Spring 5-8-2023

An Online Adaptive Machine Learning Framework for Autonomous Fault Detection

Nolan Coulter
Embry-Riddle Aeronautical University, coultern@my.erau.edu

AN ONLINE ADAPTIVE MACHINE LEARNING FRAMEWORK FOR AUTONOMOUS FAULT DETECTION

By

Nolan Coulter

A Dissertation Submitted to the Faculty of Embry-Riddle Aeronautical University

In Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy in Aerospace Engineering

May 2023

Embry-Riddle Aeronautical University

Daytona Beach, Florida

AN ONLINE ADAPTIVE MACHINE LEARNING FRAMEWORK

FOR AUTONOMOUS FAULT DETECTION

By

Nolan Coulter

This Dissertation was prepared under the direction of the candidate's Dissertation Committee Chair, Dr. Hever Moncayo, Department of Aerospace Engineering, and has been approved by the members of the Dissertation Committee. It was submitted to the Office of the Senior Vice President for Academic Affairs and Provost, and was accepted in the partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Aerospace Engineering.

DISSERTATION COMMITTEE

Hever Moncayo Digitally signed by Hever Moncayo Date: 2023.04.20 14:50:36 -04'00'

Chair, Dr. Hever Moncayo

Troy Henderson Digitally signed by Troy Henderson Date: 2023.04.17 11:14:07 -04'00'

Member, Dr. Troy Henderson

K. Merve Dogan Digitally signed by K. Merve Dogan Date: 2023.04.17 10:41:52 -04'00'

Member, Dr. Kadriye Merve Dogan

Richard S. Stansbury Digitally signed by Richard S. Stansbury Date: 2023.04.17 11:00:55 -04'00'

Member, Dr. Richard Stansbury

Maj Mirmirani Digitally signed by Maj Mirmirani Date: 2023.04.17 17:42:33 -04'00'

Member, Dr. Maj Mirmirani

Graduate Program Coordinator,
Dr. Sirish Namilae

Date

Dean of the College of Engineering,
Dr. James W. Gregory

Date

Senior Vice President for Academic Affairs and Provost,
Dr. Lon Moeller

Date

With love and gratitude, I dedicate this work to those who have been my constant source of strength and inspiration.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my committee members, Dr. Dogan, Dr. Henderson, Dr. Mirmirani, and Dr. Stansbury, for their invaluable guidance, support, and expertise throughout my research journey. Their insights and constructive feedback significantly contributed to the development of my work, and I am truly grateful for their dedication to my success.

A special thanks goes to my advisor, Dr. Moncayo, who has been a mentor since my Master's degree and throughout my graduate academic career. His unwavering support, encouragement, and wisdom have been instrumental in shaping my research and fostering my growth as a scholar. I am truly honored to have had the opportunity to learn from and work with him.

To my parents, I am forever grateful for your unwavering love and support. Your belief in my abilities and your constant encouragement have been the foundation upon which I built my academic pursuits. Thank you for always being there for me, providing a shoulder to lean on, and inspiring me to chase my dreams.

Lastly, I would like to express my heartfelt appreciation to my fiancée, Karina. Lifting me up in moments of doubt, you have been my rock and my source of strength throughout this journey. Overcoming challenges together, we have grown as a team, and your love has made all the difference. Venturing into the unknown, we embarked on this adventure hand-in-hand, and I am eternally grateful for your support, understanding, and encouragement. Every step of the way, you have been a guiding light.

ABSTRACT

The increasing complexity and autonomy of modern systems, particularly in the aerospace industry, demand robust and adaptive fault detection and health management solutions. The development of a data-driven fault detection system that can adapt to varying conditions and system changes is critical to the performance, safety, and reliability of these systems. This dissertation presents a novel fault detection approach based on the integration of the artificial immune system (AIS) paradigm and Online Support Vector Machines (OSVM). Together, these algorithms create the Artificial Immune System augemented Online Support Vector Machine (AISOSVM).

The AISOSVM framework combines the strengths of the AIS and OSVM to create a fault detection system that can effectively identify faults in complex systems while maintaining adaptability. The framework is designed using Model-Based Systems Engineering (MBSE) principles, employing the Capella tool and the Arcadia methodology to develop a structured, integrated approach for the design and deployment of the data-driven fault detection system. A key contribution of this research is the development of a Clonal Selection Algorithm that optimizes the OSVM hyperparameters and the V-Detector algorithm parameters, resulting in a more effective fault detection solution. The integration of the AIS in the training process enables the generation of synthetic abnormal data, mitigating the need for engineers to gather large amounts of failure data, which can be impractical.

The AISOSVM also incorporates incremental learning and decremental unlearning for the Online Support Vector Machine, allowing the system to adapt online using lightweight computational processes. This capability significantly improves the efficiency of fault detection systems, eliminating the need for offline retraining and redeployment.

Reinforcement Learning (RL) is proposed as a promising future direction for the AISOSVM, as it can help autonomously adapt the system performance in near real-time, further mitigating the need for acquiring large amounts of system data for training, and improving the efficiency of the adaptation process by intelligently selecting the best samples to learn from.

The AISOSVM framework was applied to real-world scenarios and platform models, demonstrating its effectiveness and adaptability in various use cases. The combination of the AIS and OSVM, along with the online learning and RL integration, provides a robust and adaptive solution for fault detection and health management in complex autonomous systems.

In conclusion, this dissertation presents a significant contribution to the field of fault detection and health management by integrating the artificial immune system paradigm with Online Support Vector Machines, developing a structured, integrated approach for designing and deploying data-driven fault detection systems, and implementing reinforcement learning for online, autonomous adaptation of fault management systems. The AISOSVM framework offers a promising solution to address the challenges of fault detection in complex, autonomous systems, with potential applications in a wide range of industries beyond aerospace.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

α_i	Lagrange multiplier
β	Coefficient sensitivities
$\boldsymbol{w}$	Support vector weighting matrix
$\boldsymbol{x}$	Data sample
γ	Q-Learning award discount factor
γ_l	Perturbation coefficients
$\overline{S}$	*Nonself* region within a self-nonself feature space
ϕ	Kernel function
π^*	Optimal Q-Learning policy function
ξ	Model slack variable
b	Bias term
C	Regularization or penalty factor
c_x	Center of a self (s) or nonself (a) detector
D_a	Detector set for nonself detectors
D_s	Detector set for self samples
E	Error support vector set
g_i	Support vector machine solution partial derivative
h_i	Support vector machine solution partial derivative
M	Margin support vector set

N	Total number of features selected
N_{fs}	Total number of feature space projections
Q	Q-learning state action value for determining best action
R	Reserve vector set
r_x	Radius of a self (s) or nonself (a) detector
R_{t+1}	Q-Learning reward at epoch
S	*Self* region within a self-nonself feature space
U	Unlearned vector set
y	Support vector machine predicted class output

1 Introduction

In this introductory chapter, the stage is set for the research presented in the dissertation by discussing the importance and challenges associated with fault detection in complex systems. The chapter begins by highlighting the need for fault detection in various applications and the significance of developing innovative solutions. Subsequently, a literature survey is performed for fault detection methodologies with an overview of the motivation behind the data-driven approach. Finally, a discussion of the relevant research questions, methods, and scope of the study follow with a detailed structure of the dissertation.

1.1 The Need for Fault Detection

Fault detection is of paramount importance in a wide range of applications, from industrial systems and vehicles to autonomous platforms and spacecraft. The capability to identify and diagnose faults in real-time enables systems to operate safely and efficiently, reduces the risk of catastrophic failures, and prolongs the lifespan of the equipment. Early and accurate fault detection can lead to timely maintenance, avoiding costly repairs and reducing the overall cost of ownership. Furthermore, fault detection contributes to improving the reliability and dependability of systems, which is essential for mission-critical applications where the cost of failure is extremely high.

The growing complexity and interconnectivity of systems have significantly increased the challenge of fault detection. Modern systems often consist of multiple interconnected subsystems that interact in intricate and nonlinear ways, making the identification of faults more difficult. In addition, the increasing reliance on autonomous systems necessitates the development of advanced fault detection methods that can operate in real-time and adapt to evolving conditions. As a result, there is a pressing need for innovative fault detection methodologies that can cope with the challenges posed by complex and dynamic systems.

Traditionally, fault detection methods have been based on model-driven approaches, which rely on mathematical models of the system's behavior to identify deviations from the expected performance. While these methods have proven effective in some applications,

they often require a deep understanding of the system dynamics and may not be suitable for systems with complex and nonlinear behavior. Moreover, model-driven approaches can be computationally expensive and may struggle to handle large amounts of data generated by modern systems.

In recent years, there has been a growing interest in data-driven fault detection methods, which leverage the wealth of data generated by systems to identify faults without requiring an explicit model of the system's behavior. Machine learning and artificial intelligence techniques have emerged as promising tools for developing data-driven fault detection methods, due to their ability to learn complex relationships and patterns from large datasets. These methods have the potential to overcome some of the limitations of model-driven approaches, offering greater flexibility and adaptability to handle the challenges posed by complex systems.

Despite the advances in data-driven fault detection, there are still several challenges that need to be addressed. One key challenge is the scarcity of labeled data, particularly for abnormal conditions. Collecting and labeling data for all possible fault scenarios can be time-consuming, expensive, and often impractical, especially for safety-critical systems where the occurrence of faults is rare. Moreover, systems may experience unforeseen faults that are not represented in the training data, leading to reduced detection accuracy. Therefore, there is a need for fault detection methods that can cope with limited and potentially noisy data, as well as adapt to new and unseen fault conditions.

Another challenge lies in the real-time requirements of many fault detection applications. The ability to identify and diagnose faults quickly is crucial for ensuring the safety and reliability of systems. However, many machine learning techniques are computationally intensive and may struggle to meet the real-time constraints imposed by some applications. Developing efficient and scalable fault detection methods that can operate in real-time is therefore a key research objective.

Finally, the increasing autonomy of systems raises the need for fault detection methods

that can adapt and learn autonomously, without requiring human intervention. In many cases, systems operate in dynamic and uncertain environments, where the conditions may change over time. Autonomous fault detection methods should be capable of adapting to these changes and updating their knowledge to maintain high detection accuracy. This necessitates the development of fault detection methods that incorporate elements of reinforcement learning and online learning, allowing them to adapt continuously as new data becomes available.

1.2 Fault Detection Methodologies

As the technology of autonomous systems advances, the need for robust, online health monitoring systems that can identify and compensate for faults in these systems grows. Faults can manifest in various forms, such as sensor errors, uncertainties in vehicle dynamics, subsystem failures, or externally induced behaviors. Despite the progression of technologies and advanced mitigation strategies, anomalies remain a common occurrence in autonomous systems. In particular, spacecraft are highly complex and isolated systems, which makes fault detection and isolation even more critical. The loss of space systems is often irreversible and typically preceded by performance degradation of system components and devices. Unforeseen circumstances and naturally occurring faults necessitate an on-board fault diagnosis system for space vehicles capable of autonomous Fault Detection, Isolation, and Recovery (FDIR) in order to maintain space operations and mitigate operational gaps as mission complexities increase.

In the field of fault detection, two main approaches are commonly employed: model-based and data-driven methods. Both approaches have their advantages and drawbacks, and selecting the most suitable method depends on the specific requirements of the system under consideration.

Model-based approaches rely on the development of a mathematical model that accurately represents the system's behavior. These models can be derived from physical laws or through system identification techniques, and they often incorporate a detailed under-

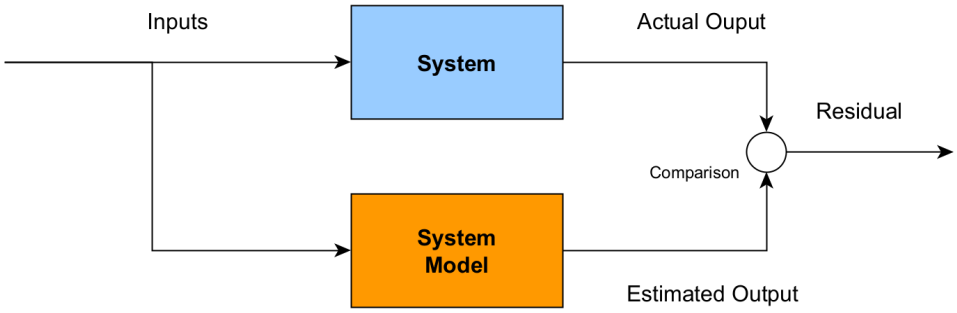


Figure 1.1 General model-based fault detection architecture.

standing of the underlying system dynamics. Figure 1.1 provides a general schematic of a model-based approach. The primary advantage of model-based methods is their ability to capture the underlying physics of the system, which can lead to more accurate and interpretable fault detection results [1]. The fault detection performance of these models, because they are based on derived system dynamics, are analytically verifiable unlike data-driven methods. However, developing an accurate model can be challenging, particularly for complex systems with many interacting components or where the system dynamics are not well understood. Additionally, model-based approaches may require substantial computational resources, which can be a concern for systems with limited processing capabilities, such as spacecraft.

On the other hand, data-driven approaches for fault detection rely on the analysis of historical data to identify patterns and relationships that may indicate faults. These methods often employ machine learning algorithms, such as neural networks, support vector machines, or artificial immune systems, to learn the complex relationships between various system parameters and fault conditions. Data-driven approaches can be particularly advantageous in situations where the system dynamics are not well understood, or where it is difficult to develop an accurate mathematical model [2]. Furthermore, these methods can often be more computationally efficient than model-based approaches, making them suitable for real-time fault detection applications. Figure 1.2 provides a general schematic of a data-driven approach.

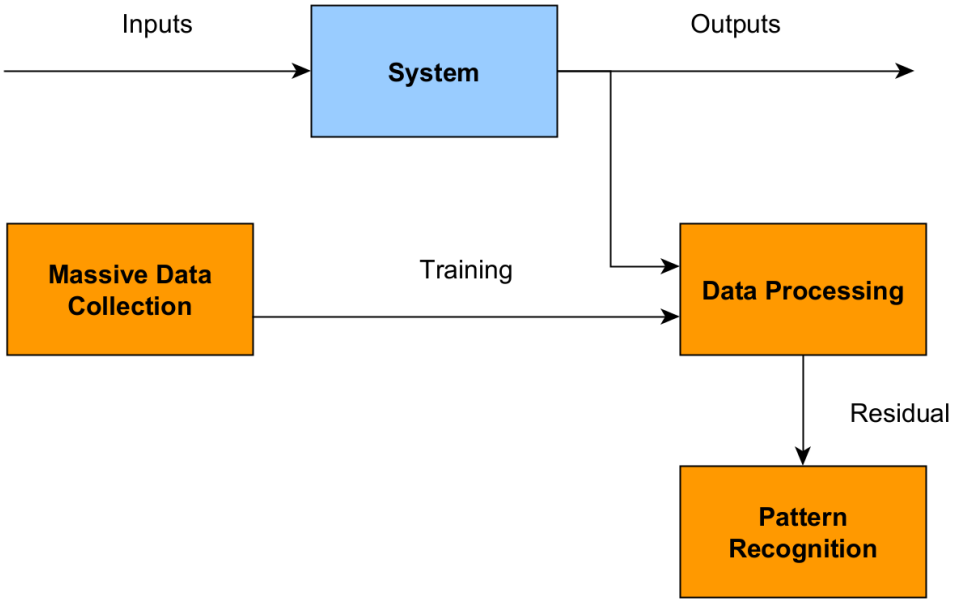


Figure 1.2 General data-driven fault detection architecture.

However, data-driven methods come with their own set of challenges. The success of these approaches heavily depends on the quality and quantity of the available data. Insufficient or unrepresentative data can lead to poor fault detection performance, as the algorithms may not be able to learn the necessary relationships between the system parameters and fault conditions. Moreover, data-driven methods often require extensive testing, simulation verification, and validation to determine the necessary features for capturing system dynamics. This can be particularly challenging for spacecraft, where obtaining sufficient fault data can be difficult due to the rarity and unpredictability of faults.

In many conventional space systems, fault detection takes place within the ground segment after engineers receive and process telemetry data from the spacecraft. This approach can create a gap in nominal operations for the space system, limiting its capabilities and allowing the anomaly to propagate and cause additional adverse effects. When developing health monitoring systems, it is crucial to strike a balance between processing requirements and response accuracy. For many space systems, a common approach involves hardware redundancy; this design concept employs identical components and sensors with outputs that are used for direct comparison. Although this method requires minimal computation, the

added hardware can be costly and contributes to the spacecraft's mass constraint.

Machine learning algorithms have been extensively explored and applied to various aerospace vehicle health monitoring systems [3]. Gao et al. [1, 2] reviewed multiple autonomous system fault detection algorithms, including both model-based and data-driven strategies. In the context of health monitoring, Kalman filters have been employed within sensor fusion networks [4]. Furthermore, Neural Networks and system model parameter estimation have been successfully applied to satellite fault detection [5, 6]. However, these fault detection algorithms necessitate prior knowledge of the vehicles they are applied to, as well as accurate system models.

Data-driven Artificial Immune System (AIS) methods have been proposed for aerospace system fault detection and isolation [7]. These health monitoring paradigms utilize the principle of self-nonself-discrimination to differentiate between nominal data entities belonging to the system's operational envelope and entities outside of it. This approach tackles the complexity and multi-dimensionality of aerospace system dynamic responses in abnormal conditions and aids spacecraft in performing real-time recovery maneuvers [8]. One challenge in implementing AIS strategies for fault detection is the need for extensive testing, simulation verification, and validation to determine the necessary features for capturing system dynamics. These algorithms often demand a large amount of data for training to ensure sufficient coverage of both nominal and off-nominal system behaviors.

Health monitoring systems have employed AIS strategies to enhance and optimize machine-learning-based classifier models. For instance, the Clonal Selection Algorithm, inspired by the immune system metaphor, has been utilized to define feature sets and perform parameter optimization for support vector machine classifiers for real-world applications [9, 10]. Moreover, the Clonal Selection Algorithm has been used to improve the performance of other AIS-inspired methods, such as the Real-Valued Negative Selection algorithm, for fault detection applications [11].

Traditionally, on-board spacecraft health monitoring systems have relied on various rule-

based techniques or offline-based machine learning approaches, which can be challenging to verify and validate. For the development of future autonomous systems, a fault detection system that balances accurate fault detection, design choices, and real-time performance is essential.

1.3 Research Questions, Methods, and Scope

The primary aim of this dissertation is to develop and evaluate an autonomous fault detection architecture that can be applied to complex systems, with a particular focus on spacecraft. This research will address the challenges associated with the design, implementation, and validation of such an architecture, taking into account the unique constraints and requirements of spacecraft and other autonomous systems. To achieve this objective, the following research questions will be investigated:

1. How can an autonomous fault detection system be designed and implemented to effectively handle the complexities and multi-dimensionality of spacecraft and other autonomous systems, while incorporating a structured implementation approach for the fault detection strategy?

2. What are the most suitable machine learning algorithms and techniques for the development of a robust, real-time fault detection system capable of handling large amounts of data without overfitting or excessive computational requirements?

3. How can an autonomous fault detection system be adapted and improved over time to maintain or enhance its performance in response to changes in system behavior or the emergence of new faults?

4. How can the performance of the proposed fault detection system be evaluated, compared to existing methods, and validated to ensure its effectiveness and reliability in real-world applications?

To address these research questions, this research will employ a combination of theoretical, computational, and experimental methods. The development of the fault detection system, the Artificial Immune System augmented Online Support Vector Machine (AISOSVM), will involve the selection and adaptation of appropriate machine learning techniques, as well as the integration of artificial immune system algorithms to enhance performance. Additionally, the research will involve designing and modeling a structured implementation approach for the fault detection strategy, ensuring seamless integration with the overall system architecture. Incremental learning and reinforcement learning approaches will be investigated for the autonomous adaptation of the system over time.

The scope of this dissertation will cover the development and evaluation of the proposed fault detection system, with a specific focus on its application to spacecraft and other autonomous systems. While the primary application domain is space systems, the proposed methodology is designed to be tailored to other autonomous platforms, such as drones, autonomous vehicles, and underwater vehicles. The performance of the proposed system will be compared to existing methods and evaluated using a combination of simulated and real-world data sets to ensure its effectiveness and reliability in real-world applications.

1.4 Dissertation Outline

This dissertation is organized into nine chapters, providing a comprehensive examination of the development, implementation, and evaluation of the proposed autonomous fault detection system for spacecraft and other autonomous platforms. The outline of the dissertation is as follows:

Chapter 1: Introduction. This chapter presents the motivation behind the research, the significance of the problem, the state of the art, and introduces the research questions, methods, and scope of the dissertation.

Chapter 2: Adaptive Fault Detection Framework. This chapter presents the integrated and structured approach for designing the fault detection design and deployment

framework, along with discussions on workflow and system modeling, data acquisition and collection, data processing, feature selection and reduction, and clustering and training techniques.

Chapter 3: Support Vector Machine. This chapter provides an overview of the support vector machine, including its background, theory, framework, and support vector classifier applications.

Chapter 4: Incremental Learning and Decremental Unlearning. This chapter discusses the concepts and algorithms related to incremental learning and decremental unlearning, focusing on the online support vector machine and model validation.

Chapter 5: Artificial Immune System Paradigm. This chapter presents the artificial immune system paradigm, discussing the overview, concepts, negative selection algorithm, clonal selection algorithm, and variable detector strategy.

Chapter 6: The AISOSVM. This chapter details the development, motivation, algorithm, implementation, and integration of the OSVM with artificial immune system algorithms.

Chapter 7: Autonomous Adaptation. This chapter investigates the use of Q-learning and its integration with AISOSVM, addressing the challenges and potential solutions related to autonomous adaptation, as well as the evaluation and validation of the approach.

Chapter 8: Results. This chapter presents the experimental setup and test scenarios, comparison with alternative approaches, application to spacecraft systems, and insights and lessons learned from the evaluation of the proposed fault detection system.

Chapter 9: Conclusion. This chapter provides a summary of the contributions of the dissertation, discusses limitations and future work, and examines the broader impact and applications of the research.

Each chapter is designed to provide a thorough understanding of the proposed autonomous fault detection system, covering the theoretical foundations, practical implementation, and evaluation of the system's performance in real-world scenarios. The dissertation will also address the challenges and opportunities associated with the development of such a system, aiming to contribute to the advancement of autonomous fault detection and management for spacecraft and other autonomous platforms.

2 Adaptive Fault Detection Framework

In this chapter, the focus is on the development and application of an adaptive fault detection framework, which is essential for the successful implementation of the proposed AISOSVM methodology. The framework is designed and modeled using Model-Based Systems Engineering (MBSE) to thoroughly characterize the framework's requirements and needs, resulting in a fully integrated design and deployment process complete with the necessary tools and process flow.

The chapter explores various aspects of the framework, including the importance of gathering training data, its sources, and applications, as well as the various data processing techniques employed during training, such as clustering and normalization. Additionally, the chapter delves into sensitivity analysis for feature selection, the significance of features, and the use of Principal Component Analysis (PCA) to reduce high-dimensional features for application in the support vector machine binary classifier.

By providing a comprehensive understanding of the adaptive fault detection framework and its critical components, this chapter sets the stage for the AISOSVM development presented in later chapters.

2.1 An Integrated and Structured Approach

Developing an effective fault detection system for autonomous platforms requires a comprehensive understanding of the various components and stages involved in the process. To ensure that the fault detection system functions optimally, it is crucial to adopt an integrated and structured approach that effectively addresses the complexities and challenges involved in the design, testing, and deployment phases.

The engineered approach offers several benefits, including better management of the various stages of the process, improved system reliability, and enhanced fault detection capabilities. This approach is underpinned by a systematic organization of the framework components and a well-defined methodology that guides the development process. By leveraging an integrated tool set, the various elements of the fault detection system are seamlessly

interconnected, allowing for more effective communication and collaboration between the components.

A structured workflow ensures that the development process is organized and follows a logical progression. It enables the efficient allocation of resources, streamlined workflows, and the identification of potential bottlenecks or issues before they escalate. Additionally, the structured approach promotes the consistent application of best practices and methodologies, which results in a more robust and reliable fault detection system.

In the context of the AISOSVM, the framework encompasses the entire process, from the initial design and modeling stages to the final deployment and evaluation of the fault detection system. By adopting this approach, the AISOSVM development benefits from a coherent and well-planned process, which ultimately leads to a more effective and reliable fault detection system for autonomous platforms.

2.2 Model-Based Systems Engineering

The development of a structured and integrated approach to designing and deploying adaptive fault management systems is essential for achieving the desired functionality and reliability of complex autonomous systems, such as spacecraft. MBSE offers a structured, systematic, and visual approach to system and process design, which is beneficial for the development of fault management systems. This enables the creation of a comprehensive system representation that captures the interactions between fault management components and the operating system, facilitating the identification and analysis of potential inadequecies and their impacts [12]. By providing a single source of truth for system design, MBSE enhances communication and collaboration between stakeholders, simplifying the design process and reducing the risk of errors and inconsistencies.

One example of a model-based fault management system in the aerospace industry is the Model-Based Off-Nominal State Identification and Detection (MONSID) tool [13, 14]. MONSID offers engineers and users a graphical interface to build system models and design and deploy model-based fault management systems, simplifying the entire process. The

MONSID tool has been successfully applied to free-flying satellites and robotic space systems. This work aims to adopt MBSE to leverage its benefits of comprehensive and verifiable design methods for data-driven fault detection strategies.

2.2.1 Arcadia Method and the Capella Tool

Capella, an open source MBSE tool, is employed to apply the Arcadia method, an MBSE methodology focused on designing complex systems and software architectures [15], to the AISOSVM Fault Management Framework design process. The Arcadia methodology is a model-based systems engineering approach that emphasizes the importance of architecture-driven and requirements-driven system and process design. This methodology provides a coherent and traceable process for the development and management of complex systems, promoting a comprehensive understanding of the system architecture and its components.

The Arcadia method comprises several stages, starting with the Operational Analysis (OA) stage, where the operational context, stakeholder needs, and operational requirements are identified. Next, the System Analysis (SA) stage further refines the functional chains and system components to achieve the desired system behavior transforming system requirements into allocated actor and system responsibilities and functions. The Logical Architecture (LA) and Physical Architecture (PA) stages are then used to define the logical and physical implementation of the system, respectively. Each stage provides a progressively more detailed representation of the system, allowing for the effective integration of system requirements, functional chains, and architectural elements [16].

By following the Arcadia methodology, the workflow and toolflow design benefits from a structured and systematic design methodology that ensures the effective integration of system requirements, functional chains, and architectural elements, ultimately leading to a more resilient and adaptable fault management system for various autonomous platforms.

2.2.2 MBSE for Data-Drive Fault Management Systems

Applying MBSE to data-driven fault management systems, such as the AISOSVM, enables a comprehensive understanding of the system's architecture and interactions between

its components. This understanding is crucial for the development of an effective fault detection and health management system, as it allows for the identification of potential failure modes, their impacts, and appropriate mitigation strategies. Furthermore, the use of MBSE promotes a systematic approach to system design [17], ensuring that all relevant aspects, such as data acquisition, processing, feature selection, and classification, are considered and integrated effectively for different operating domains and platforms.

The Arcadia method and Capella tool facilitate the modeling of the AISOSVM framework by providing a coherent process for capturing the system's operational context, requirements, functions, and architectural elements. This process ensures that the fault management system is designed to meet the specific needs of the application and adapt to changes in system dynamics or operational environment. The AISOSVM framework can be represented in the various stages of the Arcadia method, starting with stakeholder needs and system requirements. The system needs analysis and system analysis stages are used to define the system functions, such as data acquisition, processing, feature selection, and classification, as well as their interactions and relationships with other system components.

Throughout the design process, the Capella tool facilitates the creation and visualization of the AISOSVM framework's models, enabling the efficient communication and collaboration between stakeholders. Moreover, the tool supports the continuous validation and verification of the system's design, ensuring that the fault management system meets its requirements and can effectively adapt to changes in system dynamics. The result of this MBSE-driven design for the data-driven fault management approach is an integrated system model of the design approach and software components that can be reused in future system models and used for verification and validation of the AISOSVM framework implementation.

2.3 Framework Design and Modeling

The AISOSVM framework design and modeling process leverages the capabilities of MBSE, Arcadia, and Capella to create a comprehensive and adaptable fault management system for complex autonomous platforms. Utilizing MBSE in conjunction with the Arcadia

method provides a structured and systematic approach to the fault management framework design process, enabling a seamless transition from the Operational Analysis through the various levels of system and software decomposition.

The use of MBSE in the design of the AISOSVM Framework facilitated the creation of an organized and coherent workflow for fault detection system design and deployment. Additionally, it provided a foundation for the necessary tool set required to integrate the AISOSVM into various autonomous platforms. A first iteration of a fault detection framework model is developed promoting reusability across diverse platforms and process integrations for future AISOSVM framework development.

2.3.1 Operational Analysis

Operational Analysis is a crucial stage in the Arcadia method and serves as the foundation for the design and development of any system, including the AISOSVM Framework. It primarily focuses on understanding the stakeholders' needs, defining the operational context, and capturing the high-level use cases in the form of Operational Capabilities. By performing an Operational Analysis, a clear vision of the system's purpose and objectives is established while key operational entities and their interactions are defined. The Operational Analysis provides an initial perspective for the concept of operations for the fault detection framework, ensuring that the system's development aligns with its intended operational context.

The Operational Analysis began with the creation of high-level use cases in the form of Operational Capabilities. Operational Capabilities describe the fundamental abilities that a system must possess to achieve its objectives and satisfy the needs of its stakeholders. These capabilities serve as a bridge between the stakeholders' expectations and the system's functional design.

In the AISOSVM Framework, the following Operational Capabilities were identified:

- **Design and Configure Fault Management Models:** describes the process of designing and configuring the fault management models that will be utilized in the system to detect and diagnose faults.

- **Perform Real-Time Fault Management:** involves the real-time monitoring and management of faults within the system, ensuring timely detection and diagnosis.
- **Provide Data Management and Integration:** focuses on managing the data utilized by the fault management system, including data acquisition, storage, retrieval, deletion, and modification and also encompasses the integration of various data sources to support analysis.
- **Verify and Validate Fault Detection Models:** focuses on the implementation of verification and validation processes for the fault detection models, ensuring their accuracy, reliability, and effectiveness in detecting faults.

Additionally, the following Operational Entities were identified as the primary stakeholder groups for a fault management design and deployment process:

- **Autonomous System Operations Management (ASOM):** ASOM is responsible for overseeing the operation and performance of the autonomous system. It consists of engineers and system operators who design, configure, monitor, and maintain the Autonomous System and the fault management models to ensure the system's reliability and performance. ASOM interacts with the other operational entities to ensure the system operates within its intended environment while meeting operational needs and use cases.
- **Autonomous System:** The Autonomous System represents the platform or vehicle that relies on the fault management framework for its fault detection, diagnosis, and response capabilities. This entity is directly impacted by the proper functioning of the fault management framework and plays a critical role in the operational context by providing data, allowing for model deployment, and executing fault management actions based on the framework's outputs.

- **Operating Environment:** The Operating Environment encompasses the external conditions and factors that the Autonomous System operates within, including weather, terrain, and other environmental variables. The operating environment can influence the performance and behavior of the autonomous system and its fault management processes and can directly cause faults and failures to occur.

To detail the interactions between the operational entities and the activities required to fulfill these capabilities, Operational Activity Interaction Blanks (OAIBs) were created. Within each OAIB, need statements, represented as Operational Activities, were defined to capture the essential actions and processes involved in addressing the operational needs. These need statements were allocated to the appropriate Operational Entities, ensuring that all responsibilities and roles within the operational context were accurately represented.

Subsequently, the Operational Activities were transformed into Functional Chains within the Operational Architecture Blank (OAB), illustrating the interdependencies and flow of activities between the operational entities. The Operational Architecture Blank (OAB) diagram offers a high-level view of the system's operating domain and is shown in Figure 2.1. The OAB diagram presents an abstract representation of the main needs for the operational entities that must to be satisfied by the proposed system; it also illustrates the interactions of these entities as a part of the concept of operations. By capturing the needed operational behavior of the system, the OAB diagram allows stakeholders to gain a better understanding of the system and its components, facilitating communication and collaboration. The OAB diagram from Capella outlines four primary functional chains, representing the main use cases of the framework: Data Management, Real-Time Fault Management, Design Fault Management Models, and Verify and Validate System. These functional chains depict the sequence of activities required to accomplish the respective objectives.

The Operational Analysis stage effectively translates the stakeholders' needs into a structured and coherent operational context, laying the foundation for the System Analysis phase. By maintaining a clear focus on the operational needs and context, the AISOSVM Frame-

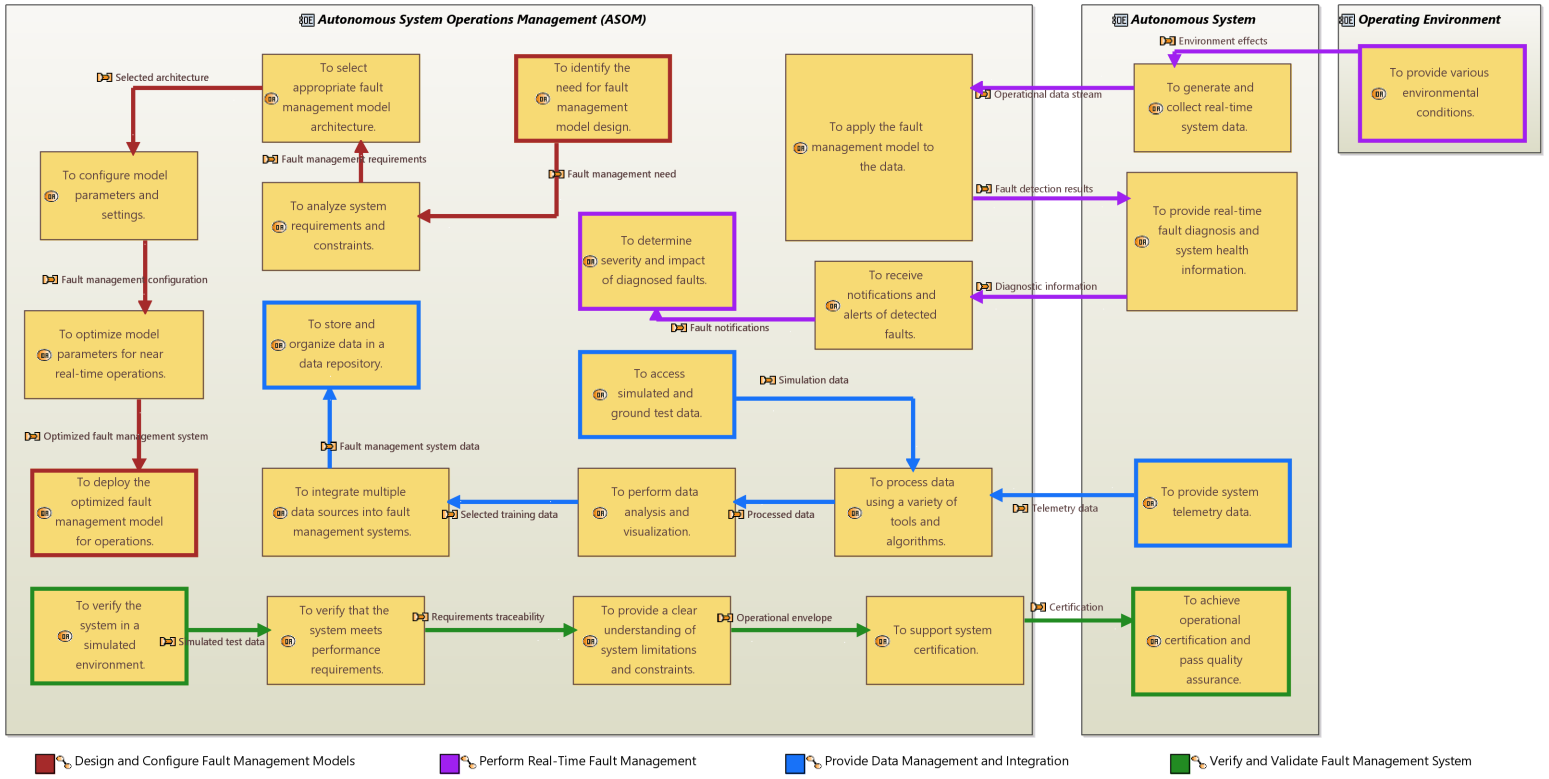


Figure 2.1 Integrated AISOSVM design and deployment approach.

work's design remains grounded in its intended application, ensuring a successful implementation and integration within various autonomous platforms. Appendix A provides other AISOSVM framework model-based views that comprise the concept of operations and derived use cases.

2.3.2 System Analysis

The System Analysis phase began with the creation of Functional and Nonfunctional Requirements for the fault management framework. These requirements were traced back to the high-level use cases (Operational Capabilities) and need statements from the Operational Analysis. Furthermore, relationships between requirements at the system level were established, indicating which system requirements were derived from other system requirements.

Three Requirements Diagrams were created in Capella: one for System Functional Requirements, one for System Nonfunctional Requirements, and one for displaying only the requirements with relationships to each other. These diagrams allowed for a clear understanding of the requirements and their interdependencies, ensuring that all aspects of the fault management framework were considered. The nonfunctional requirements diagram is shown in Figure 2.2 illustrating the Capella requirements diagramming process.

Next, Missions and Capabilities diagrams were created. Missions are derived directly from from the Operational Analysis, specifically from the defined Operational Capabilities which incorporate the needed use cases the AISOSVM Framework system must achieve. For each System Mission, a set of System Capabilities was allocated; these System Capabilities are a decomposition of the high level use case and form specific functions or needs that must be implemented into the system to achieve the respective mission. These System Capabilities can be traced back to system requirements and have been outlined as follows:

- **Configure Fault Management Models:** focuses on creating, customizing, and configuring fault management models to adapt to specific system requirements and constraints.

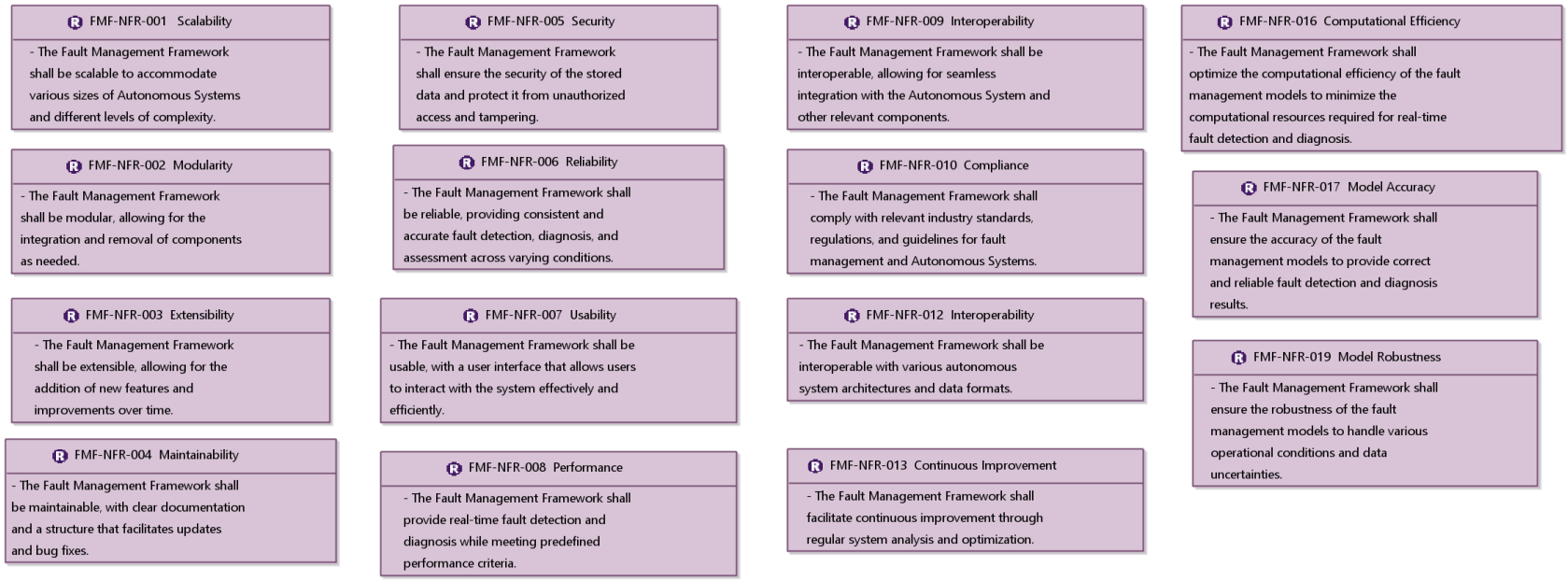


Figure 2.2 AISOSVM nonfunctional system requirements diagram.

- **Real-Time Fault Detection and Diagnosis:** encompasses the real-time detection, diagnosis, and management of faults within the system, enabling quick and accurate responses to emerging issues.

- **Data Acquisition and Integration:** involves collecting, integrating, and analyzing data from various sources, ensuring that the fault management framework has access to all necessary information for effective operation.

- **Optimize and Deploy Fault Management Models:** involves the optimization, testing, and deployment of fault management models, ensuring their accuracy, effectiveness, and seamless integration into the system.

- **Monitor System Health:** focuses on the continuous monitoring of system health, performance, and the detection of anomalies that could indicate potential faults.

- **System Certification and Quality Assurance:** includes certifying the fault management system's compliance with relevant standards and ensuring that quality assurance processes are in place.

- **Verification and Validation:** encompasses the testing, verification, and validation of the fault management models and the overall system to ensure their accuracy and effectiveness.

Each System Capability was refined further with a System Data Flow Blank (SDFB) diagram, which described the specific functions and data flows required to achieve the higher-level System Capability. The functions within each SDFB were then allocated to a System Level Actor or the Fault Management Framework System itself. This allocation process ensured that responsibilities were clearly assigned to the appropriate entity within the system. The System Level Actors, which were derived from the Operational Entities, include personnel and other entities that would directly interact with the system. These actors include:

- **Engineers:** Engineers are responsible for designing, configuring, and maintaining the fault management models. They work closely with the fault management framework to develop, optimize, and validate the models according to the specific requirements of the autonomous system.

- **System Operators:** System Operators are the personnel responsible for overseeing and managing the day-to-day operations of the autonomous system. They rely on the fault management framework to monitor system health, receive alerts, and diagnose potential issues in real-time.

- **Operating Environment:** The Operating Environment represents the external conditions and factors in which the autonomous system operates. This entity provides essential environmental data that must be integrated into the fault management framework for a more accurate and comprehensive analysis.

- **Autonomous System:** The Autonomous System is the primary subject of the fault management framework and encompasses the hardware, software, and control systems required for autonomous operation. The fault management framework is designed to monitor, detect, diagnose, and respond to faults within the Autonomous System to ensure its safe and efficient operation.

To further define responsibilities and interactions between actors, functional chains were created for each system capability in each of the contextual SDFB diagram. These functional chains provided a valuable representation of the interactions and dependencies among functions required to achieve the respective system capability.

The System Architecture Blank (SAB) diagram was then constructed, showing the system functions, System Actors, and data flow between all allocations of functions and responsibilities. This diagram provided a comprehensive understanding of the interactions between system components and the flow of information and actions within the fault management

framework. Figure 2.3 provides a reduced SAB diagram depicting the fault management configuration, optimization, and deployment use cases. Appendix A provides other AISOSVM framework model-based views that comprise the system analysis and requirements flowdown.

2.3.3 Deriving Software Components for Fault Detection

Having defined the operational and system analyses of the AISOSVM framework, the next step was to derive the specific software components required for implementing the fault detection aspect of the framework. The focus on fault detection and the online adaptation of the fault detection algorithm was the primary objective for this dissertation.

To derive the software components, a detailed analysis of the system functions, data flows, and system capabilities was performed. This analysis helped identify the specific software components needed to achieve the desired fault detection functionality. The software components were derived from the functions allocated to the Fault Management Framework within the System Architecture Blank diagram, taking into account the necessary interactions and dependencies among the components.

The derived software components include:

- **Data Acquisition and Integration:** This component is responsible for collecting and integrating data from various sources, such as sensor measurements, system logs, and environmental data. It ensures that all relevant data is made available to other software components in the framework in a timely and efficient manner.

- **Feature Extraction:** This component processes the raw data obtained from the Data Acquisition and Integration component and extracts relevant features that are useful for fault detection. It applies various signal processing and machine learning techniques to reduce the dimensionality of the data and identify patterns indicative of faults.

- **Model Training and Optimization:** This component is responsible for training, optimizing, and selecting the most suitable fault detection models using the features

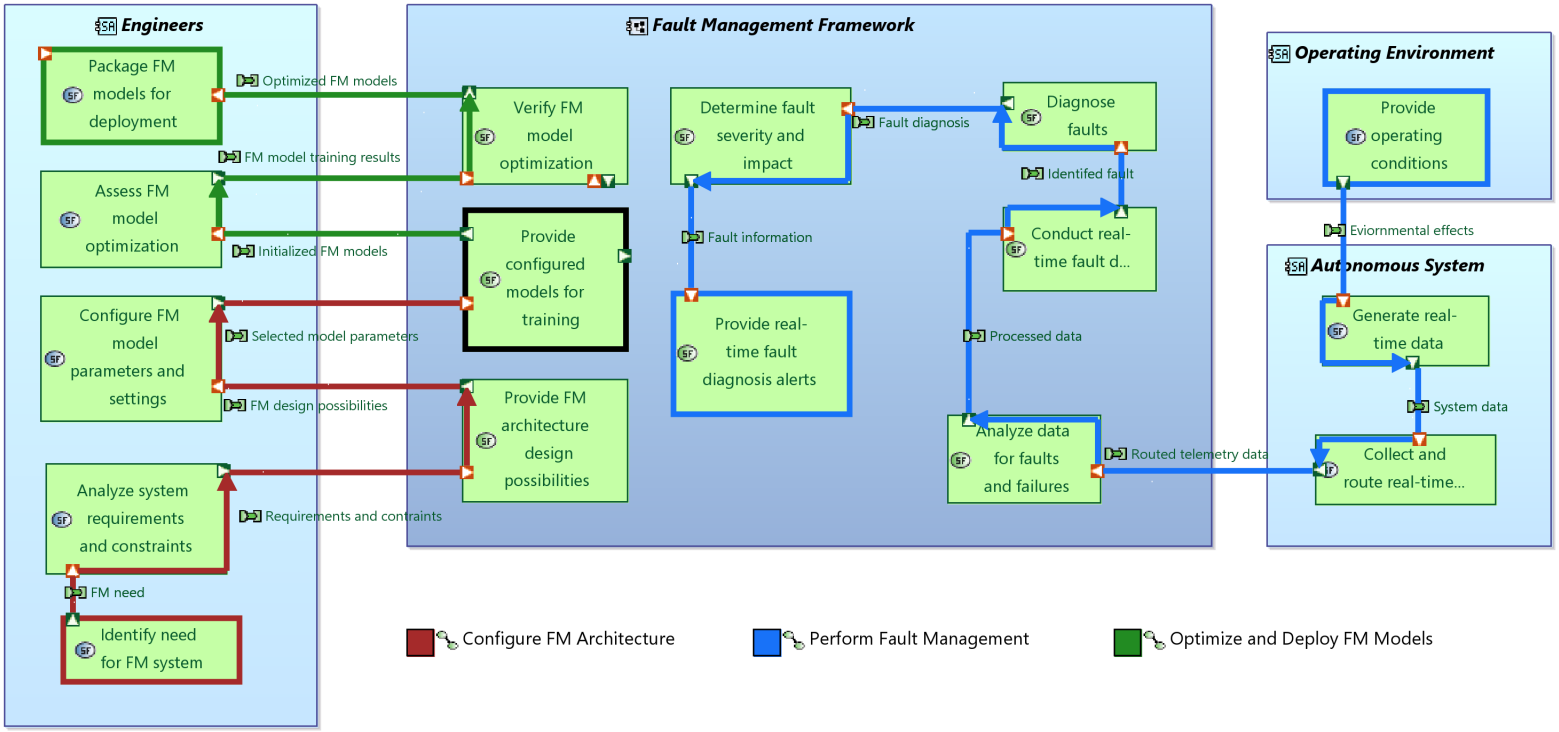


Figure 2.3 Allocated system level functions for configuring and deploying the fault management system.

extracted from the Feature Extraction component. It employs machine learning algorithms and optimization techniques to adapt the models based on the specific requirements and constraints of the autonomous system.

