Machine Learning and Coding Schemes for Fault and Cyber-attack Detection of Unmanned Aerial Vehicles

Edward Gharibian Saaki

A Thesis
in the Department of
Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of
Master of Applied Science

at Concordia University
Montréal, Québec, Canada

August 2024

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: **Edward Gharibian Saaki**

Entitled: **Machine Learning and Coding Schemes for Fault and Cyber-attack Detection of Unmanned Aerial Vehicles**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

______________________ Chair
Dr. Yousef R. Shayan

______________________ Examiner
Dr. Yousef R. Shayan

______________________ Examiner
Dr. Fariborz Haghighat

______________________ Thesis Supervisor
Dr. Khashayar Khorasani

______________________ Thesis Supervisor
N/A

Approved by ______________________
Dr. Wei-Ping Zhu, Associate Chair of Graduate Studies

__________ 2024

____
Dr. Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Machine Learning and Coding Schemes for Fault and Cyber-attack Detection of Unmanned Aerial Vehicles

Edward Gharibian

Abstract

The main objective of the thesis is to effectively detect and isolate faults and detect cyber-attacks and isolate faults from cyber-attacks for UAVs. In our first result, we leverage SVMs as a supervised machine-learning technique and Kalman filtering as a proven model-based method. We show that these methods can effectively detect and isolate faults and cyber-attacks. This thesis emphasizes the importance of hybrid approaches to fault and cyber-attacks SVMs perform equally well for the FDI cyber-attack detection on the control inputs of the ground control system. We propose to train two-class SVM only with the healthy dataset while adding a small bias to control the input of randomly selected observations. To isolate faults from cyber-attacks, we propose to use two χ^2, one on the plant side and the other on the controller side that reliably isolates faults from cyber-attacks. In our second result, we investigate the problem of stealthy caber-attack detection, where the attackers exploit full knowledge of the system to launch undetectable cyber-attacks. We study replay and covert cyber-attacks, and to make them detectable, we use coding techniques. We extend the two-way coding for MIMO discrete-time systems and study its properties using Kalman filtering for stealthy cyber-attack detection. We show that for an attacker to remove the cyber-attack impact from coded output, it is necessary to know the coding. We propose a new coding scheme using randomly generated numbers on the controller side that effectively converts coded inputs and outputs to time-varying random numbers.

Contents

List of Figures

List of Tables

Abbreviations

AE	Auto-Encoder
ANN	Autoassociative Neural Network
CNN	Convolutional Neural Networks
CPS	Cyber-Physical Systems
DL	Deep Learning
DoS	Denial-of-Service
FDI	False Data Injection
IID	Independent and Identically Distributed
LQG	Linear-Quadratic-Gaussian
LSTM	Long-Short Term Memory
LTI	Linear Time-Invariant
MIMO	Multi-Input Multi-Output
ML	Machine Learning
PCA	Principal Component Analysis
PLS	Partial List Square
RBF	Radial Basis Function
RNN	Recurrent Neural Networks
SCNN	Siamese Convolution Neural Network
SDA	Stacked Denoising Autoencoders
SISO	Single-Input Single-Output
SVM	Support Vector Machine
UAV	Unmanned Aerial Vehicles

Chapter 1

Introduction

1.1 Introduction

CYBER-PHYSICAL systems are the core of advanced manufacturing and industrial systems that integrate their physical functions within a virtual software environment. Cyber-Physical Systems (CPS) are a combination of an embedded hardware or physical system with deterministic and real-time behavior, with cloud systems that have a probabilistic behavior without strict time constraints [1].

CPS was identified as a national priority for R&D by many European countries, the United States, Japan, and some other countries due to the potential it offers to improve competitiveness and security in scientific and technological aspects [2]. The Canadian Research and Development Classification 2020 also recognized CPS as one of the Fields of Research, and the Natural Sciences and Engineering Research Council of Canada, supports many initiatives in CPSs. These facts grab the attention of scientific communities, and a considerable amount of research has been carried out in this area.

Industrial systems have long employed networked controlled systems as a part of the control loop to automate the process and improve productivity. These systems that integrate computer and networking systems to control and interact with the physical systems are known as Cyber-physical systems [3]. Examples of CPSs are smart electrical grids, intelligent transportation networks, process control systems, Unmanned Aerial Vehicles (UAV)s, etc. In recent years with the advent of the Internet technologies, many of these legacy systems are either connected to the Internet or planned to integrate with these new technologies [4], a movement that is largely referred to as Industry 4.0 [5, 6], and emergence of 5G network is anticipated to further accelerate this trend. Legacy industrial control systems generally have been designed with the assumption of operating in a secure and isolated environment, hence, no or less security measures are implemented in many of these systems that are now connected to the Internet. If for any reason, these systems need to take advantage of new and emerging technologies, they are potentially exposed to a host of cyber-attacks that might be able to exploit the vulnerabilities of these systems. A typical cyber-physical system is shown in Figure 1.1.

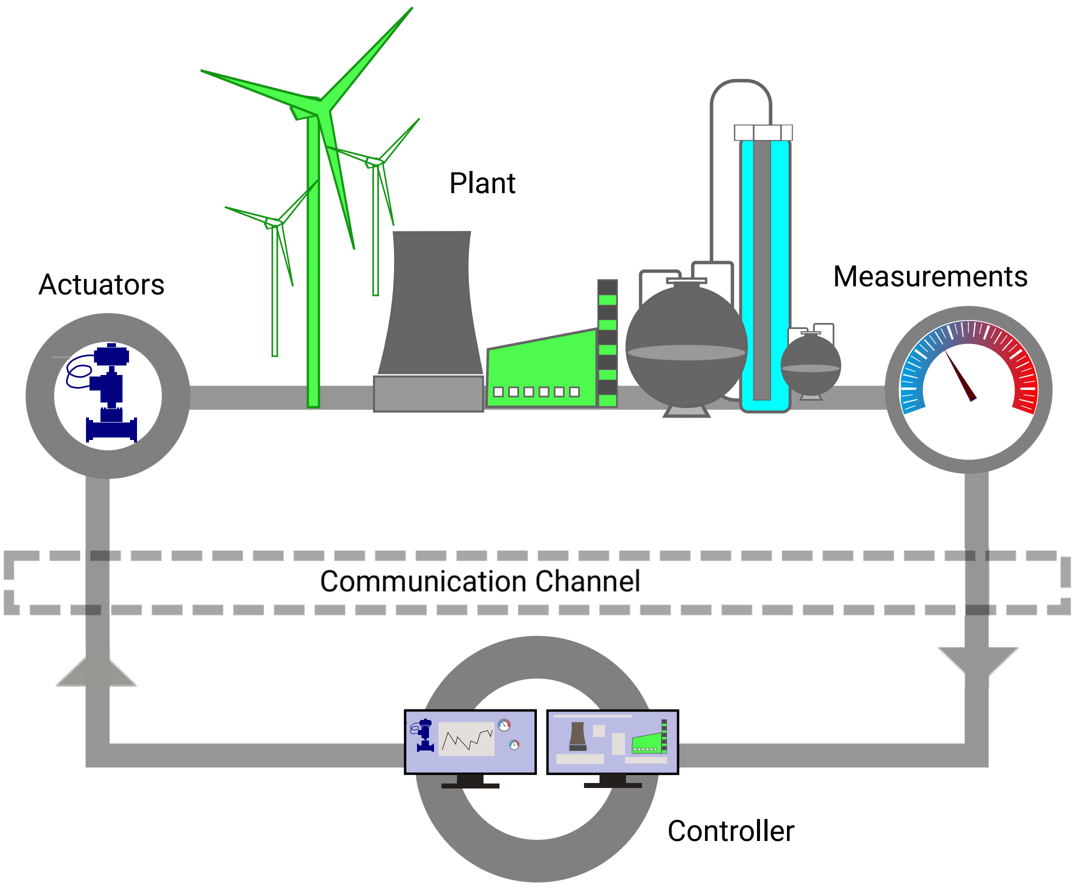


Figure 1.1: CPS control systems and cyberattack.

Future industrial systems will be connected to the Internet and will perform many tasks autonomously. However, before this futuristic view becomes a reality, several challenging issues including flexibility, robustness, fault tolerance, and resilience of these systems should be addressed.

Unmanned Aerial Vehicles (UAVs) are one of the most fascinating examples of CPSs. The UAV market has exponentially growth over the past several years and many industries are working to integrate them into their products and services. Therefore UAVs have become an attractive subject for both industry and the research community. However, due to its autonomous nature, cyber threats have been a major concern to UAV operations. The growing number of UAVs in civilian activities raises several security and privacy issues related to them. It is expected that UAV utilization will grow exponentially with a rate of 57.5% yearly up to 2028 [7]. Considering the increasing number of incidents related to UAVs, recorded daily all over the world [8], the subject of security and privacy of UAVs became more crucial.

1.2 Literature Review

Monitoring the condition of plants and equipment for detecting faults has long been challenging. Modern systems heavily rely on computer-controlled systems, and with the emergence of the so-called Industry 4.0 technologies, the vulnerabilities of those systems are now exposed to the Internet and, hence, to attackers. Proper operation of plants needs monitoring of several measurements of the plants and analysis of those measurements. However, measurement devices are prone to faults and cyber-attacks. This fact introduces a new challenge to run the systems safely and securely in the presence of faults and cyber-attacks.

Fault and cyber-attack detection and isolation methods can roughly divided into model-based, data-driven, and hybrid approaches. Model-based methods use a mathematical model of the plant to generate some analytical redundancy, which compares measurements of plants with computed values mainly from other measurements or a comparison of analytical values obtained from different sets of measurements. The resulting differences are known as residuals and indicate the presence of a fault or cyber-attack in the system. Following residual generation, some residual evaluation is required to detect, isolate, and identify the faults or cyber-attacks. Even for healthy real-life systems, there are always measurement noises and model uncertainties, which means residuals are never zero. Therefore, an empirical or theoretical threshold is necessary for deciding about the presence of faults or cyber-attacks in the system. It should be noted that, *Noise* may be removed to some extent by filtering or statistical methods, which requires statistical knowledge of noises, and *disturbance* may be handled by decoupling methods.

