcal{L}}{\partial \hat{f}_t} \cdot \frac{\partial \hat{f}_t}{\partial W_f}$$

$$\Rightarrow \boxed{\left(\frac{\partial \mathcal{L}}{\partial W_f}\right)_t = \frac{\partial \mathcal{L}}{\partial \hat{f}_t} \otimes x_t} \tag{B.20}$$

[1] In contrast to $\mathcal{L}_t$, which would indicate the loss calculated only at time t and neglecting the feedback effect on future states.

$$\left(\frac{\partial \mathcal{L}}{\partial U_f}\right)_t = \frac{\partial \mathcal{L}}{\partial \hat{f}_t} \cdot \frac{\partial \hat{f}_t}{\partial U_f}$$

$$\Rightarrow \boxed{\left(\frac{\partial \mathcal{L}}{\partial U_f}\right)_t = \frac{\partial \mathcal{L}}{\partial \hat{f}_t} \otimes h_{t-1}} \tag{B.21}$$

$$\left(\frac{\partial \mathcal{L}}{\partial b_f}\right)_t = \frac{\partial \mathcal{L}}{\partial \hat{f}_t} \cdot \frac{\partial \hat{f}_t}{\partial b_f}$$

$$\Rightarrow \boxed{\left(\frac{\partial \mathcal{L}}{\partial b_f}\right)_t = \frac{\partial \mathcal{L}}{\partial \hat{f}_t}} \tag{B.22}$$

B.2.3 Input Gate

Similarly, for the input gate, $i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \hat{i_t}} =& \frac{\partial \mathcal{L}}{\partial c_t} \cdot \frac{\partial c_t}{\partial i_t} \cdot \frac{\partial i_t}{\partial \hat{i_t}} \\ =& \frac{\partial \mathcal{L}}{\partial c_t} \circ \tanh\left(W_c x_t + U_c h_{t-1} + b_c\right) \circ (1 - i_t) \circ i_t\end{aligned} \tag{B.23}$$

Giving:

$$\left(\frac{\partial \mathcal{L}}{\partial W_i}\right)_t = \frac{\partial \mathcal{L}}{\partial \hat{i_t}} \cdot \frac{\partial \hat{i_t}}{\partial W_i}$$

$$\Rightarrow \boxed{\left(\frac{\partial \mathcal{L}}{\partial W_i}\right)_t = \frac{\partial \mathcal{L}}{\partial \hat{i_t}} \otimes x_t} \tag{B.24}$$

$$\left(\frac{\partial \mathcal{L}}{\partial U_i}\right)_t = \frac{\partial \mathcal{L}}{\partial \hat{i_t}} \cdot \frac{\partial \hat{i_t}}{\partial U_i}$$

$$\Rightarrow \boxed{\left(\frac{\partial \mathcal{L}}{\partial U_i}\right)_t = \frac{\partial \mathcal{L}}{\partial \hat{i_t}} \otimes h_{t-1}} \tag{B.25}$$

$$\left(\frac{\partial \mathcal{L}}{\partial b_i}\right)_t = \frac{\partial \mathcal{L}}{\partial \hat{i_t}} \cdot \frac{\partial \hat{i_t}}{\partial b_i}$$

$$\Rightarrow \boxed{\left(\frac{\partial \mathcal{L}}{\partial b_i}\right)_t = \frac{\partial \mathcal{L}}{\partial \hat{i_t}}} \tag{B.26}$$

The input gate controls the absorption of information into the cell state via the tanh term in Eq. B.18a. Let $a_t = \tanh\left(W_c x_t + U_c h_{t-1} + b_c\right) = \tanh\left(\hat{a}_t\right)$. Then we can also obtain:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \hat{a}_t} =& \frac{\partial \mathcal{L}}{\partial c_t} \cdot \frac{\partial c_t}{\partial a_t} \cdot \frac{\partial a_t}{\partial \hat{a}_t} \\ =& \frac{\partial \mathcal{L}}{\partial c_t} \circ i_t \circ \left(1 - a_t^2\right)\end{aligned} \tag{B.27}$$

This provides the loss gradients with respect to the cell state weights and biases:

$$\left(\frac{\partial \mathcal{L}}{\partial W_c}\right)_t = \frac{\partial \mathcal{L}}{\partial \hat{a}_t} \cdot \frac{\partial \hat{a}_t}{\partial W_c}$$

$$\Rightarrow \boxed{\left(\frac{\partial \mathcal{L}}{\partial W_c}\right)_t = \frac{\partial \mathcal{L}}{\partial \hat{a}_t} \otimes x_t} \tag{B.28}$$

$$\left(\frac{\partial \mathcal{L}}{\partial U_c}\right)_t = \frac{\partial \mathcal{L}}{\partial \hat{a}_t} \cdot \frac{\partial \hat{a}_t}{\partial U_c}$$

$$\Rightarrow \boxed{\left(\frac{\partial \mathcal{L}}{\partial U_c}\right)_t = \frac{\partial \mathcal{L}}{\partial \hat{a}_t} \otimes h_{t-1}} \tag{B.29}$$

$$\left(\frac{\partial \mathcal{L}}{\partial b_c}\right)_t = \frac{\partial \mathcal{L}}{\partial \hat{a}_t} \cdot \frac{\partial \hat{a}_t}{\partial b_i}$$

$$\Rightarrow \boxed{\left(\frac{\partial \mathcal{L}}{\partial b_c}\right)_t = \frac{\partial \mathcal{L}}{\partial \hat{a}_t}} \tag{B.30}$$

B.2.4 Aggregation and Recursion

The loss gradients over the entire training sequence can be found by summing the gradients contributed by individual time steps. For example:

$$\frac{\partial \mathcal{L}}{\partial W_c} = \sum_{i=1}^{s} \left(\frac{\partial \mathcal{L}}{\partial W_c}\right)_t \tag{B.31}$$

In order to compute the gradients for times $t < s$, however, it is necessary to determine $\frac{\partial \mathcal{L}_{t+}}{\partial h_t}$ in Eq. B.10 and $\left(\frac{\partial \mathcal{L}}{\partial c_t}\right)_{c_{t+}}$ in Eq. B.14. Since h_{t-1} is used in the activation of the input, output, and forget gates, we can solve for its effect on the loss at the current time step and those afterwards by writing:

$$\frac{\partial \mathcal{L}_{t-1+}}{\partial h_{t-1}} = \frac{\partial \mathcal{L}}{\partial \hat{o}_t} \cdot \frac{\partial \hat{o}_t}{\partial h_{t-1}} + \frac{\partial \mathcal{L}}{\partial \hat{f}_t} \cdot \frac{\partial \hat{f}_t}{\partial h_{t-1}} + \frac{\partial \mathcal{L}}{\partial \hat{i}_t} \cdot \frac{\partial \hat{i}_t}{\partial h_{t-1}} + \frac{\partial \mathcal{L}}{\partial \hat{a}_t} \cdot \frac{\partial \hat{a}_t}{\partial h_{t-1}}$$

$$\Rightarrow \boxed{\frac{\partial \mathcal{L}_{t-1+}}{\partial h_{t-1}} = U_o \frac{\partial \mathcal{L}}{\partial \hat{o}_t} + U_f \frac{\partial \mathcal{L}}{\partial \hat{f}_t} + U_i \frac{\partial \mathcal{L}}{\partial \hat{i}_t} + U_c \frac{\partial \mathcal{L}}{\partial \hat{a}_t}} \tag{B.32}$$

This provides $\frac{\partial \mathcal{L}_{t+}}{\partial h_t}$ in the next iteration of Eq. B.10 for an earlier time.