- **Real-Time Fault Detection:** This component continuously monitors the system's health by applying the optimized fault detection models to the incoming data stream. It detects potential faults and generates alerts for system operators and engineers to diagnose and respond to the issues.

- **Model Deployment and Update:** This component manages the deployment of the optimized fault detection models to the Real-Time Fault Detection component. It is also responsible for updating the models when new data becomes available or when changes in the system's operational conditions require the adaptation of the models.

These software components were designed to work together within the AISOSVM framework to provide a comprehensive fault detection solution. The components were derived based on their ability to satisfy the system functions, data exchanges, and system capabilities defined in the System Analysis. The modular design of these components allows for flexibility and adaptability in addressing the diverse fault detection needs of different autonomous systems while promoting reuse and scalability. The identified software components helped structure the implementation framework.

The structured implementation framework, as depicted in Figure 2.4, is a vital component of the AISOSVM fault management system design, as it allows for a systematic and traceable approach to system development. This framework is created through a logical decomposition of the derived system requirements and the operational architecture developed using the Capella tool and Arcadia methodology. By breaking down the system requirements and architecture into smaller, manageable elements, the framework provides a clear roadmap for implementing and deploying the AISOSVM fault management system in various autonomous platforms.

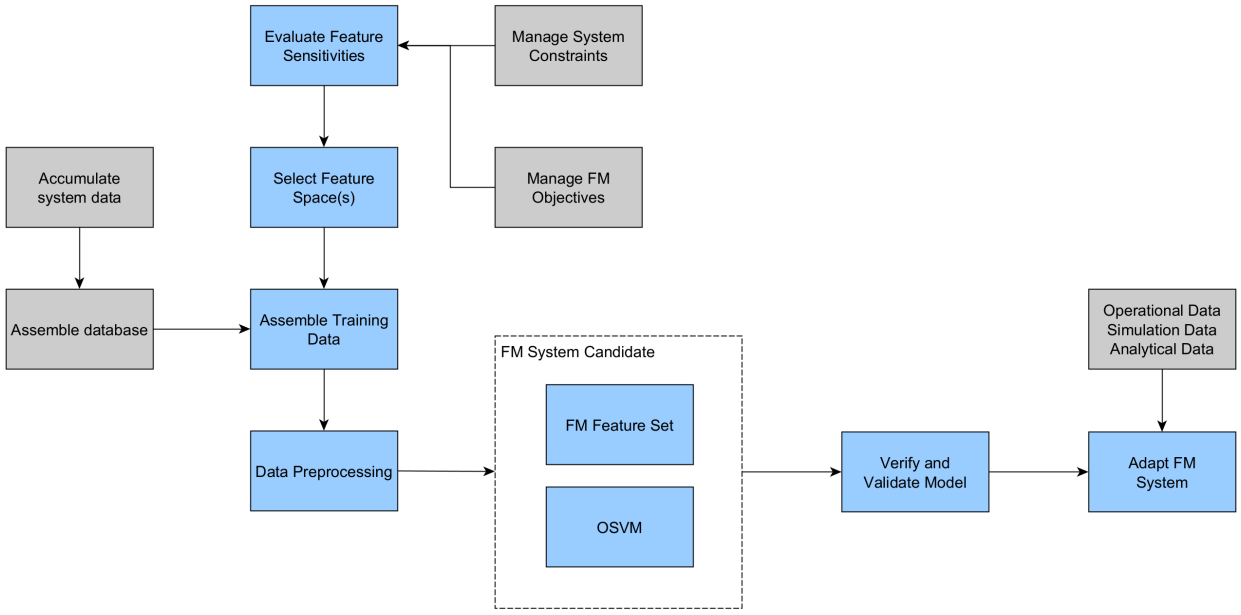


Figure 2.4 Integrated AISOSVM design and deployment approach.

One of the key benefits of this designed framework is its adaptability to various autonomous platforms. By providing a clear, step-by-step process for system development, the framework enables users to tailor the AISOSVM fault management system to specific platforms, ensuring optimal performance and seamless integration. This flexibility is critical in the ever-evolving field of autonomous systems, where technology advancements and new application domains continually introduce new challenges and requirements. Additionally, the structured implementation framework fosters effective verification and validation (V&V) of the AISOSVM fault management system. By aligning the V&V activities with the system requirements and architecture, the framework ensures that the system's performance and behavior are consistent with the intended design objectives. This comprehensive V&V process is critical in establishing confidence in the system's capabilities and its ability to detect and manage faults effectively.

The design and implementation framework modeling process leverages the strengths of MBSE, the Capella toolset, and the Arcadia method to create a structured, systematic, and visual approach to system and process design. This methodology ensures that complex aspects within the data-driven AISOSVM approach can be effectively monitored, traced, and

tailored to new platforms while maintaining the robustness required for mission-critical applications, such as spacecraft health monitoring. By adopting a model-based approach, the AISOSVM framework can harness benefits offered by model-based and data-driven fault detection methodologies, providing a foundation for implementing a comprehensive, adaptable, and reliable fault management system.

2.4 Data Acquisition and Collection

In the development of an effective fault detection system, such as the AISOSVM, data acquisition and collection play a pivotal role. The quality and quantity of data significantly influence the performance of the fault detection algorithm, as well as the accuracy and reliability of the system. This section will discuss the importance of data acquisition and collection in the context of autonomous fault detection systems, the various sources of data, and the challenges and considerations involved in obtaining relevant and representative data.

A key aspect of data acquisition and collection is ensuring that the data used for training and evaluation adequately represents the operational conditions and scenarios that the autonomous platform may encounter [18]. The data should encompass both nominal and off-nominal operating conditions, providing the fault detection system with a comprehensive understanding of the platform's behavior under various circumstances. This diverse set of data enables the fault detection system to effectively distinguish between normal and faulty conditions, thereby enhancing its fault detection capabilities.

Data can be sourced from various channels, including simulations, experimental setups, and historical records of the platform's operation. Each data source has its advantages and limitations, and the choice of the data source should be guided by factors such as the availability of data, the accuracy and reliability of the data, and the specific requirements of the fault detection system.

Simulations, for instance, offer the flexibility to generate data under controlled conditions, enabling the generation of diverse scenarios and the ability to explore edge cases. Experimental setups, on the other hand, can provide more realistic data that captures the

nuances of the platform's operation, while historical records offer valuable insights into the platform's performance over time.

One of the challenges in data acquisition and collection is ensuring data quality and consistency, as noise, outliers, and inconsistencies can adversely affect the performance of the fault detection system [19]. It is essential to adopt appropriate preprocessing techniques and data validation methods to address these issues and enhance the overall quality of the data used in the development of the fault detection strategy.

2.5 Data Processing

The performance and efficacy of a fault detection system are heavily influenced by the quality and format of the input data. Data processing, a crucial step in the development pipeline, involves transforming raw data into a format that is suitable for analysis, feature extraction, and model training.

Data processing consists of several stages that are designed to enhance the quality, consistency, and interpretability of the input data. These stages include data cleaning, data normalization, and data transformation, each of which plays a crucial role in preparing the data for further analysis and model training.

Data cleaning involves the identification and handling of missing values, noise, and outliers present in the raw data. Techniques such as filtering and outlier detection can be employed to address these issues and ensure data consistency. Data cleaning is particularly important for fault detection systems, as inconsistencies in the data can impair the system's ability to accurately identify and isolate faults.

Data normalization is a crucial step in data processing, particularly when dealing with data from diverse sources or of varying scales. Normalization techniques, such as min-max scaling and z-score normalization, help to bring the data onto a common scale, facilitating comparison and analysis [20]. Normalizing data is essential for the proper functioning of the AISOSVM, as it ensures that the features used in the model have equal significance and that the algorithm is not biased towards specific features.

Data transformation is another critical aspect of data processing, which involves converting raw data into a more suitable format for analysis and modeling. Transformation techniques such as principal component analysis (PCA), feature extraction, and feature engineering can help to reduce the dimensionality of the data, enhance its interpretability, and facilitate the identification of meaningful patterns and relationships. Data transformation is particularly relevant for fault detection systems like the AISOSVM, where the ability to identify and isolate faults is contingent upon the system's capacity to discern meaningful patterns in the input data.

Data processing is an indispensable component of the development process for fault detection systems such as the AISOSVM. By employing appropriate data cleaning, normalization, and transformation techniques, the quality and interpretability of the input data can be enhanced, enabling the AISOSVM to effectively learn to detect and isolate faults in autonomous systems.

2.6 Feature Selection and Reduction

Feature selection and reduction are crucial steps in the development of an effective fault management system, particularly for data-driven approaches. These processes involve identifying the most relevant and informative features from the collected data and reducing the dimensionality of the dataset to improve computational efficiency and mitigate the impact of the curse of dimensionality.

Principal Component Analysis (PCA) is a widely used technique for feature reduction and dimensionality reduction in machine learning and data analysis [21]. PCA works by transforming the original data into a new coordinate system, where the axes, known as principal components, capture the maximum variance in the data. The first principal component captures the largest variance, the second principal component captures the second-largest variance, and so on. By retaining only a few principal components that account for a significant portion of the total variance, PCA can effectively reduce the dimensionality of the dataset while retaining most of its essential information [22]. Within a fault management

system architecture, PCA can be employed to identify and retain the most informative features from the collected data. By transforming the original dataset into a lower-dimensional space, PCA enables the system to focus on the most critical aspects of the data, which in turn enhances the performance of the fault detection and isolation processes. Furthermore, PCA can help mitigate the risk of overfitting, as it reduces the complexity of the model by removing less relevant features.

K-means clustering is another technique employed in fault management systems for feature selection and reduction. K-means is an unsupervised learning algorithm that aims to partition a dataset into K distinct clusters based on the similarity of the data points. The algorithm works by iteratively assigning each data point to the nearest cluster centroid and updating the centroid positions based on the average of the points in the cluster [23, 24]. In the context of feature selection, K-means clustering can be used to identify groups of similar features that potentially carry redundant information. By analyzing the relationships between features within and across clusters, it is possible to identify the most representative features in each group and eliminate redundant or less informative ones. This process can effectively reduce the dimensionality of the dataset and enhance the computational efficiency of the fault management system.

The implementation of K-means clustering can aid in the identification of the most suitable feature space for the AISOSVM model. By exploring different combinations of features and assessing their impact on the clustering results, the algorithm can guide the selection of the most informative and discriminative feature set for the fault detection process.

The feature selection and reduction are essential processes in fault management systems, as they enable the system to focus on the most relevant and informative aspects of the data. By employing techniques like PCA and K-means clustering, the system can effectively reduce the dimensionality of the dataset, improve computational efficiency, and enhance the performance of the fault detection and isolation processes. These methods play a critical role in ensuring the robustness and reliability of fault management architectures in various

autonomous platforms.

3 Support Vector Machine

In this chapter, Support Vector Machines (SVMs) and their thory are introduced. SVMs are a powerful and widely used machine learning technique for classification and regression tasks. SVMs have been successfully applied in various fields, including fault detection and isolation, due to their ability to handle high-dimensional and nonlinear data. The chapter begins with the background of SVMs, providing an overview of their development, application areas, and key features. Following this, the discussion delves into the SVM framework, explaining the general pieces and process of SVMs and their use in fault detection. Lastly, the theory behind SVM classifiers is covered in depth, discussing their formulation, optimization, and essential components. The goal of this chapter is to provide a comprehensive understanding of SVMs and lay the groundwork for the development of the proposed AISOSVM methodology in the subsequent chapters.

3.1 Background of Support Vector Machines

Support Vector Machines (SVMs) are a group of learning algorithms that have gained prominence due to their ability to automatically estimate dependencies between data. Developed and expanded by Vapnik [25], SVMs are based on the statistical learning theory that were originally largely applied to applications such as text categorization, image recognition, network monitoring, among others [26, 27]. As a representative of the statistical learning theory, SVMs focus on mathematical fundamentals and have since evolved into numerous variants, addressing different problems and applications [28].

At the core of SVMs are two key characteristics: the maximal margin and the kernel method. The maximal margin is a geometric concept that aims to maximize the distance between decision boundaries and the nearest data points belonging to different classes. The kernel method, on the other hand, involves the use of kernel functions to transform data into higher-dimensional spaces where linear separation is more likely [25]. Kernel functions play a crucial role in SVMs, as they enable the approximation of real-valued functions. Some commonly used kernel functions include linear kernel, polynomial kernel, sigmoid kernel, and

radial basis function (RBF) [29, 30]. Cervantes et al. [31] provides a detailed survey of SVM kernel function implementations and how these functions have been applied to various SVM variants to address specific challenges and improve performance.

SVMs can be broadly categorized into two types: Support Vector Classification (SVC) and Support Vector Regression (SVR). SVC is utilized for classification problems, where data is divided into two or more groups based on the provided features. It has been successfully applied in areas such as text categorization and bioinformatics, [27, 32]. In contrast, SVR is employed for regression problems, where the aim is to predict continuous-valued outputs based on time-series data. SVR has been utilized in various applications, including financial market forecasting [33], electricity price prediction [34], and aircraft anomaly detection [35]. The AISOSVM developed as a part of this research focuses on the application of SVMs as a classifier for its ability to perform accurate, multi-class failure detection and diagnosis.

In recent years, SVMs have been applied to the domain of fault detection and diagnosis in complex systems such as spacecraft. Farahani and Rahimi [36] developed an anomaly detection and diagnosis approach for spacecraft control moment gyros using an optimized support vector machine model. Yao et al. [37] implemented SVMs to detect and classify failure modes of various battery pack hardware for industrial systems. Gao et al. [38] presented a fault detection and diagnosis approach for spacecraft and is combined successfully with Principal Component Analysis (PCA) for feature space reduction and optimization.

3.2 The Support Vector Machine Framework

The primary goal of the SVM framework is to find an optimal decision boundary, known as the hyperplane, which maximizes the margin between data points belonging to different classes. This is achieved through a combination of optimization techniques and kernel functions. The optimization process aims to minimize a cost function, which represents the trade-off between maximizing the margin and minimizing classification errors. The kernel function is employed to transform the input data into a higher-dimensional space, where linear separation is more likely to be achievable.

The SVM framework can be summarized in the following steps:

- Data preprocessing: Raw data is preprocessed to ensure compatibility with the SVM algorithm. This may involve normalization, feature selection, and reduction techniques.
- Kernel function selection: A suitable kernel function is chosen based on the nature of the input data and the desired output. Commonly used kernel functions include linear, polynomial, RBF, and sigmoid kernels.
- Model training: The SVM algorithm is trained on a dataset to learn the optimal hyperplane. This involves solving a convex optimization problem using techniques such as Sequential Minimal Optimization (SMO) or gradient descent.
- Model evaluation: The trained SVM model is evaluated on a validation dataset to assess its performance. Performance metrics such as accuracy, precision, recall, and F1-score are used to gauge the model's effectiveness.
- Model deployment: Once the SVM model is deemed satisfactory, it can be deployed for real-world applications such as fault detection in complex systems.

In the context of fault detection, the SVM framework can be adapted to identify and classify various types of faults in a system. The input data, which consists of features extracted from sensor measurements or system states, is preprocessed and transformed using a kernel function. The SVM algorithm then learns to distinguish between different fault classes based on the transformed data. When new data is encountered, the trained SVM model can classify it as either normal operation or a specific fault type, enabling timely fault detection and isolation. Figure 3.1 illustrates the architecture of a classical SVM classifier.

The inherent binary classification nature of SVMs poses a challenge when dealing with multi-class problems, such as fault detection and isolation in autonomous systems. In such systems, there can be multiple fault types or even combinations of faults that need to be

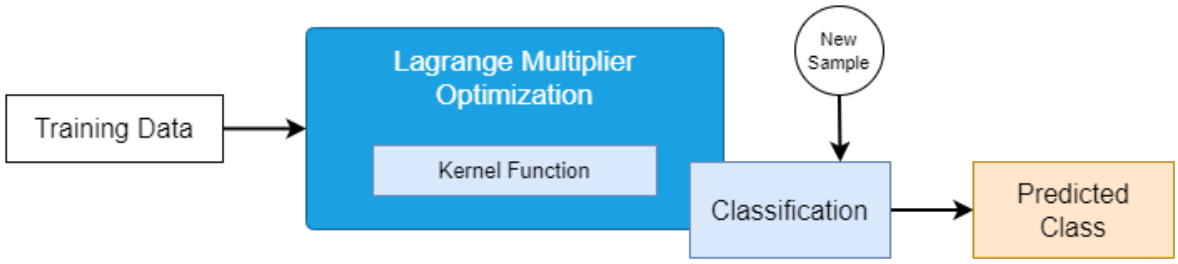


Figure 3.1 The classical SVM architecture.

identified and isolated. To address this challenge, SVMs can be extended to handle multi-class problems using techniques like one-vs-one (OvO) or one-vs-all (OvA) strategies.

In the one-vs-one (OvO) strategy, SVM classifiers are trained to distinguish between each pair of classes. For a problem involving k classes, the OvO method would require training $k(k-1)/2$ classifiers, as each classifier is responsible for discriminating between a unique pair of classes. This approach results in a large number of classifiers for problems with a high number of classes, increasing the computational complexity. However, the OvO strategy has the advantage of being less sensitive to class imbalance, as each classifier is trained on a balanced subset of the data containing only two classes.

On the other hand, the one-vs-all (OvA) strategy involves training k classifiers, where each classifier is responsible for distinguishing between one class and the rest of the classes combined. In the context of fault detection and isolation, this approach requires fewer classifiers than the OvO strategy, making it computationally more efficient. However, the OvA strategy can be more susceptible to class imbalance since each classifier is trained on an imbalanced dataset containing one class versus all other classes.

The SVM framework is a robust and versatile approach to classification and regression tasks. Its ability to handle high-dimensional data and nonlinear relationships makes it particularly well-suited for fault detection in complex systems. As the dissertation progresses, the integration of the SVM framework with the adaptive fault detection strategy and its application to spacecraft fault diagnosis will be explored in more detail.

3.3 Support Vector Machine Classifiers

The support vector machine provides a supervised learning technique for classification. For an n-dimensional space, the input vector, $\boldsymbol{x} \in R^n$, belongs to one of two classes, class 1 or class 2. From this input vector, a data set is defined $(\boldsymbol{x}_i, y_i)$, where $(i = 1...k)$ and $y \in -1, +1$ are the labels associated with both classes; a label of -1 is issued for members of the class 1 subset and a label of $+1$ is issued for members of the class 2 subset. SVM seeks to find a decision boundary such that the input data can be separated linearly with a hyper-plane given by Equation (3.1):

$$y = f(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{w}^T \boldsymbol{x} + b \tag{3.1}$$

where $\boldsymbol{w} \in R^n$ is an n-dimensional weight vector and $b \in R$ is a bias value. This equation determines the maximum margin to separate class 1 from class 2. The decision boundary is shown in Equation (3.2).

$$y_i(\boldsymbol{w}\boldsymbol{x}_i + b) \geq 1, \quad i = 1...k \tag{3.2}$$

The distance from a data point $\boldsymbol{x}$ to this hyperplane that separates the classes of data is given by:

$$||\boldsymbol{d}||_2 = \frac{|\boldsymbol{w}^T \boldsymbol{x} + b|}{||\boldsymbol{w}||_2} \tag{3.3}$$

The margin of the hyperplane is defined as the smallest distance between the hyperplane and any data point belonging to either class. The objective of support vector classifiers is to find the hyperplane that maximizes this margin. Using a soft margin concept, this can be converted to a quadratic programming problem equivalent to solving the following optimization process:

$$\text{Minimize} \quad \frac{1}{2}\langle \boldsymbol{w} \cdot \boldsymbol{w} \rangle + C \sum_{i=1}^{k} \xi_i = \frac{1}{2}||\boldsymbol{w}||^2 + C \sum_{i=1}^{k} \xi_i$$

$$\text{Subject to} \quad \begin{cases} y_i(\langle \boldsymbol{w}\boldsymbol{x}_i \rangle + b) \geq 1 - \xi_i \\ \xi_i \geq 0 \\ i = 1, 2, ..., k \end{cases} \tag{3.4}$$

with parameters $C > 0$ and $\xi_i > 0$. The slack variables ξ_i are positive definition variables, and relax the optimization process providing a fuzzy classification boundary allowing some data, like outliers, to be unclassified or classified incorrectly. The hyperparameter, C, is a penalty parameter that can be selected and tuned by the user and penalizes the SVM optimization algorithm for incorrectly classified data; the larger the value C, the higher the penalty.

The minimization problem, with its constraints, can be reformulated and solved using Lagrange multipliers.

$$\text{Minimize} \quad L(\alpha) = \sum_{i=1}^{k} \alpha_i - \frac{1}{2} \sum_{i,j=0}^{k} \alpha_i \alpha_j y_i y_j (\boldsymbol{x}_i ... \boldsymbol{x}_j)$$

$$\text{Subject to} \quad \begin{cases} \sum_{i=1}^{k} y_i \alpha_i = 0 \\ \alpha_i \geq 0 \\ i = 1, 2, ..., k \end{cases} \tag{3.5}$$

where α_i are the Lagrange multipliers that determine the optimal hyperplane by maximizing $L(\alpha)$. The values for α_i become the support vectors by solving the optimization problem. The decision boundary that separates the classes of data can be written as:

$$f(\boldsymbol{x}) = sign(\sum_{i=1}^{k} \alpha_i y_i \phi(\boldsymbol{x}_i, \boldsymbol{x}) + b) \tag{3.6}$$

$$\phi(\boldsymbol{x}, \boldsymbol{x}_i) = e^{-||\boldsymbol{x}-\boldsymbol{x}_i||^2/2\sigma^2} \tag{3.7}$$

where $\phi(\boldsymbol{x}, \boldsymbol{x}_i)$ is the kernel function. For many applications, input data is often not linearly separable; thus, to handle non-linearly separable data, the input data can be transformed to a higher dimensional space using a nonlinear transformation with the kernel function [39]. A kernel function $\phi(\boldsymbol{x}, \boldsymbol{x}_i)$ computes the inner product of two data points in the higher-dimensional space. Commonly used kernel functions include:

- Linear kernel: $\phi(\boldsymbol{x}, \boldsymbol{x}_i) = \boldsymbol{x}^T \boldsymbol{x}_i$
- Polynomial kernel: $\phi(\boldsymbol{x}, \boldsymbol{x}_i) = (\boldsymbol{x}^T \boldsymbol{x}_1 + c)^d$
- Radial basis function (RBF) kernel: $\phi(\boldsymbol{x}, \boldsymbol{x}_i) = \exp(-||\boldsymbol{x} - \boldsymbol{x}_i||^2/2\sigma^2)$
- Sigmoid kernel: $\phi(\boldsymbol{x}, \boldsymbol{x}_i) = \tanh(\kappa \boldsymbol{x}^T \boldsymbol{x}_i + \theta)$

The kernel function selected for this research effort and implementation is the well-known Radial Basis Function (RBF) shown in Equation (3.7). The constant σ is a tuning parameter that controls the width of the kernel. The RBF offers excellent generality and flexibility for regression and classification.

3.4 Model Selection and Regularization

In support vector classifiers, the choice of the kernel function and its parameters, as well as the regularization parameter C, have a significant impact on the model's performance. The regularization parameter C controls the trade-off between maximizing the margin and minimizing the classification error. A smaller value of C allows for a larger margin at the cost of allowing some misclassifications, while a larger value of C enforces a stricter separation between classes, which may result in a smaller margin.

Model selection typically involves choosing the kernel function, its parameters, and the regularization parameter C based on their performance on a validation set or using cross-

validation techniques. Commonly used cross-validation techniques include k-fold cross-validation and leave-one-out cross-validation. The optimization process implemented by the artificial immune system paradigm utilizes the feedback from this validation testing to optimize the regularization parameter and kernel parameter within the AISOSVM.

The Support Vector Classifier is implemented in the AISOSVM as it provides a powerful, versatile, and relatively low computation method for online binary classification tasks. By finding the optimal decision boundary that maximizes the margin between classes, support vector classifiers offer robust performance even in the presence of noise and outliers. The use of kernel functions enables SVMs to handle non-linearly separable data, while model selection and regularization help to prevent overfitting and ensure generalization to unseen data.

4 Incremental Learning and Decremental Unlearning

This chapter derives the concepts of incremental learning and decremental unlearning in the context of Support Vector Machines (SVMs) applied to fault detection. As systems evolve and new data becomes available, it is crucial for the fault detection model to adapt accordingly. Incremental learning allows the model to learn from new data, incorporating novel information and patterns, while decremental unlearning enables the model to forget outdated or irrelevant information. The combination of these two processes ensures that the model remains up-to-date and effective in detecting faults as the system conditions change. This chapter will provide an in-depth exploration of the mechanisms underlying incremental learning and decremental unlearning, their integration with the SVM framework, and their potential benefits for enhancing the performance of the adaptive fault detection strategy in complex systems such as spacecraft.

4.1 The online support vector machine

The Online Support Vector Machine (OSVM) is a dynamic learning method designed to adapt to changing environments and data patterns, addressing the challenges associated with online data classification [40]. One of the primary issues with online data classification is the difficulty in acquiring sufficient training data that is representative of all the underlying classification problems. As a result, a classifier trained on limited data may perform poorly when classifying a stream of data not well covered during the learning phase [41]. Additionally, the data stream's characteristics, such as the underlying data distribution, might change over time, causing the classifier built on old data to become inconsistent with new data, a condition known as concept drift.

To tackle these problems, the online support vector machine incorporates incremental learning and decremental unlearning techniques to update the model with new data while simultaneously discarding obsolete or irrelevant information. Cauwenberghs and Poggio [42] proposed the first incremental algorithm for updating an existing OSVM when new samples are acquired, providing an efficient method for incorporating new information into the exist-

ing classifier model. This algorithm adapts the classifier to changes imposed by the addition of new data without re-estimating all the model parameters from scratch. Additionally, it includes an unlearning scheme that allows selective removal of less informative patterns without reducing the classifier's quality, addressing the concept drift issue. This algorithm was expanded upon in Diehl and Cauwenberghs [43] to include more efficient adptation methods through the use of adiabatic perturbations.

The online support vector machine's ability to learn from new data and adapt to changing environments has proven valuable in applications where system conditions or data distributions may change over time, such as fault detection in spacecraft [44, 45]. However, most investigations into incremental OSVM learning algorithms have focused on the training phase rather than the larger model selection process. The model selection phase involves tuning additional variables, known as 'hyper-parameters,' to find a classifier with optimal performance for classifying previously unseen data.

4.2 Algorithm and Implementation

The Incremental Learning and Decremental Unlearning process integrated into the AISOSVM architecture follows the process and derivation of Diehl and Cauwenberghs [43]. With the established Support Vector Machine quadratic programming problem as seen in Equation 3.4, Karush-Kuhn Tucker (KKT) conditions are derived which define the solution of dual parameters for the Langrange multipliers and bias term, α_i and b, respectively. This is achieved by minimizing the set of equations:

$$g_i = \frac{\delta W}{\delta \alpha_i} = \sum_{j=1}^{k} Q_{ij}\alpha_j + y_i b - 1 \quad \begin{cases} > 0 & \alpha_i = 0 \\ = 0 & 0 \leq \alpha_i \leq C \\ < 0 & \alpha_i = C \end{cases} \tag{4.1}$$

$$h = \frac{\delta W}{\delta b} = \sum_{j=1}^{k} y_j \alpha_j = 0$$

Using the partial derivatives g_i, the training samples can be divided into three distinct groups:

1. Set M, which contains the margin support vectors that lie on the margin, $g_i = 0$.
2. Set E consisting of error support vectors that breach the margin and are defined by $g_i < 0$.
3. Set R which comprises the reserve vectors that surpass the margin $g_i > 0$
4. Set U keeps all new samples before moving through the learning process to eventually become either margin or error support vectors.

Within the process of incremental learning, new training samples with $gi > 0$ are directly assigned to R, as they inherently do not contribute to the solution.

Adiabatic increments are used when learning new samples incrementally. When incorporating unlearned samples into the solution, the objective is to maintain the KKT conditions for all previously encountered training data. The KKT conditions are maintained by varying the margin vector coefficients in response to the perturbation caused by the incremented new coefficients. As a result, elements within different vector sets might shift states, leading to incremental learning that proceeds through a series of these "adiabatic" steps. Prior to a given perturbation of the OSVM solution, the partial derivatives with respect to $\alpha_i, b : \forall i \in M$, where M is the set of margin support vectors, is:

$$g_i = \sum_j Q_{ij}\alpha_j + y_i b - 1 = 0 \quad \forall i \in M \tag{4.2}$$

$$h = \sum_j y_j \alpha_j = 0 \tag{4.3}$$

From this set of equations, they can be rearranged and reformulated to be expressed deferentially following the effects of the perturbations and incremented coefficients:

$$\Delta g_i = \sum_{k \in M} Q_{ik} \Delta \alpha_k + \sum_{l \in \mathcal{U}} Q_{il} \Delta \alpha_l + y_i \Delta b = 0 \ \forall i \in M \tag{4.4}$$

$$\Delta h = \sum_{k \in M} y_k \Delta \alpha_k + \sum_{l \in \mathcal{U}} y_l \Delta \alpha_l = 0 \tag{4.5}$$

Considering a specific alteration of the coefficients for the unlearned vectors, represented as $\Delta\alpha_l : \forall l \in U$, the goal is to identify the necessary modifications in the margin vector coefficients, represented as $\Delta\alpha_k : \forall k \in M$, and the bias Δb to ensure the KKT conditions remain satisfied for all existing data. The entire process of perturbation is governed by a perturbation parameter, p, which remains bounded from 0 to 1, as the OSVM solution evolves from its "unlearned" state to the "learned" state. When $p = 0$, the solution reestablishes the previous state before incorporating the new samples. During each perturbation step, the parameter p is incremented by the minimal value, $\Delta p_{\min}$, which triggers a change in category for at least one sample. By the time $p = 1$, all unlearned vectors are part of one of three groups: M, E, or R, ensuring both new and existing data comply with the KKT conditions.

The adiabatic changes, denoted by $\Delta\alpha_i$'s, can be described as the product of Δp and the associated coefficient sensitivities. Define $\Delta\alpha_k = \beta_k \Delta p$ for $k \in M$, $\Delta\alpha_l = \lambda_l \Delta p$ for $l \in U$, and $\Delta b = \beta \Delta p$. After incorporating these expressions into Equation 4.4 and Equation 4.5 and dividing the result by Δp, the differential KKT conditions can be expressed with respect to the coefficient sensitivities:

$$\gamma_i = \frac{\Delta g_i}{\Delta p} = \sum_{k \in M} Q_{ik} \beta_k + \sum_{l \in \mathcal{U}} Q_{il} \lambda_l + y_i \beta = 0 \ \forall i \in M \tag{4.6}$$

$$\frac{\Delta h}{\Delta p} = \sum_{k \in M} y_k \beta_k + \sum_{l \in \mathcal{U}} y_l \lambda_l = 0 \tag{4.7}$$

where γ_l is are the perturbation coefficients. These perturbation coefficients can be freely chosen. Considering that the unlearned vector coefficients will alter by at most by the penalty

factor, C, before the unlearned samples switch categories, the solution for the perturbation coefficients can be set to C such that $\lambda_l = C : \forall l \in U$. The associated coefficient sensitivities, $\beta_k, \beta : \forall k \in S$, can be derived by resolving this set of equations. Upon determining the coefficient sensitivities, it becomes possible to calculate the margin sensitivities γ_i for the error, reserve, and unlearned vectors.

From this problem statement, and using the KKT conditions, a set of rules can be derived and established for determining the data sample transitions between vectors. Table 4.1 lists these viable sample category changes that can occur during incremental learning and decremental unlearning. Within the table, the perturbation parameter Δp, is governed by the coefficient sensitivities, the margin sensitivities, and the perturbation coefficients previously defined.

Table 4.1 online support vector machine sample learning and unlearning conditions.

Training			
Initial Category	New Category	Δp	Conditional
Margin	Reserve	$-\frac{\alpha_i}{\beta_i}$	$\beta_i < 0$
Error	Margin	$-\frac{g_i}{\gamma_i}$	$\gamma_i > 0$
Reserve	Margin	$-\frac{g_i}{\gamma_i}$	$\gamma_i < 0$
Incremental & Decremental Learning			
Margin	Error	$\frac{C-\alpha_i}{\beta_i}$	$\beta_i > 0$
Unlearned	Margin	$-\frac{g_i}{\gamma_i}$	$\gamma_i > 0$
Unlearned	Error	$\frac{C-\alpha_i}{\gamma_i}$	$\gamma_i > 0$
Regularization Parameter, C, Perturbation			
Margin	Error	$\frac{C-\alpha_i}{\beta_i-\Delta C}$	$\beta_i > \Delta C$
Kernel Parameter, σ, Perturbation			
Margin	Error	$\frac{C-\alpha_i}{\beta_i}$	$\beta_i > 0$
Unlearned	Margin	$-\frac{g_i}{\gamma_i}$	$\gamma_i g_i < 0$
Unlearned	Reserve	$-\frac{\alpha_i}{\gamma_i}$	$\gamma_i < 0$
Unlearned	Error	$\frac{C-\alpha_i}{\gamma_i}$	$\gamma_i > 0$

When no initial OSVM solution is available, special attention is required. In this case, all the training samples are unlearned at the outset, and $\alpha_l = 0, b = 0 : \forall l \in U$. This leads to the immediate solution and maintenance of the KKT conditions. However, when the

coefficients of the unlearned vectors begin to increase, this condition will be violated unless $\sum_{k \in U} y_k = 0$, which results in Equation 4.8. It is common to have an unequal distribution of samples across classes, making it generally challenging to maintain the equality conditions of the KKT. The margin vector coefficients help in preserving this condition when an initial OSVM solution is already present. The margin vectors provide the necessary flexibility to adjustment the unlearned vector coefficients. One method to uphold the KKT conditions is to initiate the learning process by selecting one example from each class and developing an initial OSVM.

$$h = \sum_{j} y_j \alpha_j + C\Delta p \sum_{k \in \mathcal{U}} y_k = 0 \tag{4.8}$$

In order to now determine the coefficient sensitivities, β_k, the system of equations presented in Equation 4.6 and Equation 4.7 must be solved. The system of equations can be rewritten in a matrix form for solution as:

$$\mathbf{Q}\beta = -\sum_{l \in \mathcal{U}} \lambda_l \mathbf{v} \tag{4.9}$$

$$\beta = \begin{bmatrix} \beta \\ \beta_{s_1} \\ \vdots \\ \beta_{s_n} \end{bmatrix}, \mathbf{v}_l = \begin{bmatrix} y_l \\ Q_{s_1,l} \\ \vdots \\ Q_{s_n,l} \end{bmatrix} \tag{4.10}$$

$$\mathbf{Q} = \begin{bmatrix} 0 & y_{s_1} & \dots & y_{s_n} \\ y_{s_1} & Q_{s_1 s_1} & \dots & Q_{s_1 s_n} \\ \vdots & \vdots & \ddots & \vdots \\ y_{s_n} & Q_{s_n s_1} & \dots & Q_{s_n s_n} \end{bmatrix} \tag{4.11}$$

With these definitions in place, the sensitivities can be computed. Let $\mathbf{R} = \mathbf{Q}^{-1}$, then

$$\beta = -\sum_{lin\mathcal{U}} \lambda_l \mathbf{R}\mathbf{v}_l \tag{4.12}$$

$$\mathbf{R} \leftarrow \begin{bmatrix} & & & 0 \\ & \mathbf{R} & & \vdots \\ & & & 0 \\ 0 & \dots & 0 & 0 \end{bmatrix} + \frac{1}{\gamma_{s_{n+1}}} \begin{bmatrix} \beta_{s_{n+1}} \\ 1 \end{bmatrix} \begin{bmatrix} \beta_{s_{n+1}} \\ 1 \end{bmatrix}^T \tag{4.13}$$

When removing a margin vector sample from M, $\mathbf{R}$ as shown in Equation 4.14. Within this equation, the first element, the zeroth index, is the bias term, b.

$$R_{ij} \leftarrow R_{ij} - \frac{R_{ik}R_{kj}}{R_{kk}} \; \forall \; i,j \in M \; \cup \; \{0\}; i,j \neq k \tag{4.14}$$

4.3 Model Validation

Model validation is an essential step in the development of any machine learning model, including support vector machines (SVMs). The purpose of model validation is to assess the performance and generalization capabilities of the model when applied to new, unseen data. In this section, two prominent model validation techniques, Leave-One-Out (LOO) estimation and back-testing, will be discussed in the context of Online Support Vector Machines (OSVMs).

4.3.1 Leave-One-Out Estimation

Leave-One-Out (LOO) estimation is an essential cross-validation technique that assesses a model's ability to generalize and make accurate predictions on unseen data. The primary objective of LOO is to estimate the prediction error of a model by simulating the process of making out-of-sample predictions.

In LOO cross-validation, the model is trained on all but one data point ($n-1$ samples), with the omitted data point serving as the validation sample. This process is iteratively performed for each data point in the dataset, resulting in n separate models, where n is

the total number of samples in the dataset. For each iteration, an out-of-sample prediction is made for the omitted data point, and the error between the true label and the prediction is computed. By aggregating these errors, LOO provides an estimate of the model's generalization error, indicating its ability to perform well on unseen data.

The LOO error estimator for OSVMs can be computed using the following formula:

$$\text{LOO}i = \frac{1}{n} \sum i = 1^n L(y_i, f_{-i}(x_i)) \tag{4.15}$$

where n denotes the total number of samples in the dataset, $L(y_i, f_{-i}(x_i))$ represents the loss function evaluated on the true label y_i and the prediction $f_{-i}(x_i)$ obtained by training the model without the i-th example, and LOOi signifies the LOO error estimate. By calculating the average error across all iterations, the LOO error provides a comprehensive measure of the model's performance and generalization capabilities.

There are several benefits to using the LOO cross-validation technique. One primary advantage is that it makes maximum use of the available data for training, as each model is trained on $(n - 1)$ samples, utilizing nearly the entire dataset. This can lead to more accurate performance estimates, especially for smaller datasets.

Additionally, LOO estimation provides an unbiased estimate of the model's generalization error. Since each data point is used as a validation sample exactly once, the results are not sensitive to random partitioning or the selection of specific subsets, unlike other cross-validation methods like k-fold cross-validation.

However, LOO estimation can be computationally expensive, particularly for large datasets, as it requires training n separate models. Fortunately, for OSVMs, the LOO error estimator can be computed efficiently, as the sensitivity of the margin to the removal of each example is already available from the incremental learning and decremental unlearning process. This eliminates the need to retrain the model from scratch for each omitted example, significantly reducing the computational burden.

By providing an unbiased and comprehensive estimate of the model's generalization error,

the LOO error estimator plays a vital role in model selection and hyperparameter tuning. It allows practitioners to identify the best model configuration and monitor the model's performance over time, detecting potential issues like concept drift and overfitting.

4.3.2 Back-Testing

Another method for model validation is back-testing, which involves supplying the training vectors back into the trained OSVM model to determine the predicted outputs on those trained points. This process provides insight into the model's ability to capture the underlying patterns and relationships in the training data, the known nominal conditions, as well as its robustness against overfitting issues.

By evaluating the model's performance on the training vector sets, it is possible to assess the quality of the decision boundary and the model's ability to generalize to the current performance of the autonomous system. This can be an important diagnostic tool in identifying potential issues with the model, such as overfitting or underfitting, and informing potential improvements or adjustments to the learning process.

To perform back-testing on an OSVM model, the following steps can be followed:

- Train the OSVM on the available data using incremental learning and decremental unlearning.
- Collect the labeled data that became a part of the sample sets of the trained model. These are the data points that contribute to the decision boundary and overall best fit curve for the model.
- Supply the training vectors as input to the trained OSVM model, and obtain the corresponding predicted outputs.
- Compare the predicted outputs with the true labels of the support vectors to compute the model's performance metrics, such as accuracy.

The OSVM simplifes the applications of back-testing as it already separates the training data into three separate vectors: the margin vector, the error vector, and the reserve vector. Therefore, the back-testing metric can be directly acquired from the length of the error vector, or the number of samples misclassified.

4.3.3 Model Validation in the AISOSVM

In the context of autonomous system fault detection, rigorous model validation is particularly important, as the performance of the AISOSVM directly impacts the safety and reliability of the system. By ensuring that the AISOSVM is well-suited to the task and capable of generalizing to new, unseen data, it becomes possible to increase confidence in the system's ability to detect faults and respond accordingly. Both LOO and backtesting are performed as a part of the Clonal Selection Algorithm optimization during the AISOSVM design and training process. The Clonal Selection Algorithm uses the results of the back-testing as a part of the accuracy score within the objective function and also monitors the LOO to ensure a threshold of error estimation is not exceeded. This combined model validation produces a viable AISOSVM fault detection system with a balance in accuracy and robustness to historical and future data.

4.4 Incremental Support Vector Machine Application

The efficacy of the AISOSVM's ability to converge to a model solution, regardless of whether it is initially trained on the entire dataset or incrementally learns the dataset in parts, was validated. This adaptability is crucial in scenarios where data is acquired progressively over time, making it impractical to retrain the model from scratch each time new data becomes available. Instead, the AISOSVM can incrementally learn from the new data while maintaining its performance.

To validate the AISOSVM's capability to converge to a model solution, experiments were conducted using a representative dataset. The dataset was gathered from a spacecraft simulation environment discussed in Chapter 8. The variable detector algorithm, discussed in Chapter 5, was invoked to create the positive class of data that surrounds and separates the

samples indiciating nominal performance. The feature space belongs to the attitude control system of the spacecraft and consists of the spacecraft pitch rate, p, and the spacecraft yaw rate, r. The data was gathered from the simulation after the spacecraft was commanded to slew 15 degrees.

- The first model was trained on the entire dataset from the beginning, offline.
- The second model was initially trained on a subset of the training set. It then incrementally learned the rest of the data set, in an online fashion, one sample at a time.

The performance of both models was compared in terms of classification accuracy, detection rate, false discovery rate, and other relevant metrics. Additionally, the model size and computational complexity were analyzed to determine the efficiency of the AISOSVM approach. The offline trained AISOSVM model is presented in Figure 4.1. For the online, incrementally trained AISOSVM, the AISOSVM is illustrated after 70 sample increments. These snapshots are provided in Figure 4.2 The total number of samples learned by both models is 382.

The results demonstrated that the AISOSVM, when trained incrementally on the dataset in an online process, converged to the solution obtained when trained on the entire dataset during offline traing. Both models exhibited exact performance in terms of classification accuracy, detection rate, and false discovery rate. This finding substantiates the AISOSVM's ability to adapt to new data without the need to retrain the model from scratch, making it an attractive choice for applications where data is acquired over time. Additionally, this provides a fundamental step to for applying Reinforcement Learning for autonomous model adaptation as it provides insight into the learning process stability and its impact on the underlying OSVM model.

Additionally, the model size and computational complexity of the incrementally trained AISOSVM were found to be comparable to the model trained on the entire dataset. This

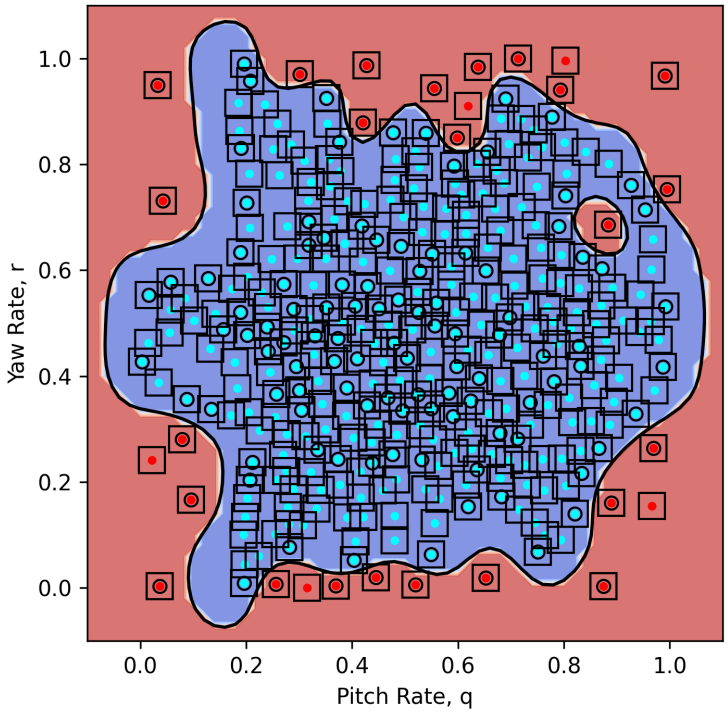
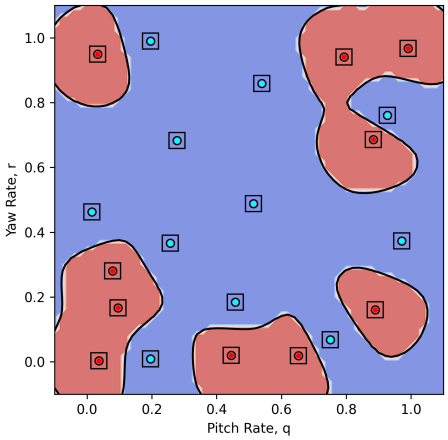
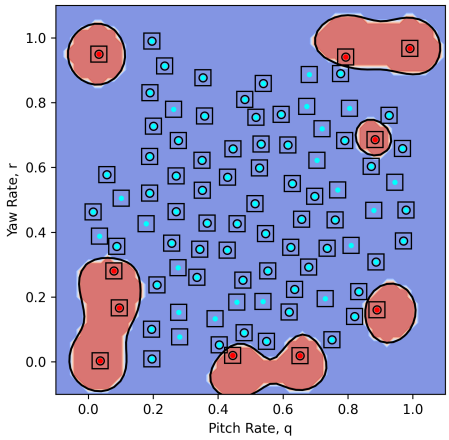


Figure 4.1 The final AISOSVM model after training offline on the entire data set. The blue samples are negative class and the red samples are the positive class of data. Black boxes indicate correct class predictions on those samples during back-testing.

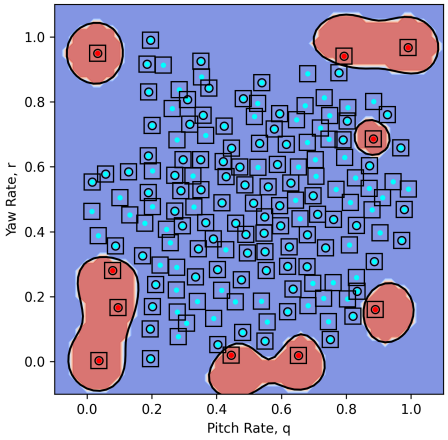
result suggests that the AISOSVM can effectively manage its resource requirements while maintaining its performance, making it a suitable solution for real-world applications where computational efficiency is crucial.



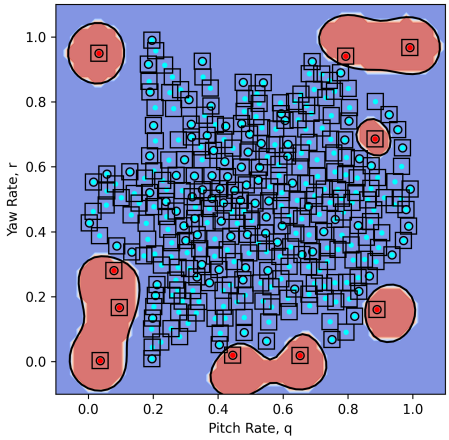
(a) Initial model.



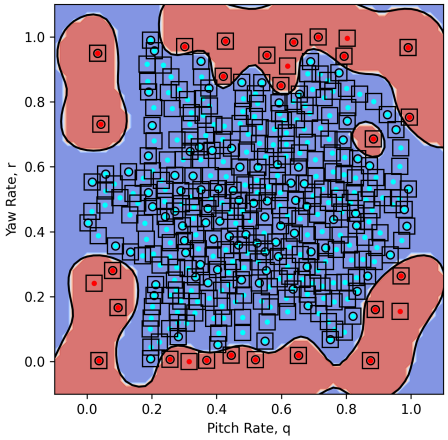
(b) After 70 samples.