1.2.1 Model-Based Fault and Cyber-attack Detection

Industrial systems, specifically complex systems, have always been prone to faults. In recent years, cyber-attacks have also become a real threat. Traditional fault detection was basically visual and sound inspection, accompanied by limit checking of process variables, etc. Due to the limitation of these methods and reliance on experts, modern methods for fault diagnosis have emerged in the industry. Modern methods use advanced signal processing, control theory, and machine learning methods to analyze process measurements. The basic idea is to use plants' inputs and outputs to calculate expected values for a healthy plant and compare them with the actual states of the plant to find out about faults in components, measurements, and actuators or even cyber-attacks. Cyber-attack can happen if an attacker who gains access to plant communication channels manipulates the measurements or control commands to harm the system. A knowledgeable attacker may design sophisticated cyber-attacks that bypass fault diagnosis systems and result in system failure. Among the well-studied fault and cyber-attack detection methods are observer-based methods, where the model of the system is used to generate residuals. These residuals are used to detect and isolate different types of faults and cyber-attacks.

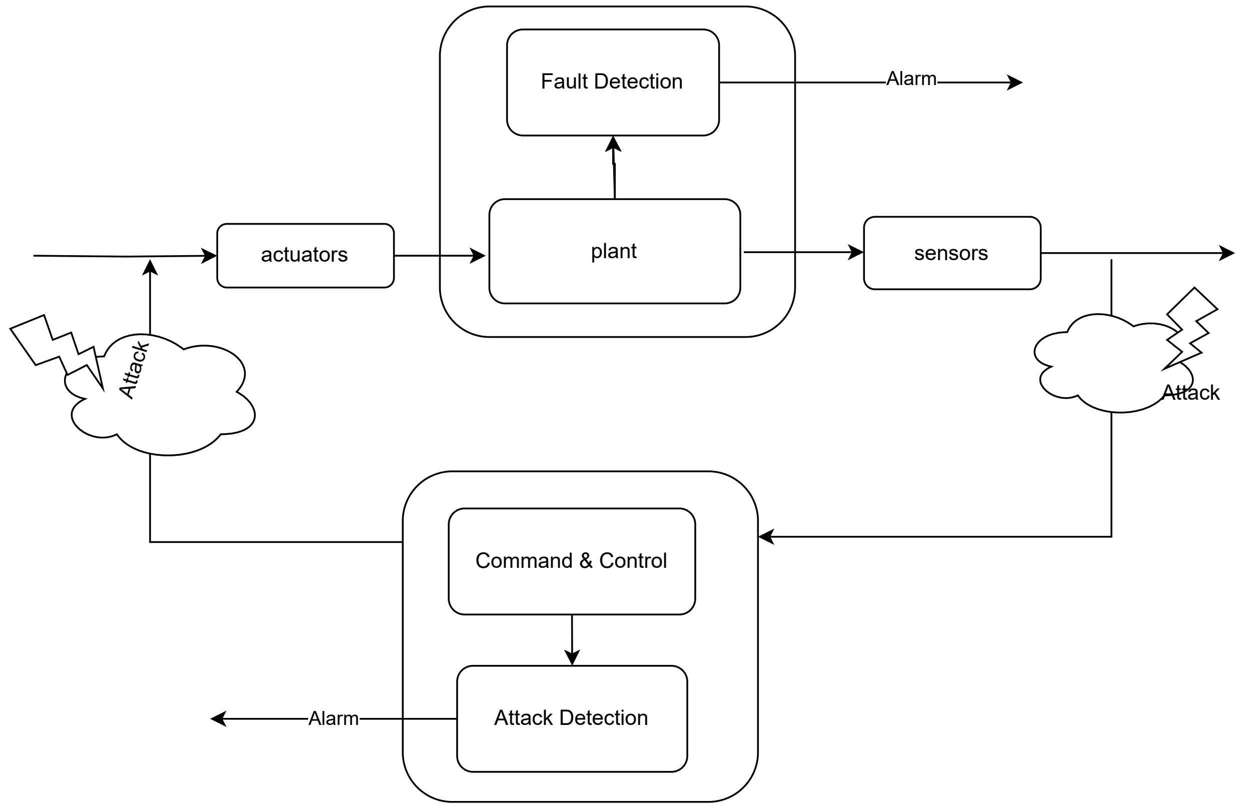


Figure 1.2: Cyber-attack detection in CPS.

Fault diagnosis of linear systems are well studied and developed in the literature. A linear system model can be represented as:

$$\begin{cases} \dot{x}(t) &= Ax(t) + Bu(t) \\ y &= Cx(t) \end{cases} \tag{1.1}$$

where x, u and y are states, inputs and outputs of the system and A, B and C are matrices of appropriate size. Residual generation is the most crucial stage of model-based fault and cyber-attack diagnosis methods. There is a wealth of literature dealing with residual generation, and various methods are proposed by researchers. Although all the methods can be adopted for discrete-time systems, it is not the case for continuous-time systems, and for example, parity relation methods are specifically designed for discrete-time systems. A linear system with fault in sensors f_s, actuators f_a and components of system f_c, is represented as below:

$$\begin{cases} \dot{x}(t) &= Ax(t) + Bu(t) + Ef_a(t) + Ff_c(t) \\ y &= Cx(t) + D_1 f_a(t) + D_2 f_s(t) \end{cases} \tag{1.2}$$

The cyber-attack and fault signals are assumed to be additive. The most important Residual generation methods with some overlap include:

1. *Kalman filtering.* The innovation or prediction error of the Kalman filter can be used as residual, which became non-zero in the presence of fault (and disturbances). Construction of residual with this method is relatively easy, but fault isolation needs a running bank of filters, and each arrival time to detect the fault.
2. *Observer* innovation is another commonly used method. Unknown input techniques may be used to decouple faults from some disturbances. In this approach, the residuals are colored and so this makes statistical methods application complicated.
3. *Parity* check generates residual vector by checking the consistency of the model with the process and can be applied to both time-domain and frequency-domain models.
4. *Parameter estimation* checks system parameters against a model of the system for the healthy situation, and deviations from the reference model, which are a type of residual, are used for the detection and isolation of faults.

The diagram in Figure 1.3 shows model-based fault diagnosis and isolation concept.

Observer-Based Methods

Observer-based Methods, also known as estimators or filters, are based on the idea of estimating the output of the system using available measurements generally though a full-order Luenberger-like observer [9–13]. Weight least square has been used widely for detection of faults, specifically for power grid state estimator [11, 12], however, it is shown that it will come short to detect intelligently designed cyber-attacks [9, 11, 12]. An example is bad data detectors in power grid which can be bypassed by such cyber-attacks.

Observer-based methods have been widely studied in literature to extend them so as to detect intelligently injected cyber-attacks. In [9] a monitor is designed that extends the Lu-

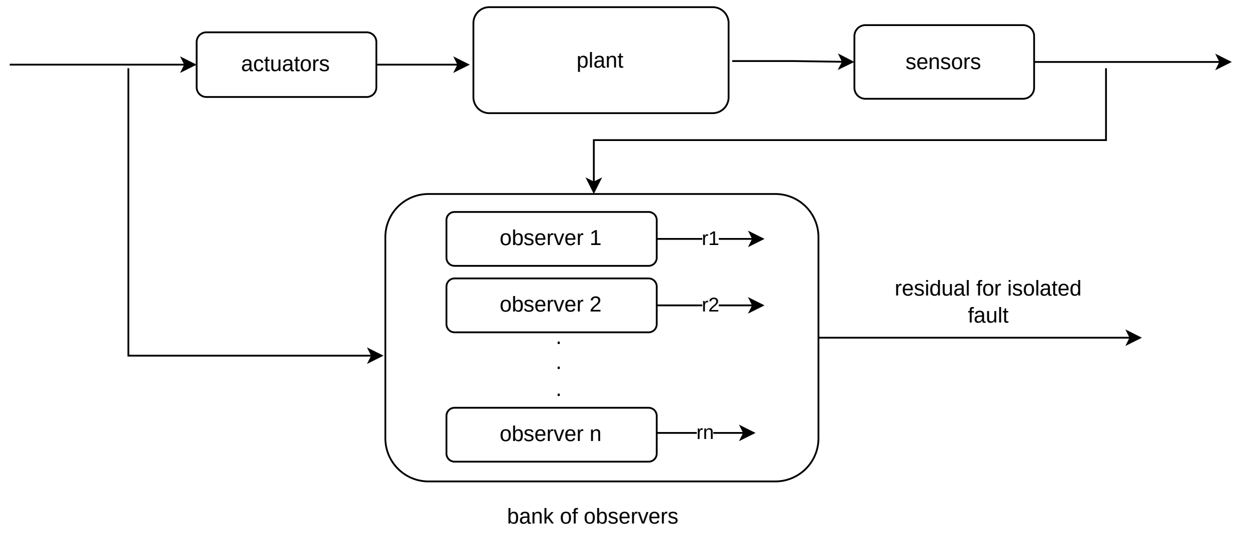


Figure 1.3: Model-based fault diagnosis and isolation.

enberger observer to detect cyber-attacks on a power grid in a centralized control room and in [10] researchers extended the method for distributed estimator in DC-micro-grid. In another research in [14], the researchers developed a Luenberger-like observer for the sparse sensor in the presence of noise and to improve the performance of the proposed algorithm, they used event-triggered approaches. Kalman filter was also applied to fault and cyber-attack detection problems for stochastic systems. In [15] necessary and sufficient condition where an attacker can destabilize a system equipped with a Kalman filter-based fault detection is studied. It is shown that for an Linear Time-Invariant (LTI) system, the controller can be attacked if the A matrix has unstable eigenvalues which are related to unobservable states of the system.