To prepare Eq. B.14 at the next time step we similarly solve:

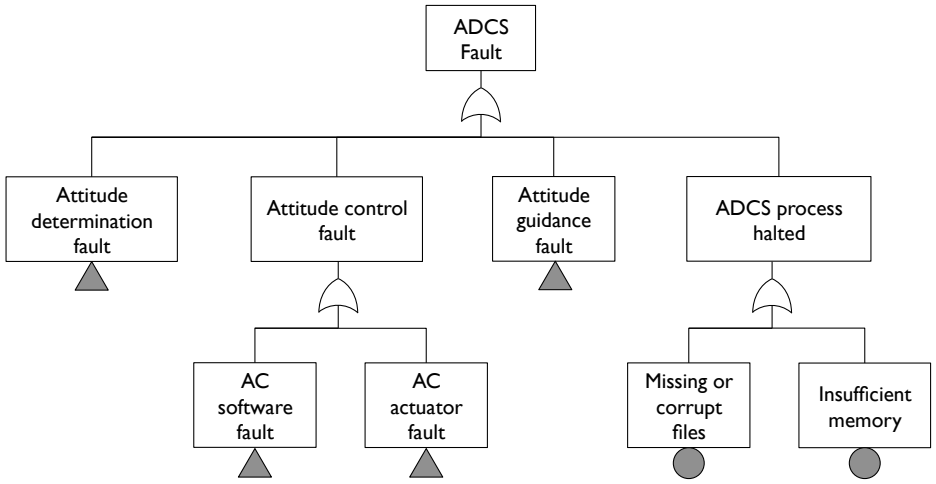
$$\left(\frac{\partial \mathcal{L}}{\partial c_{t-1}}\right)_{c_{t-1+}} = \frac{\partial \mathcal{L}}{\partial c_t} \cdot \frac{\partial c_t}{\partial c_{t-1}}$$

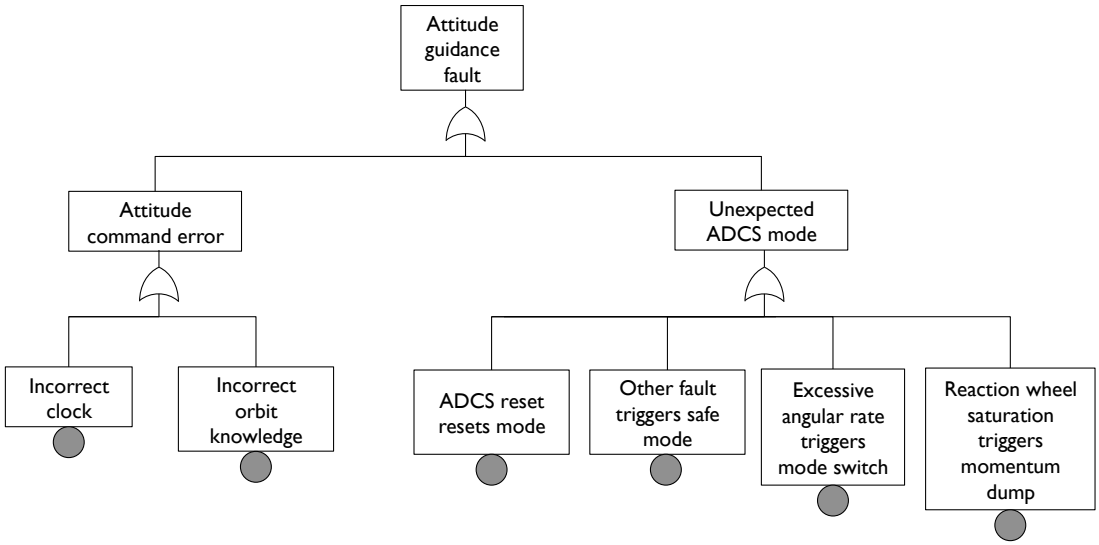
$$\Rightarrow \boxed{\left(\frac{\partial \mathcal{L}}{\partial c_{t-1}}\right)_{c_{t-1+}} = f_t \circ \frac{\partial \mathcal{L}}{\partial c_t}} \tag{B.33}$$

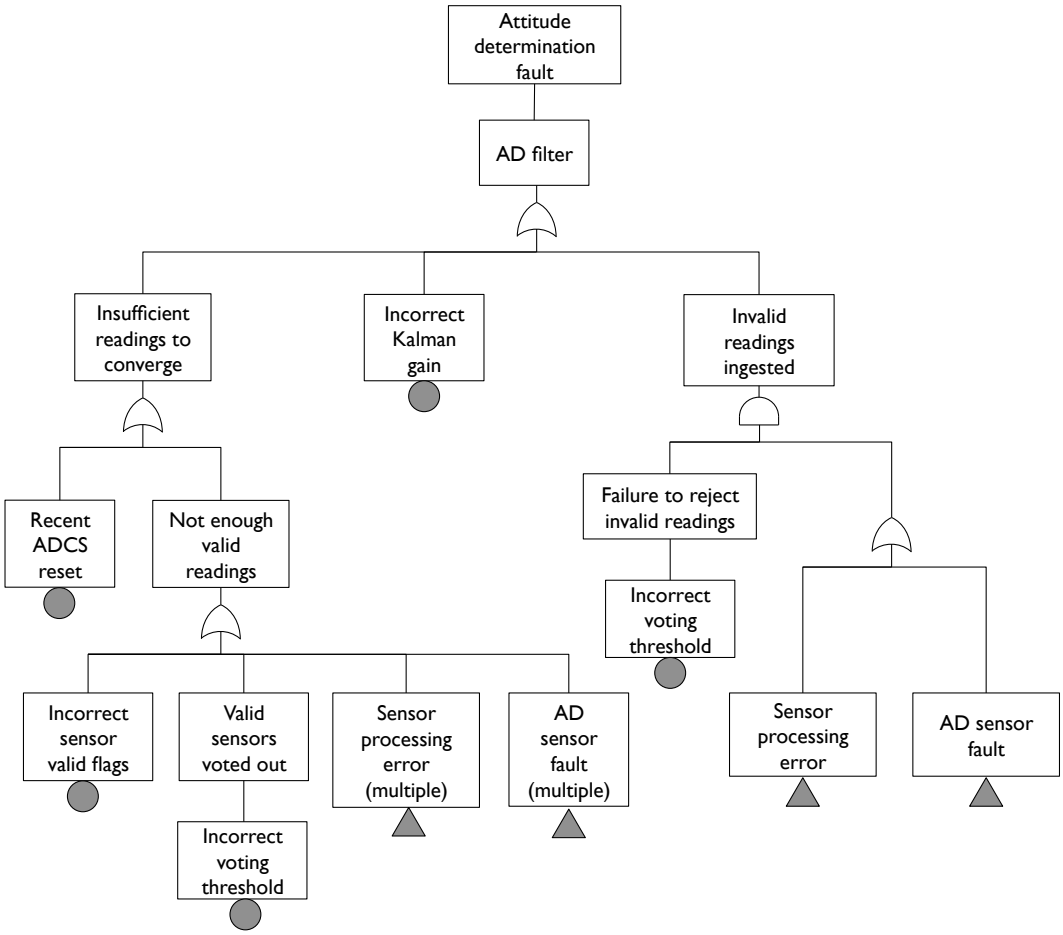
Therefore, starting at the final time step of the training sequence, we can recursively compute Eqs. B.9 − B.33 in order to generate all of the summation terms to obtain $\frac{\partial \mathcal{L}}{\partial \{W, U, b\}}$ in the manner of Eq. B.31. The gradients $\frac{\partial \mathcal{L}}{\partial \{V, b_2\}}$ are provided independently from Eqs. B.7.