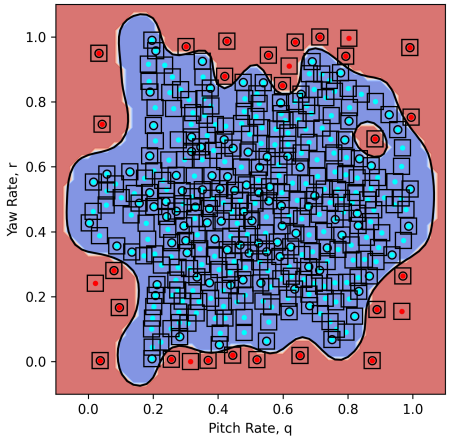
(c) After 140 samples



(d) After 280 samples



(e) After 330 samples



(f) After 382 samples

Figure 4.2 Incrementally trained AISOSVM model. In each subplot, the blue data points present the negative class and the red presents the positive class of data. Black boxes indicate correct class predictions on those samples during back-testing.

5 Artificial Immune System Paradigm

This chapter delves into the Artificial Immune System (AIS) paradigm, a bio-inspired computational framework derived from the principles and processes of the human immune system. AIS has been applied to a wide range of domains, including optimization, pattern recognition, and fault detection, due to its adaptability, robustness, and ability to learn from experience. The chapter begins by providing an overview of the fundamental concepts and components of the human immune system, followed by a discussion of how these principles have been translated into computational models. Specifically, the focus will be on the Clonal Selection Algorithm, the Negative Selection Algorithm, and the V-Detector Algorithm, examining their underlying concepts and applications. The exploration of AIS in the context of fault detection in complex systems and the AISOSVM architecture will be presented.

5.1 Overview and Concepts

The AIS paradigm is a class of biologically inspired algorithms that draw upon the principles and mechanisms of the natural immune system. These algorithms leverage the immune system's remarkable ability to distinguish between self and non-self, adapt to new threats, and retain a memory of past encounters, making them particularly well-suited for fault detection and other pattern recognition tasks.

The major concepts of AIS algorithms include:

- Self-nonself discrimination: AIS algorithms differentiate between normal system behavior, referred to as "self," and abnormal behavior, known as "non-self." By learning to recognize the patterns associated with self and non-self, the algorithms can identify and respond to faults or threats in the system.

- Affinity: The immune system uses a measure of similarity called affinity to determine how well an antibody binds to an antigen. AIS algorithms employ similar measures to assess the similarity between detectors and patterns in the data, allowing them to identify and classify faults.

- Adaptation and memory: The immune system adapts to new threats by generating new lymphocytes and maintaining a memory of past encounters. AIS algorithms incorporate mechanisms for adaptation and memory, enabling them to learn from new data and adapt to changing system behavior.

- Diversity: The immune system generates a diverse population of lymphocytes to recognize a wide range of antigens. AIS algorithms mimic this by creating diverse sets of detectors or classifiers to cover different regions of the feature space, ensuring robust fault detection.

Three widely studied AIS algorithms are the Negative Selection Algorithm (NSA), the Clonal Selection Algorithm (CSA), and the Variable Detector Algorithm (V-Detector). Each of these algorithms has been applied to fault detection tasks with varying degrees of success.

The Negative Selection Algorithm is inspired by the process of self-tolerance in the immune system, where T-cells that recognize self-antigens are eliminated during maturation. In NSA, a set of detectors is generated that do not match any self patterns, allowing them to recognize non-self patterns indicative of faults [46]. The NSA has been used for anomaly detection, intrusion detection, and fault diagnosis in various domains.

The Clonal Selection Algorithm is based on the principle of clonal selection, where B-cells that recognize antigens undergo rapid proliferation and affinity maturation. In CSA, detectors with high affinity for non-self patterns are selected and modified through a process of mutation, enhancing their ability to recognize faults [47]. The CSA has been employed in optimization, pattern recognition, and fault detection tasks.

The V-Detector Algorithm focuses on the generation of a diverse set of detectors, with each detector being responsible for recognizing different fault patterns. The V-Detector algorithm uses an iterative process to adjust the size and shape of the detectors, ensuring coverage of the feature space and robust fault detection [48, 49].

AIS algorithms offer a promising approach to fault detection by leveraging the powerful mechanisms of the natural immune system. The NSA, CSA, and VDA have been applied

to various fault detection tasks, demonstrating the potential of AIS algorithms for detecting and adapting to faults in complex systems.

5.2 Negative Selection Algorithm

The AISOSVM optimization approach is grounded in the AIS paradigm, which simulates the biological immune system's response to antigens and other threats in living organisms. The immune system can differentiate between foreign cells (*nonself*, $\overline{S}$) and the organism's natural cells (*self*, S). The AIS models the generation of T-cells by the thymus gland, which are designed to detect and identify antigens entering the body. In autonomous systems, these pathogens can represent system faults, anomalies, or cyber-attacks [11]. T-cells that do not react with the identified *self* are allowed to mature and remain active in the immune system to bind and destroy antigens. This process is known as negative selection. The AIS can be employed within a FDIR algorithm to search for abnormal conditions by comparing a current configuration of nominal features within the vehicle against the current configuration of features of the same vehicle under upset conditions [50].

In the AIS-inspired fault detection and diagnosis paradigm, the negative selection algorithm is responsible for generating antibodies or detectors for a specific system feature space. The dimension of this feature space equals the number of identifiers chosen by the designer, which can be any system measurement (e.g., system states) or computed variables (e.g., state estimates) relevant to the system's performance and having a clear impact on nominal and abnormal conditions. While this specific application focuses on a spacecraft, the algorithm is system-agnostic. The identifier values are normalized between 0 and 1 and cover the entire range of the feature space under nominal conditions. In this fault detection strategy, the *self* (S) represents the subset of the system feature space Σ corresponding to normal flight conditions, while the *nonself* ($\overline{S}$) represents the abnormal conditions. The *self* and *nonself* should be disjoint sets that span the feature space [51]:

$$\overline{S} \cap S = 0 \quad \text{and} \quad \overline{S} \cup S = \Sigma \tag{5.1}$$

The Negative Selection Algorithm (NSA) generates detectors (antibodies) from data clusters (cells). The *self* and *nonself* are represented by a set of geometrical hyperbodies, with clusters being a subregion of the *self* and detectors or antibodies being a subregion of the *nonself*. For this fault detection application, a fixed-radius NSA using hyperspheres is employed to generate the classes of data belonging to the self-nonself regions of the chosen feature space of the system. The sets of hyperspherical clusters, D_s, and antibodies, D_a, are defined as:

$$D_s = (c_s, r_s) \quad D_a = (c_a, r_a) \tag{5.2}$$

where $c_s \in R^n$ and $c_a \in R^n$ represent the hyperspace locations of the centers of the clusters and detectors, respectively, and $r_s \in R$ and $r_a \in R$ represent the radii of the clusters and detectors, respectively. In a fixed-radius NSA, each data element has the same radius as the other elements in their subspace.

The method begins by defining the *self* through the collection of a data set containing all representative *self* samples. Then, new immature detectors, or candidate detectors $\tilde{D_c}$, are randomly generated and compared with the *self* set. If $\tilde{D_c}$ overlaps an existing mature detector D or the *self*, $\tilde{D_c}$ is terminated, and a new detector is created. If the immature detector does not overlap, the detector matures and is added to the array of detectors. This process continues until a chosen threshold is reached for the number of consecutively terminated immature detectors. Detector overlapping is computed using the Minkowski distance:

$$d = (\sum_{i=1}^{k} |\beta_i - x_i|^{\lambda})^{1/\lambda} \tag{5.3}$$

where d is the distance from the center x_i of $\tilde{D_c}$ to the center of the *self* or mature detector sample, β_i, and λ is the dimension of the data set. For a two-dimensional problem, $\lambda = 2$, and the Minkowski distance becomes equivalent to the Euclidean distance. The error can be

obtained from this distance using the following equation:

$$\begin{aligned} E &= d - \gamma \\ \gamma &= r_a + r_s \end{aligned} \tag{5.4}$$

where r_a is the radius of the detector, and r_s is the radius of the *self* samples. If $E > 0$, the candidate detector D_c does not match the *self* samples or the existing detectors and will mature and be included in the detector set. If $E \leq 0$, the candidate detector overlaps an existing sample and will be terminated.

The AISOSVM is a two-class classifier and a supervised machine learning algorithm, requiring two sets of data for training the underlying online support vector machine to differentiate between the two data sets. However, generating sufficient data that captures all normal and abnormal behaviors is challenging within complex vehicle dynamics, harsh environments, and unpredictable subsystem operations. To address this, the NSA is utilized within the OSVM training and optimization process to mitigate the need for strenuous modeling and simulation testing. The NSA enables OSVM training with limited *a priori* knowledge of failure conditions, as OSVM training can now be used with nominal performance data. The NSA creates the data points that capture the *nonself*. An example of the NSA-generated antibodies surrounding the *self* data can be seen in Figure 5.1. This data was obtained from two states, yaw and roll rate, of a spacecraft operating at nominal flight conditions. For the NSA, antibodies with a fixed radius were chosen to reduce variability in the optimization process and increase OSVM training accuracy, as the OSVM hyperplane is only a function of the data point locations themselves and not the arbitrary distance of the *nonself* data point center to the nearest *self* data point.

5.3 Variable Detector Strategy

A related fault detection and diagnosis approach grounded in the AIS paradigm and self-nonself-discrimination methodology is the Variable Detector (V-Detector) algorithm. V-Detector uses a negative selection process to generate antibodies or detectors that cover the

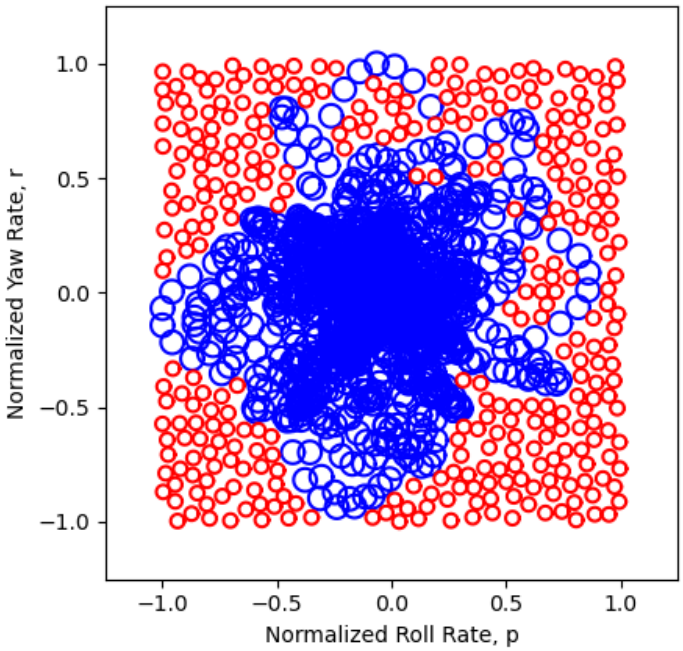


Figure 5.1 Final generation of antibodies using NSA showing *self* (blue) and *nonself* (red).

nonself region based on collected data defining the *self*, or nominal dynamics of a system. However, in contrast to the general NSA that employs a fixed radius for the generated antibodies, the V-Detector algorithm adopts a pseudo-optimization process, allowing the radius of each antibody to change to maximize *nonself* region coverage with the fewest antibodies without overlapping the *self* [52, 53].

The V-Detector strategy starts by selecting features representing the nominal dynamics of the system. These features can be sensor measurements or estimated quantities, but the chosen features should be sensitive to the system's normal or abnormal conditions. After identifying the features, the values are normalized in the range [0.0, 1.0], thereby defining the *self* space. Following the normalization of the *self* data, an initial candidate detector pool is generated with a vector of randomly selected coordinates corresponding to the center of each detector.

During each iteration, the V-Detector strategy selects a candidate detector from the candidate detector pool and evaluates if the candidate should mature and become part of

the final detector set. In the first iteration before any detectors are matured, a candidate's radius is set equal to the minimum distance to the *self* points. The distance is determined using the Minkowski distance, d, as previously defined in Equation (5.3). When a candidate detector is closest to a *self* data point, the detector radius is set to $r_a = d - r_s$, where r_s is a fixed radius value for all *self* points. Conversely, if the candidate detector is closest to a matured detector, the radius of the candidate is set to the distance d.

A candidate detector can be rejected for two reasons. If the candidate covers any part of the *self* region, it is immediately rejected. This violation is determined using the same process as the negative selection algorithm with Equation(5.4). When the candidate is closest to a matured detector, a designer-defined threshold indicates the acceptable amount of overlap between detectors. If the candidate with radius r_{a_c} overlaps a mature detector with radius r_{a_m} more than the threshold, it is rejected. The threshold, δ, defining the maximum allowable overlap is bounded from [0, 1] and calculated using Equation(5.5) [52]:

$$\delta = \frac{r_{a_c} + r_{a_m} + d}{2r'_a} \tag{5.5}$$

where $2r'_a$ is the smaller detector radius among the two detectors being compared. This candidate detector maturing process continues until either a predefined number of antibodies is reached, indicating appropriate self-nonself coverage, or until a maximum number of iterations is achieved. Once this process concludes, the resulting antibody set and corresponding radii are sorted in descending order by radius size. Figure 5.2 displays the generated antibodies with varying radius sizes surrounding the *self* for a roll rate and yaw rate feature space from a generic satellite model.

During system testing and online fault detection when the V-detector algorithm is deployed by itself, samples of the feature sets are collected, normalized, and their distances individually checked against each mature detector in the antibody set. With the antibodies sorted in descending order by radius, the incoming sample is compared to the detectors most likely to be activated first. Detector activation is determined by the distance, d, between the

sample point and a mature detector such that $d < r_a$, where r_a is the radius of the mature detector in the comparison. Due to the close clustering and overlapping of detectors, a buffer is generally used to mitigate false alarms. Once a number, n of detectors are activated in n sequential time steps, a fault is detected. When applied to the OSVM for the training process, these individual detectors are generalized to the decision boundary of the OSVM. Thus, instead of comparing each new sample to these generated detectors, the output of the OSVM determines the class of data and whether or not a detector is activated.

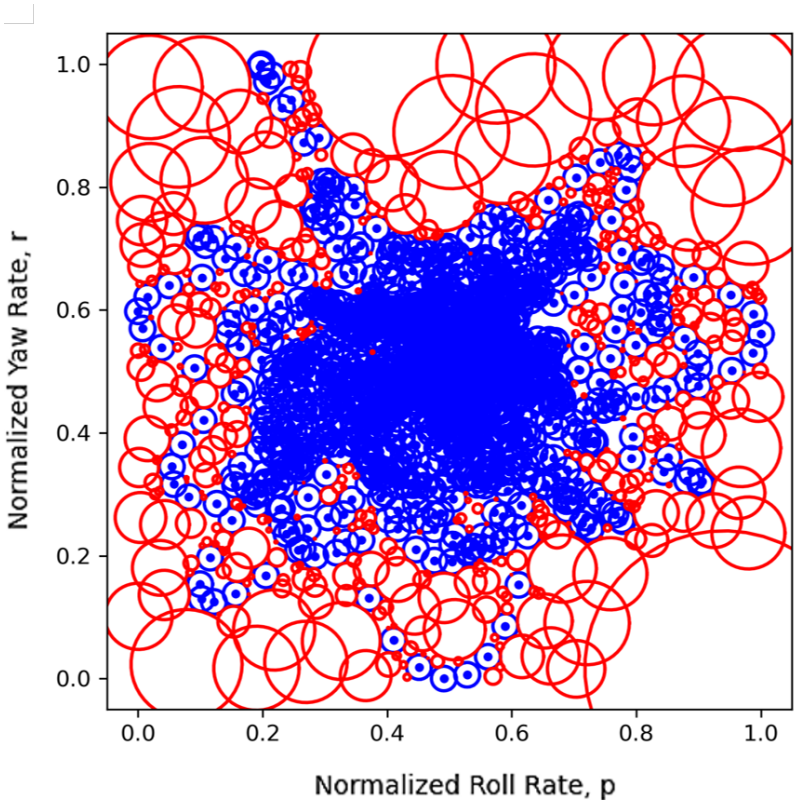


Figure 5.2 Generated detectors (red) surrounding *self* data (blue) using V-Detector the method.

5.4 Clonal Selection Algorithm

Optimizing the antibody set generated by the negative selection algorithms and the hyperparameter configuration for AISOSVM models is crucial to fault detection performance. To achieve this optimization, CSA is employed. The Clonal Selection Theory models an adaptive immune system's response. When an organism encounters an antigen or foreign

cell, B lymphocytes are produced in the bone marrow. If the B-Cells can bind to the antigen, they mature and divide, creating clones through mitosis, which forms the basis of the clonal selection theory [54]. The main features of the clonal selection theory, of which CSA is derived, follows the CLONALG implementation [55]. The CLONALG includes:

- random and accelerated genetic changes of generated cells to an antibody pattern
- proliferation of cells with bound antigens
- maintenance and retention of a single antibody pattern expressed by a differentiated cell - *clone*

CSA applies the clonal selection theory to create an immune system-inspired optimization process, which is proven to be an adaptive optimizer similar to genetic algorithms. CSA continues until a predetermined stopping threshold is reached, based on the number of generations. The Clonal Selection Algorithm is outlined in Figure 5.3, but its process is described as follows, using the strategy outlined by CLONALG [47]:

(1) Generate a binary string for each parameter that CSA will optimize. Include a maximum and minimum value for each parameter, and decode the string to a suitable figure for use in subsequent calculations.

(2) Create an initial antibody population randomly from the binary string representations. The number of antibodies generated for each generation depends on the user.

(3) Rank the antibodies using an affinity-proportionate function and clone the best n individuals, creating an initial clone set (C). Generate the number of clones for each population according to the user-defined $Clone_{rate}$, which is a percentage of the population size such that the new clones $Nc = round(Clone_{rate} \cdot P)$.

(4) Mutate the clones, C, using bit flips in their binary representation. The mutations are proportionate to the clone's affinity, with lower affinities having a higher chance of mutation. The function $exp(-Mutation_{rate} \cdot Affinity_{candidate})$ defines this probability.

(5) Create a new population of "memory cells" using the population of clones, C, which replaces the best populations of P, and the remaining population of candidates.

(6) Replace several antibodies within the new memory cell population with randomly generated candidates. The number of random candidates added to each generation is set by the user. Replace the lowest affinity antibodies with these new cells.

(7) Terminate CSA if the maximum generation limit is reached. Otherwise, use the new population to replace the antibody population from the previous iteration.

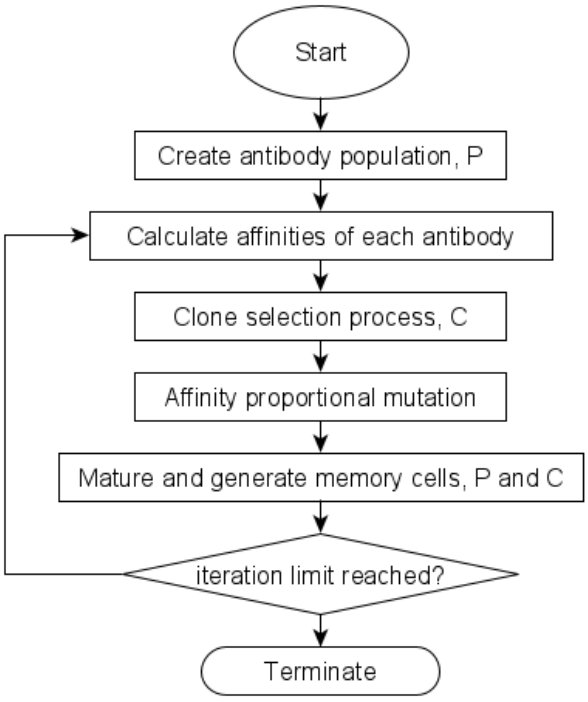


Figure 5.3 Clonal selection algorithm process.

6 The AISOSVM

In Chapter 6, the AISOSVM (Artificial Immune System Optimized Support Vector Machine) approach is presented, which combines the strengths of both artificial immune systems and support vector machines. This chapter elucidates the rationale behind integrating these two methodologies, demonstrating the benefits of leveraging the robustness and adaptability of artificial immune systems with the powerful classification capabilities of support vector machines. The AISOSVM implementation details and key components, such as the clonal selection algorithm and the variable detector algorithm, are discussed to provide a comprehensive understanding of the proposed methodology.

6.1 Development and Motivation

The AISOSVM is a novel approach to fault detection that combines the strengths of artificial immune system algorithms with the adaptive capabilities of online support vector machines. This innovative method addresses some of the key challenges faced by traditional data-driven and model-based fault detection schemes, particularly in the context of complex systems like spacecraft, where the accurate prediction of system behavior and fault propagation is often difficult. This section discusses the development and motivation behind the AISOSVM, highlighting the benefits of integrating AIS algorithms with Online SVMs and comparing this approach to the current state of the art in fault detection.

AIS algorithms are inspired by the natural immune system's ability to adapt to new and evolving threats, providing robust protection against a wide range of pathogens. In the context of fault detection, AIS algorithms offer several significant advantages over conventional methods. First, they are capable of optimizing the hyperparameters of the OSVM, ensuring that the model is well-tuned to the specific characteristics of the system being monitored. This can improve the overall accuracy and performance of the fault detection process, particularly when dealing with noisy or incomplete data.

Second, AIS algorithms can generate synthetic data to enhance class separation and mitigate the need for gathering "failure" data during the training process. This is particularly

valuable in complex systems like spacecraft, where failures can be unpredictable and difficult to replicate for training purposes. By creating synthetic data that accurately represents potential fault conditions, the AISOSVM can be trained to detect a wide range of faults, even those that may not have been observed in the real system. This ability to "learn" from synthetic data, combined with the optimization of OSVM hyperparameters, enables the AISOSVM to achieve high levels of fault detection accuracy and sensitivity, surpassing the performance of many traditional approaches that require extensive training data.

The integration of the OSVM with incremental learning and decremental unlearning adds an additional layer of adaptability to the AISOSVM, allowing it to respond to new conditions as they occur within the system and to new "nominal" conditions. This is particularly important in complex systems like spacecraft, where the operating environment and system behavior can change over time, necessitating the continuous updating of the fault detection model. By incorporating incremental learning and decremental unlearning techniques, the AISOSVM can adapt to these changes in real-time, ensuring that it remains effective even as the system evolves.

Moreover, the adaptive nature of the AISOSVM allows the system to be trained even after deployment, once "nominal" conditions have been established and verified in the operating environment. This further mitigates the need for high-quality data obtained during the training process, as the model can continue to refine its understanding of the system behavior and fault conditions after it has been deployed. In this way, the AISOSVM offers a significant advantage over traditional data-driven and model-based fault detection schemes, which often require extensive pre-deployment training and may not be able to adapt to changes in the system behavior or operating environment.

Despite the numerous advantages of the AISOSVM, it is important to acknowledge certain limitations and weaknesses inherent in the approach. One potential drawback is the computational complexity involved in optimizing the OSVM hyperparameters and generating synthetic data using AIS algorithms. This can lead to increased computational demands,

particularly in large-scale or high-dimensional systems, potentially impacting the efficiency and real-time applicability of the fault detection process. Another potential issue is the reliance on the quality of the synthetic data generated by the AIS algorithms. If the synthetic data does not accurately represent the true fault conditions, the AISOSVM may struggle to detect certain types of faults, reducing its overall effectiveness. Additionally, the adaptability of the AISOSVM, while beneficial in many respects, could lead to overfitting if not properly controlled, particularly in situations where the system is exposed to a high degree of noise or uncertainty. It is crucial to carefully balance the trade-offs between adaptability and generalization to ensure that the AISOSVM remains effective across a wide range of fault scenarios and system conditions.

In comparison to model-based fault detection approaches, the AISOSVM's data-driven nature presents some unique challenges. One significant drawback is the inability to analytically validate the AISOSVM as can be done with model-based approaches. Model-based methods often rely on well-established physical models and mathematical relationships, providing a level of certainty and confidence in their performance. In contrast, data-driven methods like the AISOSVM are heavily dependent on the quality and representativeness of the training data, which can make it difficult to guarantee the system's performance and reliability under all possible conditions. Additionally, model-based approaches are often more transparent and interpretable, allowing for easier identification of the root causes of faults and a clearer understanding of the underlying system dynamics. With the AISOSVM, the decision-making process may be more opaque, making it challenging to diagnose issues and implement effective corrective actions.

The AISOSVM represents an advancement in fault detection capabilities, offering a highly adaptive and flexible approach that combines the strengths of artificial immune system algorithms and online support vector machines. By optimizing the OSVM hyperparameters, generating synthetic data to represent potential fault conditions, and adapting to new conditions in real-time, the AISOSVM can achieve high levels of fault detection accuracy and

sensitivity, surpassing the performance of many traditional data-driven and model-based methods. As such, it represents a promising alternative for complex systems like spacecraft, where the accurate prediction of system behavior and fault propagation is often challenging or impossible. The development and motivation behind the AISOSVM underscore its potential to revolutionize fault detection in complex systems, enabling more reliable and robust monitoring of critical components and ensuring the safety and longevity of these systems.

6.2 Algorithm and Implementation

The proposed threat detection strategy utilizes a support vector machine optimized using bio-inspired algorithms that mimic the immune system response to antigens. The Clonal Selection Algorithm (CSA) performs the optimization process by mimicking the production and adaptation of immunizing cells when an antigen enters the body. Negative selection-based algorithms, namely the Variable Detector (V-Detector) algorithm, are employed during this optimization process to create the distinction between nominal and off-nominal system performance and to improve the online support vector machine training and optimization. The proposed threat detection process is outlined in Figure 6.1.

The AISOSVM architecture aims to provide a comprehensive and adaptive fault detection solution, specifically designed to address the challenges faced by complex systems. The architecture is divided into two main phases: the Offline Training phase and the Online Detection and Isolation phase.

The Offline Training phase commences with the General Training Set, a compilation of labeled data samples that encompasses normal or normal and faulty operating conditions of the system. Data acquisition is a crucial aspect of this step, as the quality and diversity of data directly influence the performance of the AISOSVM. The data can be sourced from simulations, ground test data, or historical operational data, ensuring a comprehensive representation of the system's behavior under various conditions. To facilitate the training process, this data set undergoes normalization to ensure that all features share the same scale. Following normalization, the Clonal Selection Algorithm is applied, incorporating the

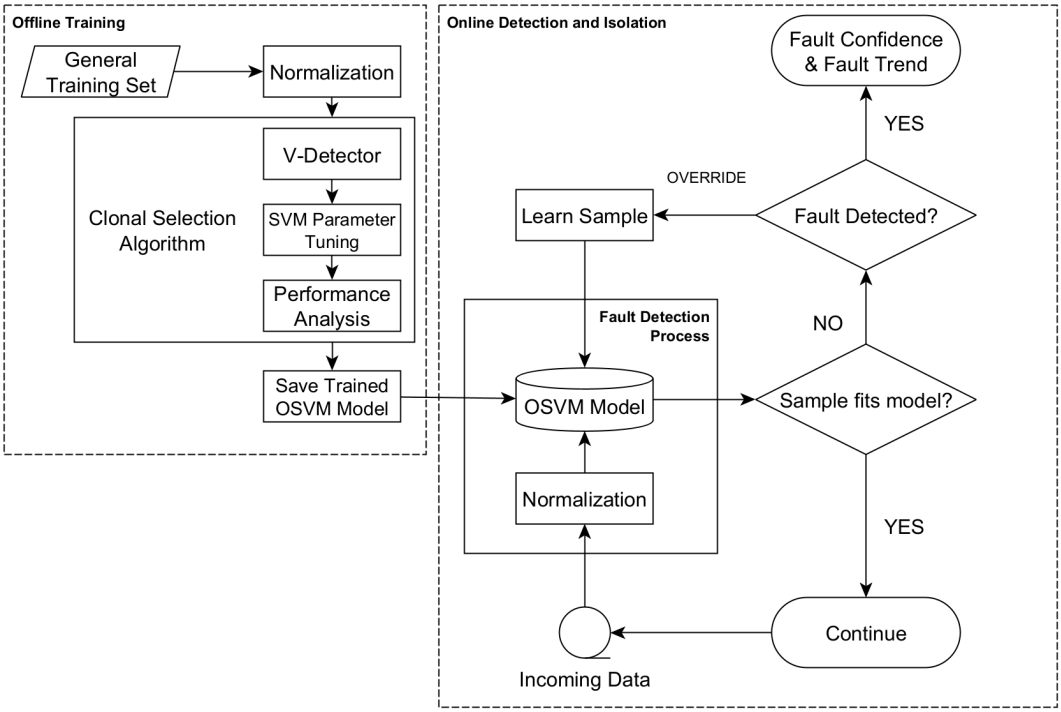


Figure 6.1 The immune system augmented fault detection architecture.

V-Detector, OSVM training, and model performance validation.

The Clonal Selection Algorithm serves as the optimization algorithm for generating an optimized OSVM. Its primary function is to optimize the OSVM hyperparameters, which in turn adjust the OSVM's performance based on the training data. Additionally, the Clonal Selection Algorithm optimizes the parameters of the V-Detector algorithm in terms of detector generation performance and detector radius generation. The V-Detector algorithm generates synthetic data points, which enhance class separation and mitigate the need for gathering failure data during the training process. This synthetic data creation is particularly beneficial in situations where obtaining failure data is difficult, expensive, or impossible. The Clonal Selection Algorithm optimizes the OSVM's hyperparameters, ensuring that the OSVM model is tailored to the training data, maximizing its performance. The Perform Analysis step evaluates the performance of the trained OSVM model on a validation set, determining its effectiveness in detecting faults.

Once the training is complete, the trained OSVM Model is saved and deployed onboard

the system for use during the Online Detection and Isolation phase.

The Online Detection and Isolation phase begins with the deployed fault detection system operating onboard the autonomous system for health monitoring. Operational data is sampled from the system sensors and data streams in near real-time. These samples are normalized to maintain consistency with the training data. The samples are then passed through the OSVM Model within the Fault Detection Process to determine the data label (i.e. nominal or abnormal). The output of the OSVM Model provides this classification. If the sample fits the model's nominal classification, the process continues with the next Incoming Data sample. If not, the system moves into a Fault Detected stage to determine if a fault is actually detected.

At this juncture, the system may be in an "Override" mode. This operating mode allows operators to make the decision to adapt the OSVM model and to learn this sample as a new nominal condition or to decrementally learn previous samples to adjust accuracy as desired. If the Override is active, the process transitions to the Learn Sample step, updating the OSVM Model with the new sample, and no fault is detected. This adaptability allows the AISOSVM to acclimate to new nominal conditions, enhancing its performance over time. Operators may want to enter this Override mode during the first period of operations once the autonomous system's performance is verified to be nominal. This will allow even a partially trained OSVM model to learn in near real-time the true nominal conditions without requiring operators in the loop. This can be especially useful for spacecraft as downlink operational data, retraining the system on the ground, and then redeploying and uplinking the updated model to the spacecraft may be restricted. If the Override is not active, the process concludes with Fault Confidence & Fault Trend, signaling a fault detection. Within this terminating step, operators can be notified of the fault with additional fault metadata corresponding to the confidence level of the classification and a quantifiable indicator for fault severity to assist with fault diagnosis processes and system recovery maneuvers.

The AISOSVM architecture offers a comprehensive and adaptive solution to fault detec-

tion, melding the power of artificial immune system algorithms with the adaptability of the online support vector machine. By incorporating the Clonal Selection Algorithm for optimizing OSVM hyperparameters and the V-Detector algorithm parameters, the AISOSVM overcomes some limitations of traditional data-driven approaches. Moreover, its ability to adapt to new conditions during the Online Detection and Isolation phase ensures a more robust and reliable fault detection system, making it an ideal candidate for mission-critical applications like spacecraft health monitoring.

6.3 Integration with Artificial Immune System Algorithms

As previously described, once the initial set of training data is acquired and normalized, CSA is invoked to generate the AISOSVM model. Within the optimization algorithm, each optimization parameter is assigned a definite range for the antibodies which are encoded into a binary string. The parameters used for the AIS optimized OSVM are the OSVM hyperparameters C and σ and the *self* and *nonself* cell radii, r_s and r_a respectively, for NSA. The CSA features and values used for optimization for all simulation test cases are given in Table 6.1. These values provided a standard and wide sweeping range for parameter tuning of the AISOSVM.

Table 6.1 Optimization parameters used within CSA for AISOSVM model generation.

Clonal Selection Algorithm (CSA)	
Number of generations	100
Population size	50
Mutation factor, $Mutation_{rate}$	-2
Clone factor, $Clone_{rate}$	30%
Number of random cells generated	10%
Online Support Vector Machine	
Penalty Factor, C	$[0.1, 50]$
RBF Kernel Parameter, σ	$[0.001, 5]$
V-Detector Algorithm	
Minimum Nonself radius, r_a	$[0.01, 0.06]$

The OSVM is trained using the sequential minimal optimization process for each CSA

population member n_b. The two OSVM hyperparameters and NSA antibody radii are used with the nominal data obtained from the spacecraft simulation to train the AISOSVM. The AISOSVM process calculates the value of the objective functions for each population member $F(n_b)$. The affinity of each n_b is obtained with the objectives of obtaining the best C, σ, r_s, and r_a combination. The multi-objective affinity is defined in Equation 6.1 and the final objective function is given in Equation 6.2:

$$\begin{cases} F_1(n_b) = 100 \times \text{auc} \\ F_2(n_b) = 100 \times (1 - \frac{\#\,\text{support vectors}}{N}) \\ F_3(n_b) = 100 \times (\frac{\text{nonself coverage}}{\text{coverage threshold}}) \end{cases} \tag{6.1}$$

$$F(n_b) = -(\alpha \cdot F_1(n_b) + \beta \cdot F_2(n_b) + \delta \cdot F_3(n_b))\,, \quad \text{for}\, \alpha > \beta\,, \delta \tag{6.2}$$

where the variable "auc" refers to the "Area Under the Curve" of the Receiver Operating Characteristic (ROC) curve, and N is the number of training data. The first two objective functions are defined to obtain the OSVM parameters, C and σ. The other two fitness functions are used to ensure the NSA generates sufficient antibodies to cover the abnormal conditions while avoiding extreme cases of overfitting. The fitness functions are combined with a weighted subtraction to create a minimization objective function, $F(n_b)$, for the AISOSVM model. The weights for this objective function can be tuned by the user to assess various model characteristics. For fault detection applications, classification accuracy remains priority over the other parameters; thus, α is selected to be largest weight. The weights used for the satellite fault detection application are shown in Table 6.2. If the population member's affinity is a good match to the antigen, that member is selected for cloning and the next generation. Using the objective function, the OSVM performance is calculated for each population member at each generation until the maximum generation is reached. After the last generation is evaluated, the best member that minimizes the objective function is chosen as the AISOSVM model for threat detection.

Table 6.2 Objective Function Weights for AISOSVM model optimization.

Objective Function Weight	Value
α	0.50
β	0.30
δ	0.20

6.4 Fault Trend Analysis

Fault trend analysis is an essential aspect of fault detection systems, as it facilitates the identification of gradual changes in the system's behavior before a critical event occurs. In this section, we discuss leveraging the prediction output from the online support vector machine as an indicator of the dynamics trend within the feature space.

The OSVM classifies samples into two classes: -1 or $+1$, representing nominal and abnormal samples, respectively. The classification is performed using Equation (3.6). However, if the sign reduction is excluded, the output provides a pseudo-distance to the decision boundary, represented as:

$$f(\boldsymbol{x}) = \sum_{i=1}^{k} \alpha_i y_i \phi(\boldsymbol{x}_i, \boldsymbol{x}) + b \tag{6.3}$$

The output from Equation 6.3 does not provide the actual distance to the support vector machine decision boundary. However, this output can still be used to produce a generic understanding of where a sample is located within the OSVM model relative to other samples and the decision boundary. Samples lying along the decision boundary have an output of zero from this equation. By analyzing this output, fault trends within the system can be deduced. The derived fault trend analysis thresholds are summarized Table 6.3.

Monitoring the output of the OSVM and analyzing the fault trends allows not only for fault or failure detection but also for anticipating their occurrence. This approach provides an additional layer of intelligence to the fault detection system, enabling operators to make informed decisions and implement preventive measures to minimize the impact of faults on the system's performance. Figure 6.2 and Figure 6.3 provides examples of this trend analysis

Table 6.3 OSVM rule-based fault trend analysis.

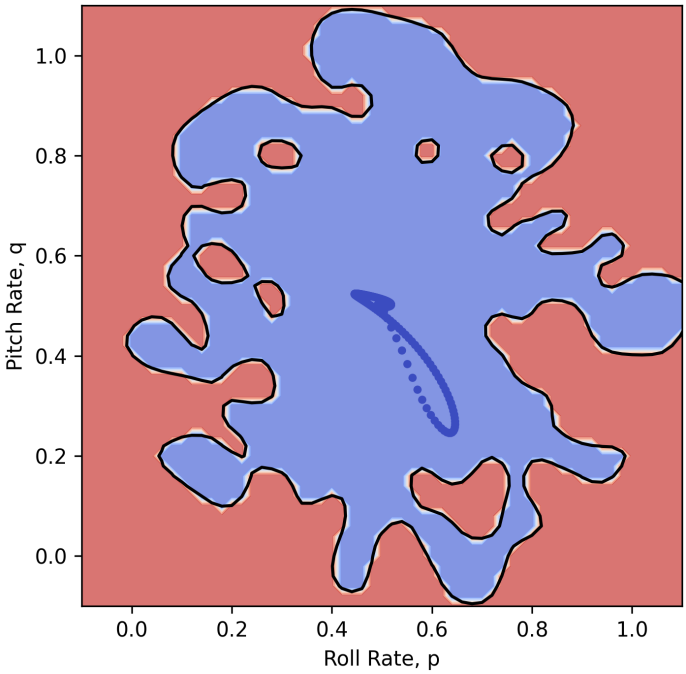
Condition	Fault Trend
$f(\boldsymbol{x}) \leq -1$	Nominal samples
$f(\boldsymbol{x}) > -1$ and $f'(\boldsymbol{x}) < 0$	Trending nominal
$f(\boldsymbol{x}) > -1$ and $f'(\boldsymbol{x}) > 0$	Trending abnormal
$f(\boldsymbol{x}) > 0$	Abnormal samples

for a nominal test and a failure test case, respectively. In these figures, yellow signifies a warning that a data sample is between 0 and -1, green indicates the sample is stables and less than or equal to -1 and red indicates a positively classified sample portraying an anomaly or failure.

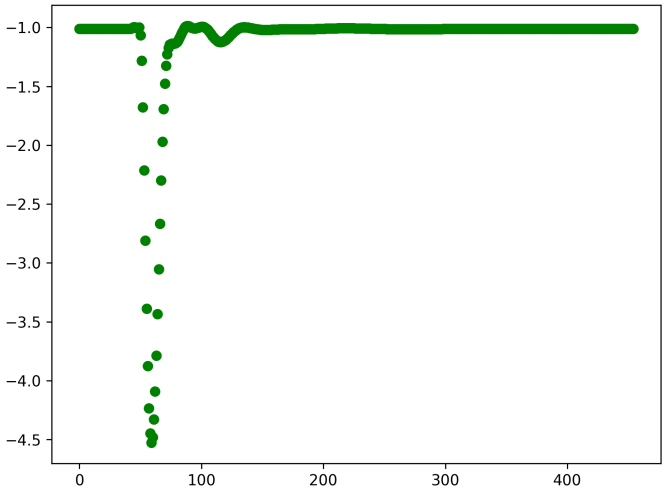
Furthermore, the ability to discern fault trends can provide valuable information for system health management, allowing for more accurate predictions of the remaining useful life of components, subsystems, and the overall system. By incorporating fault trend analysis, maintenance scheduling can be optimized, leading to reduced maintenance costs, improved system reliability, and extended system lifespan.

Incorporating fault trend analysis into the AISOSVM framework enhances its ability to autonomously adapt to changing system dynamics. By continuously monitoring the system's behavior and identifying fault trends, the AISOSVM can adjust its model parameters to maintain high detection accuracy and reduce false alarms. This adaptability makes the AISOSVM a powerful tool for fault detection in complex and dynamic systems, such as aerospace platforms.

Moreover, the fault trend analysis can be employed to detect and diagnose multiple fault types, including abrupt faults, incipient faults, and intermittent faults. Abrupt faults manifest themselves suddenly and significantly affect the system's performance. Incipient faults are characterized by a gradual change in the system's behavior and can eventually lead to catastrophic failures if left undetected. Intermittent faults are sporadic in nature and can occur at irregular intervals, making them difficult to detect and diagnose. By observing the output of the OSVM and analyzing the fault trends, the AISOSVM can effectively detect



(a) AISOSVM model with nominal data overlay.

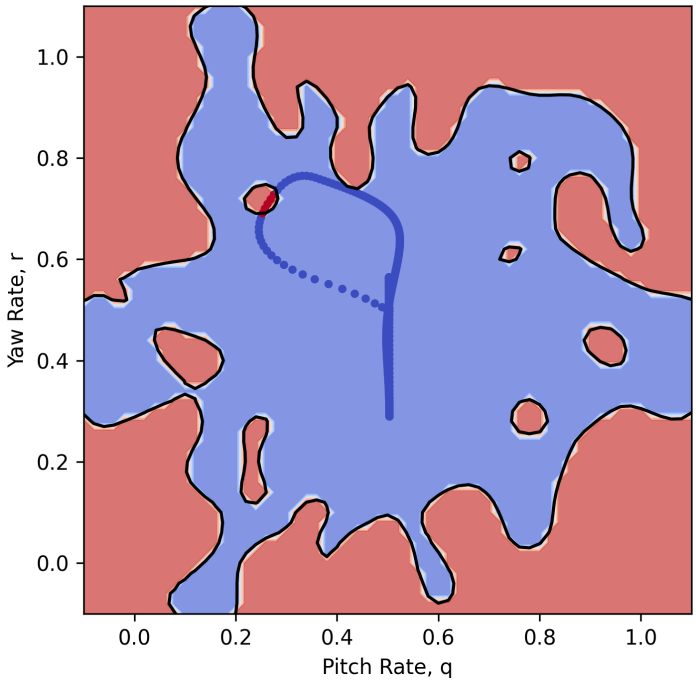


(b) Corresponding fault trend analysis.

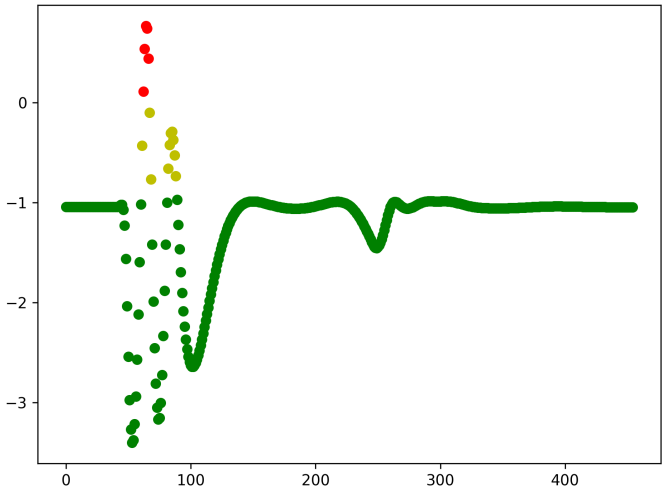
Figure 6.2 Example of the AISOSVM used for fault trend analysis for a nominal test case with no detected anomalies.

and diagnose these various fault types, allowing for timely intervention and maintenance actions to prevent further damage or system failures.

Additionally, fault trend analysis can be utilized in conjunction with other diagnostic techniques, such as feature extraction and signal processing, to enhance the overall fault



(a) AISOSVM model with failure data overlay.



(b) Corresponding fault trend analysis.

Figure 6.3 Example of the AISOSVM used for fault trend analysis for a failure test case.

detection and diagnosis capabilities of the AISOSVM framework. This multimodal approach allows for increased robustness and adaptability, as well as improved accuracy and precision in detecting and diagnosing faults.

Fault trend analysis plays a crucial role in fault detection systems by providing valuable insights into the system's dynamics and behavior. By incorporating this analysis into the

AISOSVM framework, the system's adaptability and fault detection capabilities are significantly enhanced. This approach allows for the effective detection and diagnosis of various fault types, optimization of maintenance scheduling, and improved system health management. As a result, the AISOSVM, when augmented with fault trend analysis, becomes an indispensable tool for fault detection in complex and dynamic systems.

6.5 Performance Metrics

Several performance metrics can be used to evaluate the effectiveness of the AISOSVM for fault detection in autonomous systems. These metrics are essential in providing an objective assessment of the algorithm's ability to identify faults accurately and efficiently. The chosen metrics take into consideration the different aspects of the fault detection problem, such as the balance between true and false detection notifications and the robustness of the model under different conditions.

The Receiver Operating Characteristic (ROC) is a graphical representation used to evaluate the performance of a binary classification system, such as the AISOSVM fault detection system. It displays the relationship between the true positive rate (sensitivity) and the false positive rate (1-specificity) at various threshold settings. The ROC curve is plotted with the true positive rate (TPR) on the y-axis and the false positive rate (FPR) on the x-axis. The closer the curve is to the top-left corner of the graph, the better the classification system's performance. A perfect classification system would have a ROC curve that passes through the top-left corner, indicating 100% sensitivity and 100% specificity. An example of an ROC is depicted in Figure 6.4.

The Area Under the ROC Curve (AUC) is a scalar value that summarizes the overall performance of a binary classification system. It represents the probability that a randomly chosen positive instance will have a higher score than a randomly chosen negative instance. An AUC value of 1.0 signifies perfect classification, while an AUC value of 0.5 indicates that the classifier performs no better than random chance. Higher AUC values correspond to better classification performance.

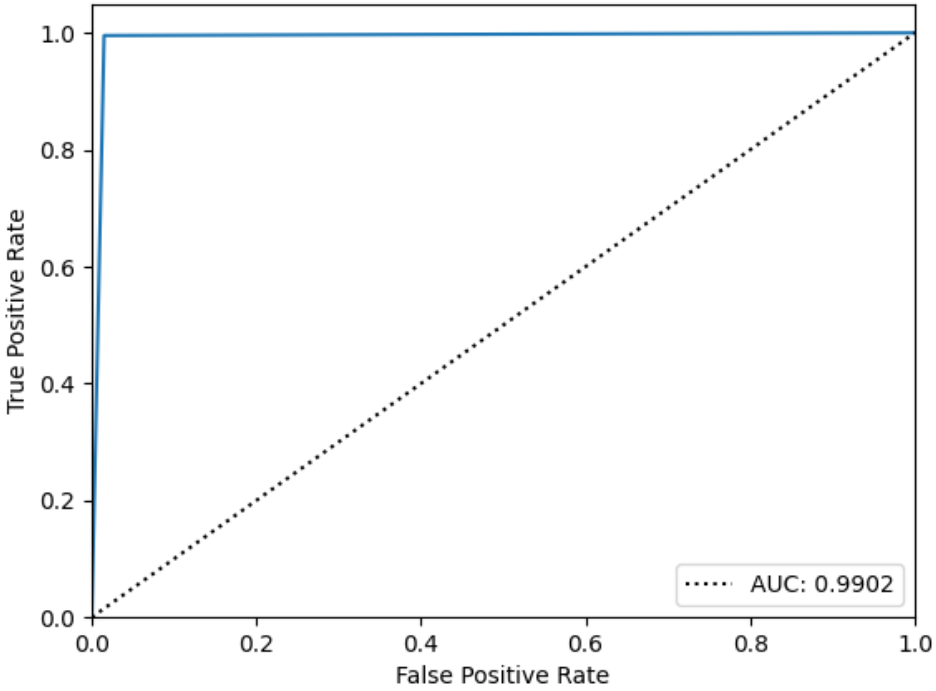


Figure 6.4 An example Receiver Operating Characterstic curve for the AISOSVM.

The True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN) are fundamental metrics in binary classification problems. These metrics provide a way to quantify the number of correct and incorrect detections made by the AISOSVM:

- True Positives (TP): The number of faults correctly identified as faults.
- False Positives (FP): The number of non-faults incorrectly identified as faults.
- True Negatives (TN): The number of non-faults correctly identified as non-faults.
- False Negatives (FN): The number of faults incorrectly identified as non-faults.