Kalman Filter

To evaluate the residuals generated by the Kalman filter, various methods have been proposed. The χ^2 detector [16] is the most notable and widely adopted method for detecting cyber-attacks and faults. This detector uses the statistical behavior of states to make decisions about the presence of faults or cyber-attacks and has been used to detect faults and false data injection cyber-attacks. In [15] researchers designed an Linear-Quadratic-Gaussian (LQG) controller and used χ^2 detector to detect False Data Injection (FDI) cyber-attacks. Authors of [17] showed that χ^2 detector could be used for detection of both faults and cyber-attacks, including Denial-of-Service (DoS) cyber-attacks, and FDI cyber-attacks, but it will fall short of detecting cyber-attacks injected to measurement when the cyber-attack signal complies with the statistical distribution of measured data. To detect this type of cyber-attack the researchers proposed a Euclidean detector. The Euclidean detector they developed, reconstructs states of the system and compares it with measurements, however, it is more resource intensive, is more sensitive to noise, and performs well when the noise parameters are known, however for unknown noise parameters, χ^2 has better performance. In [18] it is shown that the Kalman filter cannot detect cyber-attacks that have the same statistical properties as system noise. In [19] authors developed cyber-attack detection and secure state estimator for power grid based on Kalman filter.

Unknown Input Observer

Another commonly used method for residual generation is the unknown input observer (UIO), which is a special type of observer that generates robust and directional residual for detection and isolation of faults [20]. UIOs have been used to detect faults in wind turbines [21], gas turbines [22] and power systems [23]. It has also been used for cyber-attack detection of CPS's [24, 25]. A popular method to design UIO is based on linear matrix inequality (LMI) due to capabilities for dynamics systems [26]. In this approach, A bank of observers is designed for the isolation of faults and cyber-attacks.

Parity-based Methods

Parity space-based residual generation is another well-studied approach for fault and cyber-attack detection. In this method, residuals are called parity vectors. To use the parity method, the LTI system model needs to be written in discrete time. The parity approach uses past measurements to generate the residual vector, and its main advantage is its straightforward procedure [27, 28]. The authors of [29] proposed a fault diagnosis method for linear discrete systems based on a minimum error maximum probability machine (MEMPM) to improve false alarm rate while keeping high detection rate. In [30] authors used a parity-based approach to detect false data injection cyber-attacks only using local data on the DC microgrid. They formulated a multi-objective optimization problem to make detection robust to the variation of parameters in the grid.

1.2.2 Data-Driven Fault and Cyber-attack Detection

Model-based fault and cyber-attack detection requires an accurate model of the process that is not always available. Generally, process model development requires high expertise and costly experiments. On the other hand, the emergence of computer and IT technologies in industrial systems results in the availability of large amounts of industrial process data, which makes data-driven fault and cyber-attack detection a viable alternative. Data-driven or Machine Learning (ML) methods either use deep learning or statistical methods to detect deviation of a process from its normal operating conditions.

The deviation is also known as an anomaly in ML literature. Anomaly refers to the point, observation, or events in data that do not comply with normal or expected behavior, and the anomaly detection problem is to find these patterns in data. These data that do not conform with the rest of the data are also known as outliers, discordant observations, exceptions, aberrations, surprises, peculiarities, or contaminants in different application domains [31]. Anomaly detection is used in a variety of disciplines and applications such as fraud detection in financial applications, intrusion detection, fault, and cyber-attack detection in industrial and military systems, etc.

Most machine learning-based anomaly detection methods were developed during past decades, and were used in diverse research fields and application domains. Most detection techniques are developed for some specific application domains, but there are also more generic approaches.

Many factors might result in anomalies, including noise, and novelty, also known as unobserved data. However, fault and malicious activity are the main focus of this thesis. There are several challenges in finding anomalies in data including:

- Defining a normal region of data that reflects the whole normal condition is difficult, and most of the time there is no clear boundary between normal and abnormal regions.
- Malicious activities are generated by humans and try to imitate normal behavior to remain stealthy.
- Normal behavior may evolve by system input and over time.
- There might not be a clear definition for anomaly behavior in many disciplines.
- Availability of labeled data without noise also is not always possible.

Anomaly detection is not a trivial task, due to several challenges mentioned above. Anomaly detection techniques are categorized as classification-based, cluster-based, nearest neighbor-based, statistical, information theory, and spectral. Many techniques are focused on the special application domain or a single research area. Application area of anomaly detection ranges from cyber-intrusion and fraud detection to medical, image processing, industrial systems, and sensor network fault detection [31].

Types of Anomaly: An Anomaly detection problem has several different aspects, and the nature of the data itself is the most important one. The type of anomaly is another important aspect of detection. Anomalies can be classified as *point anomalies*, *contextual anomalies* and *collective anomalies* [31].

Point Anomaly is the simplest type of anomaly and refers to the case where an individual data is not conforming with the rest of the data. If the anomaly is just in a specific situation and is acceptable in other situations, it is referred to as *Contextual Anomaly*, or *Conditional Anomaly* [32]. For industrial systems, for instance, the time or environmental condition can be viewed as the context of data. An example of such an anomaly is the observation of a high temperature in records of environment temperature in winter, which can be considered normal in summertime. *Collective Anomaly* refers to the collection of data that are normal individually, but the occurrence of them together is considered an anomaly.

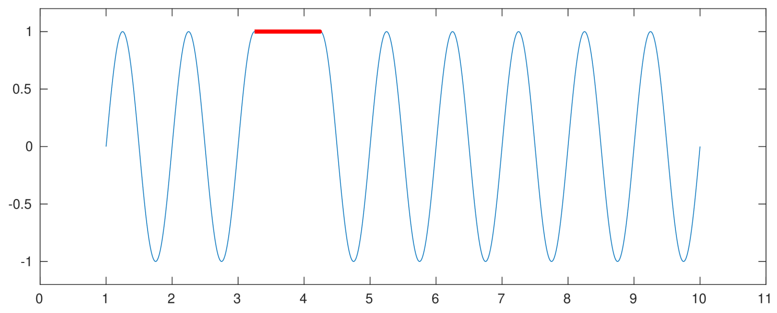


Figure 1.4: Collective anomaly.

In Figure 1.4, the bold part of the graph is considered as anomaly.

Labeled Data: The availability of labeled data is decisive in choosing of a detection method.

Supervised Anomaly Detection is practical if enough labeled data is available for both the normal and the anomalous conditions. This technique is like predictive modeling. Labeled data for normal operating conditions is often available, but it is unlikely to obtain anomalous labeled data for all behaviors. Some research uses a simulator or a computer model to train the detector to learn normal conditions. Also, there are techniques for introducing artificially generated anomalies in the normal conditions data.

Semi-supervised Anomaly Detection techniques assume only normal operating condition data is available. This approach attracted some researchers specifically for fault detection [33].

Unsupervised Anomaly Detection implicitly assumes normal data are far more frequent than anomalous data and makes the detection based on this assumption.

The most notable data-driven fault and cyber-attack detection approaches are statistical and deep learning methods. The most commonly used statistical methods in the literature for fault and cyber-attack detection include Principal Component Analysis (PCA), Partial List Square (PLS), and Support Vector Machine (SVM) [34]. The most significant deep learning methods in the literature are different variants of neural networks like Convolutional Neural Networks (CNN), Auto-Encoder (AE) Neural Networks, and Recurrent Neural Networks (RNN) [35–37].

Statistical Methods

Statistical methods have been widely used for fault and cyber-attack detection. These methods can offer excellent performance even for complex industrial systems.

PCA-based Methods: PCA projects a high dimensional data set into a low dimensional best linear manifold and has the ability to describe information variation in a dataset. This capability results in the adoption of PCA in a wide range of applications, including image processing, pattern recognition, and processes monitoring [38]. This property makes PCA one of the most widely used statistical monitoring tools, which has been successfully used for fault diagnosis of complex industrial systems [34]. Research on nonlinear PCA (NLPCA), receives attention in 1980 and can be achieved by kernel techniques, principal manifolds, Autoassociative Neural Network (ANN), or a combination of these methods. The authors of [39] developed an NLPAC method for fault diagnosis of car engines. They used ANN with three hidden layers, with a hyperbolic Lagrange activation function. They introduced a statistical local approach for PCA to detect faults. In [40], a PCA-based hybrid method is proposed for fault diagnosis of rolling bearings. They used PCA to transform original noisy data to principal components with improved noise-to-signal ratio and then used a cyclic bispectrum for detection. The authors of [41] studied the performance of PCA and SVM methods and found that in the case of the Tennessee-Eastman simulator, PCA performed better due to SVM multi-class classification difficulties.

PLS-based Methods: Although Partial least square (PLS) is not used extensively for cyber-attack detection, it is quite a popular tool for fault detection of industrial systems. Authors in [42] developed a schema for key performance indicators for diagnosis, and showed it is effective for industrial processes, while it has a simpler procedure to implement compared with the standard PLS. In [43] PLS is used for fault detection and online monitoring for a linearized industrial process and can detect actuator and sensor faults based on statistical indices. Authors in [44], used a kernel partial least squares based on an optimal preference matrix. Considering the excellent nonlinear handling of this kernel, the authors proposed a fault detection method for the aluminum electrolytic process.