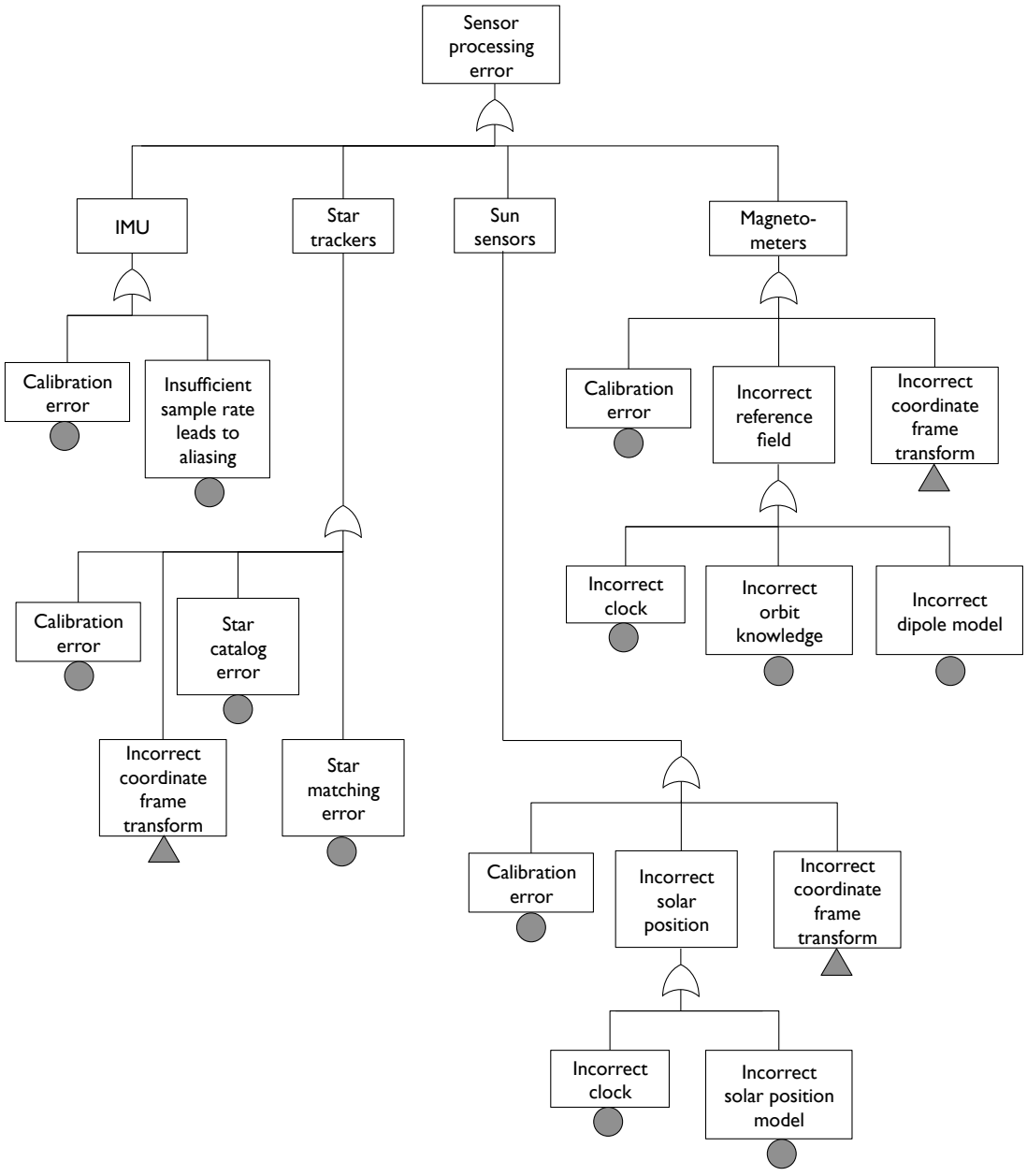
C. ADCS FAULT TREES

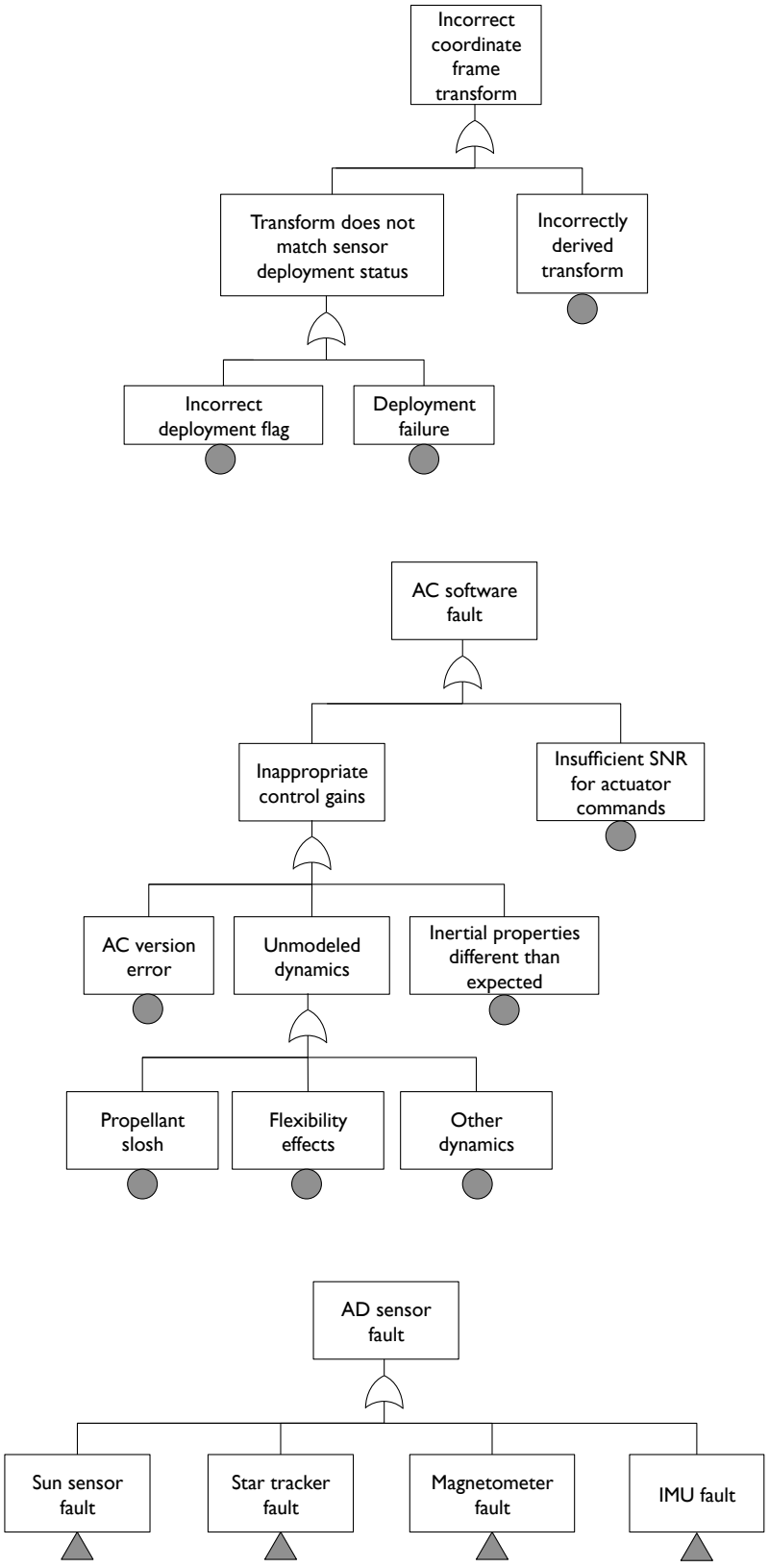
This appendix provides a fault tree covering 90 different faults for a CubeSat ADCS. The CubeSat is assumed to be Earth-orbiting. Circles located underneath events are basic events while triangles represent lower-level fault trees given on another page.