The Detection Rate (DR), False Positive Rate (FPR), and False Discovery Rate (FDR) are essential performance metrics that offer complementary insights into the effectiveness of the AISOSVM for fault detection in autonomous systems. By examining these metrics, we

can gain a deeper understanding of the trade-offs between accurate fault detection and the occurrence of false alarms.

The DR, also known as sensitivity or recall, measures the proportion of true positive detections to the total number of actual faults provided by Equation (6.4). A higher DR value indicates that the AISOSVM is successful in identifying a larger fraction of the faults present in the system. However, a high DR does not necessarily imply that the algorithm is perfect, as it might still produce a significant number of false alarms. To strike an optimal balance between accurate fault detection and the number of false alarms, we also need to consider the False Positive Rate and the False Discovery Rate.

$$DR = \frac{TP}{TP + FN} \tag{6.4}$$

False alarm metrics include the FPR and the FDR. FPR, also known as the Type I error rate or (1-specificity), is the proportion of actual negative cases that are incorrectly identified as positive by the classifier as shown in Equation 6.5. Applied to the AISOSVM fault detection system, the FPR is the ratio of false positives to the sum of false positives and true negatives given by Equation (6.6). A lower FPR indicates better performance in avoiding false alarms. The FDR, one the other hand, is the proportion of positive cases identified by the classifier that are actually false positives. In the context of the AISOSVM fault detection system, the FDR is the ratio of false positives to the sum of false positives and true positives. A lower FDR indicates better performance in minimizing false alarms. It is important to minimize these metrics, as false alarms can lead to unnecessary corrective actions, which could be resource-intensive and potentially detrimental to the spacecraft's operation.

$$FPR = \frac{FP}{FP + TN} \tag{6.5}$$

$$FDR = \frac{FP}{FP + TP} \tag{6.6}$$

When evaluating the performance of the AISOSVM, it is crucial to consider both detection rate and false alarm metrics in tandem, as they provide a more comprehensive view of the algorithm's effectiveness. A well-performing AISOSVM would ideally exhibit a high DR and a low false alarm rates, which would signify accurate fault detection with a minimal number of false alarms. However, there is often a trade-off between these two metrics, and optimizing one may come at the expense of the other. For instance, if the AISOSVM is tuned to be more conservative in its fault detection, it might result in a lower DR, as fewer true faults are detected, but also a lower FPR and FDR, as fewer false alarms are generated. Conversely, a more aggressive AISOSVM might achieve a higher DR by detecting more true faults but may also produce a higher FOR and FDR due to an increased number of false alarms.

Previously discussed in Chapter 4, to assess the robustness and generalization of the AISOSVM, Leave-One-Out (LOO) estimation and the k-folds validation method are also employed. In the LOO estimation, one data point is held out as the validation set, while the remaining data points are used for training. This process is repeated for each data point, and the average performance across all iterations is calculated. The k-folds validation method is an extension of the LOO estimation, where the dataset is divided into k equal-sized folds. In each iteration, one fold is used as the validation set, while the remaining $k - 1$ folds are used for training. The average performance across all k iterations is then calculated.

By evaluating the AISOSVM using these performance metrics, a comprehensive understanding can be obtained regarding its effectiveness and robustness in detecting faults within spacecraft systems. The performance metrics discussed in this section provide a comprehensive evaluation of the AISOSVM fault detection system's effectiveness in accurately detecting faults. The ROC curve and AUC measure the overall classification performance, while the TP, FP, TN, and FN metrics help quantify the system's accuracy, sensitivity, and specificity.

The detection rate, false alarm rates (FPR and FDR), and LOO estimation further contribute to understanding the system's performance in identifying faults and minimizing false alarms. By examining these performance metrics, we can determine the AISOSVM fault detection system's suitability for real-world applications and identify areas for potential improvement.

7 Autonomous Adaptation

This chapter delves into the crucial aspect of autonomous adaptation in the context of developing a robust and effective machine learning model for fault detection in autonomous systems. The chapter explores the integration of reinforcement learning, specifically deterministic Q-learning, with the previously discussed AISOSVM model. This combination aims to enhance the AISOSVM model's ability to adapt to changing environments and system dynamics autonomously. Furthermore, the chapter discusses the methodology for integrating Q-learning with the AISOSVM model, elucidating the underlying principles and mechanisms that facilitate the fusion of these two techniques. A foundation is constructed that outlines potential viability of integrating the reinforcement learning methods for autonomous online learning with the incremental learning process of the AISOSVM.

7.1 Reinforcement Learning with Q-Learning

Reinforcement learning (RL) is a subfield of machine learning focused on developing algorithms that enable agents to learn optimal decision-making policies through interaction with an environment. The primary objective of RL is to maximize the cumulative reward obtained by the agent as it navigates the environment and takes actions. RL has been successfully applied to various fields, such as robotics, game playing, autonomous vehicle control, and resource allocation, among others [56, 57].

One of the key concepts in reinforcement learning is the Markov Decision Process (MDP), which provides a formal framework for modeling decision-making problems in stochastic environments [58, 59]. An MDP is defined by a tuple (S, A, P, R), where S represents the set of states, A denotes the set of actions, P is the state transition probability function, and R is the reward function. The goal in an MDP is to find a policy $\pi : S \rightarrow A$, which maps states to actions, that maximizes the expected cumulative reward over time.

Q-learning is a popular model-free, off-policy reinforcement learning algorithm that has been extensively studied and widely applied to various problems. The central idea behind Q-learning is to learn the action-value function $Q(s, a)$, which represents the expected cu-

mulative reward for taking action a in state s and following an optimal policy thereafter. The action-value function is defined as:

$$Q(s,a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} | S_t = s, A_t = a, \pi^*\right] \tag{7.1}$$

where $\gamma \in [0,1)$ is the discount factor, R_{t+1} is the reward at time $t+1$, and π^* is the optimal policy.

The Q-learning algorithm iteratively updates the action-value function by performing temporal-difference (TD) updates based on the agent's experience [60]. The update rule for Q-learning is given by:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)\right] \tag{7.2}$$

where α is the learning rate, s_t and a_t are the current state and action, r_{t+1} is the immediate reward, and s_{t+1} is the next state. The term $\max_{a'} Q(s_{t+1}, a')$ represents the maximum action-value for the next state, which is used to update the current action-value estimate [61].

Q-learning has several attractive properties, such as convergence to the optimal action-value function under certain conditions, even when the agent follows an exploratory policy. This enables Q-learning to be used in settings where the agent has limited knowledge of the environment and must balance exploration (discovering new states and actions) with exploitation (choosing actions that maximize the expected reward).

There are various approaches to implementing Q-learning, depending on the nature of the state and action spaces. For discrete state and action spaces, the action-value function can be represented as a table, with entries for each state-action pair. However, for continuous or high-dimensional state spaces, function approximation techniques, such as neural networks or kernel methods, are often employed to represent the action-value function [62].

In the context of autonomous system fault detection, reinforcement learning, and specif-

ically Q-learning, can provide several benefits:

- **Adaptability.** Q-learning enables the agent to learn and adapt to the environment and system dynamics in an online fashion, making it well-suited for dealing with changing conditions and non-stationary processes.

- **Robustness.** The model-free nature of Q-learning allows it to be robust to uncertainties and inaccuracies in the environment's model, as it does not rely on prior knowledge of the state transition probabilities or reward function.

- **Scalability.** Q-learning can handle large state and action spaces when combined with function approximation techniques, enabling its application to complex systems with numerous variables and components.

- **Decision-Making.** By learning an optimal policy, Q-learning allows the agent to make autonomous decisions that maximize the expected cumulative reward, which can be useful for identifying and responding to faults in the system.

Integrating Q-learning with the AISOSVM model for fault detection in autonomous systems involves several key steps. First, the system's state representation must be designed to capture relevant information about the system's dynamics, components, and potential faults. This representation should be concise and informative, allowing the RL agent to learn an effective policy efficiently. Next, the action space must be defined, which typically includes actions the agent can take to diagnose and mitigate faults or to further explore the system.

Once the state and action spaces are defined, the Q-learning algorithm can be applied to learn the action-value function and, subsequently, the optimal policy for fault detection and response. The learning process involves the agent interacting with the environment, taking actions, observing state transitions and rewards, and updating the action-value function according to the Q-learning update rule. The agent may follow an exploratory policy, such as ϵ-greedy, to balance exploration and exploitation. This ϵ-greedy function defines a threshold

$\epsilon \in [0, 1]$ that governs the probability of choosing one method over the other. The Q-Learning process flow is provided in Figure 7.1

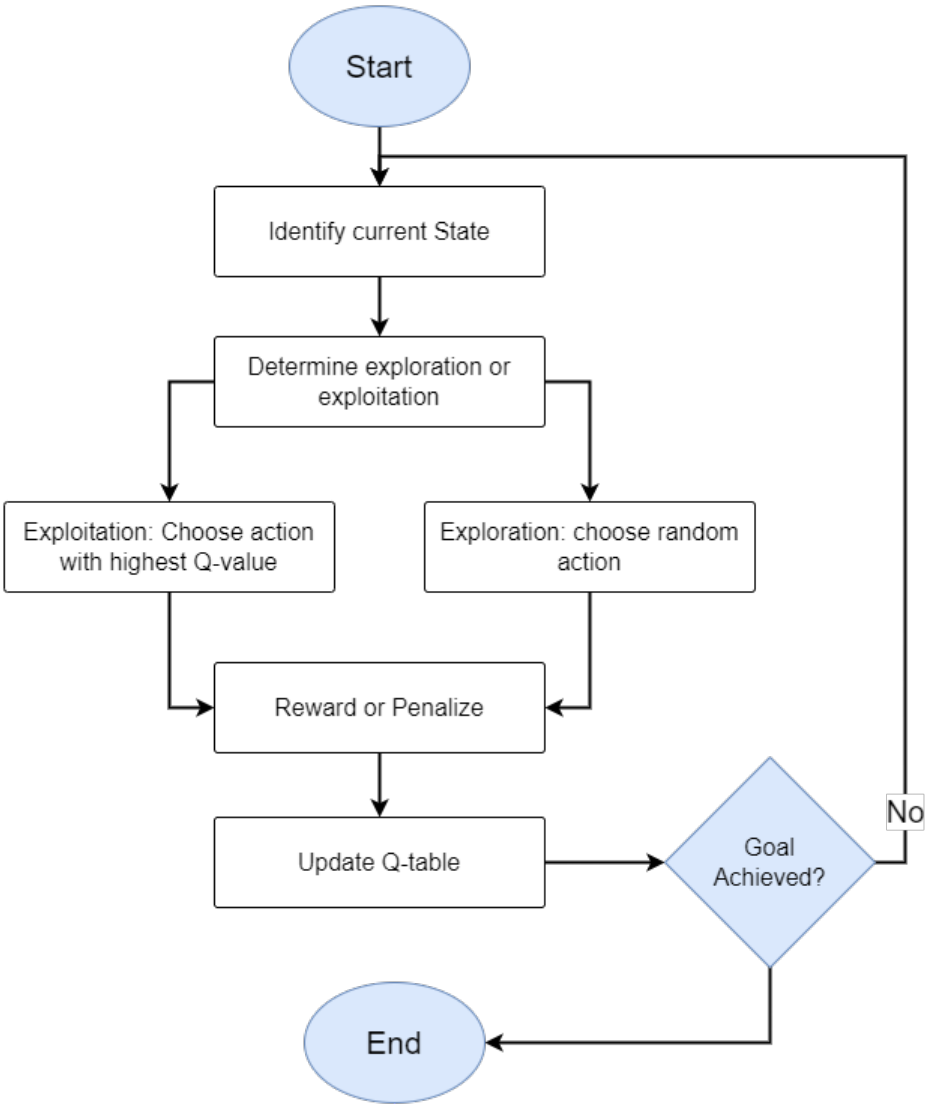


Figure 7.1 Q-learning process flowchart.

In the next section, the integration of Q-learning with the AISOSVM model will be discussed in greater detail, outlining the methodology and specific techniques used to fuse these two approaches into a cohesive framework for autonomous fault detection.

7.2 Q-Learning Integration with the AISOSVM

The integration of Q-learning with AISOSVM aims to enhance the adaptability and autonomy of the fault detection system. By incorporating Q-learning into the AISOSVM

framework, the system can autonomously adapt its parameters in response to the changing dynamics of the environment or the system itself. This integration enables the AISOSVM to learn from the system's experiences and improve its performance over time.

In the proposed AISOSVM framework, Q-learning is used to update the action-value function based on the observed system states, actions taken, and the rewards received. The rewards can be defined as the accuracy of the fault detection, the speed of detection, or a combination of both. The Q-learning algorithm explores different actions, such as adjusting the OSVM parameters, the feature selection process, or the AIS algorithms, to optimize the performance of the fault detection system. This incorporation of reinforcement learning within the fault detection system architecture is illustrated in Figure 7.2.

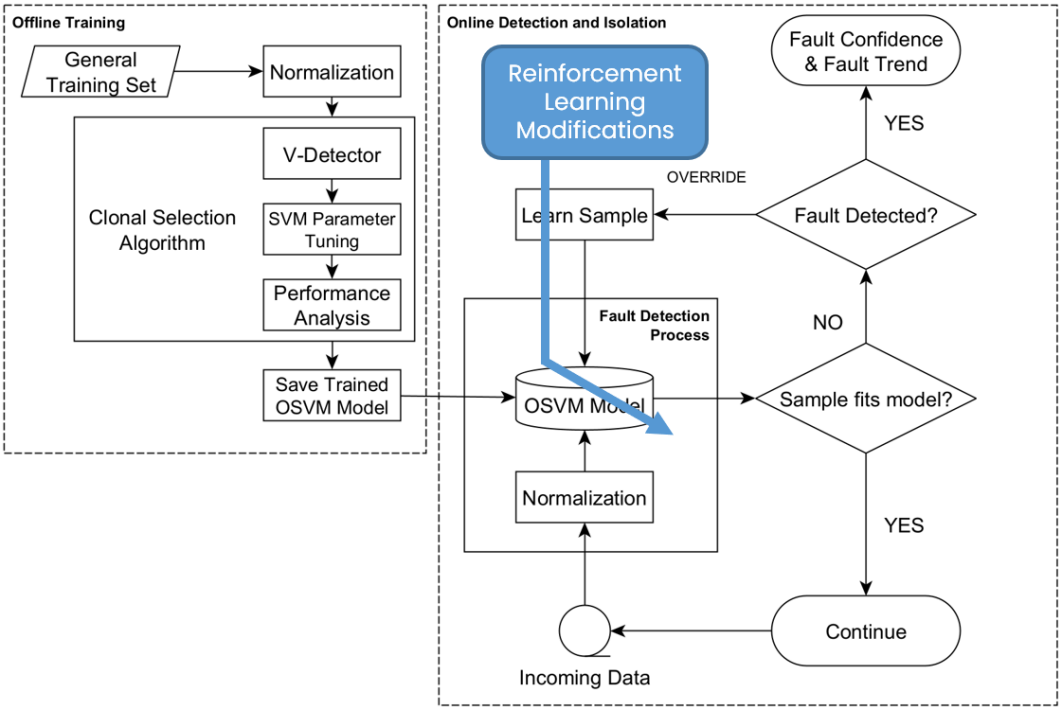


Figure 7.2 Implementation impact of Reinforcement Learning on the AISOSVM architecture.

For the AISOSVM implementation, the following actions and states could be used for the Q-Learning process:

- **Actions:**

- Perturb the regularization parameter, C
- Perturb the kenerl parameter, σ
- Learn the newly encountered sample
- Unlearn a previously encountered sample
- Do nothing

- **States:**
  - Detection Rate (DR)
  - False Discovery Rate (FDR)

Other possible states could include the false positive rate and the F1-Score. These are bounded between $[0, 1]$. Within this defined problem setup, the states within the Q-table would be a $\mathbb{R}^{n \times n}$ matrix where n is the number of states for both the Detection Rate and False Discovery Rate. For the AISOSVM, a step of 0.01 was used for the two states. The goal of the Q-Learning method is to achieve a very high detection rate (optimal is Detection Rate = 1.0) and a very low number of false alarms (optimal is False Discovery Rate = 0.0). An example of the Q-Table is provided in Table 7.1. The table is arranged with the DR state column ranging from $[0, 1]$ and the FDR column ranging from $[1, 0]$. Additionally, for each state pair, the five actions are also represented as a number between 0 and 4 mapping to one of the five actions previously discussed.

Table 7.1 Q-Table example.

Entry ID	Action	DR	FDR	Q
0	0	0.99	0.01	0.0100
1	1	0.99	0.01	-0.0100
2	2	0.99	0.01	0.0190
3	3	0.99	0.01	-0.0100
4	4	0.99	0.01	0.0271
5	0	0.99	0.00	0.0340
6	1	0.99	0.00	-0.0100

Once the Q-learning algorithm converges to an optimal policy, it provides guidance for the AISOSVM on how to adjust its parameters and strategies in different system states to achieve optimal performance in fault detection. The integration of Q-learning with AISOSVM enables the development of a more robust and adaptive fault detection system that can handle a wide range of operating conditions and fault scenarios. This autonomous adaptation capability is particularly important for spacecraft systems, as it allows them to respond to unforeseen events and maintain their functionality in the face of faults or anomalies.

7.3 Q-Learning Implementation

The primary objective of integrating Q-learning into the OSVM is to enhance the model's adaptability and maintain its performance while keeping the model size low, thus minimizing computational resources required for its management.

The Q-learning process starts with a partially trained support vector machine, providing an initial state for the algorithm. Two possible actions are defined: (1) learn the new incoming sample, and (2) do nothing. The Q-learning table converges to a specific value for the given starting point and a predetermined number of episodes, based on the following reward structure:

Action: Do nothing

- If the OSVM performance metrics for detection rate (DR) or false discovery rate (FDR) does not degrade, reward 1.
- If the OSVM's DR and FDR both worsen compared to the previous OSVM, penalize -1.

Action: Learn sample

- If the OSVM's DR and FDR improve, reward 1.5.
- If the OSVM's DR and FDR degrade, penalize -2.

The rationale behind this reward system is to encourage the OSVM to maintain its performance while minimizing the model size; this rationale is part of a "Do No Harm" policy. When performing autonomous fault management within autonomous systems, such a policy must be implemented to ensure the fault management does not adversely affect the system. A "Do No Harm" policy prioritizes the maintenance of the current OSVM's performance such that is does not deteriorate due to a learning action taken by the Q-learning algorithm. Additionally, it is crucial to avoid learning every new sample to prevent an increase in model size and subsequent computational resource requirements.

To implement Q-learning within the AISOSVM architecture, the state representation should capture the current performance of the OSVM, including its detection rate and false discovery rate. The state space can be discretized to facilitate the implementation of the Q-learning algorithm, and the Q-table can be initialized with arbitrary values. During each episode, the Q-learning algorithm follows an exploratory policy, such as the ϵ-greedy strategy, to balance exploration and exploitation. The agent interacts with the environment by selecting actions based on the current Q-table and observing the resulting state transitions and rewards. The Q-table is then updated according to the Q-learning update rule provided in Equation 7.2. Upon convergence of the Q-learning table, the learned Q-values can be used to inform the decision-making process for learning new samples within the AISOSVM framework. This adaptive, data-driven approach enables the OSVM to maintain its performance while keeping the model size low, ultimately providing an effective solution for autonomous system fault detection and adaptation.

For this initial test of the Q-Learning implementation, the parameters for the Q-Learning process is provided in Table 7.2.

The Q-Table was trained from an initial state taken from the detection rate and the false discovery rate of an initial AISOSVM model. This fault detection model was initially trained on 100 data samples with a set regularization parameter and kernel parameter. The initial detection rate was 0.72 and the original false discovery rate was 0.31. The original output

Table 7.2 Q-Learning parameters for AISOSVM integration.

Parameter	Value
Learning rate, α	0.1
Discount factor, γ	0.5
Greed factor, ϵ	0.1
Episodes	100

of this initial model is shown in Figure 7.3.

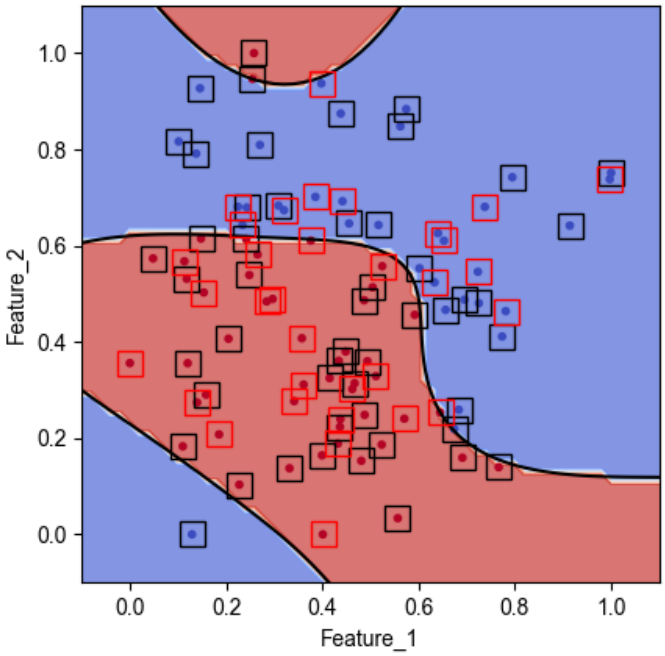


Figure 7.3 Initial AISOSVM model used for Q-Table generation.

After achieving this initial state, the Q-Learning implementation was invoked to train the Q-Table on the best actions to improve the model performance. After 100 episodes, the Q-Table displays a trend and convergence to a set of best actions for the AISOSVM model states as shown in Figure 7.4. The Q-learning process is able to converge to a reward system value that improves the performance of AISOSVM model. The Q-Values correspond to the best actions to take based on the initial starting state and any subsequent states as well. After determining the Q-Values and best actions for this initial model, the table was used

online to dictate whether a new sample should be learned and added to the model or whether no action should be taken to adapt the model. Figure 7.5 provides a map indicating the best action to take based on the maximum Q-value for an initial detection rate ("state_dr") and false discovery rate ("state_fdr"). This map showcases the Q-Learning table implementation for 50 learning actions taken across each of the 100 episodes within the training process. This map shows a clear trend emerging from the training process; the Q-Learning process is trending towards the bottom right of the map representing the goal state of the AISOSVM: 100% detection rate and 0% false alarms.

Using this Q-Table for autonomous training decisions within the AISOSVM architecture, the autonomously selected actions improved the initial model's performance by increasing the detection rate and decreasing the false discovery rate. This online AISOSVM adaptation demonstration used the maximum Q-Value to determine the best action to take for the model's given state for a single feature space. Additionally, to ensure the model actions did not get stuck in a decision loop where the only action to take was to do nothing thus never incrementing the model state, a 1% chance to take a random action was provided so long as that action did not degrade the AISOSVM performance from its current condition. From this process, the final adapted model is shown in Figure 7.6. Additionally, a Table 7.3 provides the online Q-Learning process results.

Table 7.3 Online AISOSVM adaptation results.

Initial Conditions	
Detection Rate, DR	72%
False Discovery Rate, FDR	31%
Q-Table Actions	
Nothing Actions Taken	1934
Learning Actions Taken	426
Percentage of Nothing Actions	81.9%
Percentage of Learning Actions	18.1%
Final Conditions	
Detection Rate, DR	83%
False Discovery Rate, FDR	23%

Figure 7.4 Average Q-Values for all trained state-action pairs and the initial AISOSVM model states.

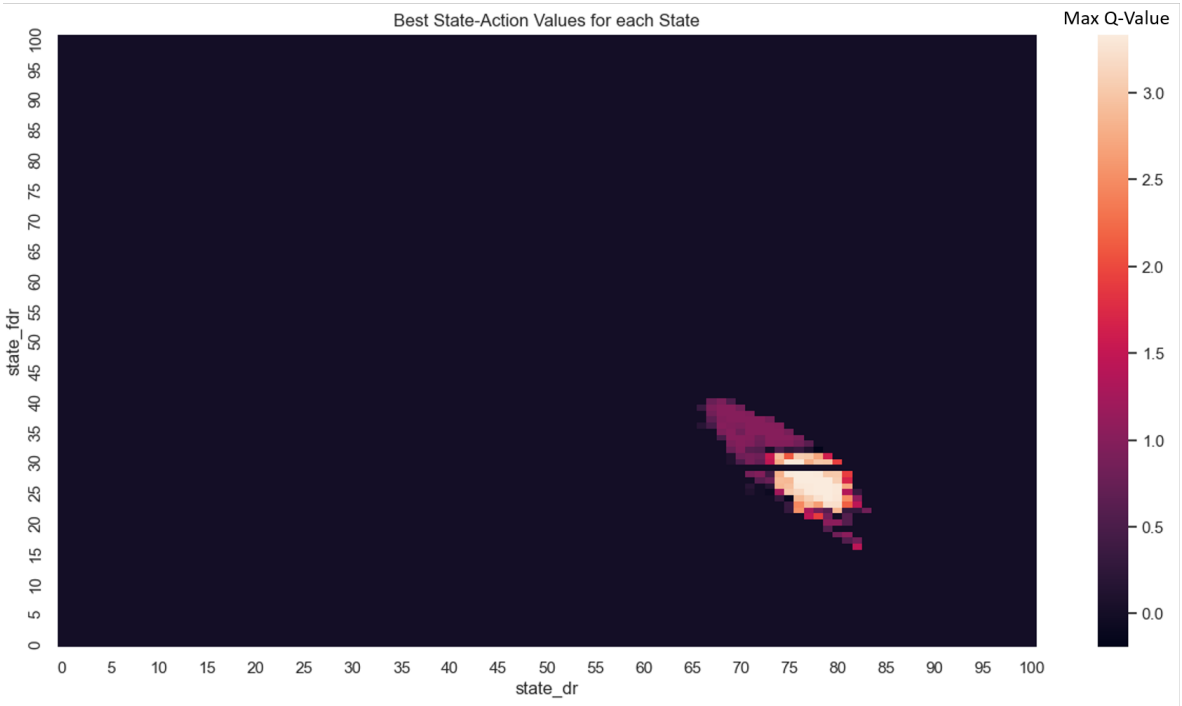


Figure 7.5 Best action Q-values based on the current AISOSVM model state.

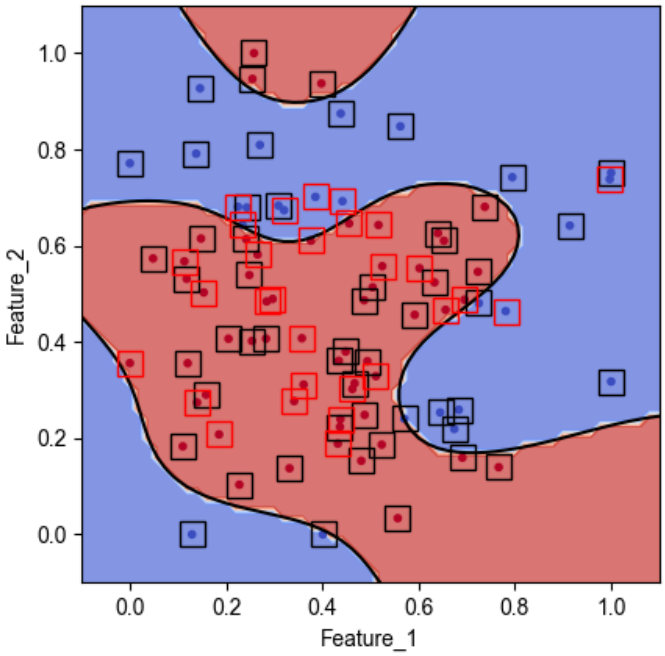


Figure 7.6 AISOSVM model adapted online using the generated Q-Table.

This demonstration only covered an initial start state for the AISOSVM due to the limitations of the deterministic learning process and the requirement to be trained on each state-action pair. Future work will investigate additional framework requirements for fully implementing the Q-Learning process online with the AISOSVM architecture. To gather a complete map of the state action feature space, the Q-Learning table may need to be trained alongside the initial AISOSVM in order to gain a greater coverage of potential states. An alternate approach could be simple rule-based systems to limit the total state action feature space to those within the surrounding area of the initial AISOSVM model and to implement the "Do No Harm" protocol to revert the model if an action is taken to the detriment of the performance. Ideally, the output of the AISOSVM offline training process should be a model with very high detection rates and very low false discovery rates. Therefore, it may be of benefit to limit the scope of the Q-Learning process to the immediate vicinity of that initial state as a single misclassified sample would not change the state parameters by a significant amount assume the AISOSVM model size is a few hundred samples.

8 Applications and Case Studies

In this chapter, we present the results obtained from the implementation and testing of the AISOSVM fault management system on different spacecraft scenarios. A comprehensive evaluation of the system's performance is conducted, focusing on its effectiveness in detecting faults in near real-time. The results are discussed in the context of the system's practical applicability and potential impact on spacecraft operations. Furthermore, we delve into the implications of the findings for the overall research objectives and explore the potential avenues for future work to enhance and expand the AISOSVM fault management system. This chapter aims to demonstrate the system's capabilities in addressing the challenges of fault detection and health management in autonomous spacecraft, validating the theoretical underpinnings and the practical implementation of the AISOSVM framework presented in the preceding chapters. The AISOSVM is experimentally evaluated in simulation using a high-fidelity spacecraft simulation environment; the AISOSVM implementation process and fault detection performance was further validated using flight data collected from a spacecraft hardware testbed.

8.1 The Simulation Environment

To validate the performance of the AISOSVM for fault detection in spacecraft systems, a series of experiments and test scenarios were designed and conducted in simulation. While the AISOSVM has the potential to be applied to any autonomous system, spacecraft applications were chosen for their criticality and the need for accurate, near real-time fault detection capabilities. Spacecraft are highly complex and isolated systems, making fault detection vital to mission success and the safety of assets. The simulations tests aimed to assess the ability of the AISOSVM to identify faults in different subsystems, as well as its capability to adapt and learn from new data in real-time.

The experiments were carried out using a comprehensive spacecraft simulation that encompassed various subsystems, including the reaction wheels and the attitude controller, to replicate a realistic operating environment. The spacecraft simulation served as the founda-

tion for the test scenarios, providing a platform for evaluating the AISOSVM's performance under various conditions. The reaction wheel model, an essential component of the spacecraft's attitude control system, was used to simulate faults in the reaction wheel assembly.

By testing the AISOSVM in these operationally reflective spacecraft scenarios, the proposed methodology's effectiveness and adaptability in addressing the challenges of fault detection could be thoroughly assessed. The spacecraft application serves as a demanding testbed, with stringent requirements for accuracy, timeliness, and robustness, highlighting the value of the AISOSVM in mission-critical environments. These experiments provide valuable insights into the potential applications of the AISOSVM not only in the field of spacecraft health monitoring but also in other autonomous systems where reliable and adaptive fault detection is of paramount importance.

8.1.1 The Simulation Environment

A spacecraft simulation environment was developed to support the design, testing, and evaluation of the fault detection concepts. As shown in Figure 8.1, the simulation was generated as a modular structure within Matlab/Simulink using the Aerospace Toolbox and Aerospace Blockset. These toolboxes provide the general framework for solving the equations of motion of a six-degree-of-freedom (6DOF) spacecraft with dynamics that include J2000 and solar radiation pressure perturbations as presented in Bate et al. [63], Vallado [64]; this environment is portable and flexible for generating system data given different control configurations and flight conditions. The simulation environment includes dynamic models for a generic microsat, the Low-Earth Orbit (LEO) space environment, autonomous Lyapunov-based flight controller, and data visualizations of the system response.

A 6DOF spacecraft simulation environment was used to gather data for fault detection algorithm training and evaluation. The simulation implements the nonlinear equations of motion for a system in Earth orbit with spherical harmonic and solar radiation pressure dynamics [64, 65]. The spacecraft model is defined with physical parameters representative of a small, 150 kg microsatellite. The spacecraft does not have propulsion capabilities for

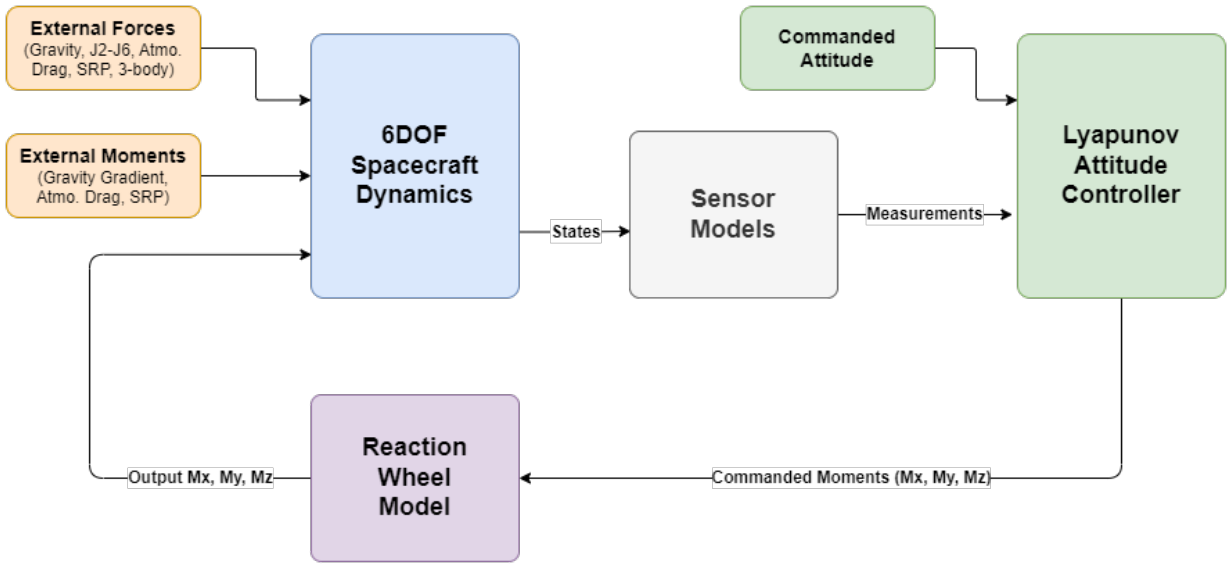


Figure 8.1 Spacecraft simulation block diagram.

translating motion. Each test case using this simulation environment was initialized with spacecraft orbital parameters corresponding to a satellite in a Sun-Synchronous Orbit (SSO). Table 8.1 summarizes these orbital parameters.

Table 8.1 Initialized spacecraft orbital parameters.

Orbital Parameter	
Periapsis altitude, km	500
Eccentricity, e	0.001
RAAN	92°
Inclination	98°
Argument of periapsis	90°

8.1.2 The Reaction Wheel Model

Reaction wheels operate by employing a rotating momentum wheel, driven by an inbuilt brushless DC motor, which interacts with the spacecraft body to either speed up or slow down the wheel, consequently altering the magnitude of the wheel's momentum. These components serve as both momentum transfer and storage devices, delivering reaction torque and conserving angular momentum [66]. Reaction wheels are frequently utilized as the actuators within the Attitude Control Systems (ACS) of small spacecraft.

The simulation reflects the use of four reaction wheels, integrated in a tetrahedral configuration for attitude control given as four commanded output torques, each with three torque components, one for each defined axis of rotation (roll, pitch, and yaw) of the reaction wheel relative to the spacecraft's body frame. Figure 8.2 shows a representation of this reaction wheel configuration. The reaction wheel model was designed and implemented using a similar approach as presented in Gutierrez Martinez [67]. Figure 8.3 provides a block diagram for the ACS model used within the simulation environment. Additionally, Figure 8.4 provides a lower level view of the individual reaction wheel models.

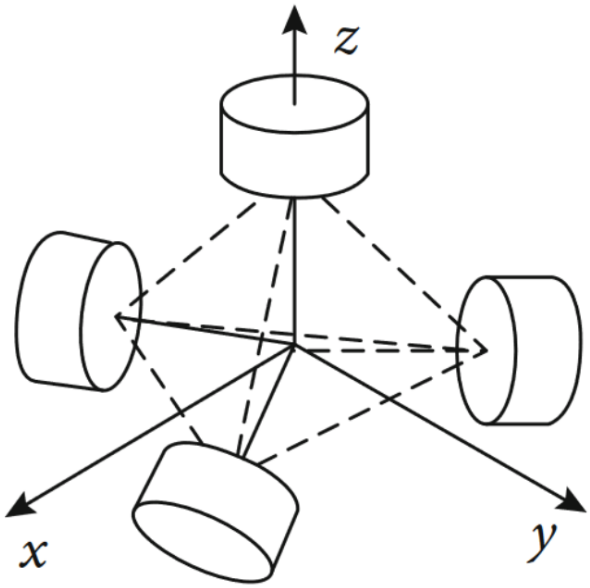


Figure 8.2 Tetrahedral reaction wheel configuration.

These reaction wheels implemented for the simulation environment and test case scenarios were modeled using Sinclair Interplanetary's 1 Nms reaction wheel by Rocket Lab. The hardware specification used for this modeling is provided by Rocket Lab [68] and summarized in Table 8.2. This reaction wheel configuration and maximum control torque reflects commercially available systems for the modeled spacecraft mass class.

8.1.3 Attitude Control Laws

The control torques themselves are generated using control laws derived from the quaternion error kinematics and a Lyapunov candidate function as defined in Equation (8.1) and

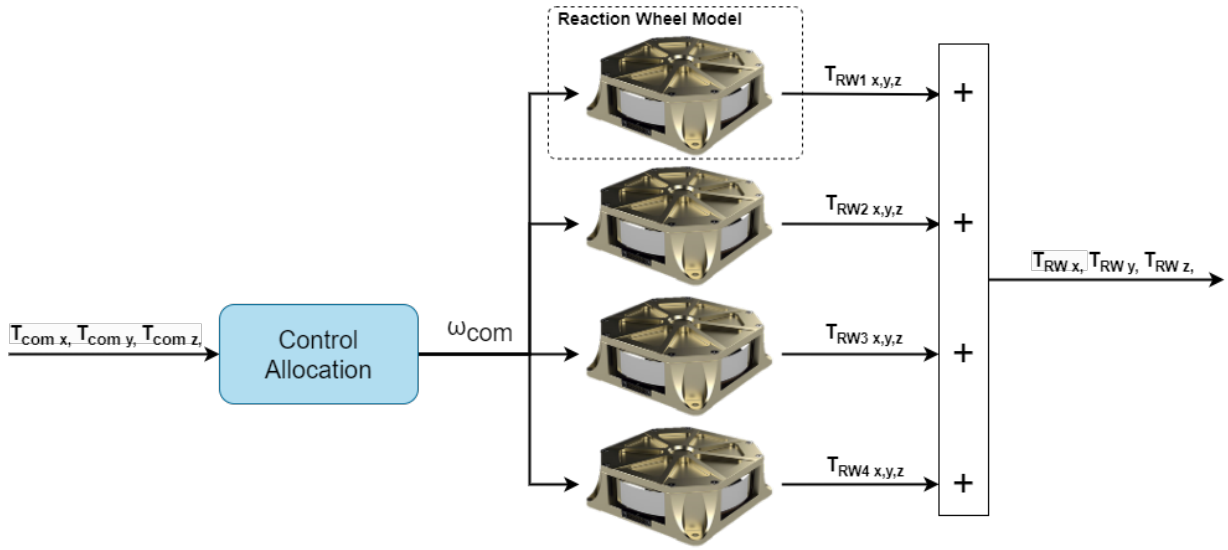


Figure 8.3 Attitude control system model block diagram.

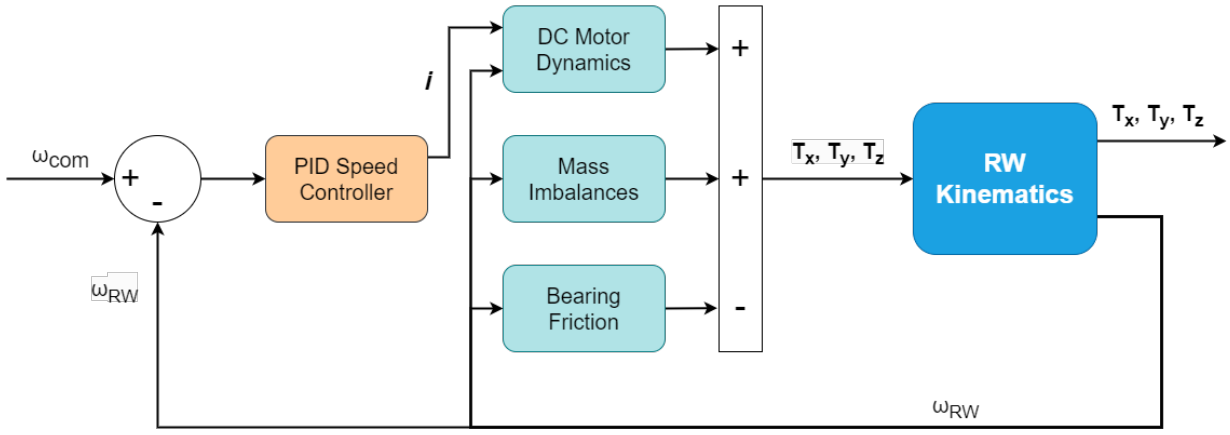


Figure 8.4 Individual reaction wheel model block diagram.

Table 8.2 Reaction wheel model specifications.

Parameter	Reaction Wheel Specification
Momentum	1.0 Nms
Torque	0.1 N-m at 0.8 Nms
Maximum Current	1.53 Amps
Supply Voltage	28 V
Power Requirement	Maximum: 43 W @ 0.6 N-m
Dimension	154 mm x 146 mm x 45 mm
Mass	1.380 kg

Equation (8.2) respectively. These control laws were derived following the process in Schaub and Junkins [69]. The Lyapunov candidate function considered was chosen to generate the torque commands that will drive both the vector and scalar portion of the error quaternion to zero and unity respectively while simultaneously nulling the error angular rates. Lastly, a saturation function is used to ensure only achievable torque commands are generated reflective of the chosen reaction wheel capabilities.

$$\dot{q}_e = \frac{1}{2}\boldsymbol{\Omega}(\boldsymbol{\omega}_e)\boldsymbol{q}_e$$

$$\boldsymbol{\Omega}(\boldsymbol{\omega}) = \begin{bmatrix} -\boldsymbol{\omega}^x & \boldsymbol{\omega} \\ -\boldsymbol{\omega}^T & 0 \end{bmatrix} \tag{8.1}$$

where $\boldsymbol{q}_e$ is the error quaternion and the quaternion is defined as $\boldsymbol{q} = [\boldsymbol{\eta}, \varepsilon]^T$ with $\boldsymbol{\eta} \; \epsilon \; \mathbb{R}^{3x1}$ and $\varepsilon \; \epsilon \; \mathbb{R}^1$. A Lyapunov Candidate is chosen as:

$$V = \frac{1}{2}\boldsymbol{\omega}_e^T \boldsymbol{K}^{-1}\boldsymbol{J}\boldsymbol{\omega} + 2\eta^T\eta + (1 - \varepsilon^2) \tag{8.2}$$

where $\boldsymbol{K} \; \epsilon \; \mathbb{R}^{3x3}$ is the positive-definite matrix of gains and $\boldsymbol{J} \; \epsilon \; \mathbb{R}^{3x3}$ is the inertia tensor matrix. From this Lyapunov candidate function, and following the Lyapunov Direct Method, the resulting control torque, T, is defined by Equation (8.3):

$$T = \boldsymbol{\omega}_B^x \boldsymbol{J}\boldsymbol{\omega}_B - \boldsymbol{J}\boldsymbol{\omega}_e^x \boldsymbol{R}_e\boldsymbol{\omega}_d + \boldsymbol{J}\boldsymbol{R}_e\dot{\boldsymbol{\omega}}_d - 2\varepsilon_e\boldsymbol{K}\boldsymbol{\eta}_e - \boldsymbol{C}\boldsymbol{\omega}_e \tag{8.3}$$

where $\boldsymbol{\omega}_e$ are the error rotation rates, $\boldsymbol{\omega}_d$ are the desired rotation rates, $\dot{\boldsymbol{\omega}}_d$ is the desired angular accelerations, $\boldsymbol{R}_e \, \epsilon \, \mathbb{R}^{3x3}$ is the rotation matrix that maps to the body-fixed reference frame, the superscript x signifies a skew operator, and $\boldsymbol{C} \, \epsilon \, \mathbb{R}^{3x3}$ is a positive-definite matrix. The resulting time-derivative of the candidate function remains a valid function as long as $\boldsymbol{K}^{-1}\boldsymbol{C}$ is positive definite. To ensure this condition, selection of these matrices can implement the following:

$$\begin{aligned} \boldsymbol{K} &= k\boldsymbol{J} \\ \boldsymbol{C} &= c\boldsymbol{J} \end{aligned} \tag{8.4}$$

where k and c are small and positive scalar values. Control torques generated by this control law for an attitude change maneuver are presented in Figure 8.5.

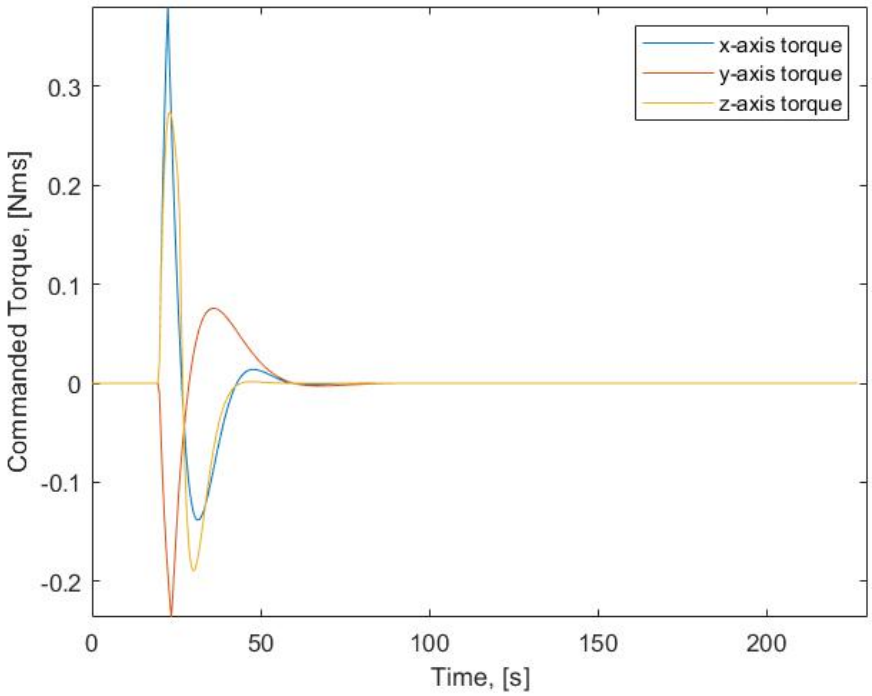


Figure 8.5 Generated control torques from Lyapunov-based controller.

8.2 Simulated Scenarios and Data Acquisition

The spacecraft simulation was used to generate the training data and to gather data representative of a failure scenario. Various test case scenarios were simulated for the generic spacecraft to perform the various analyses and to assist with designing the AISOSVM architecture. For the purpose of assessing performance, the fault detection process focused on the spacecraft's attitude dynamics. The test cases consisted of nominal operating conditions and the other representing anomalous conditions within the attitude control system. The nominal and failure test case scenarios simulated are described as follows:

- **Nominal data acquisition.** The nominal response of the system was generated by executing five hundred attitude commands within the range of $[-15, 15]^\circ$ to track a target orientation. Each simulation allowed sufficient time for the spacecraft to stabilize at the new attitude using the control system previous detailed. The nominal attitude control performance of one such nominal spacecraft simulation is presented in Figure 8.6 and Figure 8.7, which provide the spacecraft attitude tracking information and reaction wheel torques, respectively.

  The simulated nominal test cases served as training data for class 1 of the support vector machine. The five hundred simulations provided the fault detection process with over ninety-six thousand data points, capturing the typical response of the spacecraft dynamics for small to medium attitude changes.