Support Vector Machines (SVM): is a relatively new classification method, that due to nonlinear capabilities and dealing with non-Gaussian noise became popular in all types of classifications. SVM has good generalization properties and is capable of generating good models, from small samples, which makes it a proper choice for process monitoring and fault and cyber-attack detection in industrial systems [26, 45]. Authors in [46] reached a superior classification of faults with SVM compared with conventional PLS for the Tennessee-Eastman process. SVM has been combined with other methods for parameter tuning, and in [47] authors used a genetic algorithm to improve SVM tuning for a continuous stirred tank reactor and a heat exchanger. This paper achieved superior fault detection performance compared with their selected neural network classifier. In [48] authors used a distributed SVM cyber-attack detection for stealthy false data injection cyber-attack on the smart grid state estimator, and the test results on IEEE 118-bus test system showed the effectiveness of the method. In [49], GPS spoofing detection is studied for UAVs, and a method based on SVM for detecting changes in GPS signal characteristics as a feature for detection is proposed. In [50] a one-class SVM is proposed for anomaly detection in IoTs and combined with clustering and Gaussian mixture models to improve computational speed. Authors of [51] used a SVM based method for a multi-agent smart grid, in a decentralized schema to mitigate DoS cyber-attacks on load shedding, that each agent makes its own decision using anomaly detector, while final detection is done based on the consensus of all agents.

Deep Learning Based Methods

Artificial neural networks are among the best-established non-statistical methods for fault and cyber-attack detection, which according to the learning strategy employed, can be categorized as supervised or unsupervised methods. Artificial neural networks has been applied to different fault and cyber-attack detection problems, however, in the past decade the researchers have tended more toward deep neural network methods [52]. Deep Learning (DL) techniques have been extensively used for classification tasks such as image and voice recognition and are also adopted to detect faults and cyber-attacks on industrial systems. A major advantage of DL methods is the ability to automatically extract features from unlabeled data without the help of experts so that all parameters of the model including features, classification model or regression model can be trained together [36]. Among important DL techniques used for machine health monitoring systems are Auto-Encoder and its variants, Convolutional Neural Networks and Recurrent Neural Networks.

Convolutional Neural Networks (CNN) have been successfully applied in computer vision, image recognition, and pattern recognition. CNN has also been used for fault and cyber-attack detection in industrial systems. In [53] researchers used a convolutional neural network-based method to extract features from the dataset to eliminate handcraft feature selection, they proposed a method to convert signals to image (2D data) and then processed these images for fault detection. A similar approach of converting one-dimensional vibration signals to 2D images is used by [54], and then a CNN is used for gearbox fault diagnosis. In [55], dislocated time series convolutional neural networks are introduced to handle the variability of the vibration signals of electrical motors, 1D vibration signals are converted to a matrix and then a deep CNN is applied. In cyber-attack detection, authors of [56] used CNN to detect abnormality in measurements in long-period observations of smart grids to detect energy theft. In [57] a CNN-based intrusion detection method for a network of vehicles is introduced, that learns network traffic patterns and can detect malicious traffic. Authors of [58] proposed a hybrid method based on Kalman filtering and convolutional neural networks for cyber-attack detection on CPSs. They process data in three stages, first using a Gaussian mixture model that combines various data features of a physical system, then a Siamese Convolution Neural Network (SCNN) is used to detect and remove anomalies in data, and then, these data are fed to Kalman filter for further analysis and finding anomalies missed by SCNN.

Auto-Encoder Neural Networks are unsupervised learning methods comprising two phases known as encoder and decoder. The encoder transforms input data to hidden representation by nonlinear mapping, and the decoder maps hidden representation back to the original form. AEs are capable of removing noises while capturing features in data [36]. AE is used for fault and cyber-attack detection of CPSs. Authors of [59] used an AE model with one hidden layer for automatic feature extraction and removing noises from data for an induction motor. To overcome the problem of overfitting for a small dataset used in the research, they applied the dropout technique to data during training, which means randomly dropping neurons and relevant connections to prevent co-adaptation of the neurons. Authors of [60] used a similar approach with three hidden layers using Stacked Denoising Autoencoders (SDA) for fault diagnosis. In general, SDA is one of the popular approaches for fault diagnosis. Stacked autoencoder has also been used for cyber-attack detection, in [61] where false data injection cyber-attack on smart grid state estimator is studied, and an SDA method is proposed as a defense mechanism for state estimation. Authors of [62] studied a lightweight neural network known as 1D convolutional neural networks and autoencoders to study cyber-attacks on industrial control systems. They used public datasets and compared the performance of four data-driven detection methods, namely 1D CNN, shallow undercomplete autoencoders, variational autoencoders, and PCA for both time and frequency domain data, and concluded that both 1D CNN and AE based method has excellent performance. It should be noted that

AE-based methods generally work out of the box, while CNN needs more parameter tuning. They also found that if proper data preparation and feature selection are used, PCA-based methods will perform effectively. Autoencoders with Long-Short Term Memory (LSTM) based methods have also been applied to cyber-attack detection of autonomous vehicles [63]. LSTM is an extension to recurrent neural networks, that is proper for chronological sequences.

Recurrent Neural Networks (RNN) are the deepest type of neural networks and are more powerful than CNN [64]. RNN's different variant is used for a variety of anomaly detection applications including fault and cyber-attack detection in industrial systems. In [65] authors studied aircraft turbofan engines and actuator faults. They used a public dataset and labeled data using an SVM detector and later used it for the training of neural networks. They used a long-short-term memory (LSTM) neural network and compared the results with vanilla RNN, simplified LSTM, and AdaBoost LSTM, and concluded the standard LSTM-based RNN performs the best. Most DL-based methods rely on automatic feature extraction, while authors of [66] proposed a hybrid feature selection that combines handcrafted features with automatic feature extraction. They first extract features as windows of time frames, and then a neural network is applied to learn the representation of features, and finally, a supervised training layer is added to predict machine conditions. LSTM-based RNN is also used extensively for cyber-attack detection of CPSs. In [67] an LSTM-based RNN is developed for zero-day (unknown) cyber-attacks by analyzing both network packets and temporal packet behavior of the control system of a gas pipeline. They took advantage of a regular pattern of inter-device communication to generate a baseline signature and developed an LSTM to predict the next likely packet signature based on previous ones. The researchers used the SCADA dataset and showed the effectiveness of the proposed model. Authors of [68] used vanilla RNN for cyber-attack detection on smart grids. They used three datasets that contain different types of cyber-attacks including DoS and web cyber-attacks and showed that RNN offers good performance in detecting different types of cyber-attack across different datasets. Automatic dependent surveillance-broadcast is a technology aiming for safe navigation of airplanes and air traffic control management, that due to lack of security mechanism is prone to cyber-attack, in [69] an LSTM algorithm is proposed to detect anomalies between aircraft and ground control tower.

1.2.3 Defense Against Stealthy Cyber-attack

Simple cyber-attacks such as FDI and DoS cyber-attacks, can be effectively detected with data-driven and model-based methods. However, both of these methods suffer from the same fundamental limitations. This means that a sophisticated attacker with knowledge of the system and access to the communication channel can design a cyber-attack to remain stealthy. Cyber-attacks like zero-dynamics, covert cyber-attacks, and replay cyber-attacks generally cannot be detected with conventional fault detectors described so far. For that reason, researchers have adopted different approaches for detecting stealthy cyber-attacks.

Active detection methods have been introduced to improve detection of stealthy cyber-attacks. In active methods, a physical secret unknown to the attacker is introduced in the system as an authentication mechanism [70]. Examples of active methods are moving targets and watermarking. Watermarking has been widely used for replay cyber-attack detection by embedding digital signatures on signals that are used to evaluate the authenticity of received signals [71]. In a control system, this is achieved by adding a signal, unknown to the attacker to control output $u^*(k)$ in the form of $u(k) = u^*(k) + \Delta u(k)$. The effect of the watermark can then be observed in measurements, which, in the case of a replay cyber-attack, the response will belong to a previous sequence of watermarks, and hence the replay cyber-attack will be detected. In [72], researchers introduced a watermark in the form of a random sequence $\Delta u(k) \sim \mathcal{N}(0, \mathcal{F})$ added to controller output $u^*(k)$. However, this results in performance degradation of the optimal controller. To reduce the impact of watermarking in [73] a more general method with Gaussian zero mean is introduced. The optimal choice of watermark is obtained when the Kullback–Leibler divergence of compromised and the healthy residue are maximized [71].

Moving target is another active method that is designed to detect covert cyber-attacks. In this method, the defender tries to prevent an attacker from acquiring knowledge of the system model. Moving target includes an external state with linear time-varying dynamics, unknown to the attacker, as part of the authentication process. This state is affected by the system's actual states, and if an attacker injects a cyber-attack that changes the system states, it will necessarily change the external authentication state. On the other hand, as the dynamics of the system change with time, it acts as a moving target that needs to change so fast that it prevents attackers from detecting the dynamics of the external system. In [74], authors introduced extra sensors and an Independent and Identically Distributed (IID) random time-varying dynamic system for moving targets. Optimal design for moving target matrix is introduced in [75]. Moving target detection is also proposed for defending smart grid [76].

Coding methods and modulation matrix are similar concepts that change the input and output signals of a system in a way that it becomes difficult or even impossible for an attacker to identify correct values [77, 78]. Depending on how coding is implemented, coding can be considered one-way when the information in a channel flows in one direction only, or two-way when information flow is simultaneously happening in both directions. In one-way coding, the defender uses a matrix unknown to the attacker, to change the input and output of the system. For an attacker to launch a successful cyber-attack, he first needs to decode or find the coding matrix. With one-way coding matrix it is possible for an attacker to find the coding matrix by eavesdropping after a period of time. For a system with an unstable eigenvector in A matrix, and coding applied only to unstable poles, it is shown that the attacker needs at least $N \geq \max\{n, p\} - 1$ measurements of sensors and actuators to be able to calculate coding matrix, where n is a number of states and p is a number of outputs. In [78], a different class of codes, suitable for unstable dynamical systems, is introduced that is applied for state estimator coding. In this paper to code the state of the system, a matrix constructed of the last weighted state of the system is subtracted from the current state of the system. They used a linear system of the form $x_{k+1} = Ax_k + w_{k+1}$, and since the coding matrix is continuously changing, they showed that the coding matrix could not be discovered by eavesdropping. The main disadvantage of this approach is that this scheme will work only when there is a dynamic in measurement (not in steady state condition), and control command or output is not considered. Two-way coding introduced by [79], on the other hand, offers strong encryption by combining input and output information. This is a very effective method developed for communication channels with data flowing in both directions.