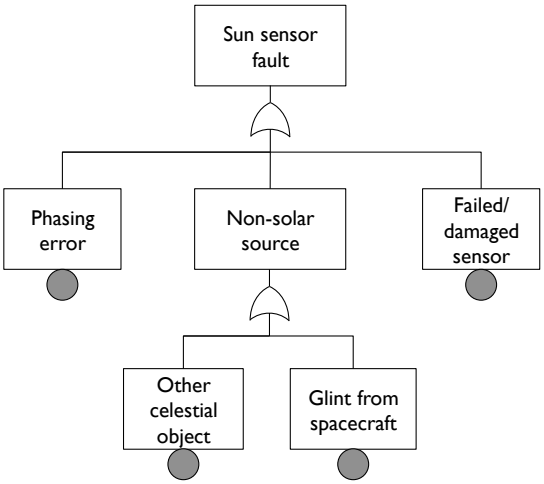


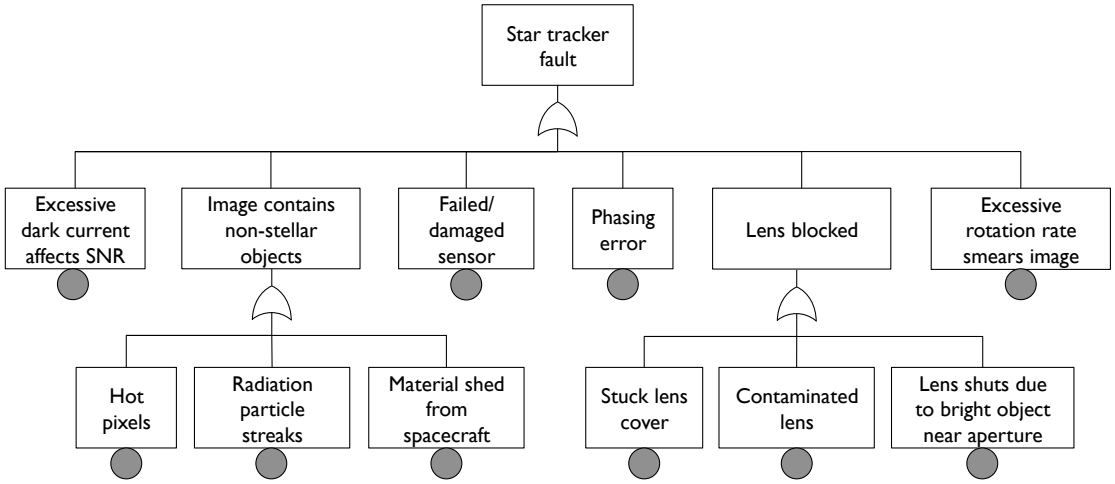


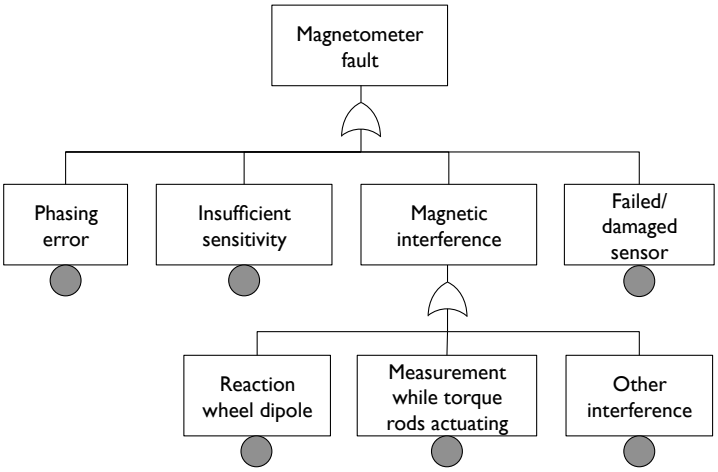

Attitude determination fault
AD filter
Insufficient readings to converge
Incorrect Kalman gain
Invalid readings ingested
Recent ADCS reset
Not enough valid readings
Failure to reject invalid readings
Incorrect voting threshold
Incorrect sensor valid flags
Valid sensors voted out
Sensor processing error (multiple)
AD sensor fault (multiple)
Sensor processing error
AD sensor fault
Incorrect voting threshold