- **Failure test scenario #1.** For AISOSVM verification and validation tests, an anomalous test case generated failure data representative of a torque saturation in one of the reaction wheels within the attitude control system. This failure was injected by using a torque saturation of 50% directly on the ACS model output on one of the three output torques. This simplified the feature space of the spacecraft to just the angular rates (p, q, r) and the attitude.

- **Failure test scenario #2.** For higher fidelity simulation test scenarios, an anoma-

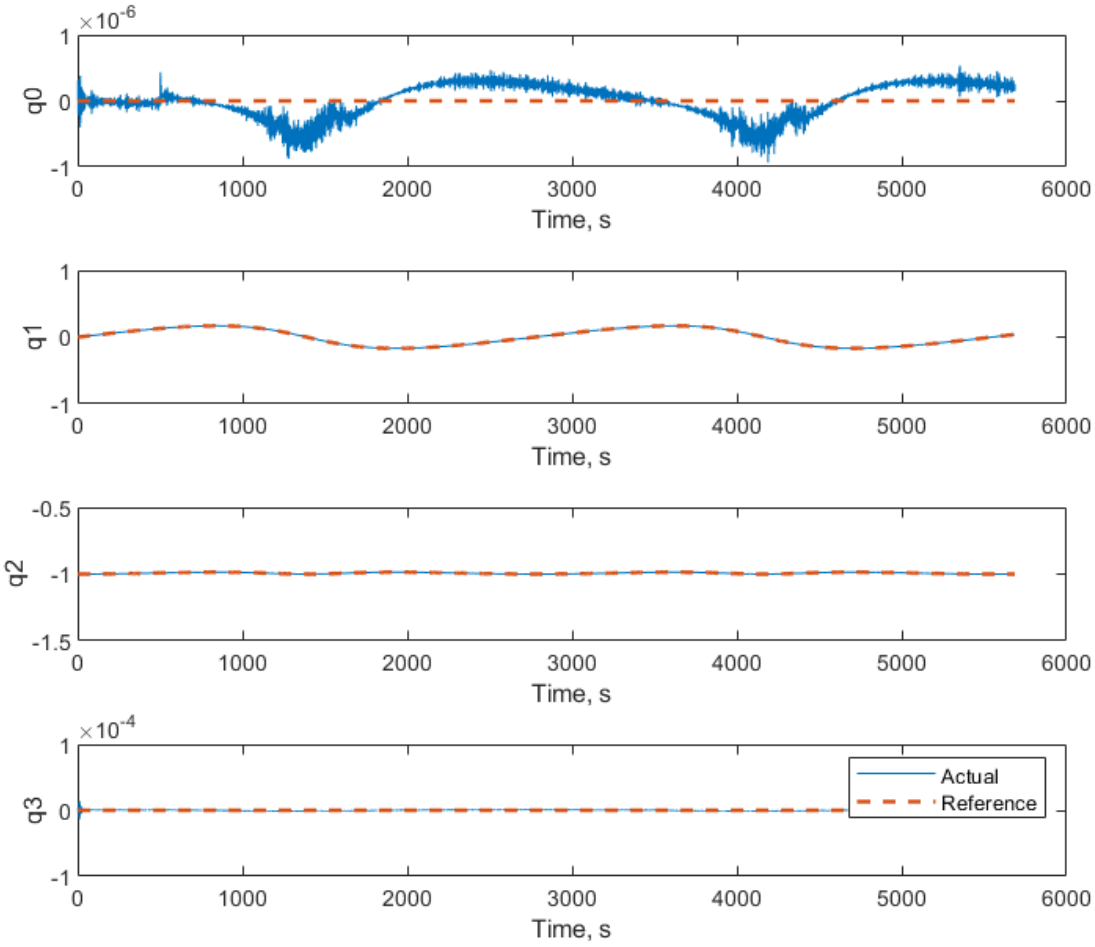


Figure 8.6 Nominal attitude control performance.

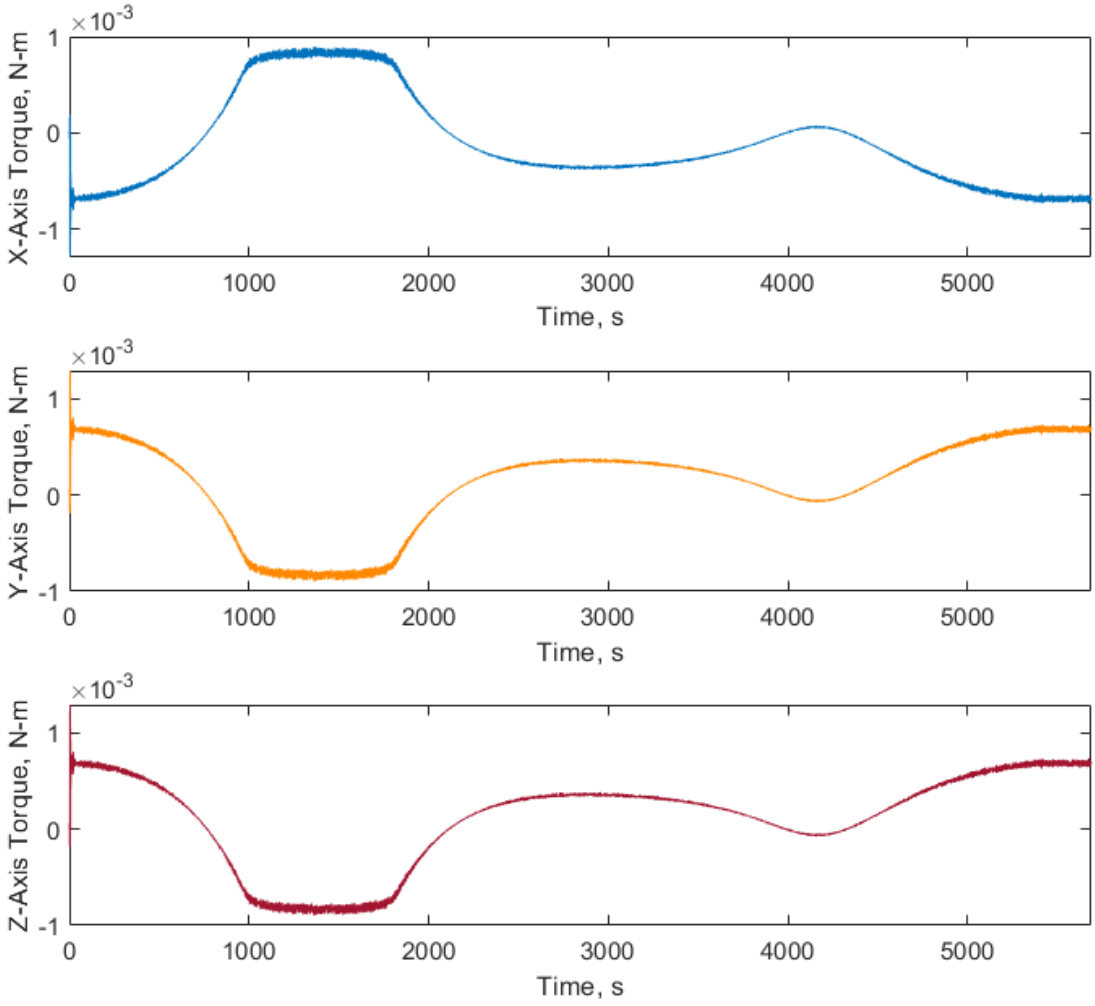


Figure 8.7 Nominal reaction wheel output torque.

lous test cases, representing a failed or saturated reaction wheel scenario, generated a second set of data points that could be compared to the trained model to evaluate fault detection performance. For these test cases, the friction of Reaction Wheel # 2 (one of four reactions wheels) was increased by 50%; this change impacts both the controller stability as well as the power supply requirements requiring both systems to become more active to account for the loss in torque output. This simulates a reaction wheel subsystem that experiences an anomaly in the bearing assembly similar to the failures experienced by the Dawn spacecraft [70]. The failure attitude control performance of one such failure mode simulation is presented in Figure 8.8 and Figure 8.9, which provide the spacecraft attitude tracking information and reaction wheel torques, respectively. As depicted, the system is able to remain stable; however, the control system responds more erratically for commanded torques and the power system must supply additional current to maintain the required torque. A comparison of the current draw is provided in Figure 8.10 demonstrating the difference between the nominal reaction wheel performance and the failure mode.

8.3 Evolution of the Negative Selection Augmentation

During the initial development stages of the AISOSVM architecture, both the fixed radius Negative Selection Algorithm and the Variable Detector algorithm were investigated for the immune system augmentation and generation of synthetic fault data for the model training process. The NSA was considered as the fixed radii detectors could potentially provide a more uniform distribution of detectors surrounding gathered *self* data. Alternatively, the V-Detector algorithm with the varying radius detectors was considered in order to maintain a small support vector machine model as the algorithm is designed to generate as few detectors as possible with the highest feature space coverage.

8.3.1 A Comparative Study

The angular rates, p, q, and r, were selected as the primary features due to their significant potential and sensitivity to attitude control system failures. The feature space can be

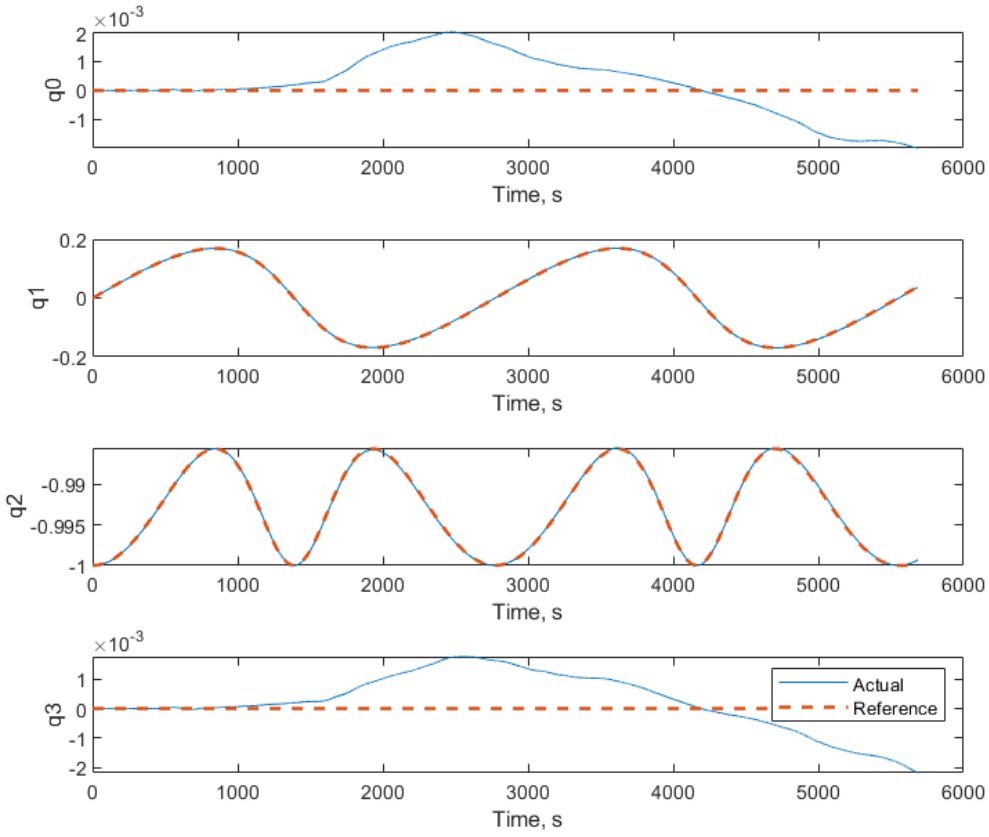


Figure 8.8 Effects of failure on attitude tracking.

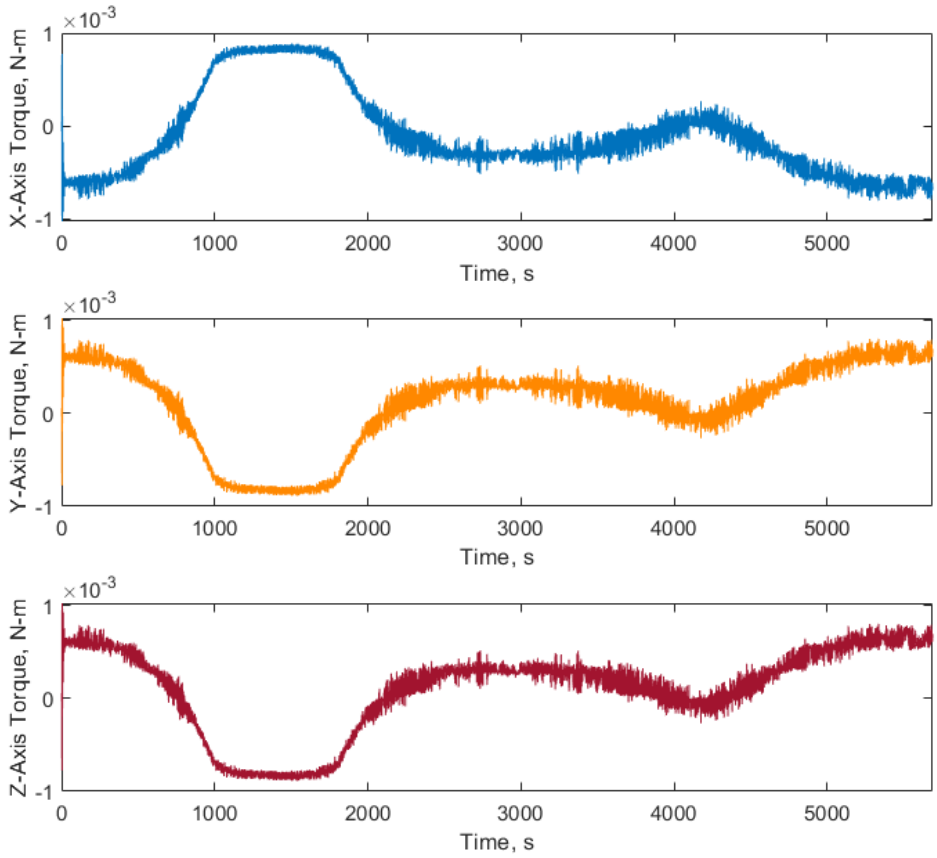


Figure 8.9 Reaction wheel output torques with increased bearing friction.

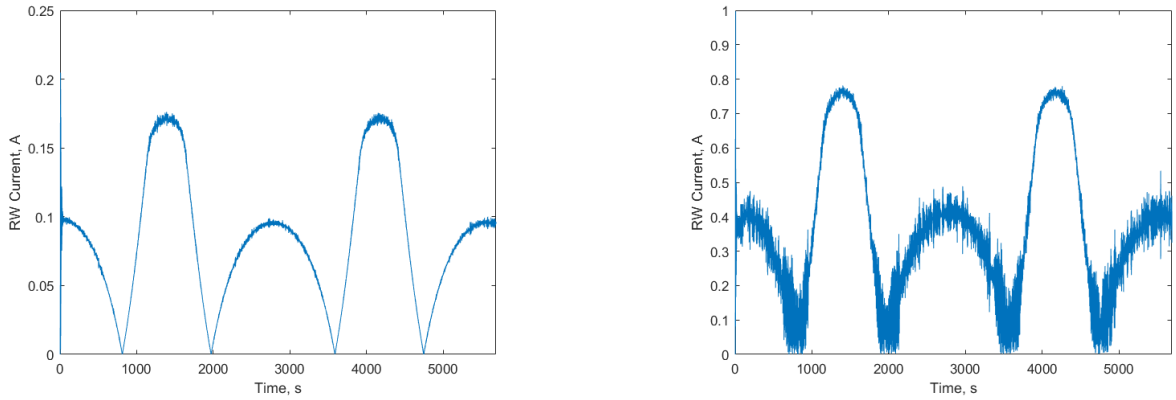


Figure 8.10 (right) The nominal current draw for a spacecraft reaction wheel and (left) the effect of the failure mode on the current draw.

further expanded and combined with other dynamics and system states as needed, as will be demonstrated in Section 8.4.

The AISOSVM training and optimization process follows the method shown in Figure 6.1. The first class of data, class -1 representing the nominal performance of the spacecraft, was obtained from the spacecraft simulation environment. The immune system strategies, CSA and NSA, generated an optimized SVM model for the fault detection applications.

As previously described, once the initial set of nominal spacecraft training data is acquired and normalized, CSA is invoked to generate the AISOSVM model. Within the optimization algorithm, each optimization parameter is assigned a definite range for the antibodies which are encoded into a binary string. The parameters used for the AIS optimized SVM are the SVM hyperparameters C and σ and the *self* and *nonself* cell radii, r_s and r_a respectively, for NSA. The CSA features and values used for optimization are given in Table 6.1.

8.3.2 Fixed Radius NSA Augmentation

The objective of the optimization is to generate a variable *self-nonself* representation while maintaining high SVM classification performance. The Clonal Selection Algorithm used four fitness parameters for optimization to generate the AISOSVM model used for fault detection. One parameter is the model's classification accuracy. This accuracy revolves around the number of *true positives*, *true negatives*, *false positives*, and *false negatives* for the classification of test data. This test data was selected as a portion of the nominal data and the generated antibodies from NSA. Other fitness parameters are the number of support vectors, the number of antibodies, and the ratio of radii for the *self* cells and antibodies. Using this process, the SVM and NSA parameters are dynamically optimized to generate an AISOSVM model that will then perform the online fault detection. The SVM training process used the sequential minimal optimization process for each CSA population member n_b. The two SVM hyperparameters and NSA antibody radii are used with the nominal data obtained from the spacecraft simulation to train the AISOSVM. The AISOSVM process calculates the value of the objective functions for each population member $F(n_b)$. The

affinity of each n_b is obtained with the objectives of obtaining the best C, σ, r_s, and r_a combination. The multi-objective affinity is defined in Equation (6.1) and the final objective function is given in Equation (6.2).

The results from the best performing AISOSVM model generation using the fixed radius detectors are presented in Figure 8.11. The figures show the output of the negative selection algorithm on the left and the final AISOSVM model on the right with the support vectors emphasized and the decision boundary between nominal (blue) and abnormal (red) classes. In the NSA plot, the red circles depict the generated antibodies to cover the abnormal subregion while the blue circles are the normalized *self* training data obtained from the spacecraft simulation. The features are comprised of the angular rates with chosen feature spaces being *p vs. r*, *p vs. q*, and *q vs. r*. Together, these three feature spaces are used to classify incoming data as either nominal or abnormal. The optimized parameters for each feature space are provided in Table 8.3.

Table 8.3 Optimized AISOSVM parameters for the fixed radius NSA fault detection feature spaces.

Parameter	Feature Space		
	p vs. q	p vs. r	q vs. r
C	15.24	229.72	204.40
σ	0.004	0.005	0.005
r_s	0.010	0.010	0.010
r_a	0.058	0.014	0.020

8.3.3 V-Detector Augmentation

The AISOSVM with V-Detector augmentation, is expected to generate a finer *self-nonself* boundary by varying the size of each individual detector allowing the algorithm to achieve greater detector coverage than the fixed radius method. This fault detection architecture uses CSA to generate an optimized SVM model for the health monitoring application. To generate the AISOSVM + VD system, the same CSA optimization parameters are maintained, as shown in Table 6.1. CSA is used to optimize the SVM hyperparameters C and σ, and

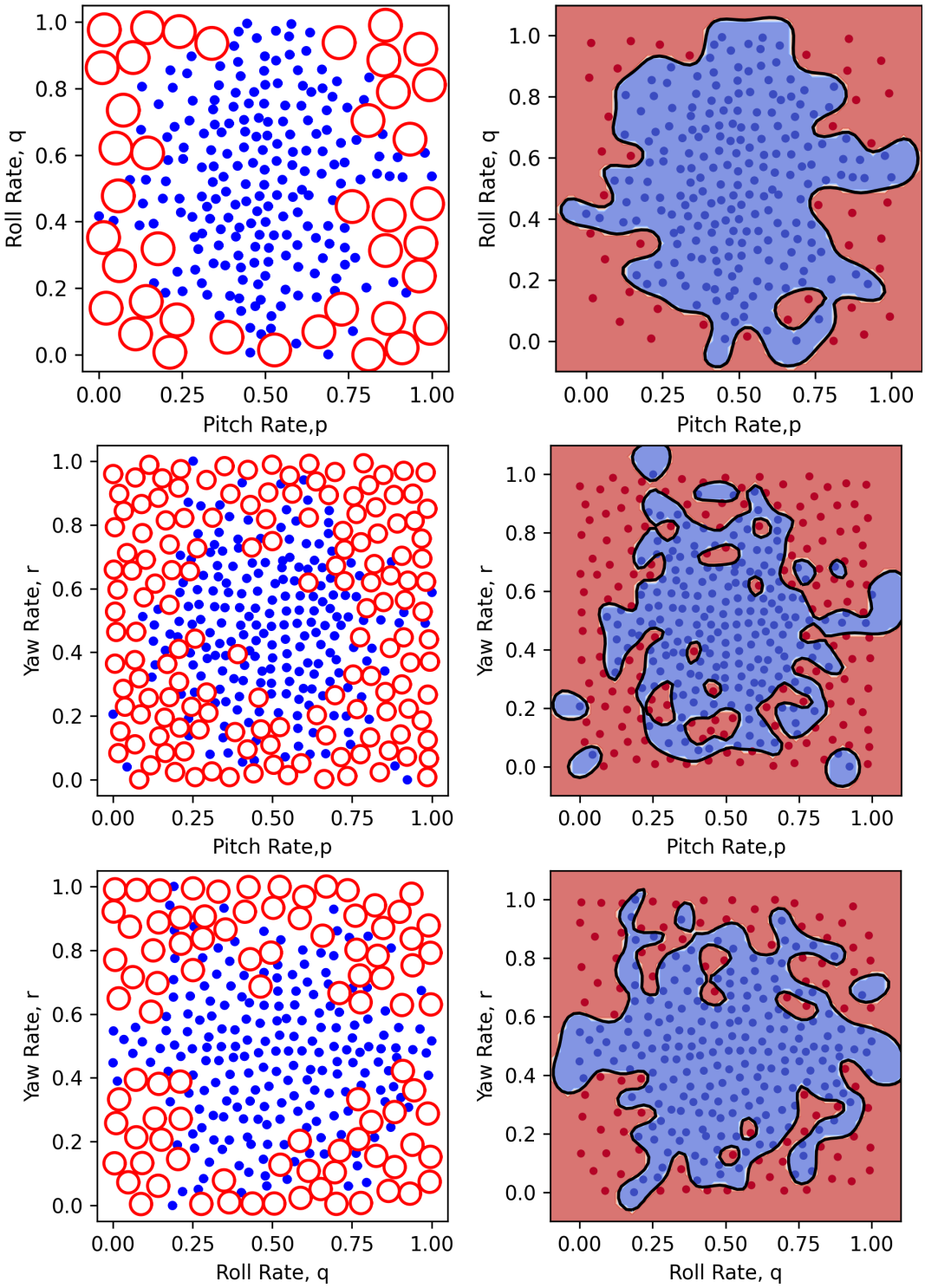


Figure 8.11 AISOSVM with the fixed radius NSA *self-nonself* generation (left column) and corresponding trained SVM with overlayed data (right column) for the angular velocity feature set.

the *self* and *nonself* cell radii, r_s and r_a, respectively, for the V-Detector process. The V-Detector method generates the variable sized antibodies around the *self* data points which are given a fixed radius as part of CSA. The r_a in the V-Detector algorithm corresponds to a minimum radius that can be given to a generated antibody. Within the CSA process, V-Detector is still able to change the radii of each individual antibody. Additionally, because the V-Detector is wrapped into the CSA optimization process and coupled with the Support Vector Machine, a lower generation threshold can be used since the support vector machine fault detection performance does not require strict coverage like the standalone V-Detector strategy due to its ability to generalize with few data points.

The results from the best performing AISOSVM + VD model generation are presented in Figure 8.12. The figures show the output of the V-Detector on the left and the final AISOSVM + VD model on the right with the support vectors emphasized and the decision boundary between nominal (blue) and abnormal (red) classes. The Variable Detector plot shows that the generated antibodies are generally able to cover more of the *nonself* space in comparison to the fixed radius NSA. The following figures show the chosen features used for spacecraft fault detection; the features are comprised of the angular rates with chosen feature spaces being *p vs. r*, *p vs. q*, and *q vs. r*. The optimized parameters for each feature space are provided in Table 8.4.

Table 8.4 Optimized AISOSVM + VD parameters for the fault detection feature spaces.

Parameter	Feature Space		
	p vs. q	p vs. r	q vs. r
C	208.02	145.35	193.04
σ	0.005	0.003	0.003
r_s	0.010	0.010	0.010
$r_{a_{min}}$	0.011	0.012	0.012

8.3.4 Fixed Radius NSA Augmentation Performance

The spacecraft simulation environment generated the nominal and failure data to evaluate the capabilities of the AISOSVM for fault detection. The AISOSVM was trained using

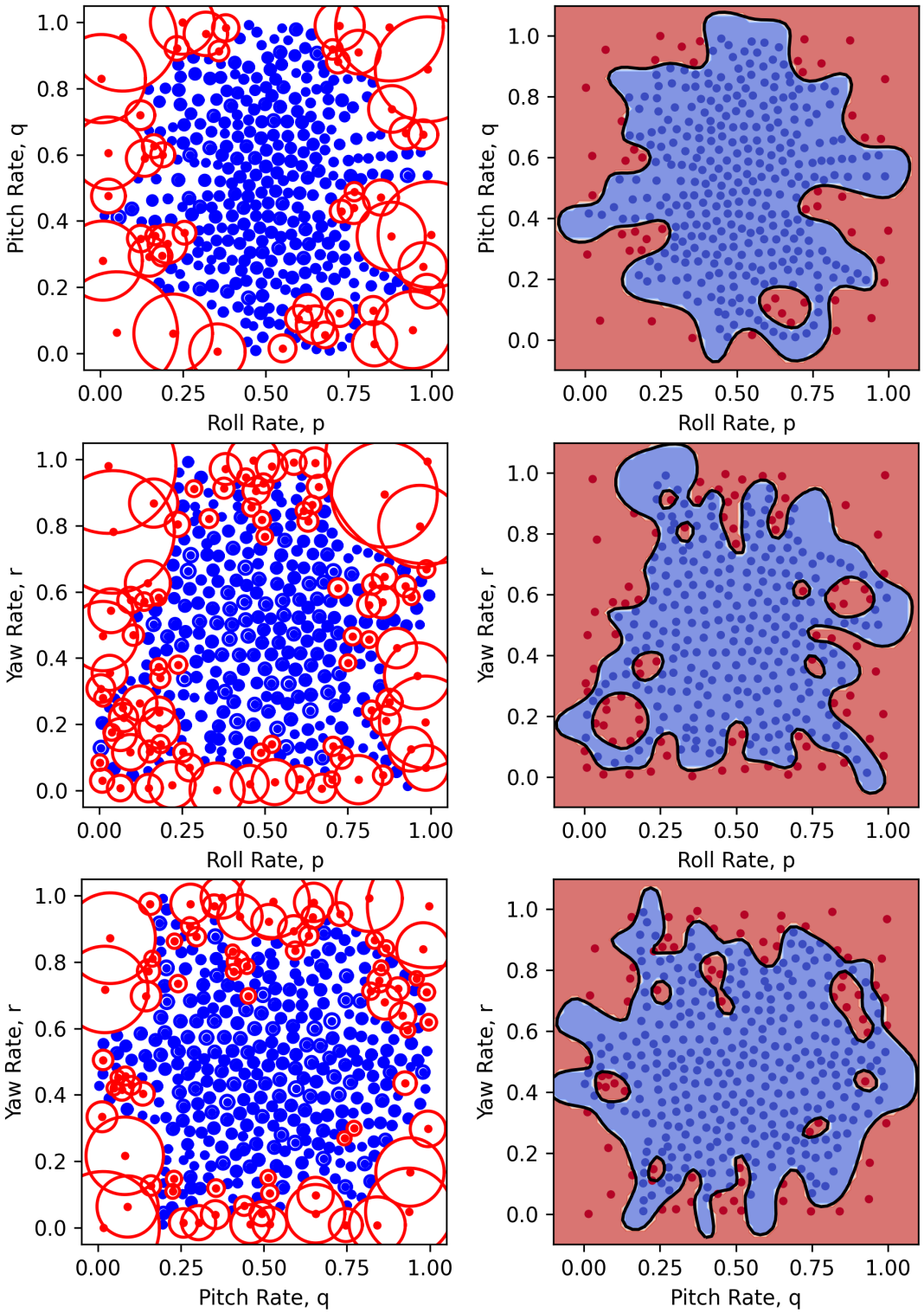


Figure 8.12 AISOSVM with the V-Detector *self-nonself* generation (left column) and corresponding trained SVM with overlayed data (right column) for the angular velocity feature set.

this simulation data to generate the model for fault detection offline. Once optimized, the anomalous data generated by Failure Scenario #1 was injected into the AISOSVM in an online scenario. The failure data for this demonstration was generated using the spacecraft simulation and saturating the output torque of the ACS by 50%.

For both nominal and anomalous test scenarios, the AISOSVM classified the data providing fault detection capabilities when the anomalous data exceeded the pre-trained boundaries that enveloped the nominal performance for the roll, pitch, and yaw control systems. The AISOSVM was tested with using the same model parameters for each feature space as provided in Table 8.3. Figure 8.13 shows the AISOSVM models with the blue and red regions highlighting the *self* and *nonself* subspaces, respectively, for each feature space. The injected anomalous data from a single failure simulation is overlayed to show the general detection process for the health monitoring system. For each feature space for each ADCS failure scenario, the AISOSVM flagged the failure occurrence when the performance of the spacecraft went outside of the *self* boundary. Each feature spaces were able to recognize the anomaly within the initial seconds of the commanded attitude change.

8.3.5 V-Detector Augmentation Performance

The AISOSVM + VD was trained using the same process and the AISOSVM and V-Detector systems involving offline training and online detection. This architecture generated a hybrid model with optimized detectors and support vector machine parameters. To evaluate the affects of the hybrid architecture, the anomalous data generated by the simulation was injected into the AISOSVM + VD. Again, this failure data for this demonstration was generated using the spacecraft simulation and Failure Scenario #1 which saturated the output torque from the ACS by 50%.

Similar to the AISOSVM with the fixed radius detectors, the AISOSVM + VD classified the nominal data providing fault detection capabilities when the anomalous data exceeded the pre-trained boundaries that enveloped the nominal performance for the roll, pitch, and yaw control systems. This health monitoring system used the same optimization parameters

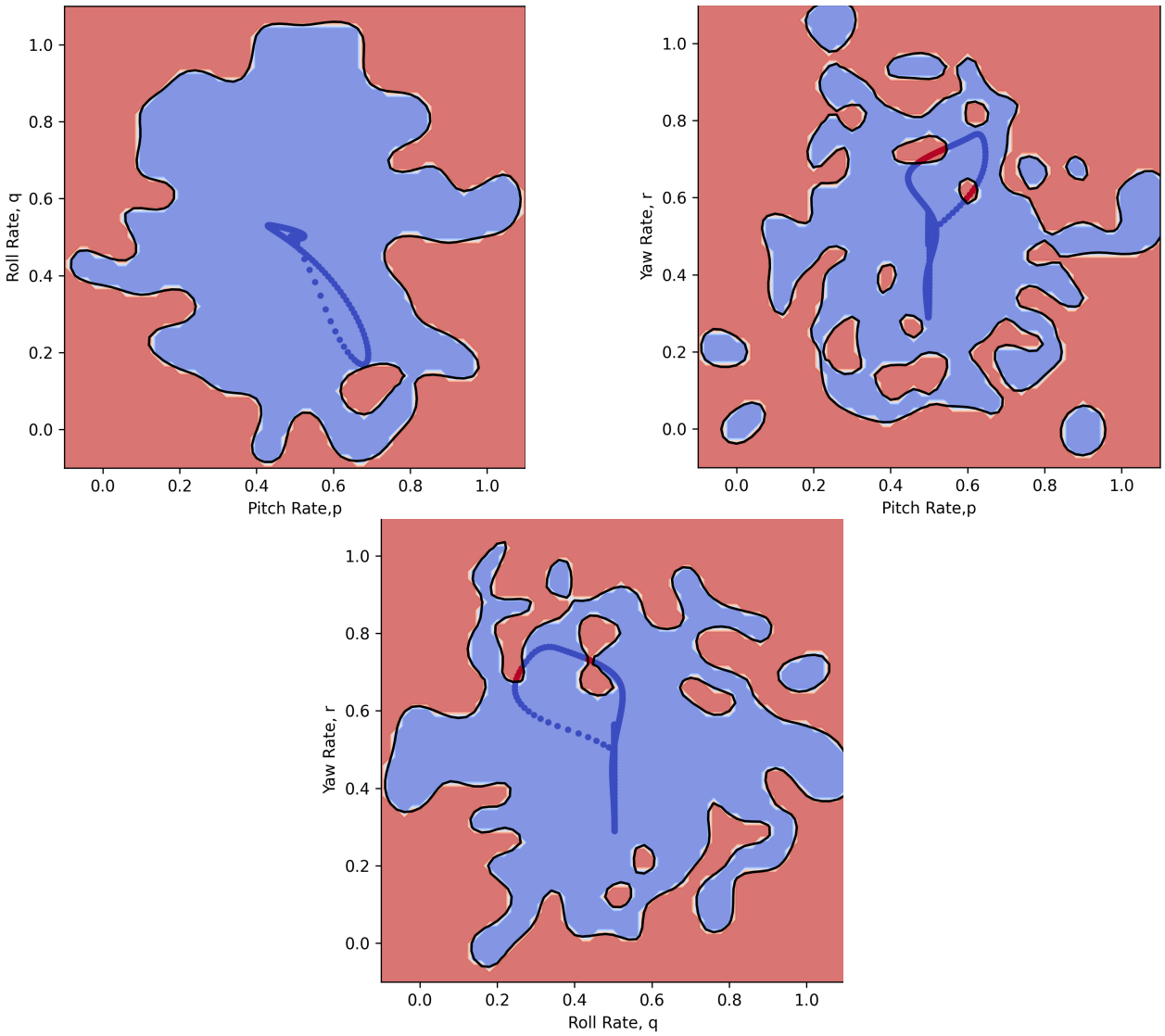


Figure 8.13 The fixed radius NSA AISOSVM model with failure data overlay.

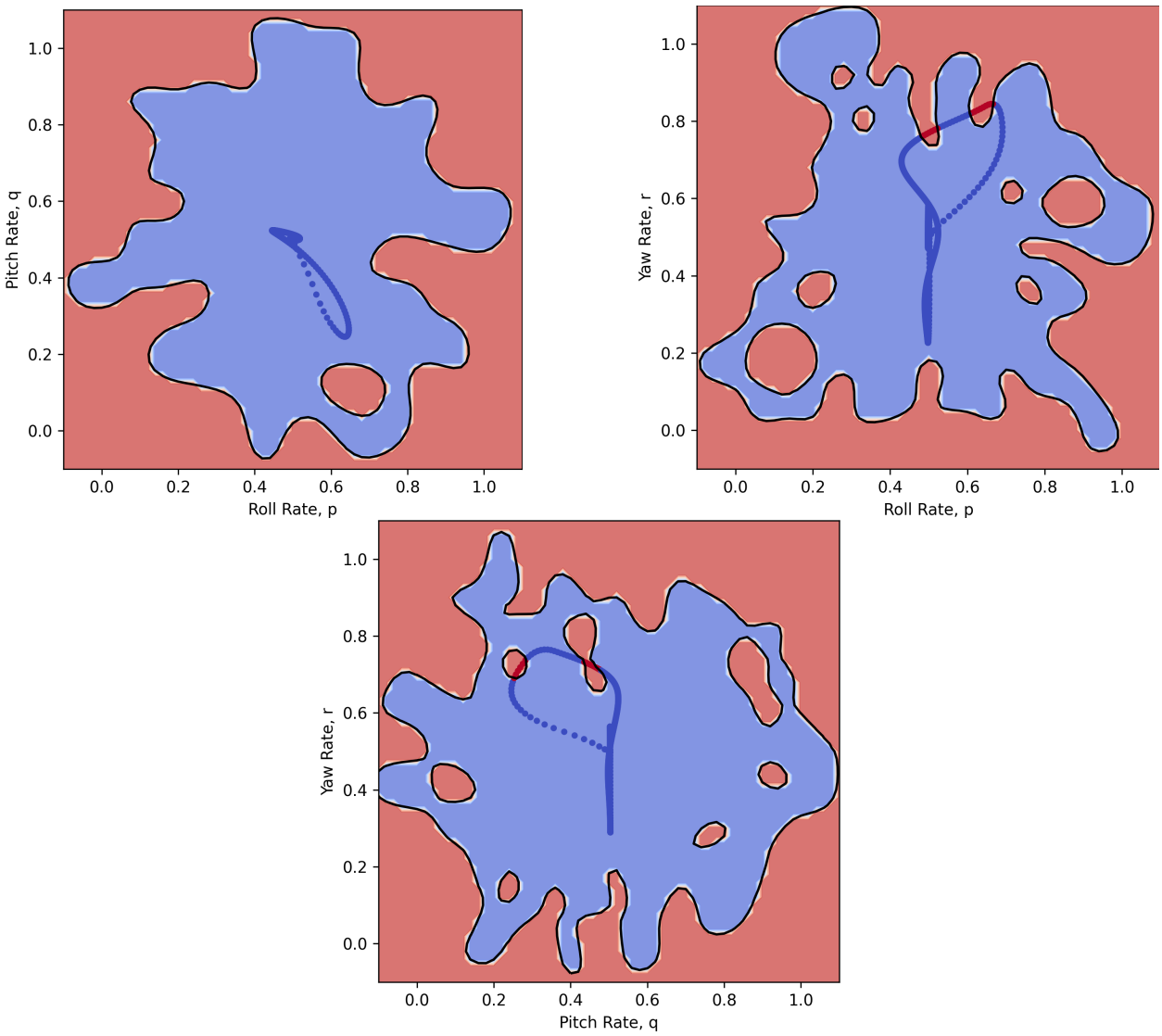


Figure 8.14 The AISOSVM + VD model with failure data overlay.

for each feature space as provided in Table 8.4. Figure 8.14 shows the AISOSVM + VD model with the injected anomalous data from a single failure simulation overlayed to show the general detection process for the health monitoring system.

8.3.6 Design Selection of V-Detector Algorithm

The results of the fault detection systems after following the offline training and online detection phases show that the immune system augmented machine learning strategies are capable of accurate fault detection. The optimization process performed by the Clonal Selection Algorithm allows the various performance tuning parameters to be adjusted to

account for user defined priorities such as model size and accuracy.

It is important to note, although not directly compared, a standalone machine learning classifier would be inadequate in comparison to the immune system augmented approaches. As these are data-driven methods, two classes of data are needed in order to train the classifier for nominal and abnormal dynamics. Without the immune system augmentation, this would require significant simulation time in order to model and generate data for failure modes. Additionally, since most systems have numerous, near-infinite failure behaviors, it is difficult to verify and validate that the simulated failure conditions adequately capture the failure dynamics that would be present in an operational scenario.

Table 8.5 summarizes and compares the average performance of the two immune system augmented approaches across the three feature spaces. Although both methods performed fairly well with near zero false alarms, the V-Detector immune system augmented machine learning algorithm performed better overall with higher detection rates, and even lower false alarms, at the expense of a slightly slower training time. The V-Detector was able to mitigate the issues that can occur from overfitting and this comparative study verified the originally perceived benefits of the V-Detector augmentation. The issue of overfitting can start be seen in the p-r and q-r feature spaces as discontinuities in the *self-nonself* regions become more apparent and common. The detection rate of both algorithm implementations is low due to the nature of the tested failure. The output torque of the spacecraft model, along a single body axis, was limited by 50%. Event though this is clearly the beginning of a more catastrophic failure, the spacecraft was still able to achieve the commanded attitude with only a small deviation from the norm and then was capable of settling at the commanded attitude. However, despite the platform remain stable with the injected failure, the AISOSVM models were able to successfully detect and flag the difference in dynamics.

Surprisingly, the V-Detector was also able to provide a higher granularity and resolution in detector generation in comparison to the fixed radius NSA. Post analysis suggests that the ability for the V-Detector to reduce the size of generated detectors where permitted

allows a very high coverage around and within the *self* region where the NSA is limited to all the same size for all detectors. This means that the resolution and coverage metric within the CSA optimization process is conflicted with the desire for a smaller model, which is weighted higher. The "Model Size" metric in Table 8.5 refers to the number of data points that create the model for online detection. This metric correlates to processing requirements for online detection. Based on the results, the AISOSVM + VD approach performed better in comparison to the AISOSVM with the fixed radius NSA at the expense of additional training time. All models were trained and simulations run on Windows 10 on a Lenovo ThinkPad X1 Extreme laptop with an Intel(R) Core(TM) i7-8750 CPU at 2.20 GHz.

Table 8.5 Comparison of average model performances and characteristics.

	AISOSVM	AISOSVM + VD
Detection Rate	9.17%	11.83%
False Discovery Rate	0.30 %	0.06%
Model Size	212	162
Training Time, [s]	2216	2663

8.4 Spacecraft Attitude Control System Simulation Results

In this section, the proposed AISOSVM fault detection approach is applied to a higher fidelity model of a spacecraft attitude control system with individual reaction wheels. Spacecraft attitude control systems, and their reaction wheels, play a critical role in maintaining the desired orientation of a spacecraft, ensuring the proper functioning of its onboard instruments, and enabling accurate navigation. The inherent complexities and uncertainties in the spacecraft's environment, coupled with the limited computational resources available on-board, make it essential to employ efficient and adaptive fault detection and control techniques.

The AISOSVM model was trained using the nominal test case data. Features were selected that best represent the measureable dynamics of the system and that are deemed or expected to be sufficiently sensitive to the nominal and abnormal operating conditions. The ten primary features used within this application are shown in Table 8.6. The main

features of interest correspond to the Reaction Wheel #2 states. It is this reaction wheel that is failed according to the Failure Scenario #2 described previously. It is important to note that the performance of any data-driven fault management system is only as good as the coverage of the features' historical data to the nominal and failure conditions as well as the sensitivity of those features to the failure modes the fault management system designer wishes to detect.

Table 8.6 Selected features and states for the spacecraft and attitude control system.

	Feature	Desription
1	ω_{RW_2}	Reaction Wheel #2 Speed
2	T_{RW_2}	Reaction Wheel #2 output torque
3	i_{RW_2}	Reaction Wheel #2 power bus draw
4	$q_{0_{error}}$	Scalar quaternion error
5	$q_{1_{error}}$	q_1 quaternion error
6	$q_{2_{error}}$	q_2 quaternion error
7	$q_{3_{error}}$	q_3 quaternion error
8	p	Spacecraft roll rate
9	q	Spacecraft pitch rate
10	r	Spacecraft yaw rate

The AISOSVM was trained using the framework outlined in Figure 6.1. This process generated binary classifiers for fault detection based on the nominal data obtained through simulation data acquisition and the generated antibodies produced by the V-Detector algorithm. For fault detection, the AISOSVM was constructed using a One-vs-One configuration. This implies that each feature is combined with every other feature to produce a feature space for fault detection. In this configuration, the number of features required for full maximum possible coverage is provided by

$$N_{fs} = \frac{N(N-1)}{2} \tag{8.5}$$

where N_{fs} is the total possible number of feature space projections and N is the number of states or features. For this application, a total of 45 possible projections could be considered. However, based on inspection, this number of projections was down selected to 12 to remove

insensitive pairs or projections that would not directly provide meaning data for the given failure mode being tested, namely a RW wheel output torque failure caused by increase wheel friction.

Once the AISOSVM models were trained, they were back-tested and analyzed through Leave-One-Out estimation to ensure model robustness and accuracy. After training, the selected feature spaces were validated using simulation data from a nominal test scenario. This validation test data was not used within the original AISOSVM training process. Figure 8.15 through Figure 8.18 shows the validation test results of several of the more sensitive and relevant feature spaces for the spacecraft ACS and reaction wheel systems. The AISOSVM models correctly classified the validation data as nominal will nearly 100% accuracy for each model. Only the $q_{3_{error}}$ vs i_{RW_2} feature space contained any false alarms. Table 8.7 provides the individual AISOSVM model performance metrics. It is important to note that the provided LOO estimation is obtained from the AISOSVM training data, not this validation data, to provide insight into the model's robustness to unseen data and a probability of incorrectly classifying new data.

Table 8.7 AISOSVM model performance metrics for the ACS validation test.

Feature Space	AUC Score	FPR	LOO Estimation
ω_{RW_2} vs. T_{RW_2}	1.0	0.0%	0.0235
ω_{RW_2} vs. i_{RW_2}	1.0	0.0%	0.0498
ω_{RW_2} vs. p	1.0	0.0%	0.0379
ω_{RW_2} vs. q	1.0	0.0%	0.0556
ω_{RW_2} vs. r	1.0	0.0%	0.0376
T_{RW_2} vs. i_{RW_2}	1.0	0.0%	0.1094
i_{RW_2} vs. $q_0 Error$	1.0	0.0%	0.0000
i_{RW_2} vs. $q_1 Error$	1.0	0.0%	0.0355
i_{RW_2} vs. $q_3 Error$	0.998	0.132%	0.0777
i_{RW_2} vs. p	1.0	0.0%	0.0610
i_{RW_2} vs. q	1.0	0.0%	0.0716
i_{RW_2} vs. r	1.0	0.0%	0.0764

After successful model validation, the same AISOSVM models and feature spaces were

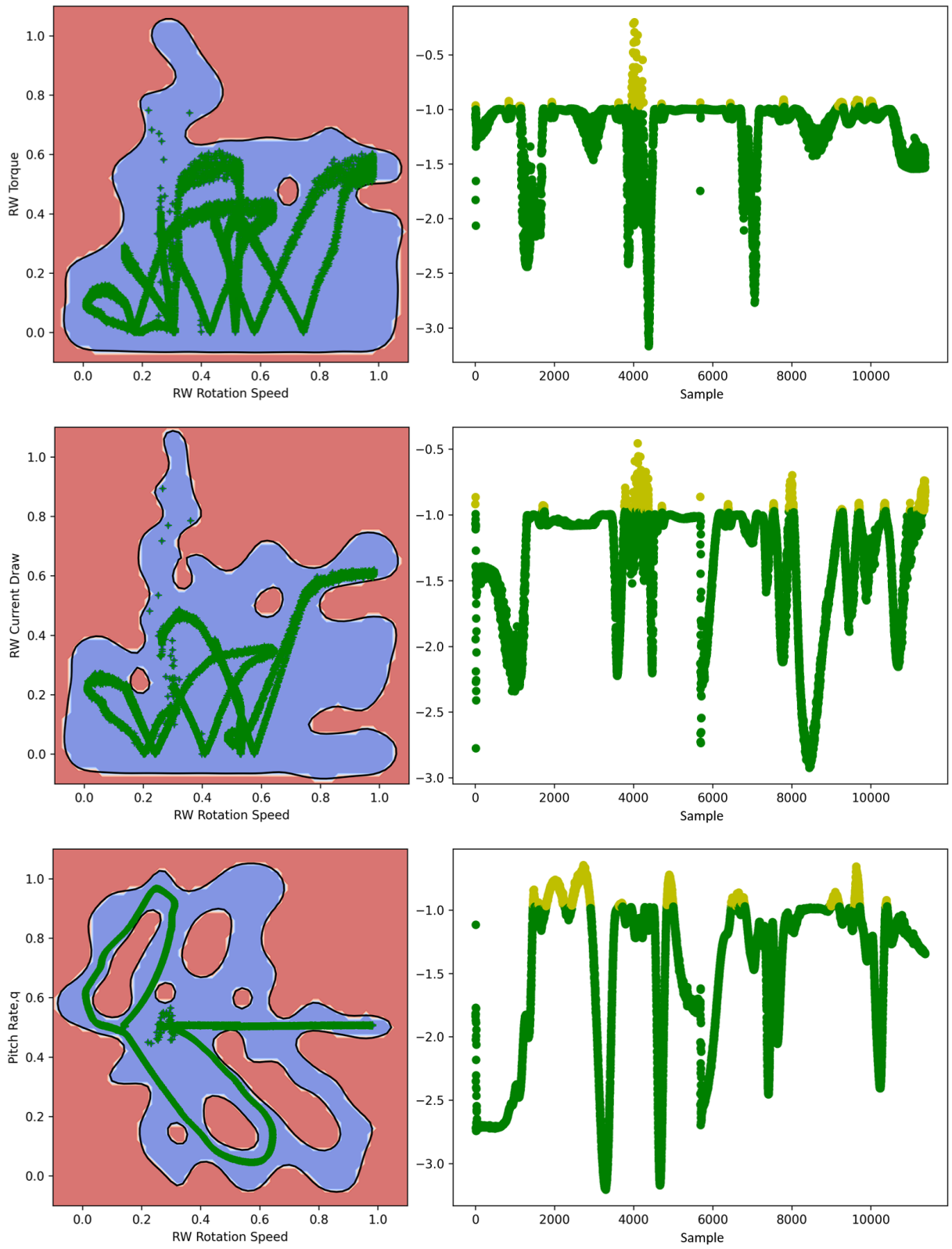


Figure 8.15 Trained AISOSVM models with nominal validation data overlay in green (left) and fault trend outputs (right).