1.2.4 Cyber-attack Detection Challenges

To analyze cyber-physical system security and vulnerabilities, most researchers study some specific cyber-attacks and their effects on the system under consideration.

Denial of Service (DoS) cyber-attack is the most common cyber-attack in the real world, where an adversary jams a communication channel. DoS cyber-attacks block communication between controller and plant and do not need knowledge of system dynamics [5]. DoS cyber-attacks' objective is disruption of communication, and attackers do not try to stay stealthy, so its detection is relatively easy.

False data injection (FDI) cyber-attack is another common type of cyber-attack that has been extensively studied in the literature. False data injection, in general, is considered as some random additive value injected in the input or output of the system. Depending on the cyber-attack signal it might be possible to detect these cyber-attacks with a fault detection system [9], however, a sophisticated attacker with knowledge of the system can design cyber-attacks that remain undetectable by the detector while harming the system. For instance, if the system has invariant zeros, an attacker can inject its cyber-attack on inputs of plant in a way that excites zero-dynamics of the system and if the unobservable poles of the system are also unstable, can make the system unstable without being detected [9]. However, even for a strongly observable system that does not have any invariant zero, if the number of input and output signals exceeds some limit, which is more than half of sensors for LTI systems, it would be possible to design cyber-attacks, which are undetectable by monitoring systems [80, 81].

For strongly detectable systems with no invariant zero, it is still possible to design a stealthy cyber-attack. For instance, a covert cyber-attack is a powerful stealthy cyber-attack, where an adversary with knowledge of the system and access to inputs and outputs can inject cyber-attacks on input signals and remove cyber-attack effects on outputs, hence, remains undetectable by monitoring systems [82]. Another cyber-attack on CPS that has been the subject of much research is a replay cyber-attack, where an adversary records sensor measurements of the system and then injects these data into the output of the system instead of actual data while manipulating input commands to achieve their objectives [9]. These types of cyber-attacks cannot be detected with fault monitoring systems and extra measures need to be taken to detect the cyber-attack.

1.2.5 Discussion

The current methods for detecting cyber-attacks and faults in industrial systems involve monitoring the system's measurements and control inputs and comparing them to normal behavior that is defined by a model based on physical laws such as observers or by machine learning techniques. However, both model-based and data-driven methods have limitations due to data noise and uncertainty [83]. Behavior-based methods do not use the system model but try to infer cyber-attacks or anomalies based on misbehavior discovered in the data. In [84] researchers use Kullback–Libeler divergence to detect cyber-attacks on agents based on statistical distribution of data. These methods can be effective for detecting unforeseen zero-day cyber-attacks, but in general, are less effective compared with signature-based methods due to weak accuracy and difficulty of triggering an alarm at the right time [85].

Most model-based cyber-attack and fault detection methods generate a residual by comparison of the actual states of a system and expected plant states. Data-driven methods also can be trained to model the system's normal behavior, like one class SVM or neural network-based methods, however, the majority of data-driven methods use classification techniques to detect and isolate (classify) cyber-attacks and faults from normal operating conditions. The classification method uses labeled training data to learn a model from data, and then classify the test data in any of the recognized classes. Deep learning models can extract features from data and so can be used as unsupervised classifiers. In deep learning methods, the data is fed to a model like a black box, and the output of this model is deciding about the presence of a cyber-attack or fault. One problem with this approach is the problem of overfitting data. Most researchers test their proposed techniques using publicly available datasets, with no real-world experiments and do not present an effective validation method for their studies. The validation and success metrics are a general shortcoming of the cyber-attack detection research community due to the lack of real-world and practical data on cyber-attacks on control systems [86]. Overfitting is a problem with data-driven methods and specifically neural networks [87, 88] that need careful training and testing of models to make sure the model is working properly.

Overfitting is a problem with all data-driven methods, but complex models like deep neural networks are more prone to the problem. Moreover, deep learning methods behave more like a black box, and it is difficult to interpret the results. It seems that simpler methods with fine-tuning can perform as well [62] for fault and cyber-attack detection.

1.3 Statement of Problem

Both machine-learning methods and model-based methods have their own advantages and disadvantages. While deep learning methods can extract features and classify the faults simultaneously since they behave like a black box, interpretation of results is difficult, and in general, they suffer from overfitting problems.

Statistical methods like PCA for feature extraction and SVM for fault and cyber-attack detection can perform as well as deep learning methods. SVM as a classification method has the advantage of using only data points near the boundary, so it is computationally efficient and can perform well on fewer data points. Moreover, it can handle Gaussian and non-Gaussian noises, while using kernel tricks also can handle nonlinearities in data [89]. SVM is successfully applied to different industrial systems for fault and cyber-attack detection.

However, passive detectors, like model-based and deep learning-based methods, that analyze information of the process and controller to make decisions about cyber-attacks cannot detect stealthy cyber-attacks like zero-dynamics, replay, and covet cyber-attacks because those cyber-attacks hide cyber-attack impact from the measurements. Researchers introduced several methods for detecting these types of cyber-attacks. For instance, moving targets and watermarking were introduced to detect replay and covert cyber-attacks. However, performance degradation is inevitable in watermarking. Coding that acts as lightweight encryption can prevent cyber-attacks like covert and zero-dynamics. When there is no cyber-attack, coding is transparent to the controller and plant. However, a cyber-attack results in modified decoded measurements that make the cyber-attack detectable.

In this thesis, in Chapter 3, fault and cyber-attack detection using SVM on a UAV are studied. Faults on actuators, false data injection cyber-attacks on control inputs, and replay cyber-attacks are studied. SVM is used as a detector for both faults and cyber-attacks.

In Chapter 4, two-way coding schemes for Multi-Input Multi-Output (MIMO) systems are developed as a cyber-attack detection method. Combined with a Kalman filter-based detector, proposed coding schemes can detect covert and replay cyber-attacks. The Kalman filter is used as a model-based method to analyze coding scheme impact when inputs and outputs are under cyber-attack however, the coding can be used with other model-based and data-driven methods as well.

1.4 Contributions of thesis

In Chapter 3 of this thesis, we explored the application of support vector machines (SVM) as a robust tool for detecting and isolating faults and cyber-attacks on UAV actuators. A crucial step for machine learning methods, including SVMs, is data preparation and feature engineering. Features engineering, extraction, and noise removal from data can be challenging and generally take more than half the time of machine learning projects. We showed that SVMs can effectively detect and isolate faults of UAVs; however, the choice of features is crucial for the success of the models. Our contribution to fault detection in this thesis is as follows:

- We found that, for UAVs, the Principal Component Analysis (PCA) can effectively extract the best set of features for fault diagnosis of actuators. However, we must exclude the control input of the actuator from the analysis. Then, we must add the control inputs to the PCs to construct the training dataset.
- Kalman filter estimated states offer the best choice of features, eliminating the need for feature extraction and engineering. However, since control inputs impact the states of the UAV, to diagnose actuator faults, control input effects must be estimated. We achieved this by directly including the control inputs as features or via the estimated states of an a priori estimator. The latter has the advantage of estimating unmeasurable states that might be essential for the fault diagnosis of complex systems. However, for UAVs, both approaches work equally well. We also found that residual generation degrades the performance of SVM detection.
- We use a bank of SVMs in two steps for fault isolation of actuators. First, we trained three SVMs to detect any of the three actuator's faults. We trained another three SVM, one for each actuator, to isolate two different types of faults of each actuator. SVMs are known to perform poorly with large datasets, the use of bank os SVMs allows us to

divide the training dataset into smaller sets that had a sizable impact on the training and tuning performance of SVMs. Another advantage of the bank of SVMs is the prediction performance of new observations, as we execute fewer predictions relative to the multi-class SVM to isolate faults.

Compared with traditional χ^2, SVM showed excellent performance in detection. And in contrast to χ^2, a bank of SVM can isolate faults.

Attack detection of UAVs is another challenging research area we studied in Chapter 3. Researchers have studied machine learning-based false data injection cyber-attack detection for different plants; however, there is limited literature on the use of SVM for FDI and replay cyber-attack detection on UAVs. We employed the Kalman filter estimated states to train the SVM model for FDI cyber-attack detection. Our contributions to cyber-attack detection and isolation are as follows:

- One-class SVM is a good choice for anomaly detection. However, it performs poorly with noisy data. Due to the noises of measurements and the unknown and changing effect of wind on UAVs, the one-class SVM predicts many false positives with healthy UAV data. On the other hand, we have defined a model for FDI cyber-attacks on UAV actuators. Therefore, we used the healthy dataset for training, similar to one-class SVM, however, we defined a margin around healthy data based on the noises and disturbances of data and added some bias on randomly selected healthy control input. Our SVM trained with this data could detect all simulated cyber-attacks with negligible false positives with healthy data.
- A fault in the system will trigger the cyber-attack detector we designed since it verifies current states relative to the expected states. To isolate cyber-attacks from the simultaneously occurring faults, we propose to use a pair of χ^2. We calculate a χ^2 based on the information on the plant side and another by the controller side information. We showed that an SVM trained with the proposed method with the two χ^2 can detect and isolate cyber-attacks from faults.
- For replay cyber-attack detection, we used physical watermarking. We showed that an SVM combined with signal smoothing and a search algorithm can detect the replayed cyber-attack on outputs of UAV with high precision.