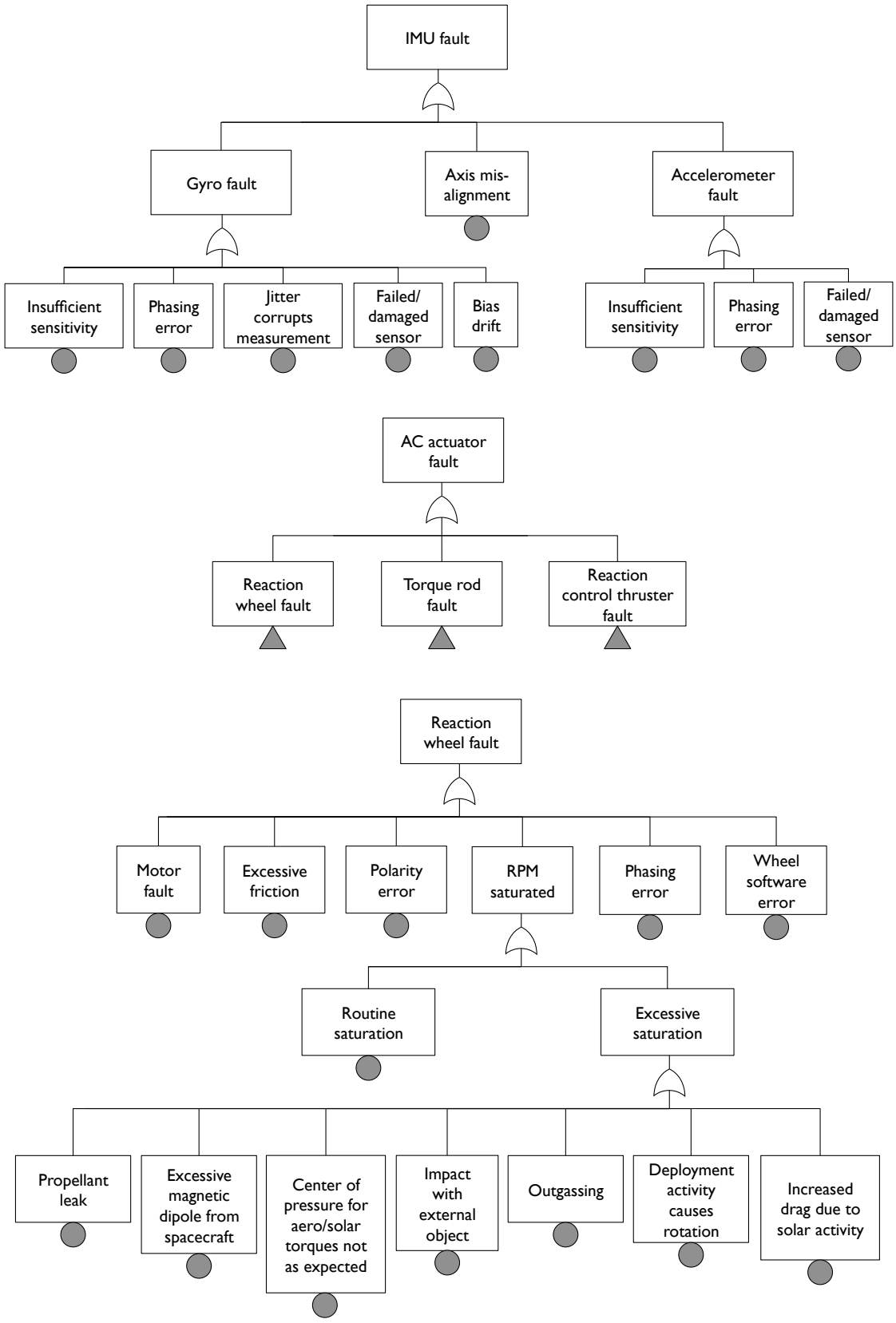

Sensor processing error
IMU
Star trackers
Sun sensors
Magneto-meters
Calibration error
Insufficient sample rate leads to aliasing
Calibration error
Star catalog error
Incorrect coordinate frame transform
Star matching error
Calibration error
Incorrect reference field
Incorrect coordinate frame transform
Incorrect clock
Incorrect orbit knowledge
Incorrect dipole model
Calibration error
Incorrect solar position
Incorrect coordinate frame transform
Incorrect clock
Incorrect solar position model

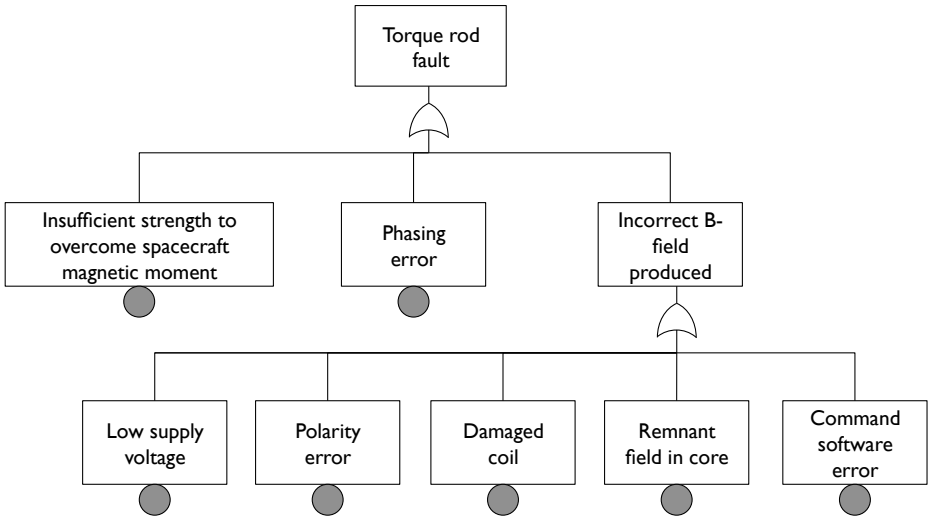

Incorrect coordinate frame transform
Transform does not match sensor deployment status
Incorrectly derived transform
Incorrect deployment flag
Deployment failure
AC software fault
Inappropriate control gains
Insufficient SNR for actuator commands
AC version error
Unmodeled dynamics
Inertial properties different than expected
Propellant slosh
Flexibility effects
Other dynamics
AD sensor fault
Sun sensor fault
Star tracker fault
Magnetometer fault
IMU fault

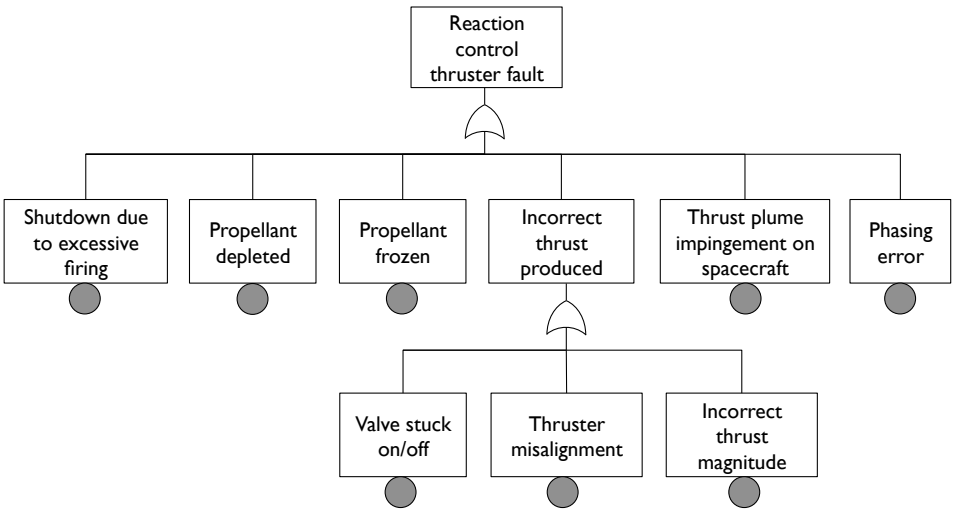

Sun sensor fault
Phasing error
Non-solar source
Failed/ damaged sensor
Other celestial object
Glint from spacecraft


Star tracker fault
Excessive dark current affects SNR
Image contains non-stellar objects
Failed/ damaged sensor
Phasing error
Lens blocked
Excessive rotation rate smears image
Hot pixels
Radiation particle streaks
Material shed from spacecraft
Stuck lens cover
Contaminated lens
Lens shuts due to bright object near aperture


Magnetometer fault
Phasing error
Insufficient sensitivity
Magnetic interference
Failed/ damaged sensor
Reaction wheel dipole
Measurement while torque rods actuating
Other interference


IMU fault
Gyro fault
Axis mis-alignment
Accelerometer fault
Insufficient sensitivity
Phasing error
Jitter corrupts measurement
Failed/ damaged sensor
Bias drift
Insufficient sensitivity
Phasing error
Failed/ damaged sensor
AC actuator fault
Reaction wheel fault
Torque rod fault
Reaction control thruster fault
Reaction wheel fault
Motor fault
Excessive friction
Polarity error
RPM saturated
Phasing error
Wheel software error
Routine saturation
Excessive saturation
Propellant leak
Excessive magnetic dipole from spacecraft
Center of pressure for aero/solar torques not as expected
Impact with external object
Outgassing
Deployment activity causes rotation
Increased drag due to solar activity