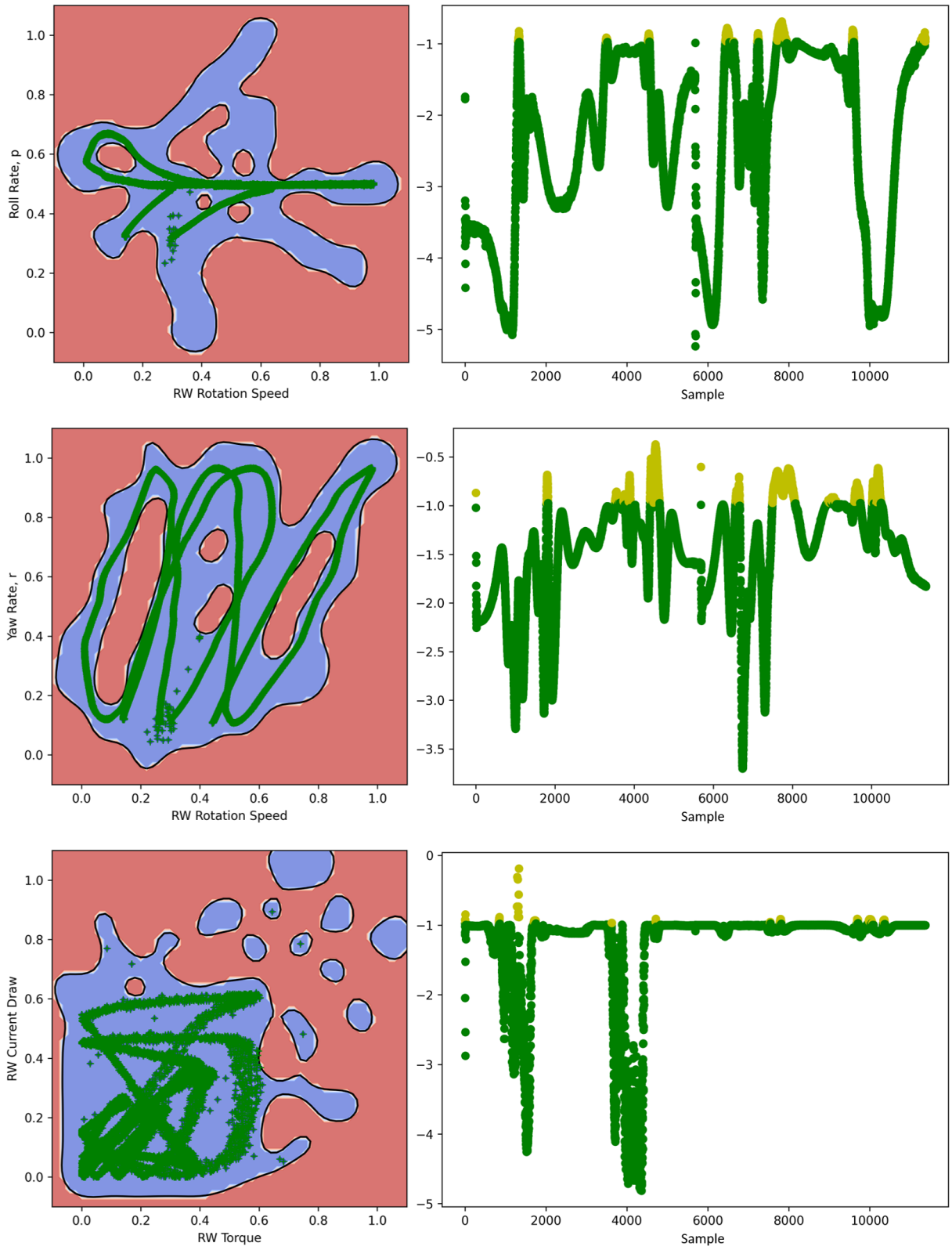


Figure 8.16 Trained AISOSVM models with nominal validation data overlay in green (left) and fault trend outputs (right).

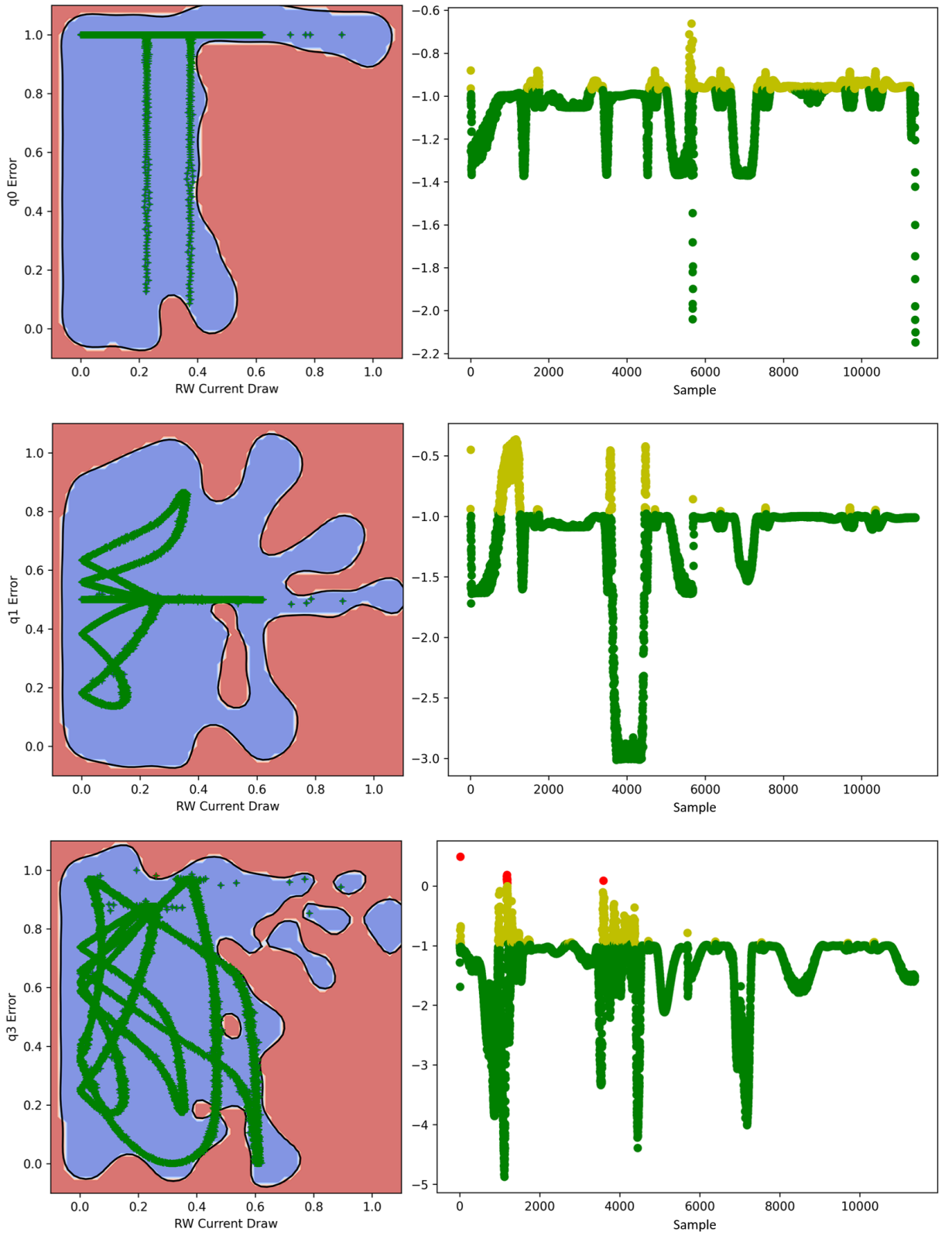


Figure 8.17 Trained AISOSVM models with nominal validation data overlay in green (left) and fault trend outputs (right).

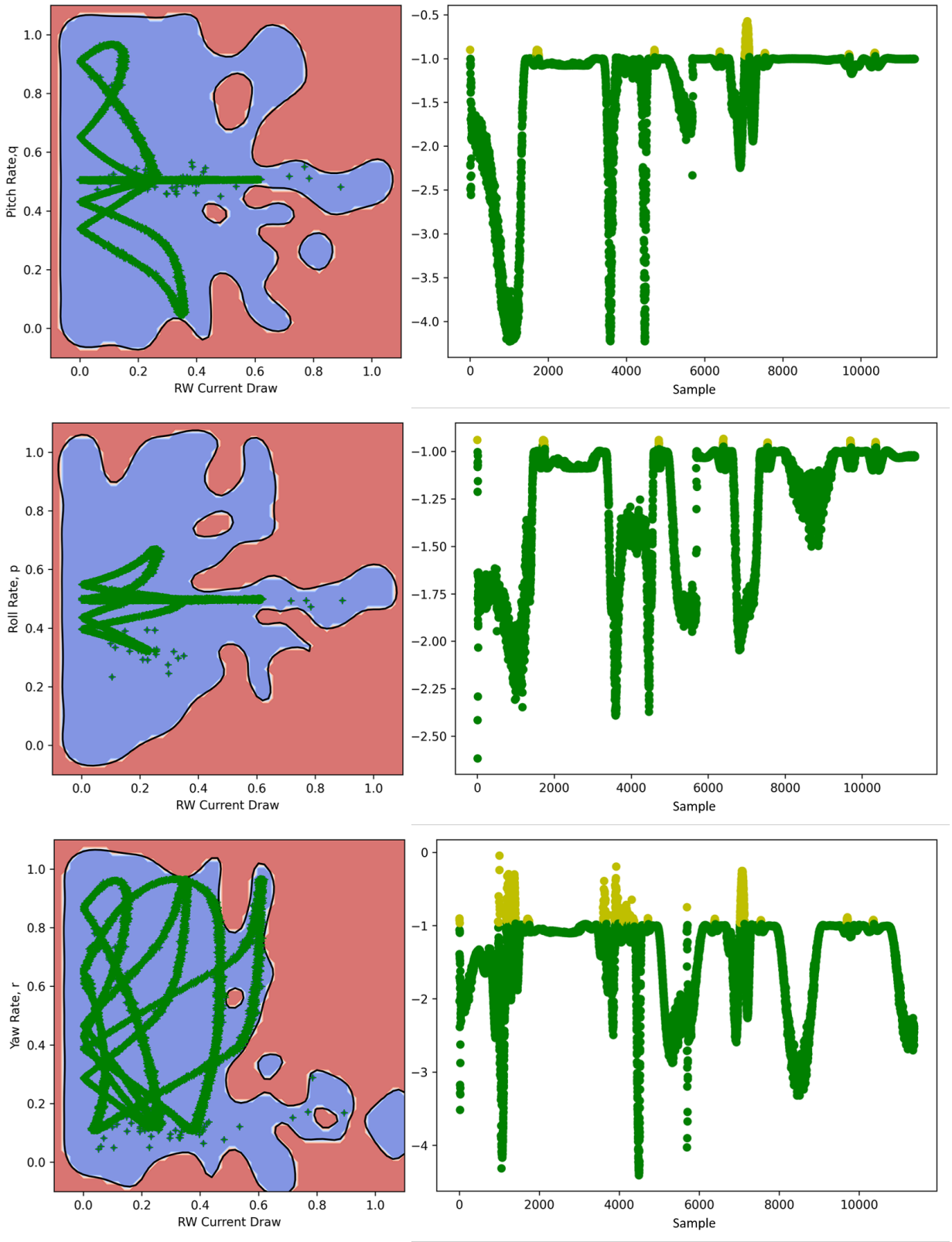


Figure 8.18 Trained AISOSVM models with nominal validation data overlay in green (left) and fault trend outputs (right).

used to classify the reaction wheel failure as described in the failure test scenario #2. The most sensitive feature projects revolved around the relationships between the attitude dynamics and the reaction wheel speed. Figure 8.19 through Figure 8.22 provides the classification results and fault trend analyses for each of the primary selected feature space projections. There was a clear emergent behavior in the spacecraft dynamics after the reaction wheel friction was increased. For the commanded attitude change, the spacecraft was not able to achieve the nominal rotation rates due to the slower reaction wheel rotation and reduced torque output. Additionally, for the i_{RW_2} vs. ω_{RW_2} projection, the reduced wheel speed caused by the increased bearing friction resulted in a shift in the required current draw to achieve the desired rotation speed. The results of the experiments showed that the proposed AISOSVM approach was successful in detecting faults in the spacecraft attitude control system.

The injected fault of increased bearing friction provided a noticeable yet fairly benign failure mode. The spacecraft with its redundant reaction wheels and robust control system was able to maintain stability and achieve the commanded orientation. The fault detection system was still able to detect the abnormal dynamics. Additionally, the fault trend analysis for many of the feature space sets reflected these dynamics with large positively identified data samples occurring at both slew maneuver actuation peaks where the required reaction wheel torque peaked. Table 8.8 provides a summary of the AISOSVM failure data performance metrics. Overall, the AISOSVM approach maintained its performance in terms of fault detection and false alarm rate throughout the incremental learning process. The application to a spacecraft attitude control system demonstrates its potential for efficient and adaptive fault detection and control. The adaptability and computational efficiency of the approach make it a promising solution for spacecraft attitude control systems and other complex, resource-constrained environments.

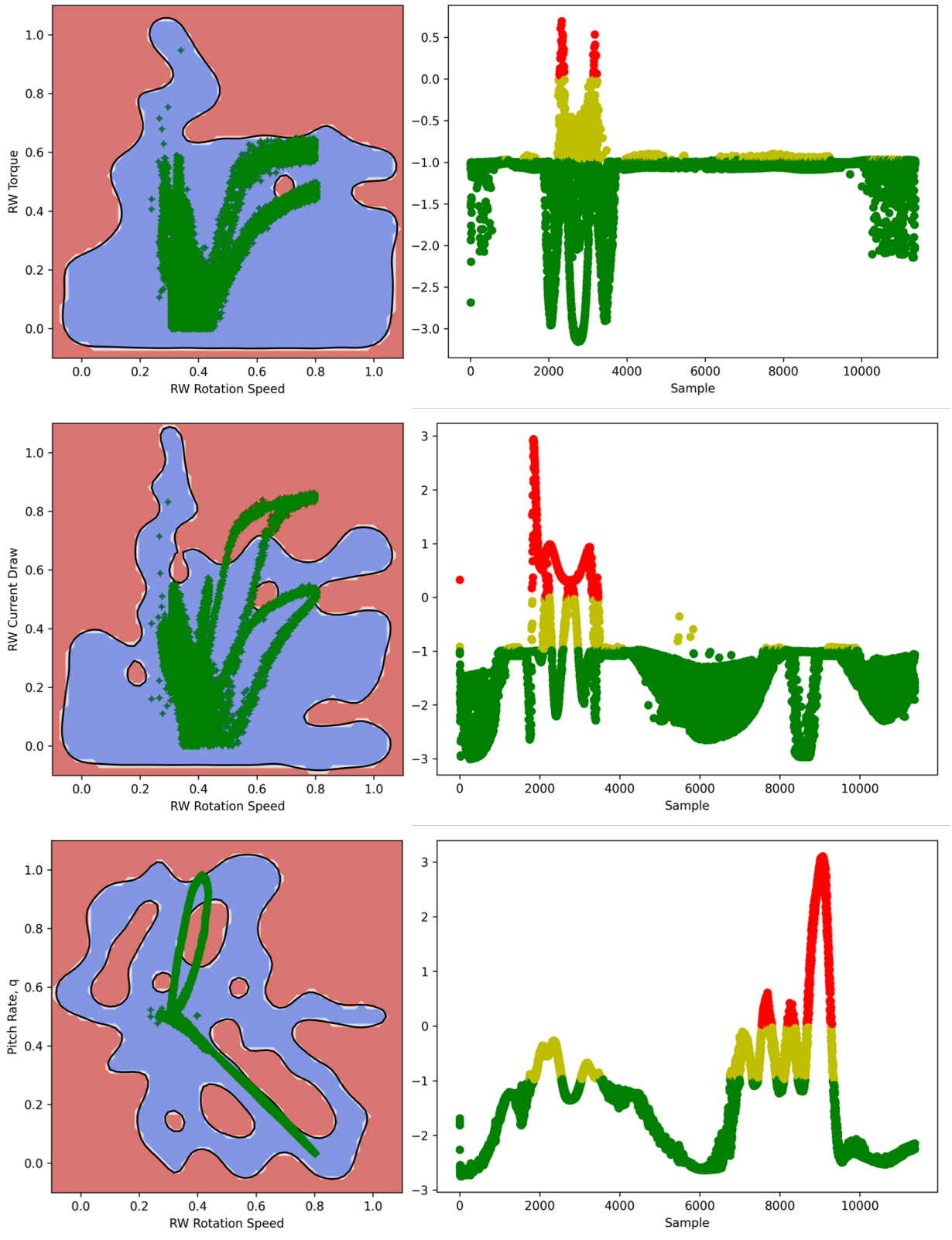


Figure 8.19 Trained AISOSVM models with failure data overlay in green (left) and fault trend outputs (right).

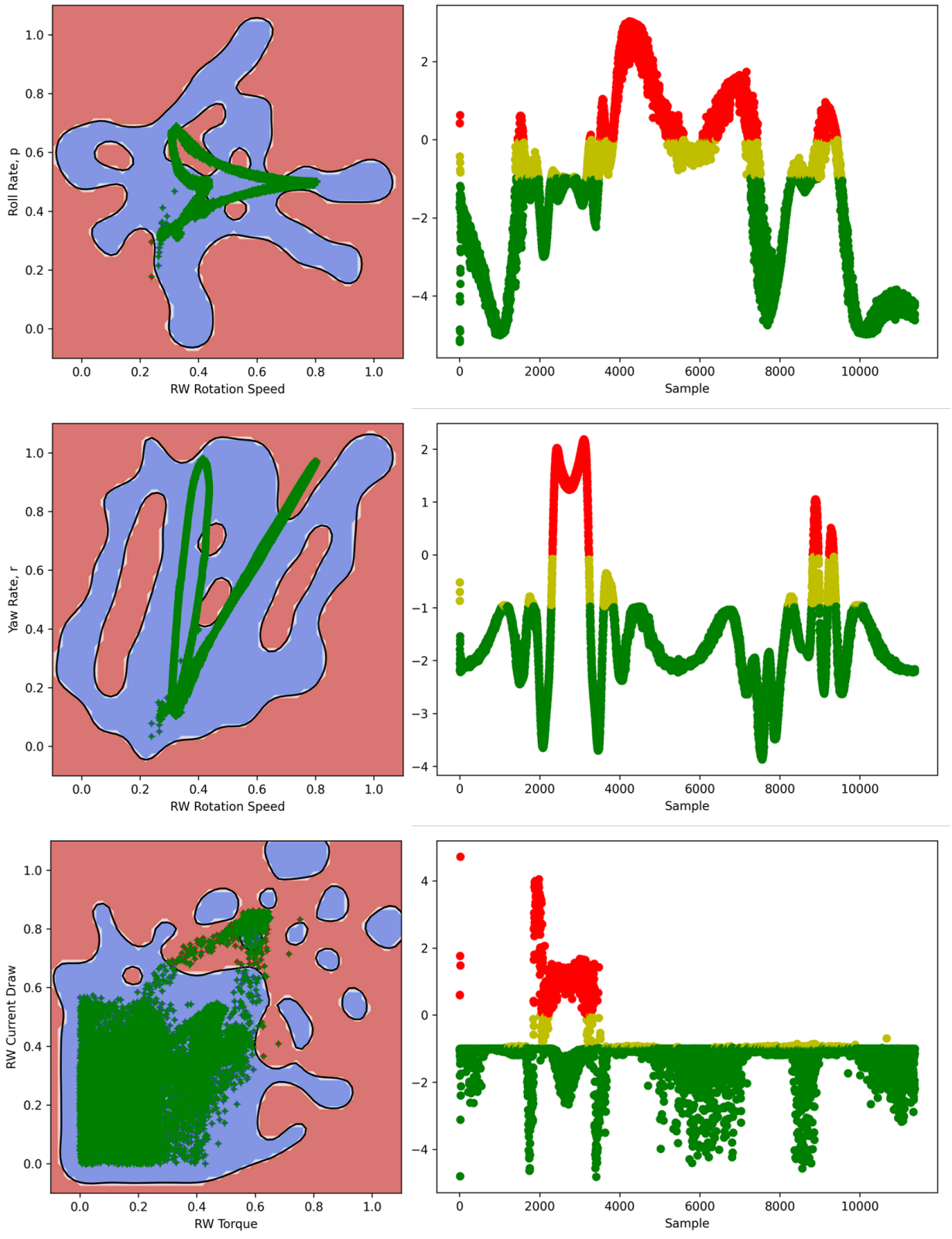


Figure 8.20 Trained AISOSVM models with failure data overlay in green (left) and fault trend outputs (right).

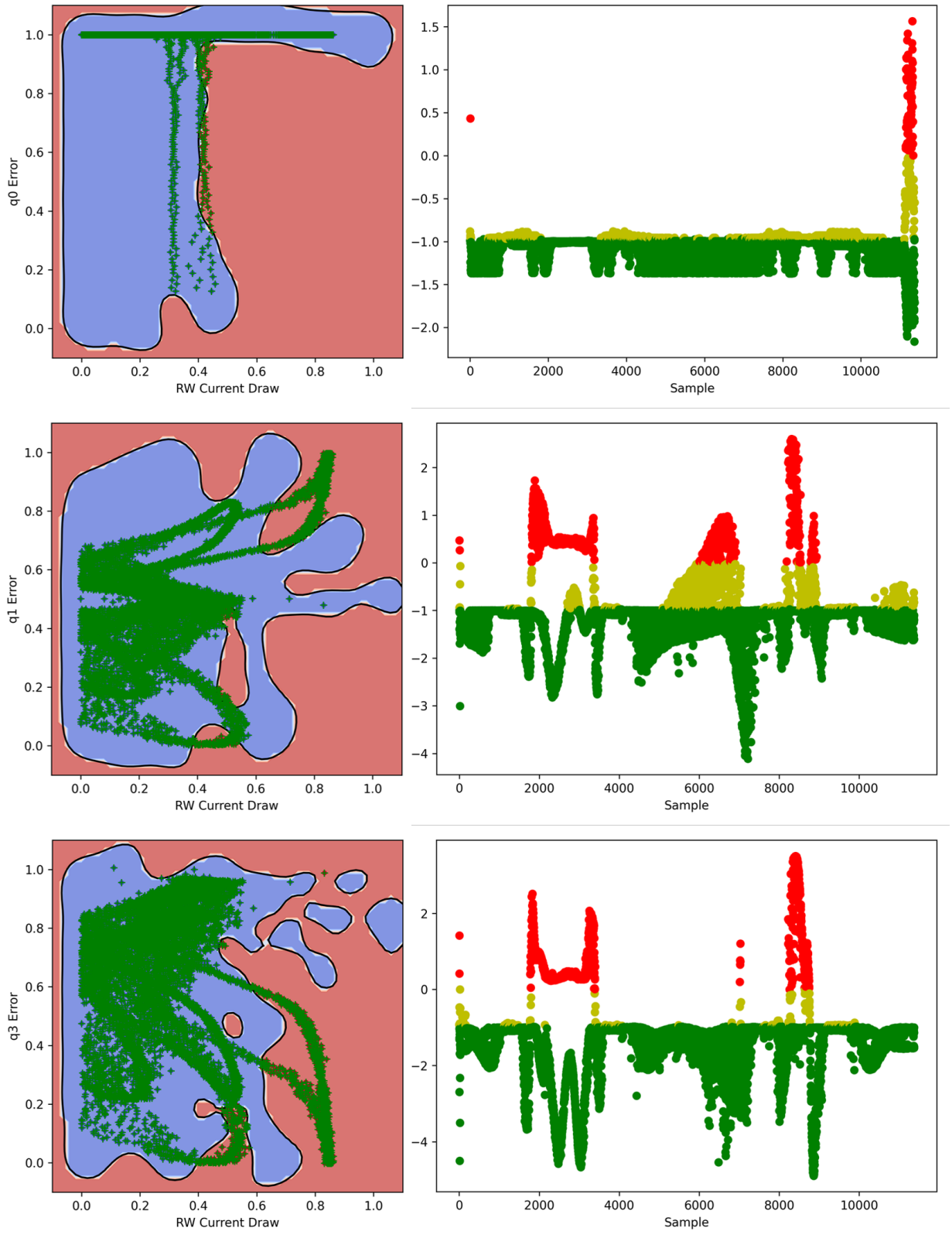


Figure 8.21 Trained AISOSVM models with failure data overlay in green (left) and fault trend outputs (right).

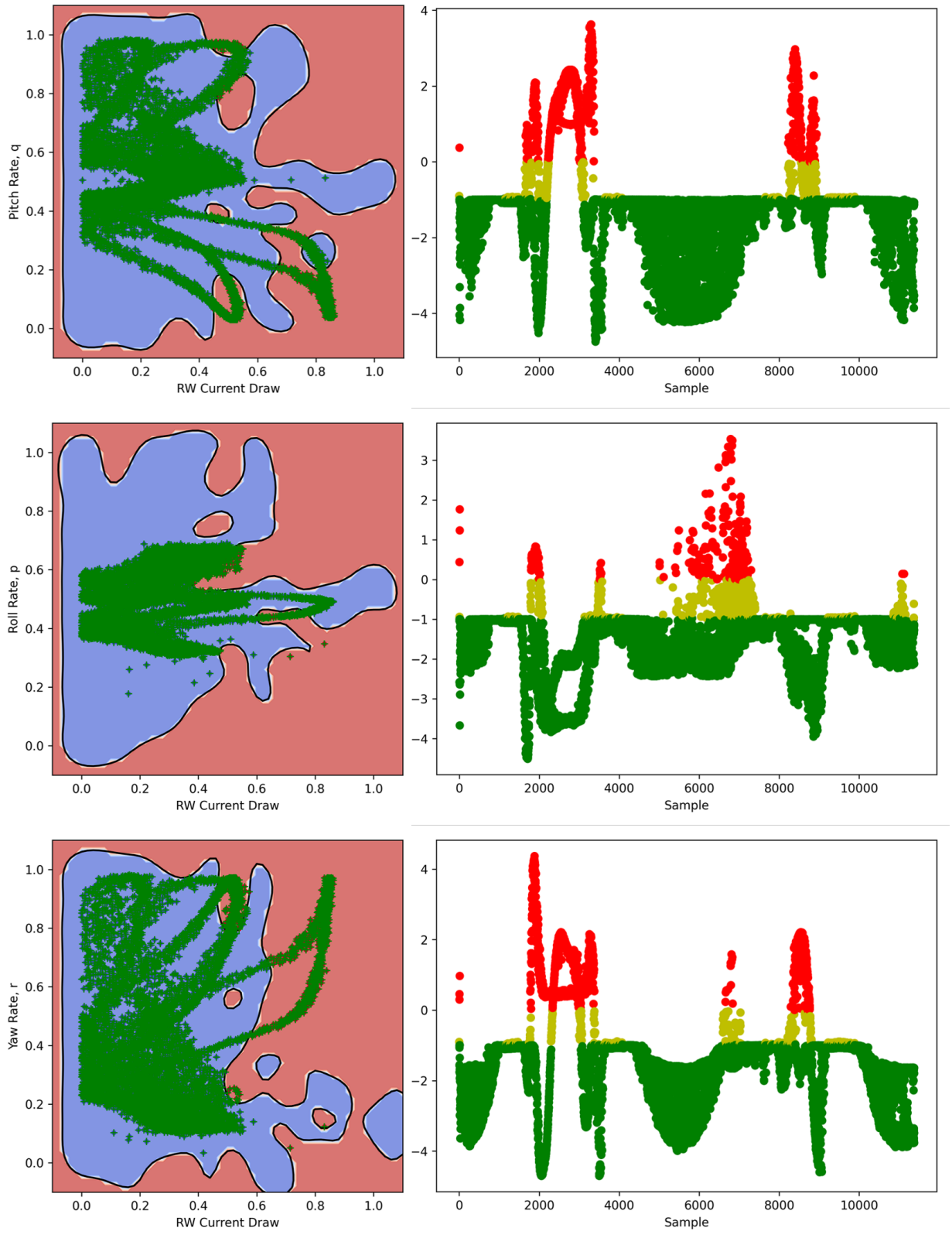


Figure 8.22 Trained AISOSVM models with failure data overlay in green (left) and fault trend outputs (right).

Table 8.8 AISOSVM model performance metrics for the ACS failure test.

Feature Space	DR	TP	FN
ω_{RW_2} vs. T_{RW_2}	0.439%	50	11322
ω_{RW_2} vs. i_{RW_2}	7.773%	884	10488
ω_{RW_2} vs. p	7.686%	874	10498
ω_{RW_2} vs. q	28.095%	3195	8177
ω_{RW_2} vs. r	10.026%	1140	10232
T_{RW_2} vs. i_{RW_2}	6.006%	683	10689
i_{RW_2} vs. $q_0 Error$	0.600%	68	11304
i_{RW_2} vs. $q_1 Error$	9.647%	1097	10275
i_{RW_2} vs. $q_3 Error$	9.207%	1047	10325
i_{RW_2} vs. p	11.089%	1261	10111
i_{RW_2} vs. q	2.515%	286	11086
i_{RW_2} vs. r	11.819%	1344	10028

8.5 Spacecraft Testbed Experimental Setup

A high fidelity spacecraft tested, the Extreme Access System (EASY) Spacecraft, was used to test the AISOSVM algorithms for operational system verification and validation. The EASY Spacecraft Testbed is a concept spacecraft designed at the Advanced Dynamics and Control Laboratory (ADCL) of Embry-Riddle Aeronautical University, FL, with the primary purpose of supporting the development of innovative autonomous space exploration spacecraft for in situ resource utilization in environments such as asteroids, where gravitational force is minimal. Figure 8.23 provides a picture of this testbed configuration. The EASY Spacecraft is mounted on a three-degree-of-freedom gimbaled platform, enabling free motion in roll, pitch, and yaw axes. This setup simulates full attitude control and angular rate regulation in microgravity environments, facilitating the testing of trajectory tracking and recovery from tumbles or other abnormal conditions that may arise in space.

The EASY Spacecraft features 24 solenoid-valves acting as thrusters, with sixteen fixed thrusters arranged in pairs and eight Thrust Vectoring Control (TVC) thrusters placed in pairs around the z-axis. The fixed thrusters are organized in horizontal and vertical configurations, enabling yaw and pitch rotations, respectively. The TVC thrusters are connected to a servo motor mechanism, allowing each pair to rotate around its position for additional

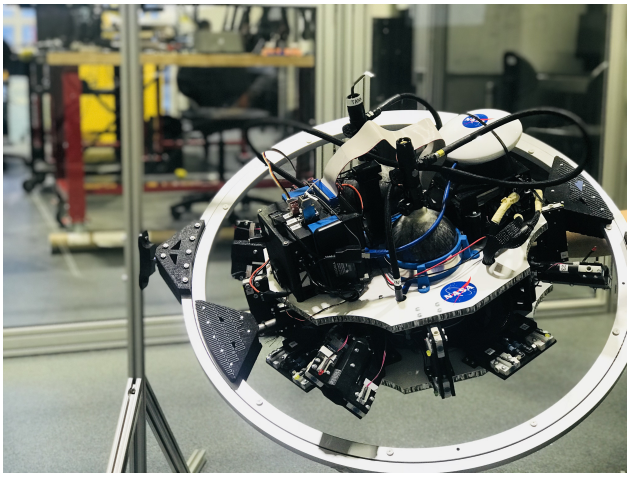


Figure 8.23 Extreme Access System (EASY) Spacecraft Test Bed.

yaw rotation control.Figure 8.24 shows the location of each thruster around the x and y axes of EASY.

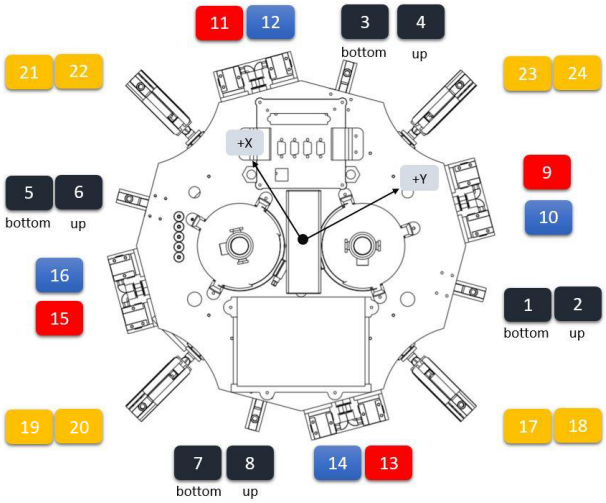


Figure 8.24 Location of the thrusters in the x and y axes. Yellow for TVC; red and blue for horizontal configuration, left and right side respectively; dark blue for vertical configuration, denoting both bottom or up.

The propulsion system of the spacecraft relies on compressed air stored in two high-pressure reservoirs, which can hold up to 4500 psi of pressure. Four pressure regulators reduce the pressure from the reservoirs to the desired operating pressure of 130 psi for the thrusters. Pressure sensors monitor the system's pressure levels, and relief valves are employed for safety in case of over-pressure. The opening and closing of the solenoid valves

are regulated by Pulse Width Modulated (PWM) signals from the digital IO pins on the onboard computer. Figure 8.25 provides a schematic of this cold gas thruster configuration and propellant feed lines.

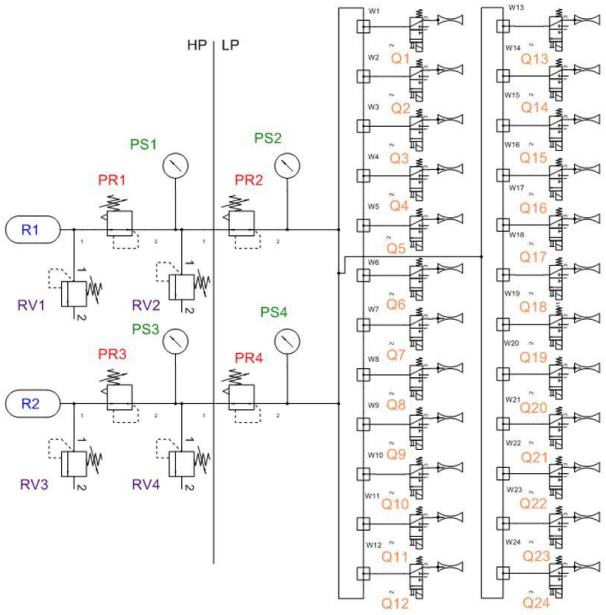


Figure 8.25 Propulsion System.

The flight computer used for the EASY Spacecraft is the PC/104 PCM-3355, which is integrated with the Emerald MM-4M-Port serial module and the Onyx MM Digital I/O module. This integration allows the spacecraft to include serial and analog input modules, along with digital I/O, which are used to actuate each of the solenoid valves to regulate the appropriate amount of air required for attitude control. An Inertial Measurement Unit (IMU) from Microstrain is employed to provide accurate measurements of attitude and angular rates, which are necessary for the controllers. The Microstrain IMU communicates with the flight computer through an RS232 communication protocol. A robust Nonlinear Dynamic Inversion (NLDI) controller is provides the command control for attitude tracking as described in [7].

The hardware components were tested individually to ensure full functionality before being mounted and integrated into the EASY Spacecraft. The testbed is connected to the

host computer via a high data rate Wi-Fi connection, which is crucial for online tuning of the controllers and signal monitoring. Figure 8.26 illustrates the EASY Spacecraft and lab control workstation setup.

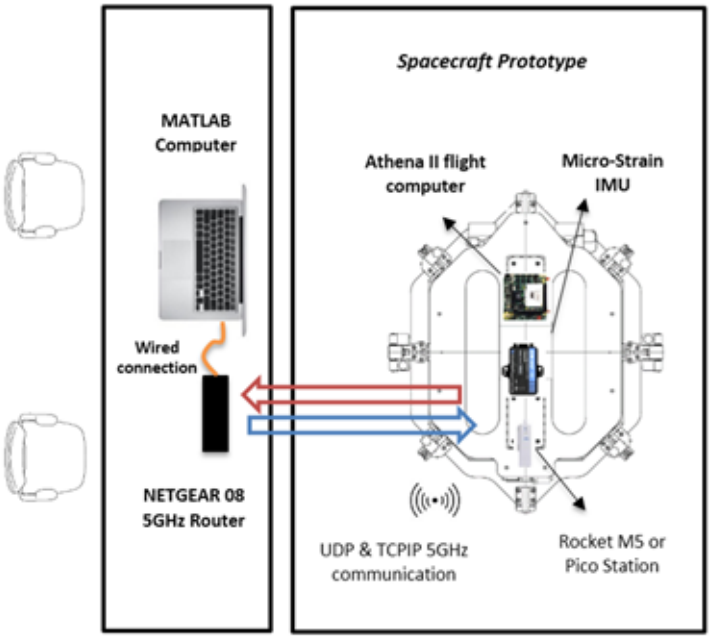


Figure 8.26 Schematic of Test Bed and Hardware used on EASY.

8.6 Spacecraft Testbed Data Acquisition

The application process of the AISOSVM fault detection system began with data accumulation for offline training. This involved conducting several nominal test cases with the testbed, enabling the creation of a training vector. This series of nominal test cases required the spacecraft to execute various roll maneuvers ranging between 28 and 35 degrees. It is important to note that the pitch and yaw axes of the spacecraft gimbal were locked in order to improve repeatability of the tests. The purpose of the tests was to generate a diverse set of data, representing a range of normal operating conditions for the spacecraft to perform this maneuver. This information was vital in the creation of a robust training vector for the AISOSVM, which would be essential for accurate fault detection during real-world operations.

Throughout the nominal test cases, several key data streams were recorded to provide a comprehensive understanding of the spacecraft's behavior during these maneuvers. These data streams created the selected features to represent the nominal dynamics of the testbed. Eighteen primary features used within this application are shown in Table 8.9.

Table 8.9 Selected features and states for the EASY Spacecraft.

	Feature	Desription
1	p	Roll Rate, deg/s
2	ϕ	Roll angle, deg
3	T_1	Thruster 1 Actuation
4	T_2	Thruster 2 Actuation
5	T_3	Thruster 3 Actuation
6	T_4	Thruster 4 Actuation
7	T_5	Thruster 5 Actuation
8	T_6	Thruster 6 Actuation
9	T_7	Thruster 7 Actuation
10	T_8	Thruster 8 Actuation
11	T_9	Thruster 9 Actuation
12	T_{10}	Thruster 10 Actuation
13	T_{11}	Thruster 11 Actuation
14	T_{12}	Thruster 12 Actuation
15	T_{13}	Thruster 13 Actuation
16	T_{14}	Thruster 14 Actuation
17	T_{15}	Thruster 15 Actuation
18	T_{16}	Thruster 16 Actuation

Once the nominal data had been gathered, the AISOSVM was trained on the selected feature spaces to ensure the model's adaptability to new and unseen scenarios. This training process created 153 possible feature space representations and enabled the AISOSVM to develop a comprehensive understanding of the spacecraft's normal behavior and, subsequently, enhance its ability to detect any deviations from this norm, indicating potential faults. This feature space was reduced to the first 10 features for their sensitivity to roll thruster failures. Thrusters T_9 through T_{16} provide yaw control. Since only a roll maneuver is commanded and the yaw gimbal is locked for this test scenario, these features are disregarded simplifying the fault detection test.

To test the AISOSVM's fault detection capability, a failure test case was conducted to simulate a fault within the spacecraft's attitude control thruster system. In this scenario, Thruster 1 was intentionally turned off, preventing the EASY spacecraft from utilizing a commanded roll maneuver. The roll rate, roll angle, and commanded thruster actutation for T_1 through T_8 are provided in Figure 8.27, Figure 8.28, Figure 8.29, and Figure 8.30 for a nominal and failure test case for comparison. The nominal data was acquired from a 32 degree roll maneuver test case scenario and the failure data was acquired from a 35 degree commanded roll maneuver with the injected thruster error in T_1.

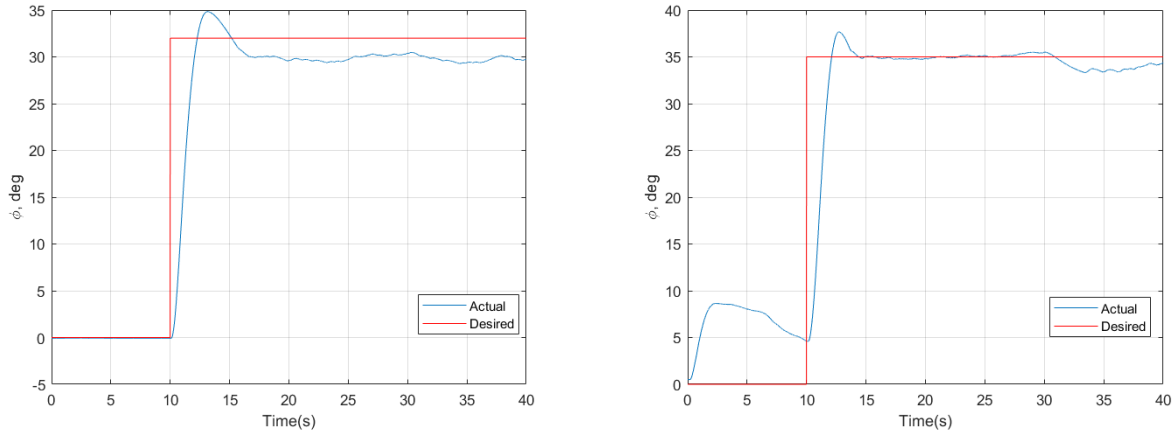


Figure 8.27 Nominal EASY spacecraft roll angle test data (left) and injected failure test data (right).

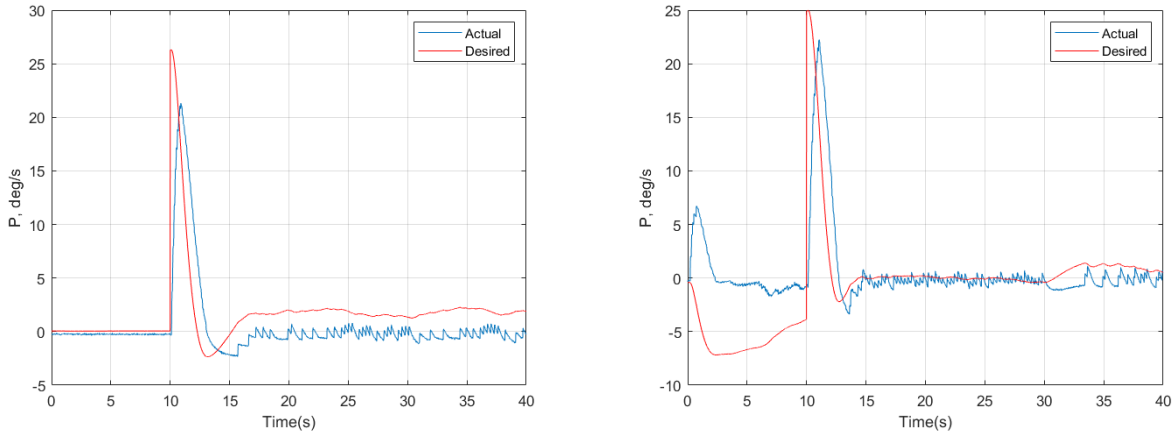


Figure 8.28 Nominal EASY spacecraft roll rate test data (left) and injected failure test data (right).

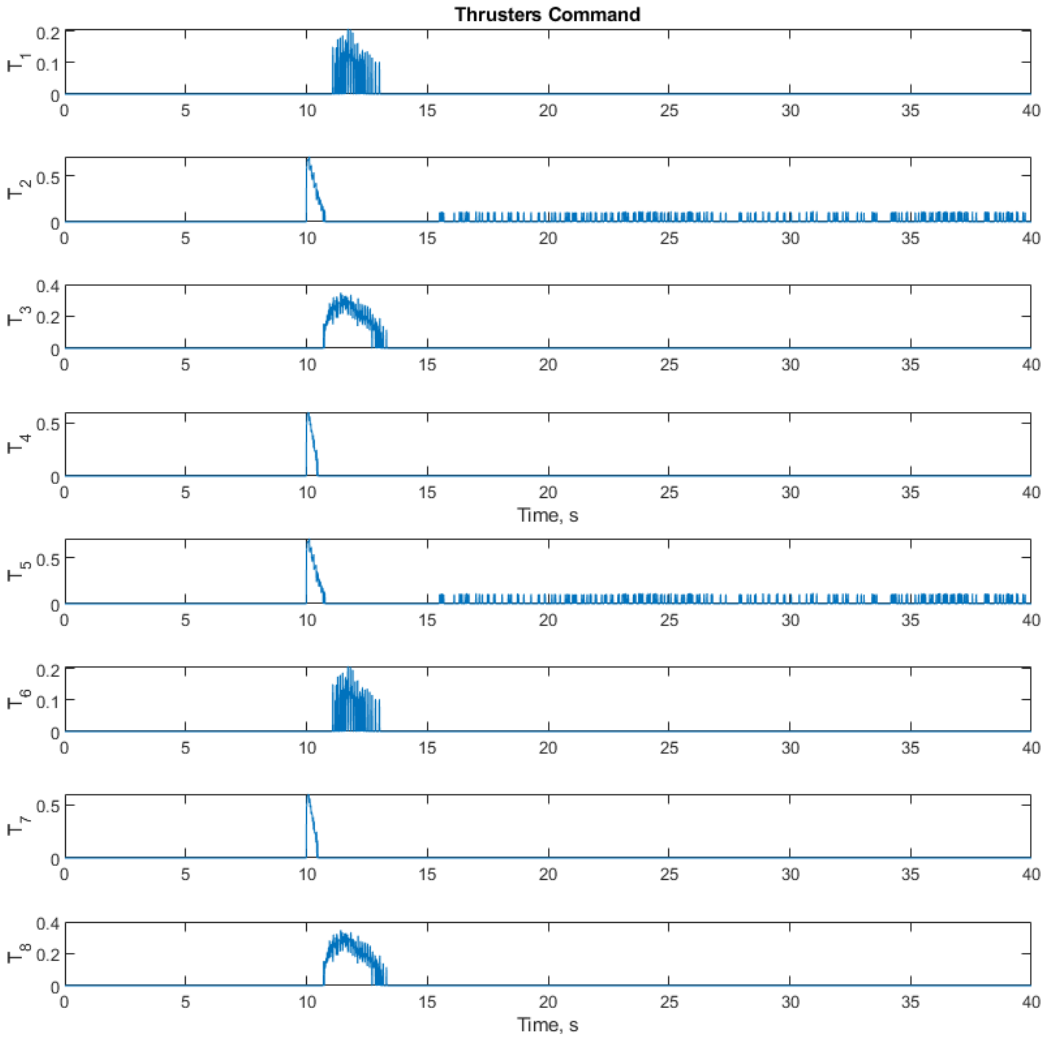


Figure 8.29 Nominal thruster actuation data.