In Chapter 4, we investigated the problem of covert and replay cyber-attack detection. Our contributions to stealthy cyber-attack detection are as follows:

- We extended the two-way coding schemes proposed by [79] for single-input single-output (SISO) systems for multi-input multi-output (MIMO) systems and analyzed them for discrete-time systems.
- We proposed a coding scheme with random numbers that transform coded inputs and

output into random numbers and effectively hide the dynamics of the system. This makes it impossible to find the coding matrix with eavesdropping. We proved that this coding schema is effective for covert and replay cyber-attacks.

- We also proved that our proposed method in Chapter 3 for the isolation of faults from cyber-attacks with two χ^2 works equally well with the coding schema.

We proved that our proposed coding scheme is effective for covert and replay cyber-attack detection. Moreover, with the proper choice of coding matrix, the two-way coding could eliminate the cyber-attacks compatible with the system noise. It is noteworthy that cyber-attacks on two-way coding inputs will impact the outputs. Hence, zero-dynamics cyber-attacks would also be detectable with our coding schemes.

Chapter 2

Background Information

2.1 Introduction

THE most common way to protect internet-connected systems is through well-studied encryption and authentication mechanisms. However, encryption is not always feasible due to computational overhead and expensive and powerful hardware requirements or for analog signals on the field level. For instance, in the case of small UAVs, due to limited hardware capabilities and battery capacity, encryption is not always feasible. Since UAVs always have some level of autonomy, and their communication with the ground control stations is wireless, unencrypted communication can expose them to cyber-attacks. And if knowledgeable attackers succeed in bypassing those security measures with whatever method, they can harm the systems and even cause severe physical damage.

The communication network is the main security threat to CPS [27, 90], which makes the CPS vulnerable to cyber-attacks such as DoS and false data injection cyber-attacks. Cyber-attacks can originate from either cyber or physical layers [11, 80]. In the field of control engineering, the control system is considered the link between cyber and physical layers of the CPS, that needs to be secured. A control system usually comprises four components: 1) physical plant, 2) sensors to measure and monitor physical plat's states, 3) a controller that based on the measurements and control law generates output command and 4) actuators that receive control output and modify plants states based on these commands.

Most security and anomaly detection methods developed so far require access to measurement and output commands to analyze and detect suspicious behavior. These methods are, to a large extent, based on fault-detection theories developed over the past few decades. Equipment failure is either random or developed gradually and follows physical laws. However, Cyber-attacks are often unbounded and stealthy, which requires additional measures to

Table 2.1: Detectability of cyber-attack depending on trust in components [86].

Component Trust			Detection	Comment
Controller	Sensor	Actuator	possibility	
✓	-	-	-	Bad actuator and bad sensing
-	-	✓	-	False sensing justifies bad controls
-	✓	-	~	Cyber-attack effects observable
✓	-	✓	✓	Cyber-attack effects observable
✓	✓	-	✓	Cyber-attack effects observable
-	✓	✓	✓	Bad command detection
✓	✓	✓	✓	No cyber-attack possible

✓= trusted/detection possible, - = untrusted/detection not possible, ~= cannot detect zero-dynamics cyber-attacks.

be detected.

When it comes to evaluating the vulnerability of a system and detecting cyber-attacks, it's crucial to determine which components of a control system can be trusted. An attacker can compromise sensors, actuators, or the controller, so it's essential to identify which parts are reliable. Controllers usually have a hierarchical structure, with the field level being the lowest level of automation and the supervisory control system at the top level. Implementing anomaly detection systems at the both field and supervisory levels is recommended to improve cyber-attack detection.

One of the challenges of implementing monitoring systems for the field layer is the need for inspection of low-level data, where signals exchange without standard units of measurement and protocols are proprietary for different devices and not publicly available. In [86], researchers analyzed the detectability of cyber-attack based on trusted components, which is summarized in Table 2.1.

This analysis indicates the importance of having at least two trusted components in a control system to detect cyber-attacks. If only sensors are trusted, cyber-attacks can still be detectable, except for zero-dynamics cyber-attacks, which also need knowledge of outputs. It is worth noting that this analysis assumes the attacker has access to all sensors and or actuators.

Researchers, independently in [80] and [81], show that there are theoretical limits on the detectability of cyber-attacks when a certain number of sensors and actuators are under cyber-attack. For strongly observable LTI systems, for both discrete-time and continuous-time systems, it is proven that if only less than half of sensors and actuators are under cyber-attack, it would be possible to detect cyber-attacks. Strongly observable here means that it is possible to reconstruct all states of the system using a sequence of measurements.

In other words, these systems do not possess zero-dynamics.

Cyber-attack detection methods are categorized differently in literature, either based on control system architecture, like centralized and decentralized [27, 91], or based on the use of historical data as stateless or stateful, and also based on the model of the system as statistical and model-based systems [86]

One general classification of detection methods is by considering detection techniques such as Machine Learning (ML) based methods, model-based methods, and hybrid methods. Machine learning or data-driven methods are broad categories that include statistical methods and several other methods like classification and neural networks.

Model-based methods generate a model of a system either through physical laws or observation and system identification methods. A model-based algorithm predicts a future measurement y_{k+1} based on available measurements y_k and control signal u_k. If only the outputs of a system are available, then regression models can offer a good estimation. Model-based methods and some statistical methods calculate a residual r_k based on the differences between measured value y_k and expected value $\hat{y}_k$. These residuals are then compared with a threshold to detect cyber-attack or fault [27, 91]. To raise the alarm, one might use a stateless or stateful approach. In a stateless approach, an alarm triggers whenever the test result deviates from a specified threshold, that is, $|y_k - \hat{y}_k| = r_k \geq \tau$. In stateful approaches, an additional parameter S keeps track of historical changes of r_k, and an alarm is triggered if $S \geq \tau$ [86].

Machine Learning methods do not need to generate residuals and can directly classify data and map a relationship among parameters or may use heuristic algorithms to model the system [27]. However, in hybrid approaches, ML-based methods, such as classification methods, may be combined with residual generation to isolate faults and cyber-attacks.

In general, one of the limitations in the cyber-attack detection of control systems, specifically for ML-based methods, compared to IT systems, is the lack of available data from cyber-attacks in the real world. Most available data are specific to certain domains, and it is unclear if they can be a reliable measure for evaluating a defense algorithm.

Types of Faults in Industrial Systems

A fault is a deviation from the nominal behavior of plant components' performance, sensor measurements, or actuators' response to the control command. Nominal here means the performance characteristics intended by the designer. While *cyber-attack* ranges from small perturbations of the system aiming to degrade the performance of a system to abrupt changes in system parameters to breakdown a system.

Fault types can be summarized as follows [28]:

- *Additive process faults* are unknown inputs acting on plant states that cause changes in the outputs of a plant independent of known inputs.
- *Multiplicative process faults* cause changes in the plant's outputs that depend on known inputs. These sorts of faults are best described as deterioration of plant parts.
- *Sensor faults* are discrepancies between measurements and actual values of plants, which are generally considered additive.
- *Actuator faults* are described as discrepancies between input command and actuator's actual output and generally are considered additive.

Component faults can be detected by monitoring measurements of a system and looking for deviation from expected values obtained from the model of the system or statistical and machine learning methods to classify the system behavior. Faults in sensors and actuators also follow similar approaches.

Types of Cyber-attacks in Industrial Systems

Cyber-attacks can target sensors and actuators of a system. In the literature, generally, the control system is assumed to be trusted so that it is possible to implement detection algorithms. The following are the most notable cyber-attacks that may affect a system [92]:

- *False Data Injection cyber-Attacks* are a type of cyber-attack where an attacker tampers measurements or actuator commands of the system, and while it may be stealthy, in general, knowledge of the dynamics of the systems is not assumed.
- *Daniel of Service Cyber-attacks* or jamming cyber-attacks compromise the availability of resources in the systems by blocking communication channels.
- *Zero-dynamics Cyber-attacks* affect systems with unstable modes that have invariant zeros. An cyber-attacker with knowledge of system dynamics can design an cyber-attack to excite zero dynamics of the system while the output remains unchanged and undetected.
- *Covert Cyber-attacks* are a type of stealthy cyber-attack where the attacker injects some command to disrupt the system while removing the effect of the cyber-attack by modifying measurements to remain undetected.
- *Replay Cyber-attacks* are the combination of two cyber-attacks. First, play back previously recorded measurements of the system related to a previous state of the system, and second, injection of some control signals to disrupt the system.

The incidence of cyber-attacks and system faults can cause variations in system states and measurements, leading to deviations from expected operating conditions. The most distinct characteristic of an cyber-attack is the intelligent human factor behind the cyber-attack, who may manipulate the signals in real time to remain stealthy. Conversely, faults are unlikely to happen simultaneously and instantaneously on several inputs and outputs, which is another distinction between cyber-attack and fault.

If the attackers know the dynamics of the systems and have access to both inputs and outputs, they may remove or compensate for the cyber-attack effect on measurement, making it difficult or impossible to detect the cyber-attack. These types of cyber-attacks are collectively known as stealthy cyber-attacks. Covert, replay, and zero-dynamics cyber-attacks are among the stealthy cyber-attacks. Zero-dynamics cyber-attacks are exceptions that only require access to inputs of the system. However, the system should have unobservable and unstable modes.

The assumptions of fault detection that are true for cyber-attack detection are:

1. The faults and cyber-attacks are not present initially.
2. Despite that disturbance can be considered additive, the detection method objective is to detect faults and cyber-attacks and discard disturbances effect.
3. Any noise of measurement, actuator, or plant is assumed to be zero mean, and non-zero mean noises are considered fault, cyber-attack, or disturbances.
4. Discrepancies between model and actual plant behavior due to modeling errors or uncertainties can adversely affect detection performance and may be confused with multiplicative disturbances.