Torque rod fault
Insufficient strength to overcome spacecraft magnetic moment
Phasing error
Incorrect B-field produced
Low supply voltage
Polarity error
Damaged coil
Remnant field in core
Command software error


Reaction control thruster fault
Shutdown due to excessive firing
Propellant depleted
Propellant frozen
Incorrect thrust produced
Thrust plume impingement on spacecraft
Phasing error
Valve stuck on/off
Thruster misalignment
Incorrect thrust magnitude

D. ADDITIONAL EXAMPLES

D.1 Fault Simulator Examples

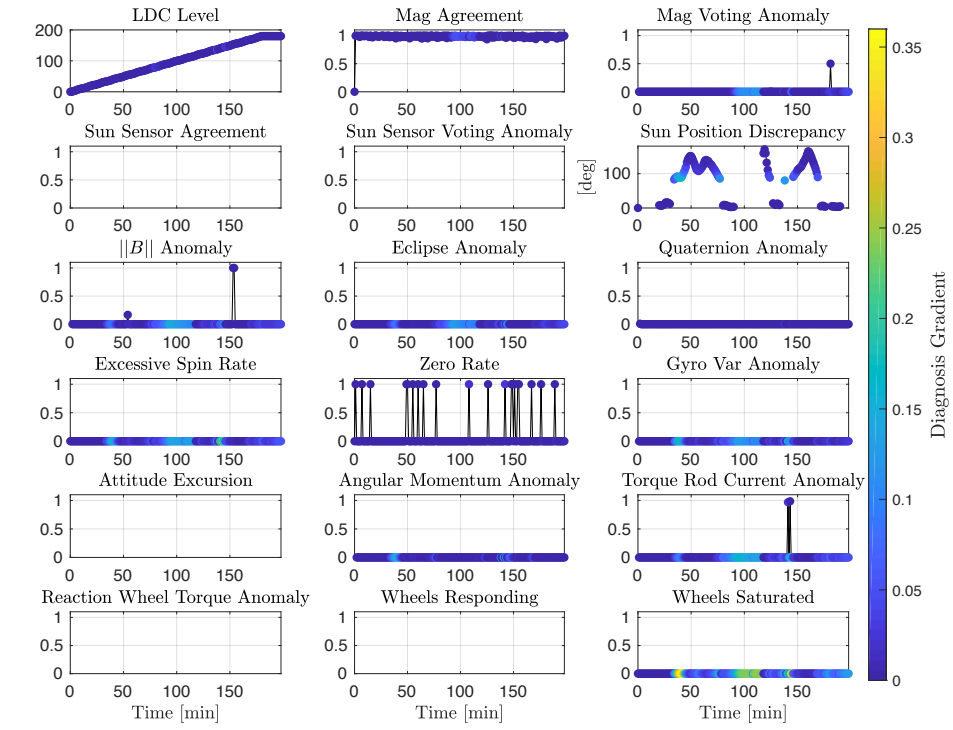
D.1.1 False Sun Error

Figure D.1 showcases the LSTM's ability to isolate an intermittent fault. Whether the sun sensors register a non-solar source depends on the spacecraft's attitude. In this example the attitude is uncontrolled (detumble mode), allowing the fault to appear and disappear at irregular intervals.

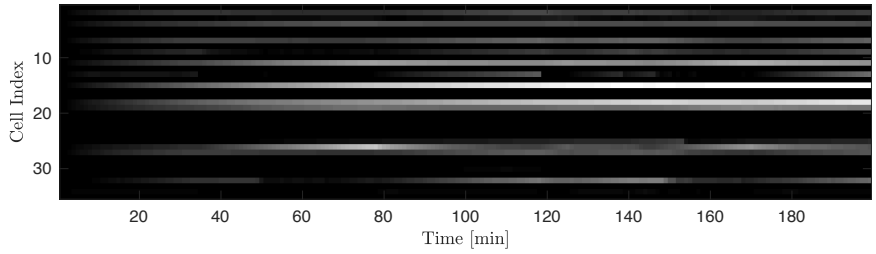
D.1.2 Gyro Phasing Error

This example reverses the X and Z-axis gyros and simulates the resulting ADCS behavior in no torques mode. Figure D.2 shows the resulting anomalies alongside the LSTM network's diagnosis. Although the network initially classifies the fault as a gyro calibration error, it eventually converges to the correct "Gyro phasing error" diagnosis. This is an interesting example because the difference between the two faults is extremely subtle when operating in a mode with an uncontrolled attitude. In fact, on first inspection it is not at all clear how the LSTM network manages to reject the initial gyro calibration error and achieve the correct diagnosis. This is where the gradient with respect to the inputs proves useful. It reveals that the three most important signals to the network's ultimate diagnosis are:

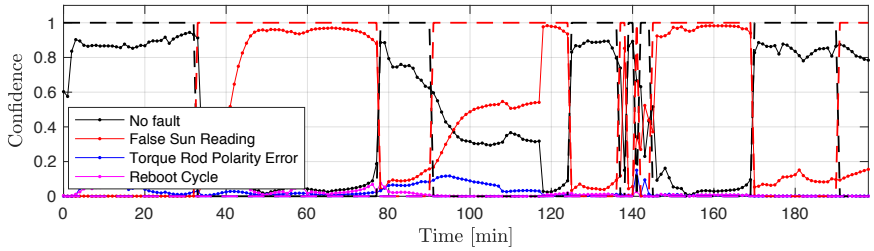
- The zero rotation rate
- The quaternion anomaly
- The gyro variance anomaly



(a) Anomaly inputs



(b) LSTM cell state evolution



(c) Fault confidence outputs

Figure D.1: Isolating cases when the sun sensors are registering a non-solar bright object. The LSTM network is able to identify the affected intervals even though the fault is highly intermittent.

Focusing on these signals explains the network's decision. Zero rotation rates are unlikely to occur under a gyro calibration error because this fault most often leads to higher-than-actual estimates of the rotation rate. The diagnosis therefore quickly switches once the zero rate first appears in Fig. D.2(a). The lack of a gyro variance anomaly despite the consistent quaternion-B discrepancy helps to further distinguish the gyro phasing error from a calibration error or attitude jitter.