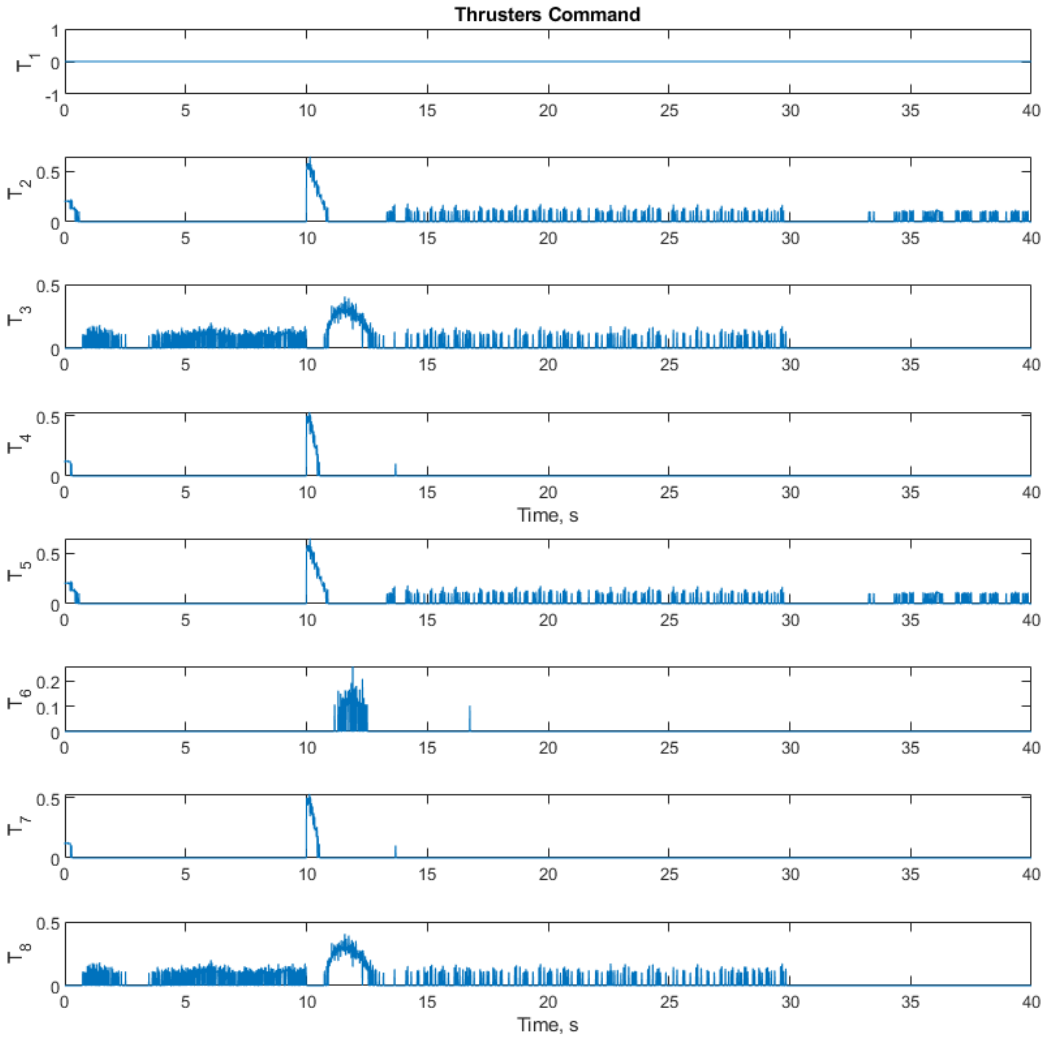


Figure 8.30 Thruster actuation data with injected failure in T_1.

8.7 Spacecraft Testbed Implementation Results

The AISOSVM models were validated with a nominal test data set acquired from four separate experiments with the EASY Spacecraft performing roll maneuvers between 28 - 35 degrees. This validation data was acquired from a 32-degree roll maneuver that the AISOSVM was not previously trained on. The results for twelve of the most sensitive features are presented in Figure 8.31 through Figure 8.34 along with the trend analysis for each of the feature spaces. For the EASY Spacecraft measurements, the AISOSVMs were able to create *self-nonself* spaces to separate the nominal and abnormal dynamics of the spacecraft; however, due to the sparse nature of the training data, which contained only about 30 seconds of data for four separate, unique maneuvers, the generated AISOSVM models suffered partially from over-fitting creating a restrictive decision boundary. This created more instances of false alarms previously unseen in the simulation experiments where significantly more training data could be acquired. Despite the sparsity of the training data, the AISOSVM models still showed good performance in correctly classifying the nominal data with a false positive rate (FPR) consistently less than 1.0% for the feature spaces. The performance metrics for these feature spaces are summarized in Table 8.10.

Table 8.10 AISOSVM model performance metrics for the EASY Spacecraft validation test.

Feature Space	AUC Score	FPR	LOO Estimation
p vs. ϕ	1.0	0.0%	0.0382
p vs. T_1	0.993	0.687%	0.1170
p vs. T_2	0.998	0.245%	0.0993
p vs. T_3	0.997	0.245%	0.0854
p vs. T_4	0.996	0.437%	0.1225
p vs. T_5	0.998	0.250%	0.0893
p vs. T_6	0.993	0.656%	0.1009
p vs. T_7	0.997	0.312%	0.1618
p vs. T_8	0.999	0.125%	0.0800
p vs. T_9	0.999	0.125%	0.1314
ϕ vs. T_3	0.996	0.437%	0.0881
ϕ vs. T_8	0.998	0.156%	0.0994

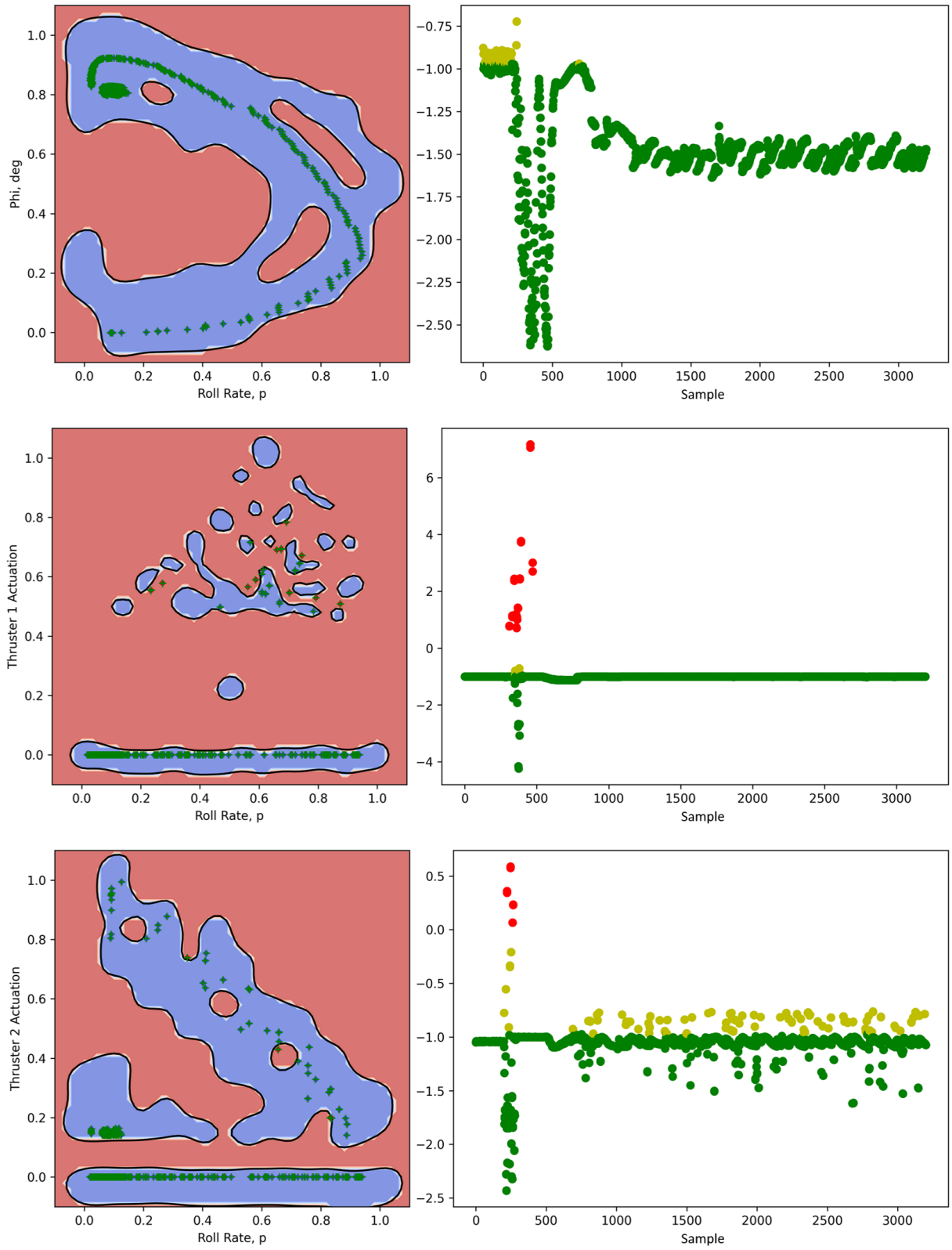


Figure 8.31 Trained AISOSVM models with EASY Spacecraft nominal data overlay in green (left) and fault trend outputs (right).

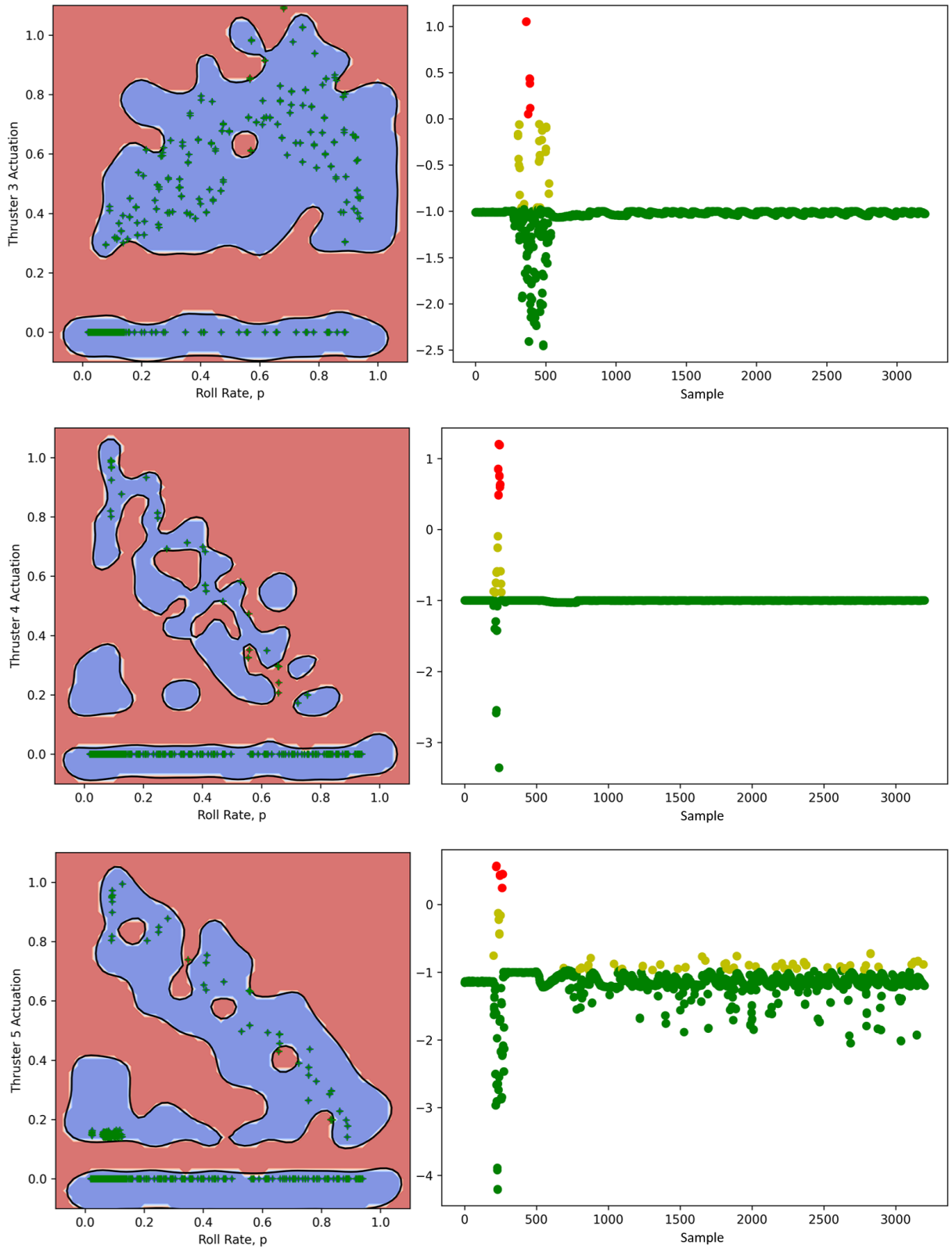


Figure 8.32 Trained AISOSVM models with EASY Spacecraft nominal data overlay in green (left) and fault trend outputs (right).

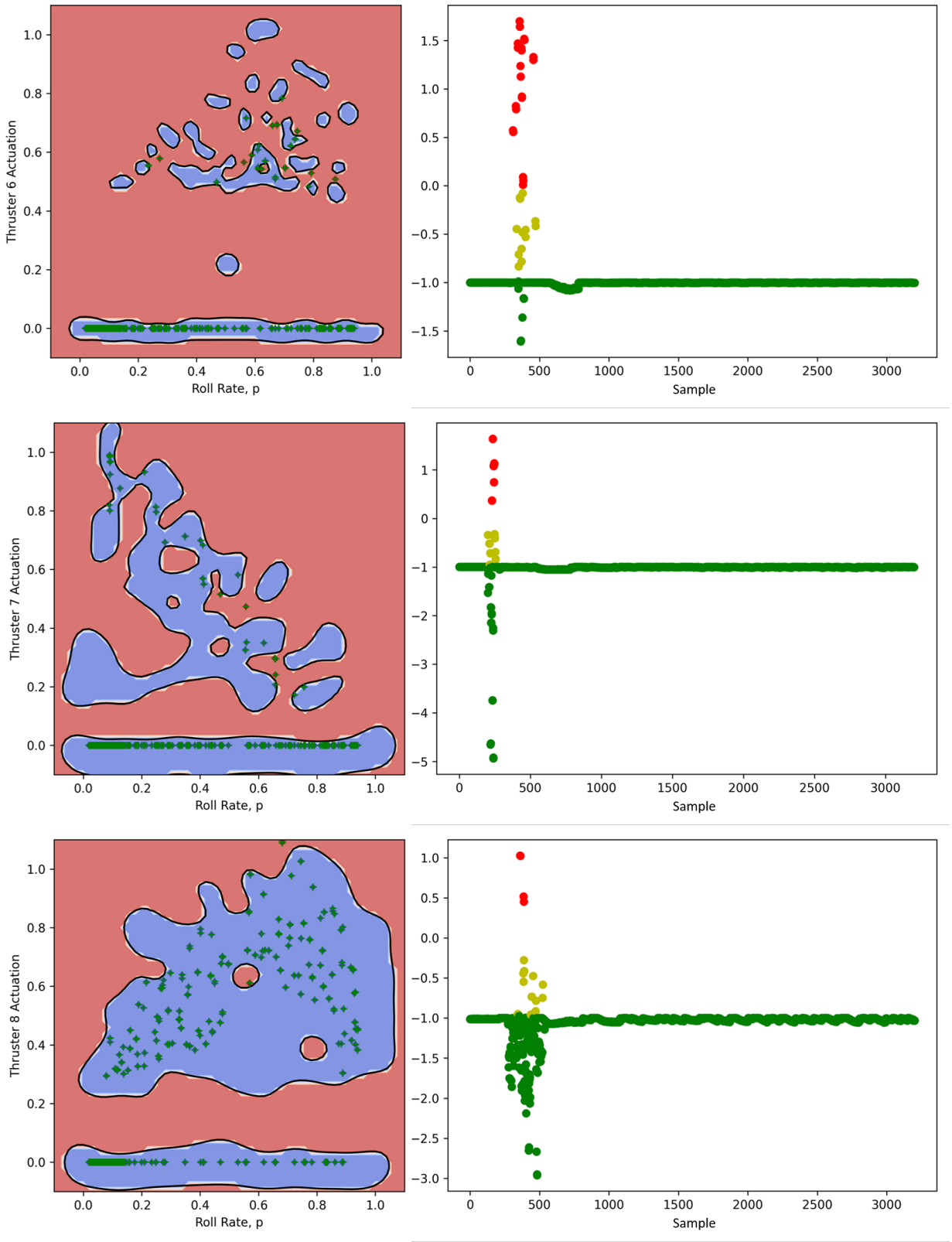


Figure 8.33 Trained AISOSVM models with EASY Spacecraft nominal data overlay in green (left) and fault trend outputs (right).

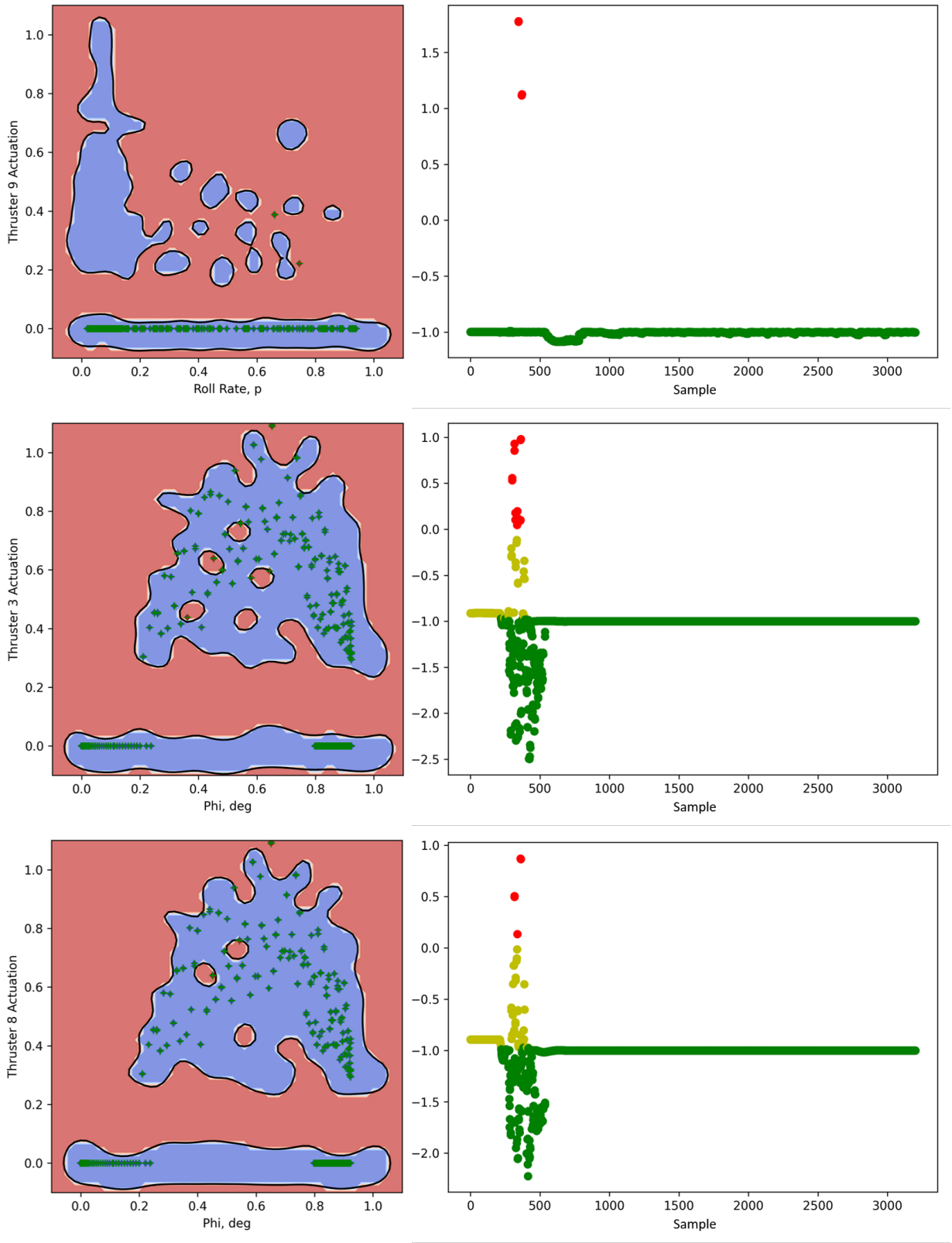


Figure 8.34 Trained AISOSVM models with EASY Spacecraft nominal data overlay in green (left) and fault trend outputs (right).

The trained AISOSVM models were applied to the EASY Spacecraft failure data acquired from injecting a complete failure into Thruster 1, T_1. Each of the AISOSVM models were able to detect the thruster failure, each with detection rates, DR, greater than 6.0% with several feature spaces achieving a detection rate above 10.0%. These performance metrics for the failure test case are summarized in Table 8.11. Additionally, the AISOSVM model data and the trend analysis are provided in Figure 8.35 through Figure 8.38 for the selected feature spaces.

It is important to note that the EASY Spacecraft, despite having a failed thruster, was able to both remain stable and achieve the commanded attitude with little deviation from nominal. This is most likely an emergent behavior due to the pitch and yaw gimbal axes being locked while still using redundant thrusters. Had the gimbals not been locked, the failed thruster could have caused a more significant deviation in the nominal dynamics of the testbed. This effect is reflected in the AISOSVM detection rates where only a small percentage of the failure data was correctly classified as a failure mode because the EASY spacecraft was able to recover quickly.

Table 8.11 AISOSVM model performance metrics for the EASY Spacecraft failure test.

Feature Space	DR	TP	FN
p vs. ϕ	10.278%	329	2872
p vs. T_1	6.748%	216	2985
p vs. T_2	7.685%	246	2955
p vs. T_3	10.552%	341	2860
p vs. T_4	7.591%	243	2958
p vs. T_5	7.591%	243	2958
p vs. T_6	7.622%	244	2957
p vs. T_7	7.310%	234	2967
p vs. T_8	10.153%	325	2876
p vs. T_9	7.248%	232	2969
ϕ vs. T_3	13.651%	437	2764
ϕ vs. T_8	14.902%	477	2724

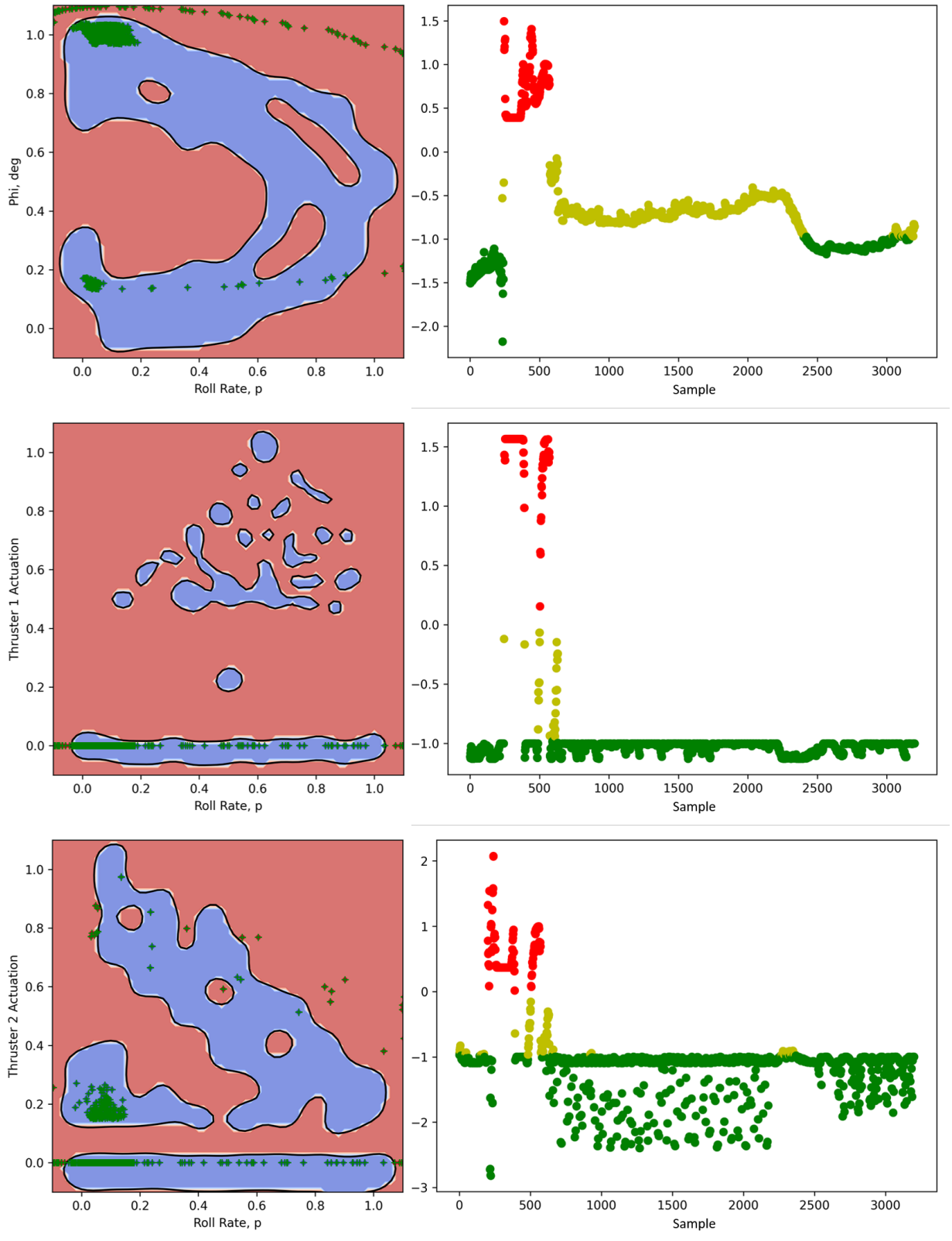


Figure 8.35 Trained AISOSVM models with EASY Spacecraft failure data overlay in green (left) and fault trend outputs (right).

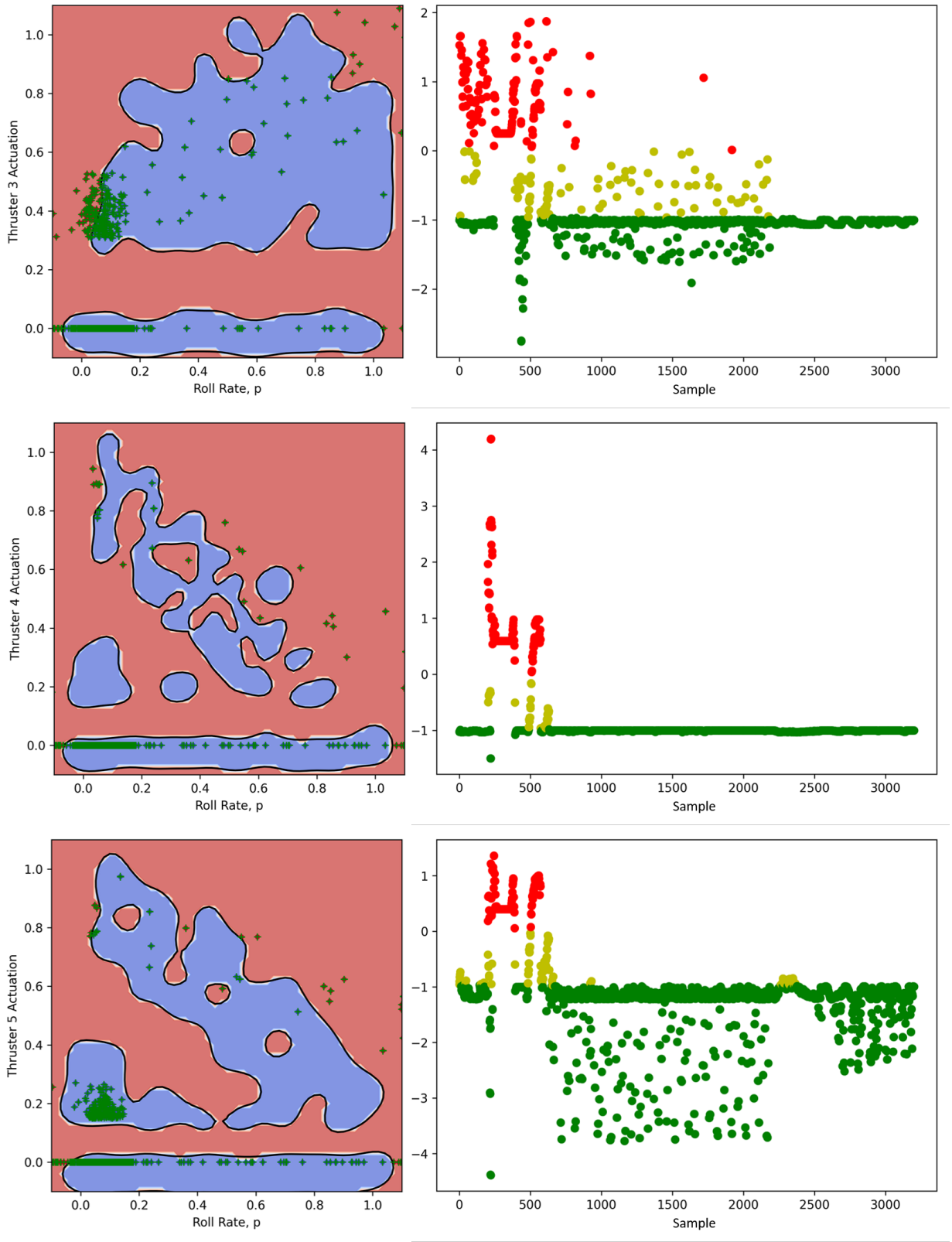


Figure 8.36 Trained AISOSVM models with EASY Spacecraft failure data overlay in green (left) and fault trend outputs (right).

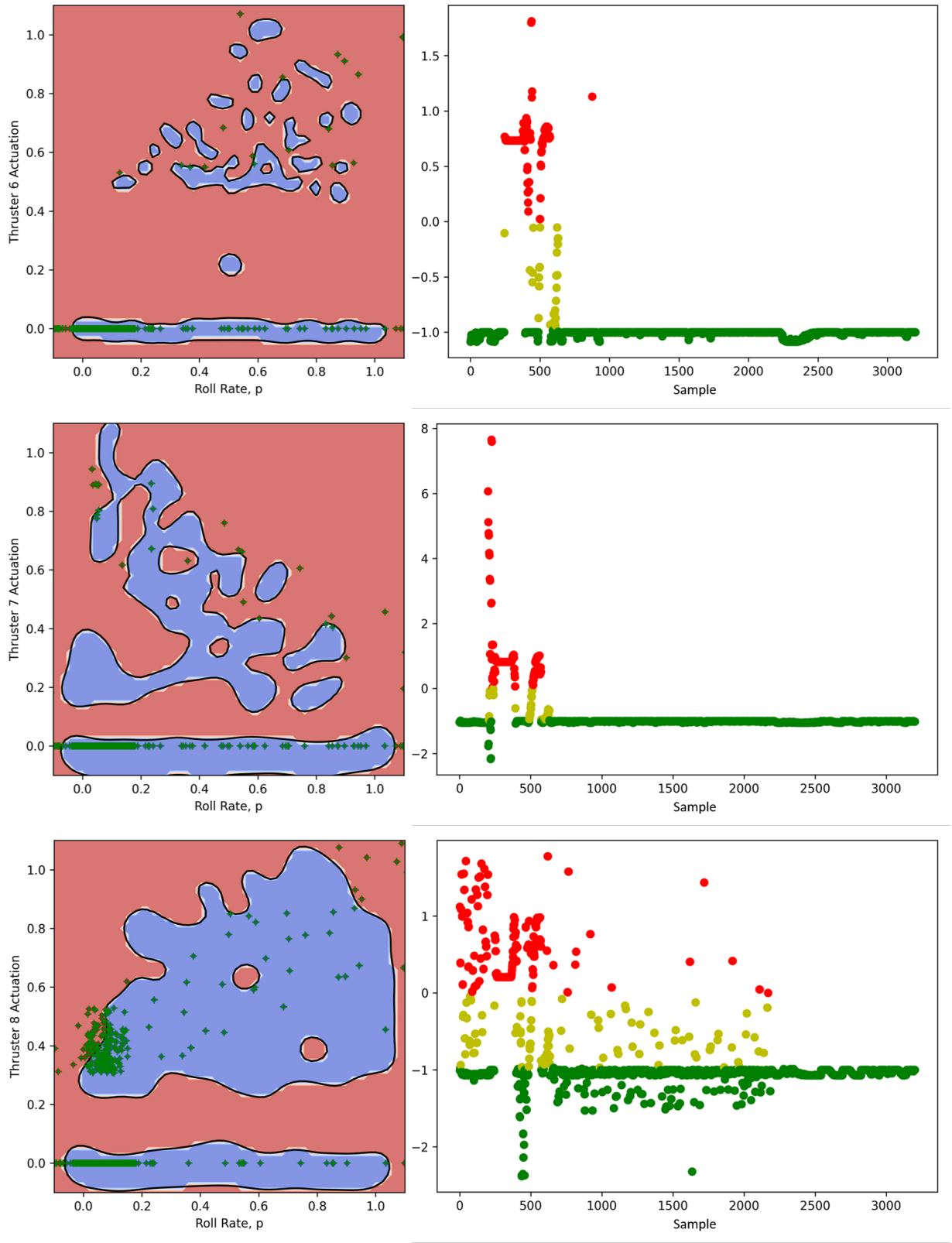


Figure 8.37 Trained AISOSVM models with EASY Spacecraft failure data overlay in green (left) and fault trend outputs (right).

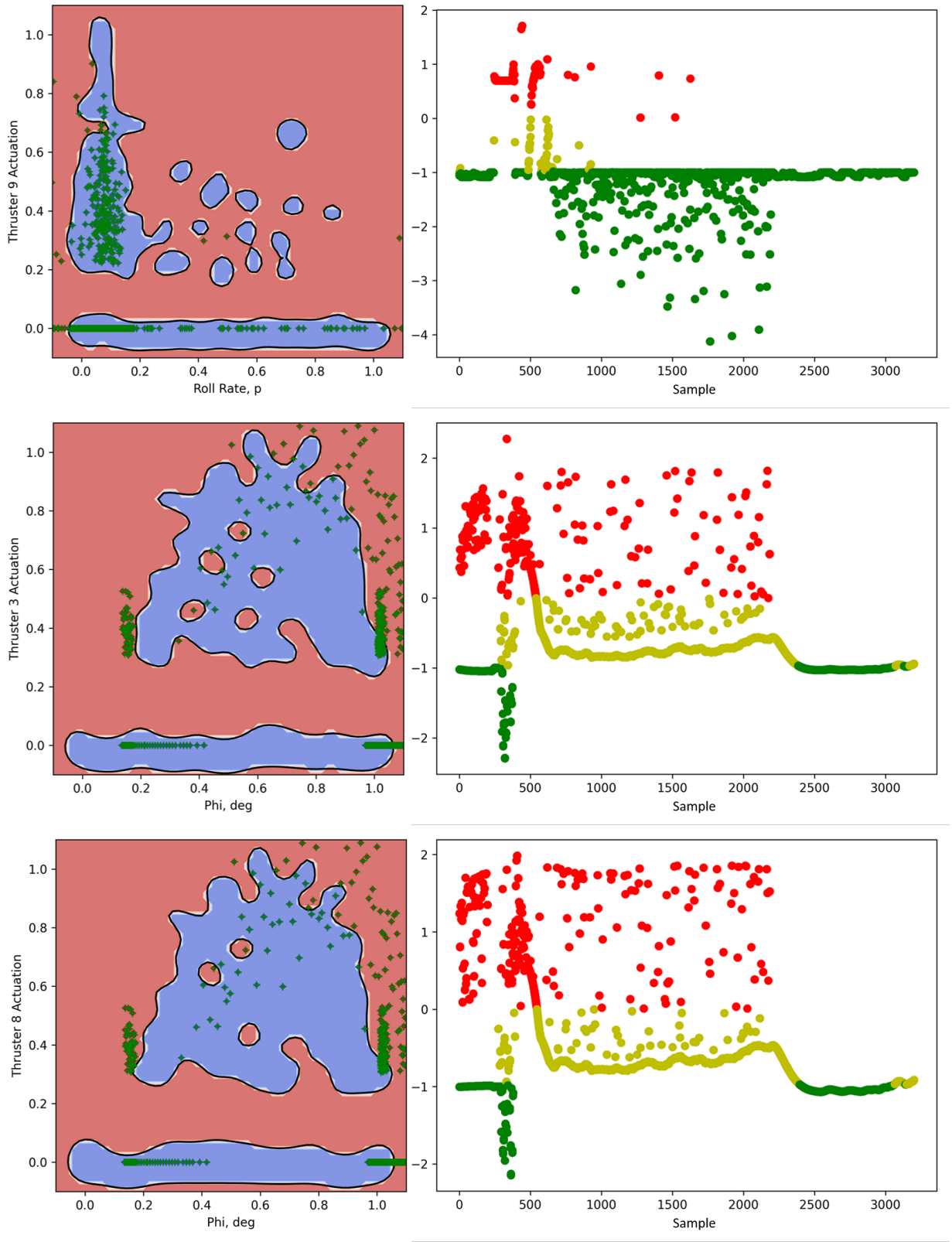


Figure 8.38 Trained AISOSVM models with EASY Spacecraft failure data overlay in green (left) and fault trend outputs (right).

9 Conclusion

This final chapter serves as a summary of the research presented throughout the dissertation, emphasizing the key findings and contributions of the Artificial Immune System augmented Online Support Vector Machine (AISOSVM) fault management system in addressing the challenges of fault detection and health management for autonomous spacecraft. The chapter highlights the significance of the study and its impact on the field, along with the potential implications for practical applications in the aerospace industry. Additionally, the limitations of the current research and the AISOSVM framework are acknowledged, providing a basis for identifying areas that warrant further investigation. Lastly, future research directions are proposed, offering a roadmap for continuous improvement and expansion of the AISOSVM fault management system, ultimately aiming to enhance the robustness and reliability of autonomous spacecraft operations.

9.1 Summary of Contributions

This dissertation has made several key contributions to the field of fault detection and health management for autonomous spacecraft through the development and implementation of the AISOSVM fault management system. The primary contributions have expanded the understanding and application of immune system paradigms in fault detection and have demonstrated the effectiveness of an adaptive architecture. Furthermore, this work has laid a strong foundation for the integration of reinforcement learning in future research to further enhance fault detection capabilities.

The main contributions of this dissertation are:

- Integration of the immune system paradigm to augment fault detection
- Development of an adaptive fault detection architecture
- Development of a structured and integrated approach for the design and deployment of a data-driven fault detection system

- Foundational work for integrating reinforcement learning for online, autonomous fault detection system adaption
- Application of the designed data-driven fault detection framework to real world scenarios and platform models

The integration of the immune system paradigm has been a crucial contribution to augmenting fault detection capabilities. By incorporating the Artificial Immune System (AIS) paradigm with the Online Support Vector Machine (OSVM), the AISOSVM framework has significantly improved the efficiency and accuracy of fault detection in autonomous spacecraft. This innovative approach has demonstrated the value of incorporating bio-inspired techniques to enhance fault detection and health management systems.

The development of an adaptive fault detection architecture has been another significant contribution. The AISOSVM framework, which combines the AIS paradigm and OSVM, has shown its adaptability to different training data and system configurations. This adaptive nature ensures accurate and efficient fault detection across various spacecraft scenarios, highlighting the potential of this approach in practical aerospace applications.

The creation of a structured and integrated approach for the design and deployment of a data-driven fault detection system is another key contribution. By utilizing Model-Based Systems Engineering (MBSE) and the Capella tool with the Arcadia methodology, a systematic design process for the AISOSVM fault management system has been established. This approach allows for a comprehensive understanding of the system requirements, operational architecture, and design, leading to an efficient tailoring process for various autonomous spacecraft platforms.

The foundational work for integrating reinforcement learning for online, autonomous fault detection system adaption has been an important contribution. Although not fully developed within the scope of this dissertation, the groundwork laid in this research paves the way for future studies to incorporate reinforcement learning into the AISOSVM framework. This integration would allow for continuous online learning and adaptation, further enhancing the

performance and applicability of the fault detection system in a wide range of autonomous systems.

An additional vital contribution of this dissertation is the application of the designed data-driven fault detection framework to real-world scenarios and platform models. By applying the AISOSVM framework to a variety of spacecraft scenarios and models, including the reaction wheel model and the attitude controller, this research has demonstrated the practical applicability and effectiveness of the developed approach. These real-world tests have not only validated the performance of the AISOSVM fault management system but have also showcased its adaptability and robustness when faced with complex and dynamic operational environments. This practical application of the AISOSVM framework contributes to bridging the gap between theoretical research and real-world implementation, ultimately promoting the adoption of advanced fault detection and health management systems in the aerospace industry.

Together, these contributions represent a significant advancement in the field of fault detection and health management for autonomous spacecraft, providing a robust, adaptive, and scalable solution that can be tailored to various platforms and operational scenarios.

9.2 Limitations and Future Work

While the AISOSVM-based fault detection system presented in this dissertation has demonstrated promising results, there are some limitations that should be acknowledged. Additionally, these limitations present opportunities for future research to address and further improve the capabilities of the fault detection and health management system.

1. Fault Isolation: The current AISOSVM framework focuses primarily on fault detection, without providing a comprehensive strategy for fault isolation. In real-world applications, isolating the root cause of a fault is essential for the rapid diagnosis and resolution of system issues. Future work could expand the AISOSVM framework to incorporate fault isolation strategies, potentially by integrating additional machine learning algorithms or model-based approaches.

2. Online Learning: While the AISOSVM architecture does allow for adaptation based on operator intervention, it currently does not support fully autonomous online learning. The integration of reinforcement learning or other online learning techniques could enable the AISOSVM to adapt autonomously, enhancing its performance and further reducing the reliance on human intervention.

3. Multi-Model Testing: The AISOSVM framework has been applied and validated using specific spacecraft scenarios and platform models. However, to establish the broader applicability of the framework, it is necessary to test its performance on a diverse range of autonomous systems, spanning various domains and industries. Future work could investigate the implementation and adaptation of the AISOSVM framework for different autonomous systems, such as unmanned aerial vehicles, robotic systems, and power grids.

4. User Interface and System Integration: The current framework does not include the fully envisioned tool and workflow. Additional work can be performed such as the development of a user interface or provisions for seamless integration with existing system models. To facilitate broader adoption and ease of use, future work could focus on the development of a graphical user interface (GUI) and the creation of application programming interfaces (APIs) for seamless integration with other software and system models.

5. Scalability: The AISOSVM framework has been demonstrated to work effectively for the considered spacecraft scenarios, but its performance in larger-scale systems with increased complexity remains to be evaluated. Future research could explore the scalability of the AISOSVM framework, assessing its ability to handle systems with larger numbers of sensors, features, and operational modes.

By addressing these limitations and exploring future research directions, the AISOSVM framework can continue to evolve and improve, further solidifying its potential as a valuable

fault detection and health management tool for a wide range of autonomous systems.

9.3 Broader Impact and Applications

The AISOSVM-based fault detection system developed in this dissertation has the potential to significantly impact various industries and applications beyond spacecraft fault management. By leveraging the inherent adaptability and robustness of the artificial immune system paradigm, the AISOSVM framework can be applied to a wide range of autonomous systems and domains, enhancing their reliability and overall performance. In this section, we discuss some of the broader impacts and potential applications of the AISOSVM framework.

Unmanned Aerial Vehicles (UAVs): The reliability and safety of UAVs, commonly known as drones, are crucial concerns in their operation, especially in applications such as package delivery, infrastructure inspection, and search and rescue missions. The AISOSVM fault detection framework can be employed to monitor the health of UAVs in real-time, identifying faults and enabling prompt remedial actions, ultimately improving the safety and efficiency of UAV operations.

1. Robotics and Automation: As the adoption of robotics and automation continues to grow in various industries, such as manufacturing, agriculture, and healthcare, the need for effective fault detection and health management systems becomes increasingly critical. The AISOSVM framework can be applied to monitor the performance of robots and automated systems, ensuring their reliability and minimizing downtime due to unexpected faults.

2. Smart Grids and Energy Systems: With the increasing complexity of power grids and the integration of renewable energy sources, the demand for robust fault detection and health management systems in the energy sector is growing. The AISOSVM framework can be utilized to monitor the health of power grids and energy systems, allowing for rapid identification and resolution of faults, which in turn helps maintain the stability and efficiency of the grid.

3. Transportation Systems: Modern transportation systems, such as trains, buses, and autonomous vehicles, require reliable fault detection to ensure safety and efficiency. The AISOSVM framework can be implemented to monitor the health of these transportation systems, detecting faults and potential failures in real-time, thereby reducing the risk of accidents and improving overall system performance.

4. Industrial Internet of Things (IIoT): The proliferation of connected devices and sensors in industrial settings presents both opportunities and challenges. One significant challenge is the effective management and monitoring of these devices to ensure their proper functioning. The AISOSVM framework can be employed to monitor the health of connected devices within the IIoT ecosystem, providing early warning of faults and enabling preventative maintenance.

The applicability of the AISOSVM framework to these diverse domains highlights its potential to revolutionize fault detection and health management across various industries. By further developing and refining the AISOSVM framework, we can contribute to safer, more reliable, and more efficient autonomous systems, ultimately leading to a more sustainable and connected world.

REFERENCES

[1] Gao, Z., Cecati, C., and Ding, S. X., "A Survey of Fault Diagnosis and Fault-Tolerant Techniquesc - Part I: Fault Diagnosis With Model-Based and Signal-Based Approaches," *IEEE Transactions on Industrial Electronics*, Vol. 62, No. 6, 2015, pp. 3757–3767.

[2] Gao, Z., Cecati, C., and Ding, S. X., "A Survey of Fault Diagnosis and Fault-Tolerant Techniques - Part II: Fault Diagnosis With Knowledge-Based and Hybrid/Active Approaches," *IEEE Transactions on Industrial Electronics*, Vol. 62, No. 6, 2015, pp. 3768–3774.

[3] Marzat, J., Piet-Lahanier, H., Damongeot, F., and Walter, E., "Model-based fault diagnosis for aerospace systems: a survey," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, Vol. 226, 2012, pp. 1329–1360. https://doi.org/10.1177/0954410011421717.

[4] Rahimi, A., Kumar, K. D., and Alighanbari, H., "Fault estimation of satellite reaction wheels using covariance based adaptive unscented Kalman filter," *Acta Astronautica*, Vol. 134, 2017, pp. 159 – 169. https://doi.org/https://doi.org/10.1016/j.actaastro.2017.02.003.

[5] Chen, Y., and Lee, M., "Neural networks-based scheme for system failure detection and diagnosis," *Mathematics and Computers in Simulation*, Vol. 58, No. 2, 2002, pp. 101 – 109. https://doi.org/https://doi.org/10.1016/S0378-4754(01)00330-5.

[6] Jiang, T., Khorasani, K., and Tafazoli, S., "Parameter Estimation-Based Fault Detection, Isolation and Recovery for Nonlinear Satellite Models," *IEEE Transactions on Control Systems Technology*, Vol. 16, No. 4, 2008, pp. 799–808.

[7] Garcia, D. F., Perez, A. E., Moncayo, H., Rivera, K., and Betancur, Y., "Spacecraft Heath Monitoring Using a Biomimetic Fault Diagnosis Scheme," *Journal of Aerospace*

Information Systems, Vol. 15, No. 7, 2018, pp. 396–413. https://doi.org/10.2514/1.I010612.

[8] Coulter, N., and Moncayo, H., "Comparison of Optimal and Bioinspired Adaptive Control Laws for Spacecraft Sloshing Dynamics," *Journal of Spacecraft and Rockets*, Vol. 57, No. 1, 2020, pp. 12–32. https://doi.org/10.2514/1.A34473.

[9] Ding, S., and Li, S. X., "Clonal selection algorithm for feature selection and parameters optimization of support vector machines," 2009, pp. 17–20. https://doi.org/10.1109/KAM.2009.86.

[10] Aydin, I., Karakose, M., and Akin, E., "A multi-objective artificial immune algorithm for parameter optimization in support vector machine," *Applied soft computing*, Vol. 11, No. 1, 2011, pp. 120–129.