2.2 Cyber-attack on UAVs

Unmanned Aerial Vehicles (UAVs) have been widely used in military applications for several decades, but their utilization in commercial and non-military cases is still in progress [93]. Some potential use cases for UAVs in the non-military sector include environmental monitoring, monitoring forests for fire, aerial surveillance and mapping, road traffic monitoring, agriculture applications, ad hoc networks, etc. It is anticipated that commercial uses of UAVs grow exponentially, making them an essential part of various industries and society. The market size of UAVs was \$13.44 billion in 2020 and is projected to grow at an annual rate of 57.5% to about \$130 billion in 2025 [7]. On the other hand, the security of UAV operations has become an important subject. In recent years, drones or UAVs have raised several security and privacy issues, resulting in extensive discussion regarding the integration of UAVs into national airspace. This subject become a major challenge that requires serious attention in both academic research and commercial sectors.

UAVs can classified into two categories: fixed-wing aircraft and rotorcraft. Fixed-wing aircraft also may be categorized by size. Generally, UAVs with a wingspan of fewer than 5 ft^2 are battery-powered and do not need a runaway for take-off or landing. These types of UAVs have small payloads, which results in severe limitations on the use of sensors and capabilities of the installed computer on the UAV. Hence, the design operation and security concerns of these UAVs are challenging.

Another notable challenge with small UAV flights is wind. Typically, small UAVs travel at a speed of 30 to 60 km/h, and since wind speed at the level of flight of UAVs is almost always more than 15 km/h, the effective maneuver of UAVs can be challenging. For instance, typical trajectory tracking methods for robots assume the robot is in a particular location at a specified time, which is not always correct for UAVs, primarily due to wind and wind gusts. However, path-following methods are effective, as those methods keep the vehicle on a desired path cite[93].

The control system of UAVs, including autopilot, relies on measurement data acquired from onboard sensors, which include accelerometers, rate gyros, pressure sensors, magnetometers, and GPS receivers. Small UAVs use cheaper components, which means there are more chances of fault and failure in those components.

In this thesis, a nonlinear model of UAV developed by the University of Minnesota in 2012 is used to generate data and validate the results of the detection method. The simulator was developed with Matlab/Simulink, and it is a high-fidelity nonlinear model of a small UAV.

The theoretical basis of the UAV simulator is discussed later in this chapter. The model parameters were obtained by experiments and then validated using the system identification process [94]. Uncertainty of parameters of modeling were employed in the model

2.2.1 UAV Nonlinear Modeling

Different coordinate systems are used in the modeling and analysis of UAVs and it is crucial to know how to transform between them. The use of multiple systems is necessary due to various factors summarized as follows:

- Derivation of equations of motion is done in a fixed ground coordinate system known as inertial reference frame. However, when analyzing UAV motion, a coordinate system centered on UAV mass center and aligned with UAV position is more appropriate. This coordinate system is known as body-fixed frame.
- Aerodynamic forces applied to UAV frame can be easily described in body-fixed reference coordinate.
- While some sensors like accelerometers and rate gyros make measurements with respect to the body frame, GPS sensor output is with respect to the fixed ground coordinate system (inertial frame).
- Flight trajectories, and map information are in the ground coordinate system.

A thorough discussion on different coordinate systems, and conversion between them is presented in [93].

UAV Specification

The aircraft used for Matlab/Simulink simulation is a commercial Radio-Controlled (RC) plane UltraStick 25E. For simulation Matlab/Simulink aerospace toolbox/blockset is used to develop nonlinear 6-DOF aircraft dynamic models. The environment is also modeled using this blockset.

The plane control surface are rudder, elevator, aileron and flap, and use Hitec servos for actuation. The propulsion system is 600 watts E-Flite electric motor drive. The wingspan of UAV is 1.2 meter, and wing reference area is 0.32 m^2.

The flight avionics system installed on UAV board are:

- Angular rates: $p(deg/s)$, $q(deg/s)$ and $r(deg/s)$
- Accelerations: $a_x(g)$, $a_y(g)$ and $a_z(g)$
- Magnetic fields: $H_x(gauss)$, $H_y(gauss)$ and $H_z(gauss)$
- Airspeed and barometric altitude: $V_s(m/s)$ and $h(m)$
- GPS velocities (ENU format) and positions: $v_e(cm/s)$, $v_n(cm/s)$, $v_u(cm/s)$ and $p_x(10^{-7}deg)$, $p_y(10^{-7}deg)$, $p_z(m)$.

Aerodynamic Forces and Momentum Acting on UAV

The aerodynamic forces acting on UAS during flight are result of air pressure that acts on UAV body and is proportional to dynamic pressure of air and shape and attitude of UAV. Dynamic pressure itself is a function of relative speed of UAV to wind or airspeed and air density given by $Q = \frac{1}{2}\rho V_a^2$.

The dynamics of UAV may be described by forces and momentum acting on UAV in three directions also referred to as six degree of freedom equations as shown in Figure 2.1.

The equation of forces acting on UAV can be expressed as:

$$\begin{aligned} f_X &= QSC_X \\ f_Y &= QSC_Y \\ f_Z &= QSC_Z \end{aligned} \tag{2.1}$$

where Q is dynamic pressure, S is planform area of UAV and C_X, C_Y and C_Z are aerodynamic forces coefficients given by:

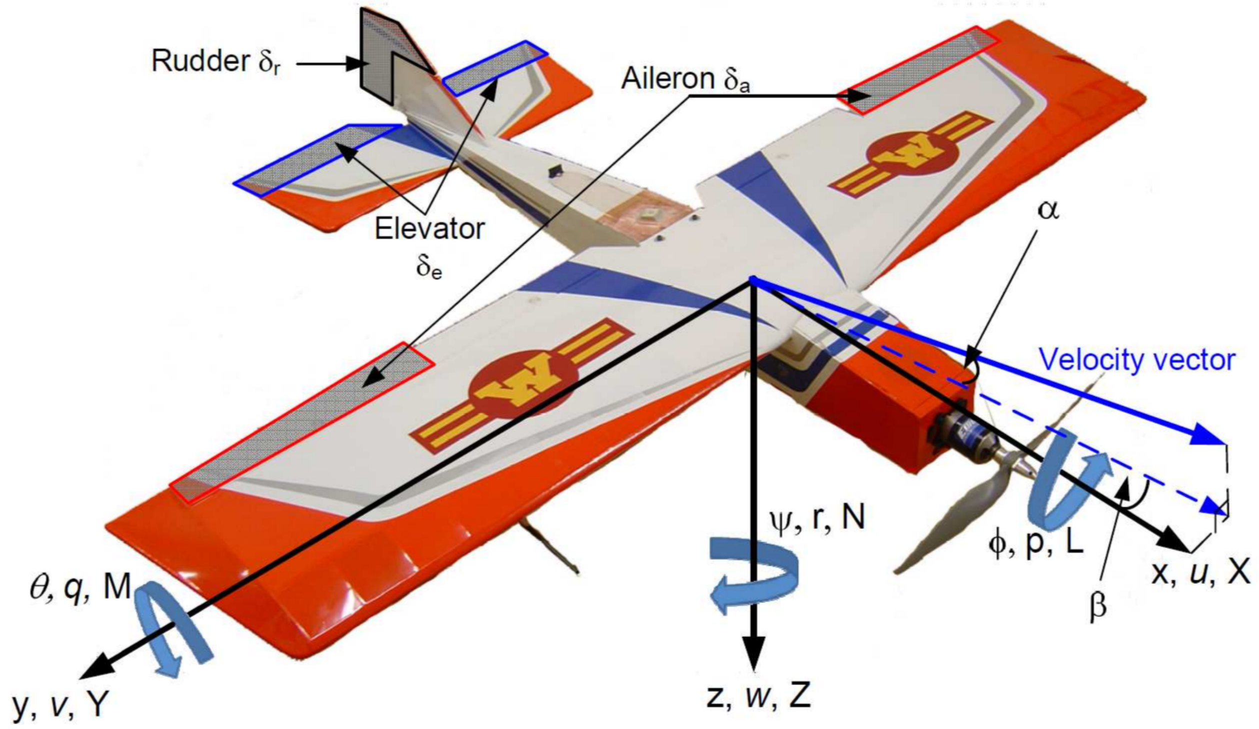


Figure 2.1: Forces and moments acting on UAV in aircraft body frame.

$$\begin{aligned} C_X &= C_L \sin(\alpha) - C_D \cos(\alpha) \\ C_Z &= -C_D \sin(\alpha) - C_L \cos(\alpha) \\ C_Y &= C_{Y_\beta}\beta + C_{Y_{\delta r}} + \frac{b}{2V_a}(C_{Y_p} + C_{Y_r} r) \end{aligned} \tag{2.2}$$

where C_L and C_D are lift and drag coefficients respectively. Lift and drag coefficients are primarily affected by angle of attack α, however pitch q and elevator angle δ_e also has influence. Assuming small perturbation, the drag and lift coefficients can be considered linear and when linearizing, only first terms will be considered. The result will be:

$$\begin{aligned} C_L &= C_{L_0} + C_{L_\alpha} + C_{L_\alpha}\alpha + C_{L_{\delta_e}}\delta_e + \frac{c}{2V_a}(C_{L_{\dot\alpha}} + C_{L_q} q) \\ C_D &= C_{D_0} + C_{D_{\delta_e}}\delta_e + C_{D_{\delta_r}}\delta_r + \frac{(C_L - C_{L_{min}})}{\pi.c.AR} \end{aligned} \tag{2.3}$$

where C_{L_0} and C_{D_0} are values of lift and drag coefficients when $\alpha = 0$, $\delta_e = 0$ and $q = 0$.