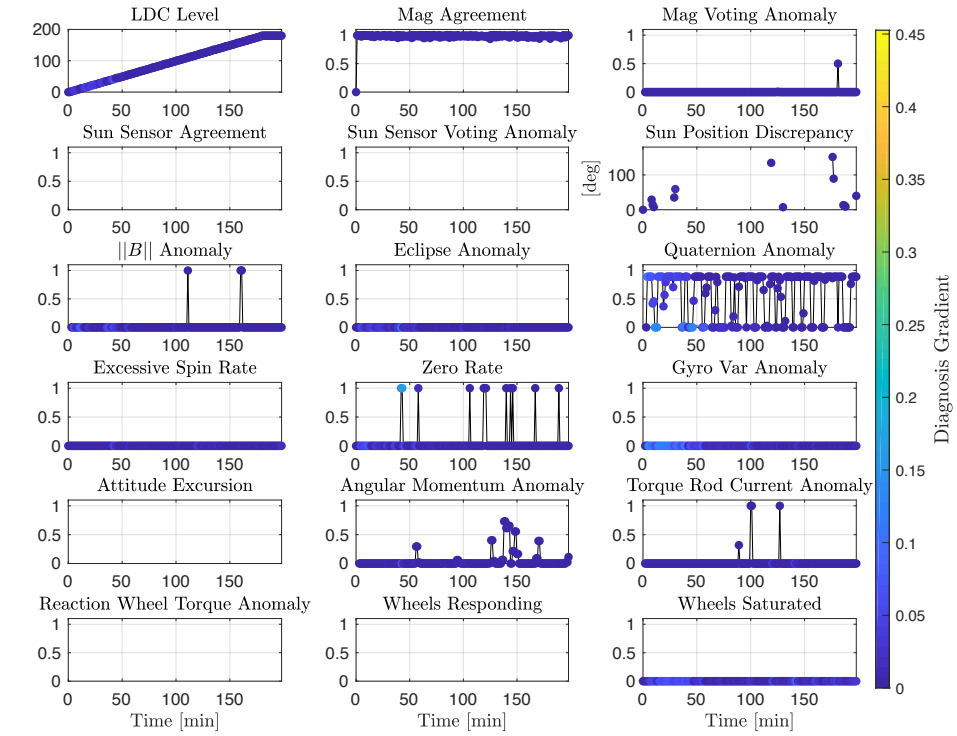
D.2 LightSail 2 Examples

D.2.1 Reaction Wheel Fault

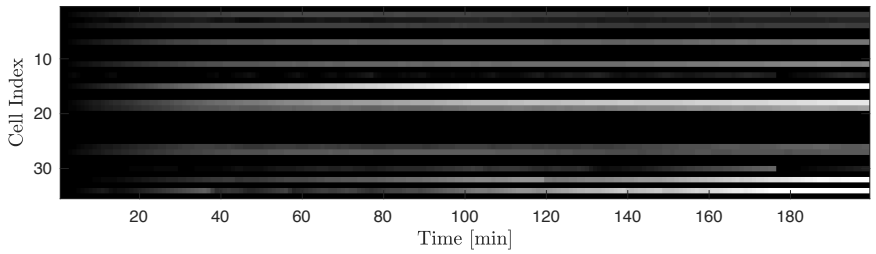
Figure D.3 applies the LSTM network to diagnose the issues that were encountered with LightSail 2's reaction wheel during testing of the spacecraft's solar sailing mode prior to sail deployment. The wheel actuates only sporadically, causing torque anomalies and a lack of response to commands. The LSTM network interprets these anomalies as the result of a reaction wheel torque fault. This does not match the expected diagnosis of "unresponsive reaction wheel," but in reality the fault is more aptly described as a combination of both faults. The pattern of anomalies does not fit fully with either case. As a result, the LSTM's neurons accumulate little activation (visible by the dark colors) and the confidence of all fault hypotheses decays to nearly zero. When this occurs, it suggests that the anomalies may be due to a new fault that the network has not learned to diagnose in training.

D.2.2 Nominal Solar Sailing

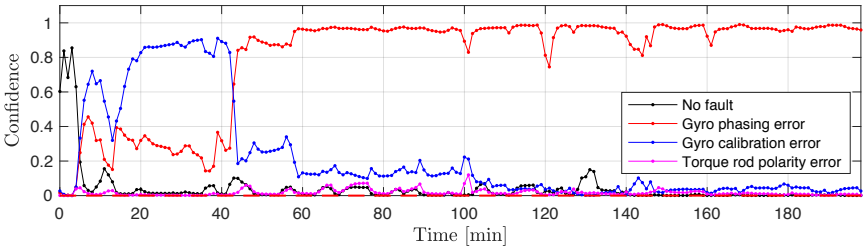
Finally, Fig. D.4 shows a lengthy interval of mostly nominal solar sailing. Except for a brief discrepancy between the two magnetometers, the LSTM network correctly recognizes the nearly 24 hours of nominal performance.



(a) Anomaly inputs

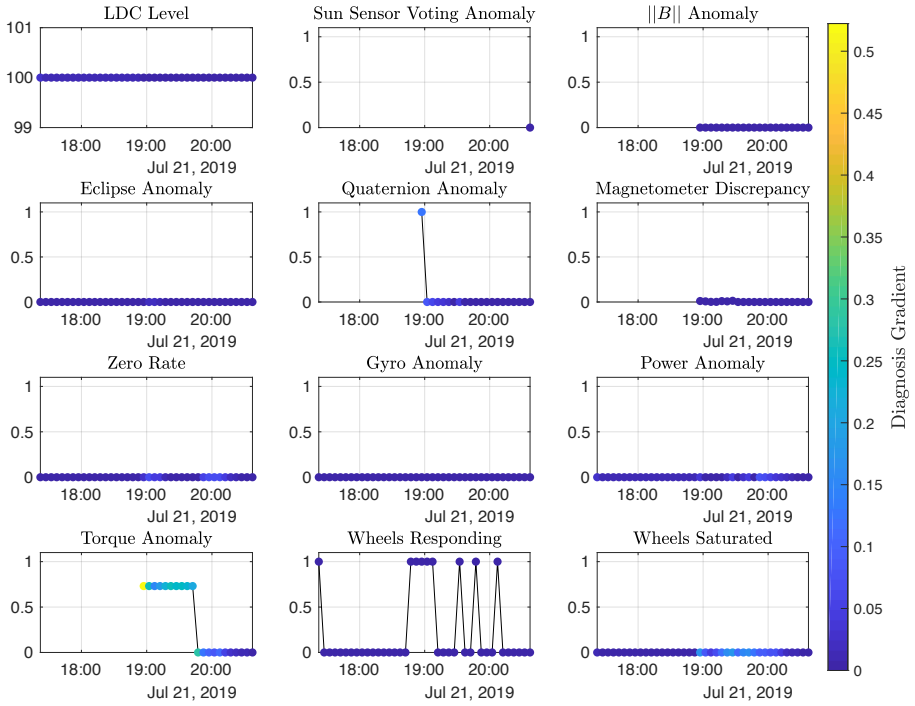


(b) LSTM cell state evolution

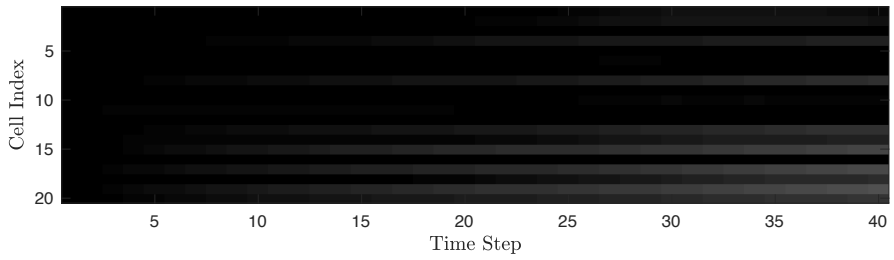


(c) Fault confidence outputs

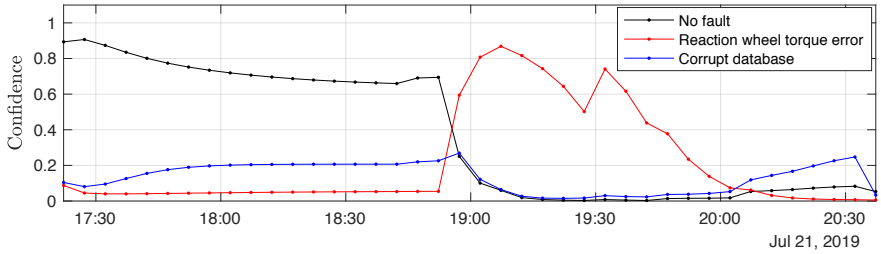
Figure D.2: Isolating a gyro phasing error while in no torques mode. Inspection of the diagnosis gradient reveals how the LSTM network is able to isolate the correct diagnosis from similar faults.