[11] Zhang, R., and Xiao, X., "A Clone Selection Based Real-Valued Negative Selection Algorithm," *Complexity*, Vol. 2018, 2018. https://doi.org/10.1155/2018/2520940.

[12] Yang, L., Cheng, N., and Li, Q., "The Design of Integrated Navigation System Based on MBSE," *2018 IEEE CSAA Guidance, Navigation and Control Conference (CGNCC)*, 2018, pp. 1–5. https://doi.org/10.1109/GNCC42960.2018.9018855.

[13] Mackey, R., Nikora, A., Altenbuchner, C., Bocchino, R., Sievers, M., Fesq, L., Kolcio, K. O., Litke, M. J., and Prather, M., "On-Board Model Based Fault Diagnosis for CubeSat Attitude Control Subsystem: Flight Data Results," *2021 IEEE Aerospace Conference (50100)*, 2021, pp. 1–17. https://doi.org/10.1109/AERO50100.2021.9438342.

[14] Feather, M. S., Kennedy, B., Mackey, R., Troesch, M., Altenbuchner, C., Bocchino, R., Fesq, L., Hughes, R., Mirza, F., Nikora, A., Beauchamp, P., Doran, P., Kolcio, K. O., Litke, M. J., and Prather, M., "Demonstrations of System-Level Autonomy for Spacecraft," *2021 IEEE Aerospace Conference (50100)*, 2021, pp. 1–18. https://doi.org/10.1109/AERO50100.2021.9438427.

[15] Voirin, J.-L., *Model-based system and architecture engineering with the arcadia method*, Elsevier, 2017.

[16] Roques, P., *Systems architecture modeling with the Arcadia method: a practical guide to Capella*, Elsevier, 2017.

[17] Henderson, K., and Salado, A., "Value and benefits of model-based systems engineering (MBSE): Evidence from the literature," *Systems Engineering*, Vol. 24, No. 1, 2021, pp. 51–66. https://doi.org/https://doi.org/10.1002/sys.21566, URL https://incose.onlinelibrary.wiley.com/doi/abs/10.1002/sys.21566.

[18] Banaee, H., Ahmed, M. U., and Loutfi, A., "Data mining for wearable sensors in health monitoring systems: a review of recent trends and challenges," *Sensors*, Vol. 13, No. 12, 2013, pp. 17472–17500. https://doi.org/10.3390/s131217472.

[19] Katal, A., Wazid, M., and Goudar, R. H., "Big data: Issues, challenges, tools and Good practices," *2013 Sixth International Conference on Contemporary Computing (IC3)*, 2013, pp. 404–409. https://doi.org/10.1109/IC3.2013.6612229.

[20] Cartocci, N., Jaoude, N. A., Daigavane, P. M., and Verma, P., "A Comprehensive Case Study of Data-Driven Methods for Robust Aircraft Sensor Fault Isolation," *Sensors (Basel, Switzerland)*, Vol. 21, No. 5, 2021, p. 1645. https://doi.org/10.3390/s21051645.

[21] Jolliffe, I. T., "Principal Component Analysis, 2nd Ed." *Springer Series in Statistics*, 2002. https://doi.org/10.1007/b98835.

[22] Zhao, R., Yan, R., Chen, Z., Mao, K., Wang, P., and Gao, R. X., "Deep learning and its applications to machine health monitoring," *Mechanical Systems and Signal Processing*, Vol. 115, 2019, pp. 213–237.

[23] Chapagain, P., Timalsina, A., Bhandari, M., and Chitrakar, R., "Intrusion Detection

Based on PCA with Improved K-means," *Innovations in Electrical and Electronic Engineering: Proceedings of ICEEE 2022, Volume 2*, Springer, 2022, pp. 13–27.

[24] Sinaga, K. P., and Yang, M.-S., "Unsupervised K-means clustering algorithm," *IEEE access*, Vol. 8, 2020, pp. 80716–80727.

[25] Vapnik, V., "The Nature Stat. Learning Theory," , 1995.

[26] Bennett, K. P., and Campbell, C., "Support Vector Machines: Hype or Hallelujah?" *SIGKDD Explor. Newsl.*, Vol. 2, No. 2, 2000, p. 1–13. https://doi.org/10.1145/380995.380999, URL https://doi.org/10.1145/380995.380999.

[27] Joachims, T., "Text categorization with support vector machines: Learning with many relevant features," *Machine Learning: ECML-98: 10th European Conference on Machine Learning Chemnitz, Germany, April 21–23, 1998 Proceedings*, Springer, 2005, pp. 137–142.

[28] Vapnik, V., "Statistical Learning Theory," , 1998.

[29] Burges, C. J., "A tutorial on support vector machines for pattern recognition," *Data mining and knowledge discovery*, Vol. 2, No. 2, 1998, pp. 121–167.

[30] Kasnavi, S. A., Aminafshar, M., Shariati, M. M., Kashan, N. E. J., and Honarvar, M., "The effect of kernel selection on genome wide prediction of discrete traits by Support Vector Machine," *Gene Reports*, Vol. 11, 2018, pp. 279–282.

[31] Cervantes, J., Garcia-Lamont, F., Rodríguez-Mazahua, L., and Lopez, A., "A comprehensive survey on support vector machine classification: Applications, challenges and trends," *Neurocomputing*, Vol. 408, 2020, pp. 189–215. https://doi.org/https://doi.org/10.1016/j.neucom.2019.10.118, URL https://www.sciencedirect.com/science/article/pii/S0925231220307153.

[32] Jha, C. K., and Kolekar, M. H., "Cardiac arrhythmia classification using tunable Q-wavelet transform based features and support vector machine classifier," *Biomedical Signal Processing and Control*, Vol. 59, 2020, p. 101875.

[33] Yang, H., Chan, L., and King, I., "Support Vector Machine Regression for Volatile Stock Market Prediction," *Intelligent Data Engineering and Automated Learning — IDEAL 2002*, edited by H. Yin, N. Allinson, R. Freeman, J. Keane, and S. Hubbard, Springer Berlin Heidelberg, Berlin, Heidelberg, 2002, pp. 391–396.

[34] Sansom, D., Downs, T., and Saha, T., "Evaluation Of Support Vector Machine Based Forecasting Tool In Electricity Price Forecasting For Australian National Electricity Market Participants," *Journal of Electrical and Electronics Engineering, Australia*, Vol. 22, 2002.

[35] Lee, H., Li, G., Rai, A., and Chattopadhyay, A., "Real-time anomaly detection framework using a support vector regression for the safety monitoring of commercial aircraft," *Advanced Engineering Informatics*, Vol. 44, 2020, p. 101071.

[36] Farahani, H. V., and Rahimi, A., "Fault diagnosis of control moment gyroscope using optimized support vector machine," *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, IEEE, 2020, pp. 3111–3116.

[37] Yao, L., Fang, Z., Xiao, Y., Hou, J., and Fu, Z., "An Intelligent Fault Diagnosis Method for Lithium Battery Systems Based on Grid Search Support Vector Machine," *Energy*, Vol. 214, 2021, p. 118866. https://doi.org/https://doi.org/10.1016/j.energy.2020.118866, URL https://www.sciencedirect.com/science/article/pii/S0360544220319733.

[38] Gao, Y., Yang, T., Xing, N., and Xu, M., "Fault detection and diagnosis for spacecraft using principal component analysis and support vector machines," *2012 7th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, 2012, pp. 1984–1988. https://doi.org/10.1109/ICIEA.2012.6361054.

[39] Ibrahim, S. K., Ahmed, A., Zeidan, M. A. E., and Ziedan, I. E., "Machine Learning Techniques for Satellite Fault Diagnosis," *Ain Shams Engineering Journal*, Vol. 11, 2020, pp. 45–56. https://doi.org/10.1016/j.asej.2019.08.006.

[40] Fong, S., Luo, Z., and Yap, B. W., "Incremental learning algorithms for fast classification in data stream," *2013 International Symposium on Computational and Business Intelligence*, IEEE, 2013, pp. 186–190.

[41] Krempl, G., Žliobaite, I., Brzeziński, D., Hüllermeier, E., Last, M., Lemaire, V., Noack, T., Shaker, A., Sievi, S., Spiliopoulou, M., et al., "Open challenges for data stream mining research," *ACM SIGKDD explorations newsletter*, Vol. 16, No. 1, 2014, pp. 1–10.

[42] Cauwenberghs, G., and Poggio, T., "Incremental and decremental support vector machine learning," *Advances in neural information processing systems*, Vol. 13, 2000.

[43] Diehl, C. P., and Cauwenberghs, G., "SVM incremental learning, adaptation and optimization," *Proceedings of the International Joint Conference on Neural Networks, 2003.*, Vol. 4, IEEE, 2003, pp. 2685–2690.

[44] Liu, J., and Zio, E., "An adaptive online learning approach for Support Vector Regression: Online-SVR-FID," *Mechanical Systems and Signal Processing*, Vol. 76, 2016, pp. 796–809.

[45] Lawal, I. A., and Abdulkarim, S. A., "Adaptive SVM for data stream classification," *South African Computer Journal*, Vol. 29, No. 1, 2017, pp. 27–42.

[46] Ren, Y., Wang, X., and Zhang, C., "A Novel Fault Diagnosis Method Based on Improved Negative Selection Algorithm," *IEEE Transactions on Instrumentation and Measurement*, Vol. 70, 2021, pp. 1–8. https://doi.org/10.1109/TIM.2020.3031166.

[47] Brownlee, J., *Clever Algorithms: Nature-Inspired Programming Recipes*, 1st ed., Lulu.com, 2011.

[48] Singh, K., Kaur, L., and Maini, R., "A survey of intrusion detection techniques based on negative selection algorithm," *International Journal of System Assurance Engineering and Management*, 2022, pp. 1–11.

[49] Ji, Z., and Dasgupta, D., "V-detector: An efficient negative selection algorithm with "probably adequate" detector coverage," *Information sciences*, Vol. 179, No. 10, 2009, pp. 1390–1406.

[50] Garcia, D. F., Perez, A. E., Moncayo, H., Rivera, K., Betancur, Y., DuPuis, M., and Mueller, R. P., "Spacecraft heath monitoring using a biomimetic fault diagnosis scheme," American Institute of Aeronautics and Astronautics Inc., 2018, pp. 396–413. https://doi.org/10.2514/1.I010612.

[51] Davis, J., Perhinschi, M. G., and Moncayo, H., "Evolutionary Algorithm for Artificial-Immune-System-Based Failure-Detector Generation and Optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 33, No. 2, 2010, pp. 305–320. https://doi.org/10.2514/1.46126, URL https://doi.org/10.2514/1.46126.

[52] Dasgupta, D., Krishnakumar, K., Wong, D., and Berry, M., "Immunity-based aircraft fault detection system," *AIAA 1st Intelligent Systems Technical Conference*, 2004, p. 6277.

[53] Perhinschi, M. G., Moncayo, H., and Davis, J., "Integrated framework for artificial immunity-based aircraft failure detection, identification, and evaluation," *Journal of Aircraft*, Vol. 47, No. 6, 2010, pp. 1847–1859.

[54] De Castro, L., and Von Zuben, F., "Learning and Optimization Using the Clonal Selection Principle," *Evolutionary Computation, IEEE Transactions on*, Vol. 6, 2002, pp. 239 – 251. https://doi.org/10.1109/TEVC.2002.1011539.

[55] De Castro, L. N., and Von Zuben, F. J., "The clonal selection algorithm with engineering applications," *Proceedings of GECCO*, Vol. 2000, 2000, pp. 36–39.

[56] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al., "Human-level control through deep reinforcement learning," *nature*, Vol. 518, No. 7540, 2015, pp. 529–533.

[57] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D., "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[58] Si, J., Barto, A. G., Powell, W. B., and Wunsch, D., *Handbook of learning and approximate dynamic programming*, Vol. 2, John Wiley & Sons, 2004.

[59] Khan, S. G., Herrmann, G., Lewis, F. L., Pipe, T., and Melhuish, C., "Reinforcement learning and optimal adaptive control: An overview and implementation examples," *Annual reviews in control*, Vol. 36, No. 1, 2012, pp. 42–59.

[60] Sutton, R. S., and Barto, A. G., *Reinforcement learning: An introduction*, MIT press, 2018.

[61] Watkins, C. J., and Dayan, P., "Q-learning," *Machine learning*, Vol. 8, 1992, pp. 279–292.

[62] Arulkumaran, K., Deisenroth, M. P., Brundage, M., and Bharath, A. A., "A brief survey of deep reinforcement learning," *arXiv preprint arXiv:1708.05866*, 2017.

[63] Bate, R. R., Mueller, D. D., White, J. E., and Saylor, W. W., *Fundamentals of astrodynamics*, Courier Dover Publications, 2020.

[64] Vallado, D. A., *Fundamentals of astrodynamics and applications*, Vol. 12, Springer Science & Business Media, 2001.

[65] Hughes, P. C., *Spacecraft attitude dynamics*, Courier Corporation, 2012.

[66] Al-Zyoud, I.-D., and Khorasani, K., "Detection of actuator faults using a dynamic neural network for the attitude control subsystem of a satellite," *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, Vol. 3, 2005, pp. 1746–1751 vol. 3. https://doi.org/10.1109/IJCNN.2005.1556144.

[67] Gutierrez Martinez, T. A., "Health Management and Adaptive Control of Distributed Spacecraft Systems," Doctoral dissertations and master's theses, Embry-Riddle Aeronautical University, 2022. URL https://commons.erau.edu/edt/707.

[68] Lab, R., "Reaction Wheels," , n.d. URL https://www.rocketlabusa.com/assets/Uploads/RL-RW-1.0-Data-Sheet.pdf.

[69] Schaub, H., and Junkins, J. L., *Analytical mechanics of space systems*, Aiaa, 2003.

[70] Mizukami, M., and Nakazono, B., "Dawn Spacecraft Reaction Control System Flight Experience," *50th AIAA/ASME/SAE/ASEE Joint Propulsion Conference*, 2014, p. 3591.

PUBLICATIONS

Journals

- **Coulter, Nolan** and Hever Moncayo. "Data-Driven Approaches for Spacecraft Fault Detection Systems," AIAA Journal for Spacecraft and Rockets [To be submitted]

- Moncayo, Hever, William Engblom, Cindy Nshuti, and **Nolan Coulter**. "Performance Analysis of Flight Control Laws Applied to a Simplified Analog of the Dual-Aircraft Platform Concept." Journal of Aerospace Engineering 35, no. 4 (2022): 04022051.

- **Coulter, Nolan**, and Hever Moncayo. "Comparison of optimal and bioinspired adaptive control laws for spacecraft sloshing dynamics." Journal of Spacecraft and Rockets 57, no. 1 (2020): 12-32.

Conference Proceedings

- Mahsa, Jalali, **Nolan Coulter**, Hevery Moncayo, and Mehrdad Saif. "Emerging Data-Based and Model-Based Approaches for Fault Detection and Identification of Connected Vehicles." IEEE SMC. 2023

- Gutierrez, Tatiana, **Nolan Coulter**, Hever Moncayo, Yashwanth Kumar Nakka, Changrak Choi, Amir Rahmani, and Akshita Gupta. "Distributed Health Management for Resilient Multi-agent Collaborative Spacecraft Inspection." In AIAA SCITECH 2023 Forum, p. 0129. 2023.

- **Coulter, Nolan**, and Hever Moncayo. "Artificial immune system optimized support vector machine for satellite fault detection." In AIAA SCITECH 2022 Forum, p. 1713. 2022.

- Gutierrez, Tatiana, Andrei Cuenca, **Nolan Coulter**, Hever Moncayo, and Brock Steinfeldt. "Development of a Simulation Environment for Validation and Verification of Small UAS Operations." In AIAA SCITECH 2022 Forum, p. 0060. 2022.

- **Coulter, Nolan**, and Hever Moncayo. "An Online Machine Learning Paradigm for Spacecraft Fault Detection." In AIAA Scitech 2021 Forum, p. 1339. 2021.

- Verberne, Johannes, Yomary A. Betancur, Karina P. Rivera Lopez, **Nolan Coulter**, and Hever Moncayo. "Comparison of MRAC and L1 adaptive controllers for a gimbaled mini-free flyer." In AIAA SciTech 2019 Forum, p. 1290. 2019.

- **Coulter, Nolan**, Hever Moncayo, and William A. Engblom. "Development and demonstration of a flight simulator for the dual-aircraft platform concept." In 2018 Modeling and Simulation Technologies Conference, p. 3886. 2018.

- **Coulter, Nolan**, Hever Moncayo, and William A. Engblom. "Evaluation and Comparison of Sailing Flight Optimization Algorithms for a Stratospheric Dual Aircraft Platform Concept." In 2018 Modeling and Simulation Technologies Conference, p. 3887. 2018.

A AISOSVM Model-Base Engineeirng Views

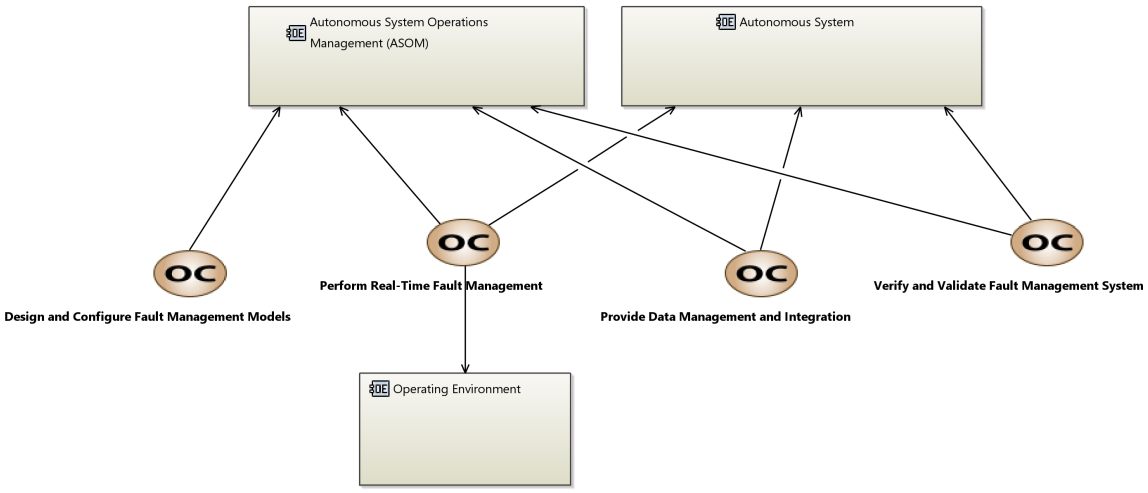


Figure A.1 Operational Capabilities diagram outlining fault management use cases.

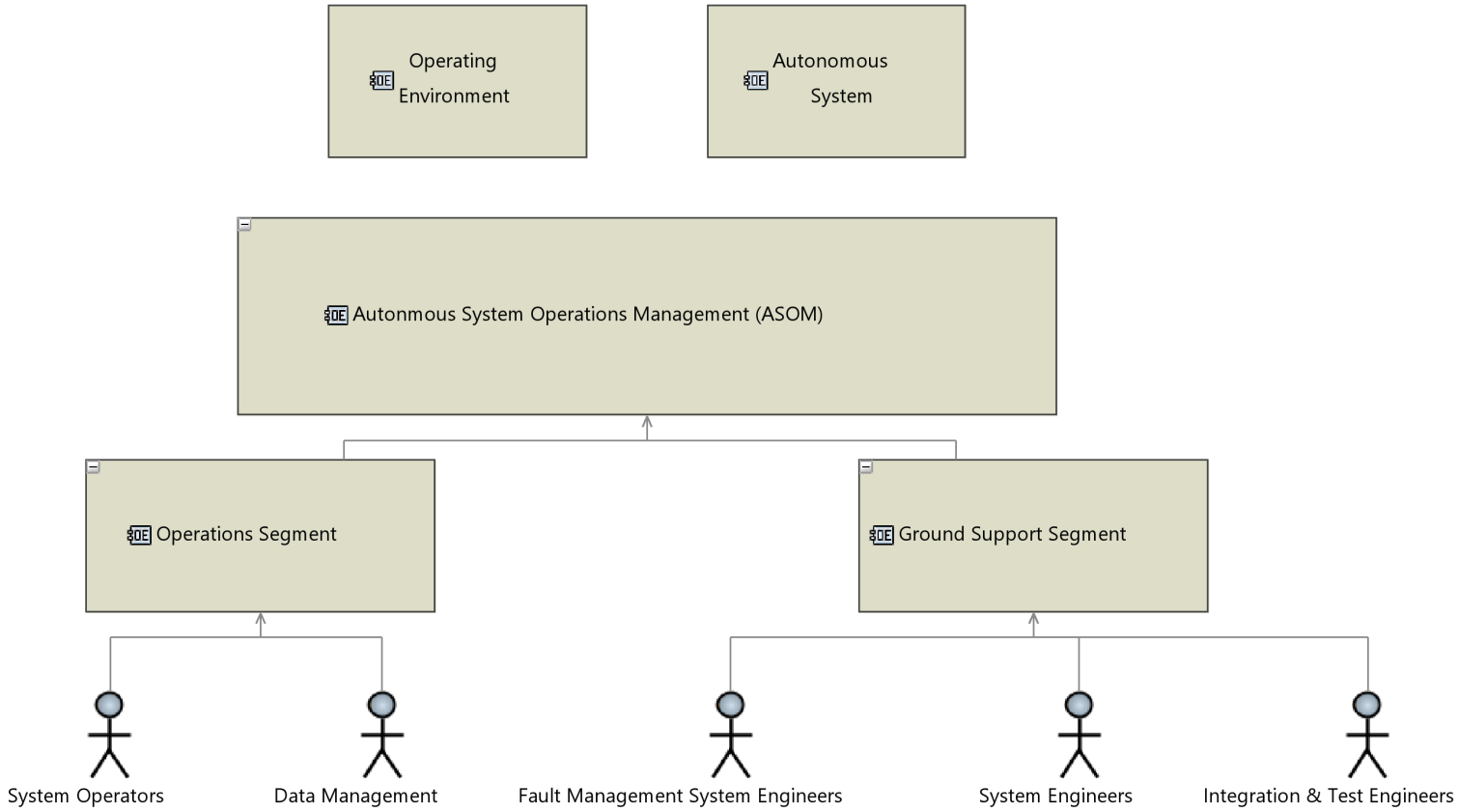


Figure A.2 Operational stakeholder breakdown for the fault management framework.

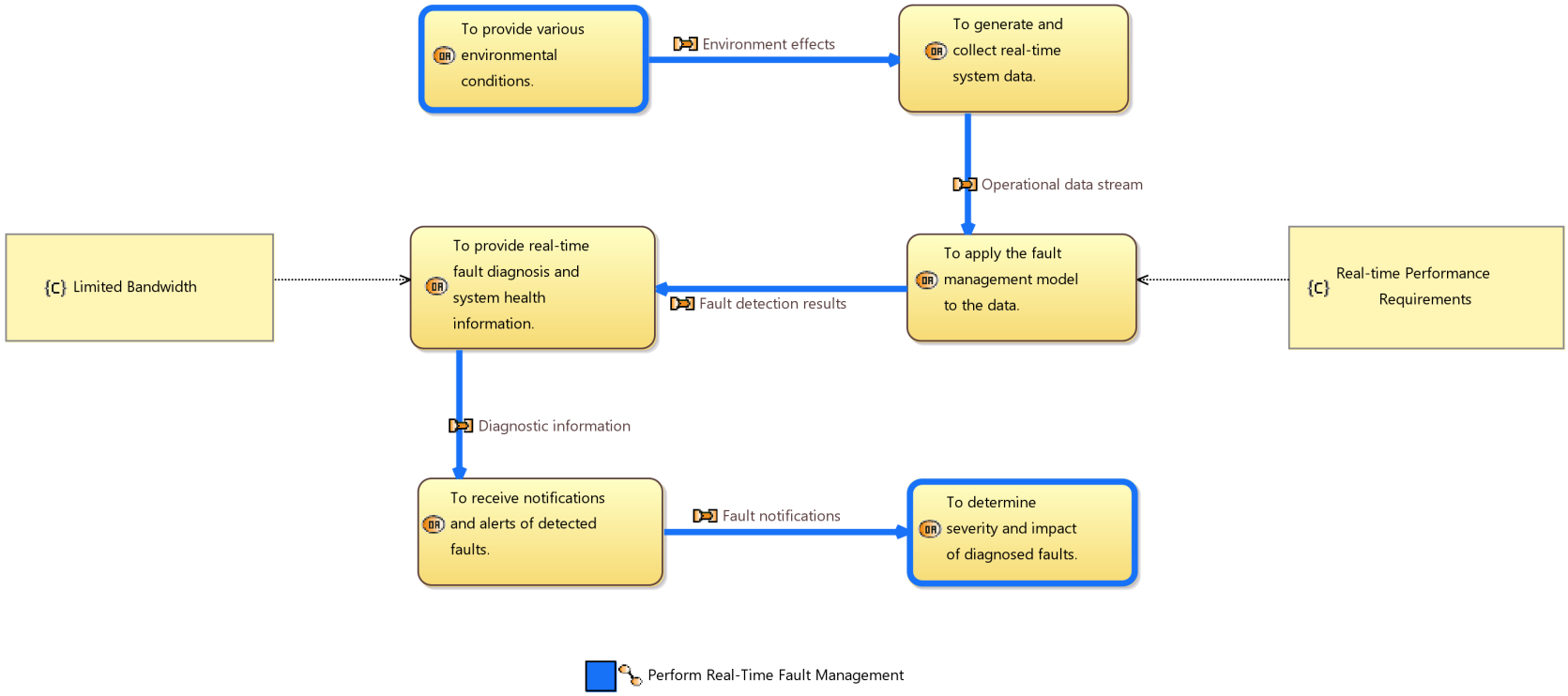


Figure A.3 Perform Real-Time Fault Management use case need statements.

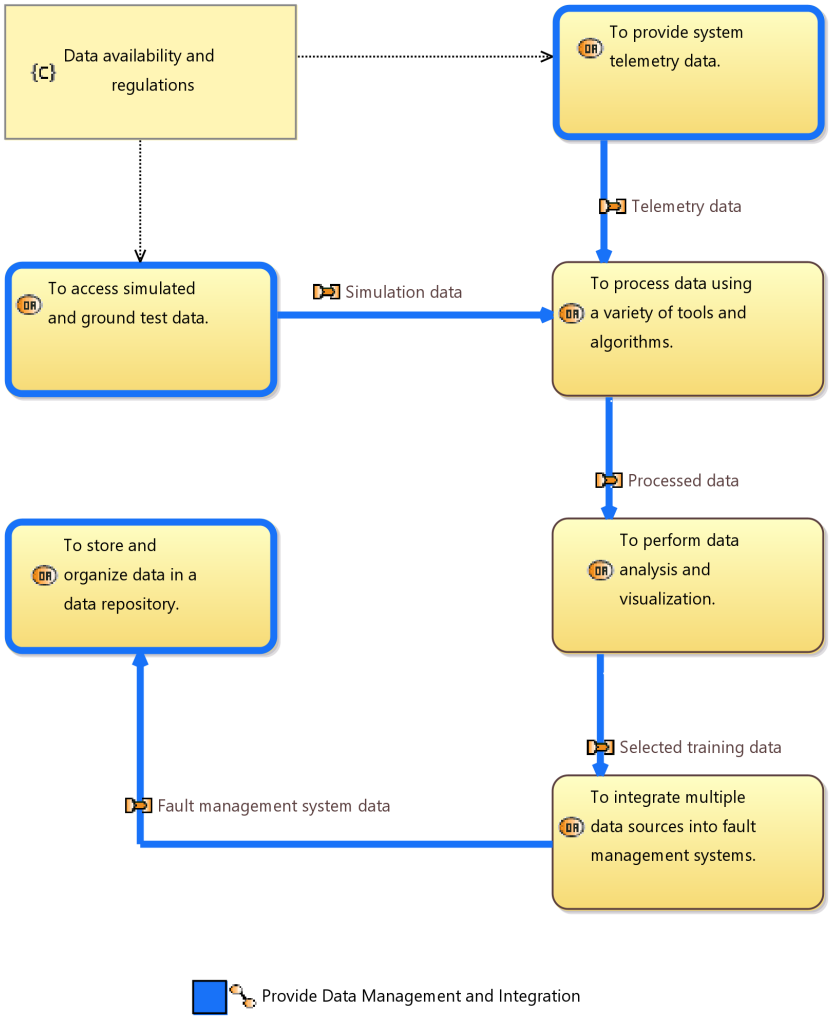


Figure A.4 Provide Data Management use case need statements.

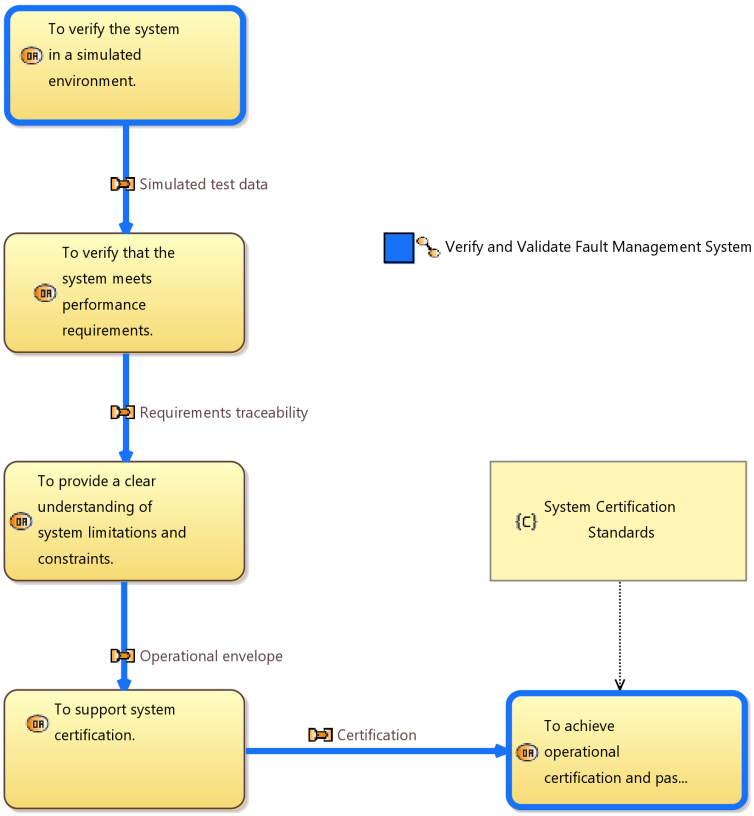


Figure A.5 Verify and Validate fault management models use case need statements.

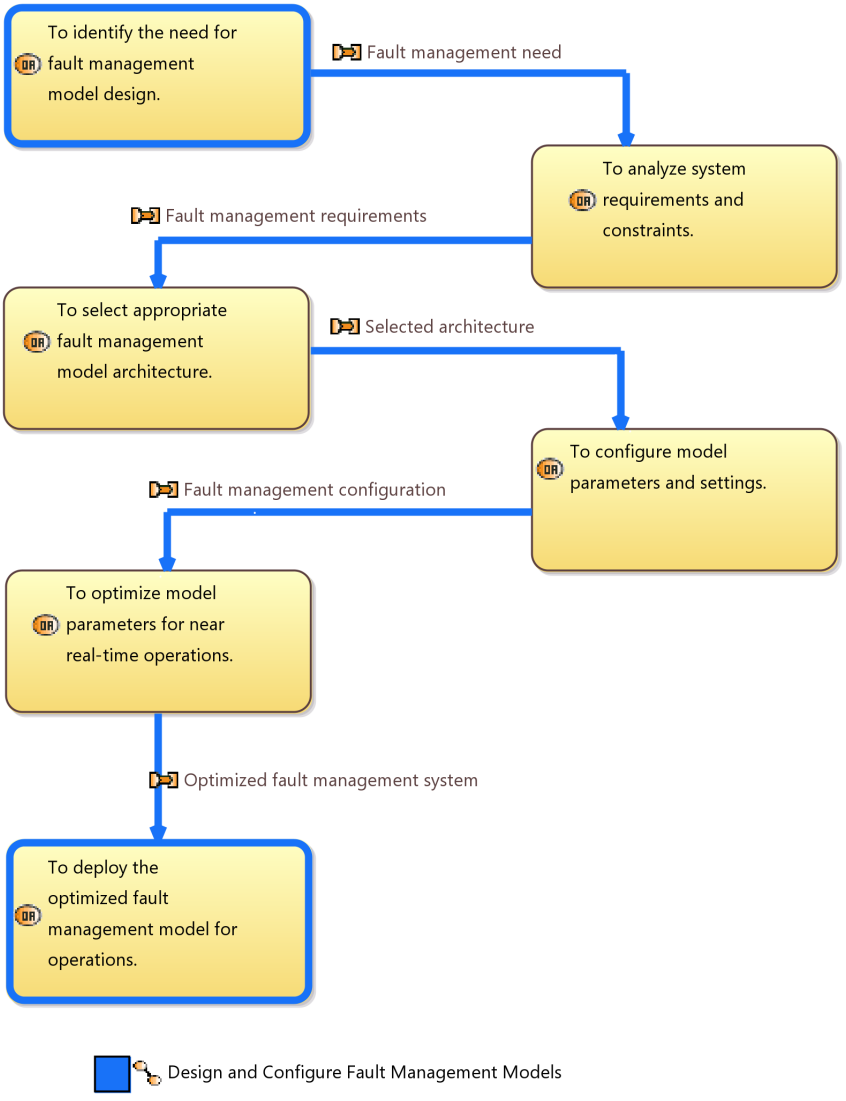


Figure A.6 Configure fault management models use case need statements.

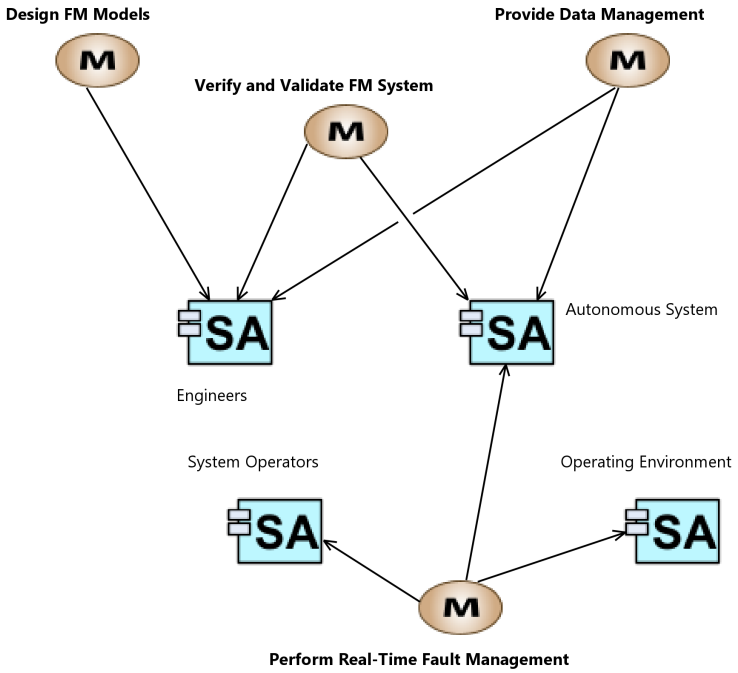


Figure A.7 System Missions Blank [MB] diagram.

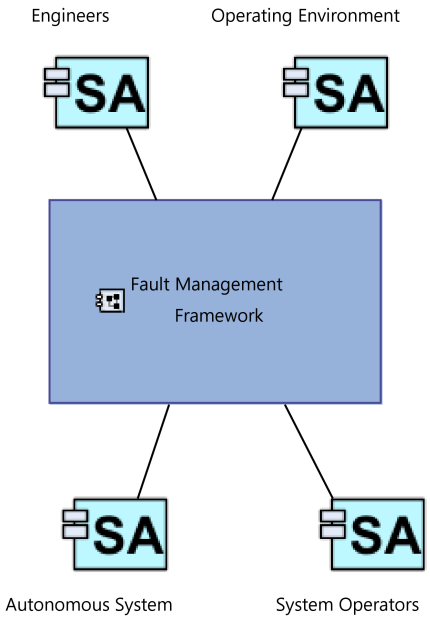


Figure A.8 System Actors for the Fault Management Framework System.

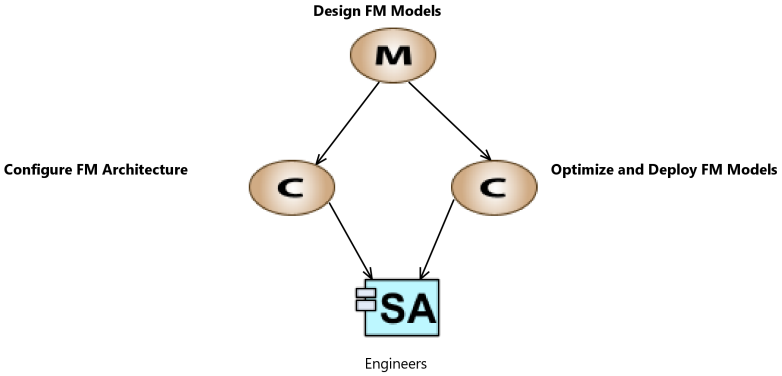


Figure A.9 Design Fault Management Models system capabilities diagram.

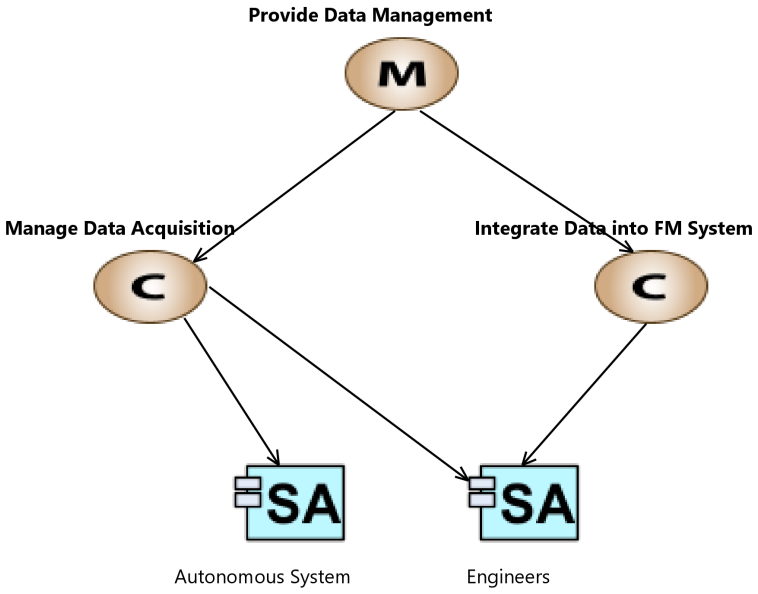


Figure A.10 Data Management system capabilities diagram.

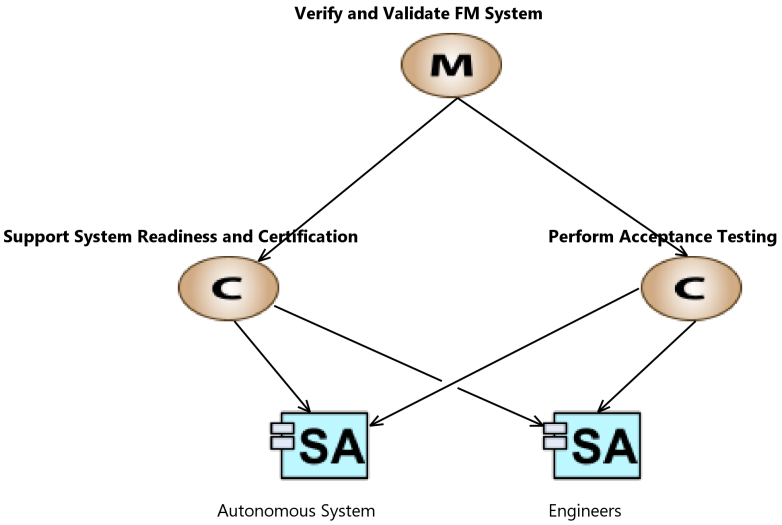


Figure A.11 Verify and Validate Fault Management system capabilities diagram.

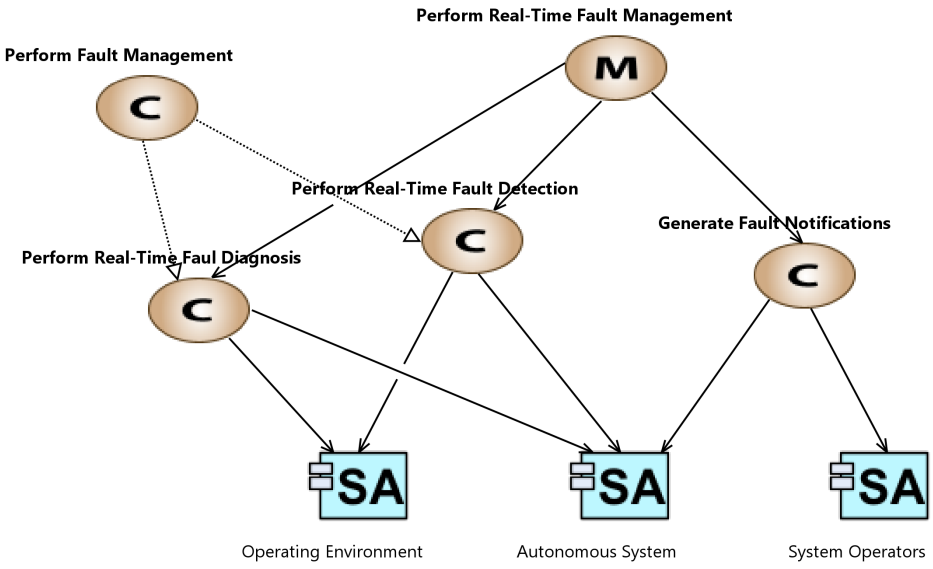


Figure A.12 Perform Real-Time Fault Management system capabilities diagram.

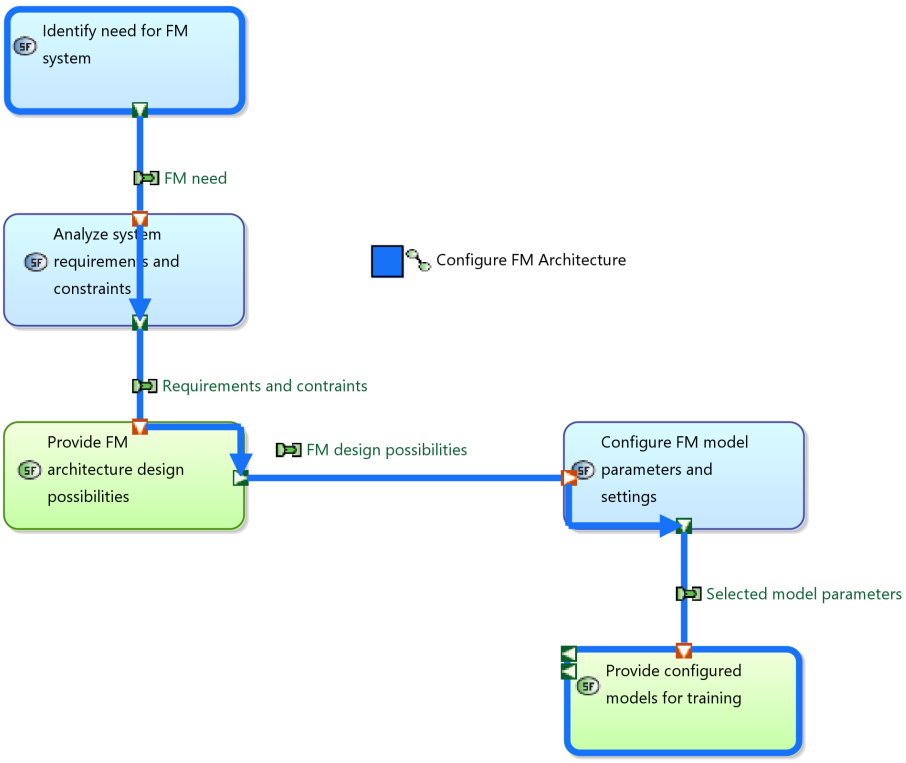


Figure A.13 Configure Fault Management Architecture system functions and data flow diagram.

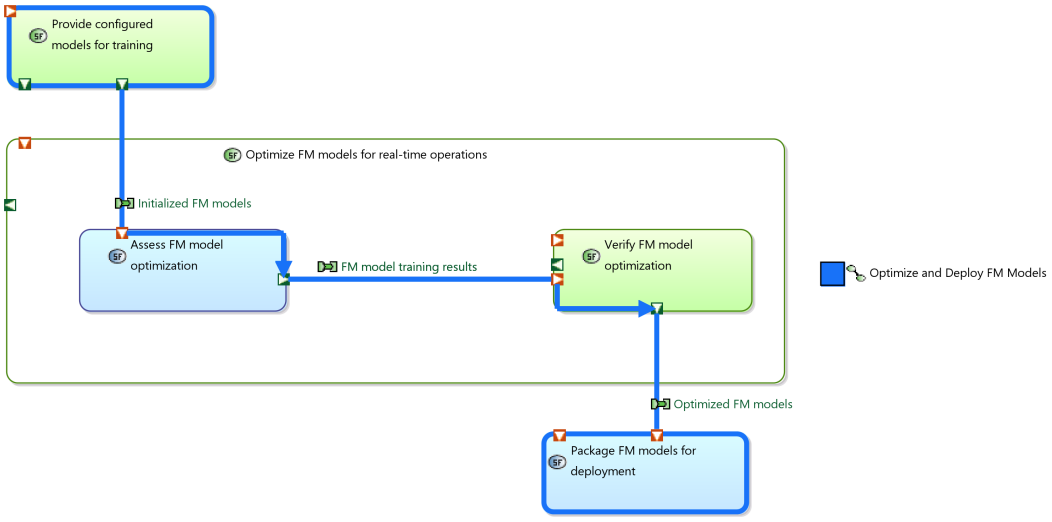


Figure A.14 Optimize and Deploy Models system functions and data flow diagram.

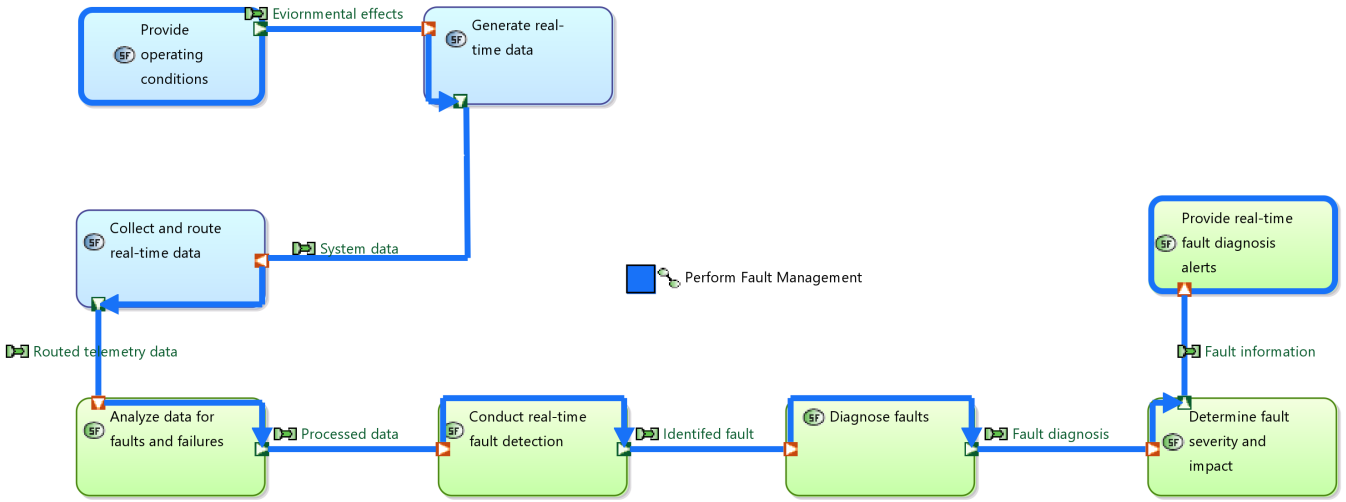


Figure A.15 Perform Fault Management system functions and data flow diagram.