(a) Anomaly inputs

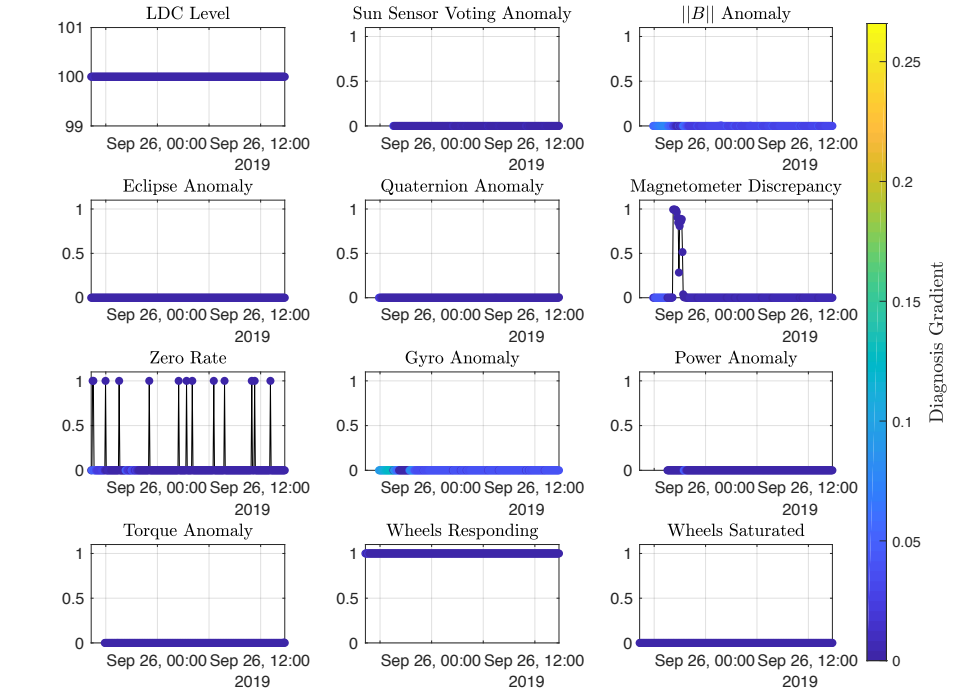


(b) LSTM cell state evolution

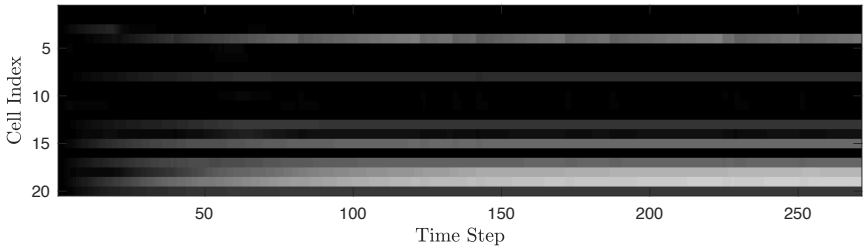


(c) Fault confidence outputs

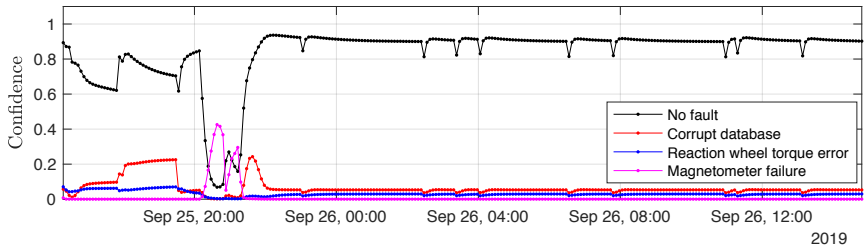
Figure D.3: Anomalies and fault confidences when processing the reaction wheel troubles encountered during LightSail 2's checkout.



(a) Anomaly inputs



(b) LSTM cell state evolution



(c) Fault confidence outputs

Figure D.4: Successful recognition of one of LightSail 2's best days of solar sailing.

VITA

Justin Rhys Mansell

Justin Mansell is a doctoral candidate at Purdue University. His research interests include space mission design, machine learning, and space flight operations. He has completed internships at Idaho National Laboratory and Maxar technologies where he investigated methods of streamlining radioisotope generator fueling and mission planning for Canadarm2. Since 2018, he has served as the flight mechanics engineer for the LightSail 2 solar sail satellite. His hometown is Calgary, Canada.

Education

Ph.D. Aeronautics and Astronautics, Purdue University, August 2020. Thesis title: "Deep Learning Fault Protection Applied to Spacecraft Attitude Determination and Control."

M.S. Aeronautics and Astronautics, Purdue University, May 2017. Thesis title: "Adaptive Continuation Strategies for Indirect Trajectory Optimization."

B.Sc. Physics, University of Calgary, June 2015.

Journal Publications

J. Mansell, N. Kolencherry, K. Hughes, A. Arora, H.S. Chye, K. Coleman, J. Elliott, S. Fulton, N. Hobar, B. Libben, Y. Lu, J. Millane, A. Mudek, L. Podesta, J. Pouplin, E. Shibata, G. Smith, B. Tackett, T. Ukai, P. Witsberger, S. Saikia, "Oceanus: A multi-spacecraft flagship mission concept to explore Saturn and Uranus," *Advances in Space Research*, vol. 59, no. 9, pp. 2407–2433, 2017.

J.R. Mansell and M.J. Grant, "Adaptive Continuation Strategy for Indirect Hypersonic Trajectory Optimization," *Journal of Spacecraft and Rockets*, vol. 55, no. 4, pp. 818–828, 2018.

J.R. Mansell, S. Dickmann, and D.A. Spencer, "Swarm optimization of lunar transfers from Earth orbit with operational constraints," *The Journal of the Astronautical Sciences*, pp. 1–22, 2019.

D.A. Spencer, B. Betts, J.M. Bellardo, A. Diaz, B. Plante, and J.R. Mansell, "The LightSail 2 solar sailing technology demonstration," *Advances in Space Research*, accepted 22 June 2020.

Conference Publications

J.R. Mansell, J. Berry, J. Quint, and M.J. Wang, "Optimizing MMRTG Fueling and Testing For Future Campaigns," Nuclear and Emerging Technologies for Space 2018, Las Vegas, NV Feb. 26 – Mar. 1, 2018.

J.R. Mansell, S. Dickmann, and D.A. Spencer, "Swarm Optimization of Lunar Transfers From Earth Orbit with Radiation Dose Constraints," 29th AAS/AIAA Space Flight Mechanics meeting, AAS 19-244, Ka'anapali, HI, 13–17 January, 2019.

J.R. Mansell and D.A. Spencer, "Data-driven Fault Detection and Isolation for Small Spacecraft," 70th International Astronautical Congress, Washington, DC, 21–25 October, 2019.

J.R. Mansell, D.A. Spencer, B. Plante, A. Diaz, J.M. Bellardo, and B. Betts, "Orbit and Attitude Performance of the LightSail 2 Solar Sail Spacecraft," AIAA SciTech Forum, AIAA 2020-2177, Orlando, FL, 6–10 January, 2020.