Apart from those three forces acting on UAV, there are three momentum equations about body of UAV given by:

$$
\begin{aligned}
L &= QSbc_l \\
M &= QSbc_m \\
N &= QSbc_n
\end{aligned}
\tag{2.4}
$$

where c_l, c_m and c_n are non-dimensional momentum coefficients given by:

$$
\begin{aligned}
c_l &= c_{l_\beta}\beta + c_{l_{\delta_a}}\delta_a + c_{l_{\delta_r}}\delta_r + \frac{b}{2V_a}(c_{l_p}p + c_{l_r}r) \\
c_m &= c_{m_0} + c_{m_\alpha}\alpha + c_{m_{\delta_e}}\delta_e + \frac{\bar{c}}{2V_a}(c_{m_\alpha}\dot{\alpha} + c_{m_q}q) \\
c_n &= c_{m_\beta}\beta + c_{m_{\delta_a}}\delta_a + c_{m_{\delta_r}}\delta_r + \frac{b}{2V_a}(c_{m_p}p + c_{m_r}r)
\end{aligned}
\tag{2.5}
$$

Dynamics of UAV

The dynamics of UAV, also known as momentum equations are calculated by the angular rate equations are given by:

$$\dot{p} = \frac{QSb}{I_{xx}}c_l - \frac{I_{zz} - I_{yy}}{I_{xx}}qr + \frac{I_{xz}}{I_{xx}}qp + \frac{I_{xz}}{I_{xx}}\dot{r} \tag{2.6}$$

$$\dot{q} = \frac{QSc}{I_{yy}}c_m - \frac{I_{xx} - I_{zz}}{I_{yy}}pr - \frac{I_{xz}}{I_{yy}}(p^2 - r^2) + \frac{I_p}{I_{yy}}\omega_p r \tag{2.7}$$

$$\dot{r} = \frac{QSb}{I_{zz}}c_n - \frac{I_{yy} - I_{xx}}{I_{zz}}pq - \frac{I_{xz}}{I_{zz}}qr + \frac{I_{xz}}{I_{zz}}\dot{p} - \frac{I_p}{I_{zz}}\omega_p q \tag{2.8}$$

Kinematics of UAV

UAV kinematics is set of equation that defines UAV rotational motion and can be described based on body angular rates (p, q and r), Euler angles (ϕ, θ and ψ) and aerodynamic angles of UAV (α, β and γ) and is given by:

$$\begin{aligned}
\dot{\phi} &= p + \tan(\theta)(q\sin(\phi) + r\cos(\phi)) \\
\dot{\theta} &= q\cos(\phi) - rsin(\phi) \\
\dot{\psi} &= \frac{q\sin(\phi) + r\cos(\phi)}{\cos(\theta)} \\
\theta &= \gamma + \alpha\cos(\phi) + \beta\sin(\phi)
\end{aligned} \tag{2.9}$$

UAV Inertia Modeling

The inertia model deals with geometric and physical parameters of UAV, the model defines relationship between aircraft mass, its center of gravity and moment of inertia coefficients which is presented as below matrix:

$$I = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix} \tag{2.10}$$

The diagonal terms are known as *moments of inertia* and off-diagonal terms are known as *product of inertia*. It is assumed that UAV is symmetric in respect to x-z plain and so the terms $I_{xy} = I_{yx} = 0$ and $I_{yz} = I_{zy} = 0$, hence the matrix can be simplified as:

$$I = \begin{bmatrix} I_{xx} & 0 & -I_{xz} \\ 0 & I_{yy} & 0 \\ -I_{zx} & 0 & I_{zz} \end{bmatrix} \tag{2.11}$$

Propulsion System Model

In small UAVs that incorporate propeller propulsion system, the torque of propeller can affect the system dynamics. This is due to the large size of propeller relative to aircraft. To handle this effect in the model, momentum due to operation of propulsion system I_p, should be added to aircraft momentum as shown in the equations (2.7) and (2.8). This momentum is given by $I_p = I_{motor} + I_{propeller}$ [94]. The model aircraft used for simulation has an outrunner motor that major part of it is rotating. From law of conservation of momentum one can write the proper equation as:

$$(I_{motor} + I_{propeller})\dot{\omega_p} = T_{motorr} - T_{propeller} \tag{2.12}$$

where I_{motor} and $I_{propeller}$ are moment of inertia of motor and of propeller respectively, and T_{motor} and $T_{propeller}$ are torque generated by motor to its shaft and propeller respectively. Hence I_p is calculated as:

$$I_p = I_{motor} + I_{propeller} = \frac{T_{motorr} - T_{propeller}}{\dot{\omega_p}} \tag{2.13}$$

To calculate I_p we need to T_{motorr} and $T_{propeller}$. These torques are calculated as explained below.

Propulsion Motor

The motor shaft torque can be presented as:

$$T_{motor} = \frac{P_0}{\omega_p} \tag{2.14}$$

where P_0 is motor shaft power and ω_p is rotational speed of propeller.

Propeller Characteristics

The trust generated by motor is proportional with power of motor or torque applied to propeller (T_p), rotational speed of propeller (ω_P), propeller trust (F_p), propeller diameter (R), air density and air speed (V_a). The propeller performance then can be described by the set of three equations:

$$J = \frac{\pi V_a}{\omega_p R} \tag{2.15}$$

$$C_T = \frac{F_p \pi^2}{4\rho R^4 \omega_p^2} \tag{2.16}$$

$$C_P = \frac{T_p \pi^3}{4\rho R^5 \omega_p^2} \tag{2.17}$$

where J is known as advanced ratio, C_T is coefficient of thrust and C_P is coefficient of power. Therefor we have:

$$T_{propeller} = T_p = \frac{4C_P \rho R^5 \omega_p^2}{\pi^3} \tag{2.18}$$

This provides the propeller thrust (F_p) and propeller torque (T_p) shown by $T_{propeller}$ in equation (2.13), at different advance ratios condition during the simulation.

Actuators Model

Control surfaces of modeled UAV in simulator are driven by micro servo motor using PWA operated at $45Hz$, and its rotational ratio is $500ged/s$. The time delay of actuator is modeled as a single period of PWM or about $22ms$.

This Matlab/Simulink block of actuators implements actuator models for the aileron, elevator, rudder, and flap. The throttle does not have a servo however, its position is limited. Although physically, there are two aileron and flap servos, they are treated as one in model. This is due to modeling approach and assumption of symmetry of aircraft. Implementation of any single control surface in model requires significant work on the aerodynamic models to take the effect of unsymmetrical body into account.

2.3 System, Fault and Cyber-attack Model

In this thesis, we assumed a discrete-time linear time-invariant (LTI) system for fault and cyber-attack detection. LTI models are widely studied in the literature and have a proven history for dynamic systems analysis. Faults considered are both additive and multiplicative on actuators. Cyber-attacks studied in this thesis include false data injection cyber-attacks on actuators, replay cyber-attacks, and covert cyber-attacks. In Chapter 3, we studied false data injection cyber-attacks on actuators and replay cyber-attacks. In Chapter 4, we investigated covert cyber-attacks and replay cyber-attack detection. For FDI cyber-attacks, we assumed the attacker had access to the input channel but did not know the system dynamics. For replay cyber-attacks, we assumed the attackers have full access to both the input and output channels. And for covert cyber-attacks, the assumptions are that the attackers have full knowledge of the system dynamics and have access to input and output channels. The system model is represented in Figure 2.2.

In our analysis, we used a linear model of the UAV. The procedure for linearization of the model of UAV can be found in [93]. We discretized the continuous time LTI UAV model with the assumption of a zero-order hold. The numerical matrices of the linear discrete-time model are presented in Section 3.2. It is assumed that the plant is equipped with Kalman filter

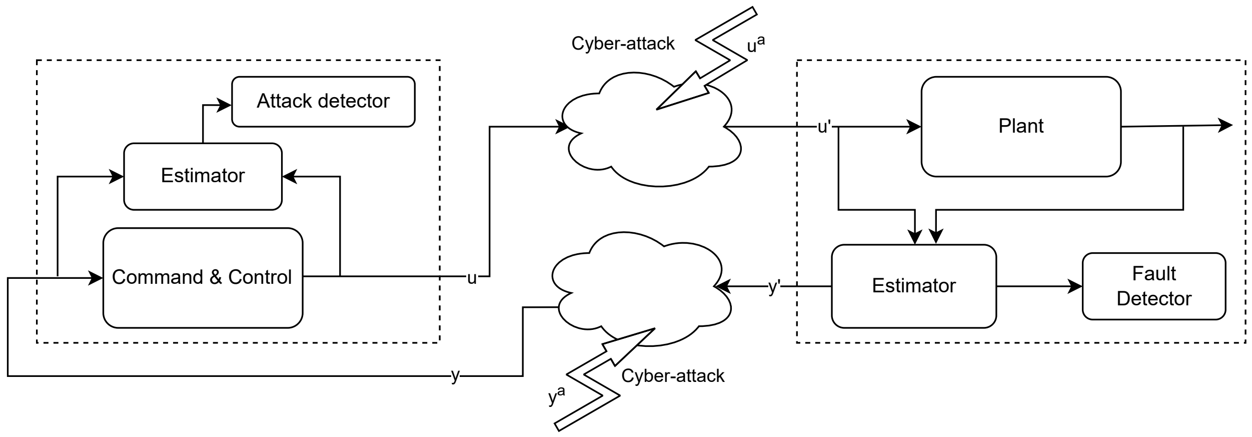


Figure 2.2: Overall CPS and cyber-attack model.

based estimator and fault detector, and hence sends the best estimation of measurements to command-and-control. The Linear model considered is:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) + w(k) \\ y(k) &= Cx(k) + \nu(k) \end{aligned} \tag{2.19}$$

where $x_k \in \mathbb{R}^n$ is UAV state vector, $y(k) \in \mathbb{R}^m$ and $u(k) \in \mathbb{R}^p$ are respectively UAV measurements and controller input vectors, $w(k) \sim \mathcal{N}(0, Q)$ and $\nu(k) \sim \mathcal{N}(0, R)$ are zero mean Gaussian disturbance and noise respectively, A, B and C are matrices with appropriate dimensions.

2.3.1 Fault Model

The faults that might happen in control surfaces (actuators) of a small UAV mainly fall in two categories [95, 96]:

- *Total loss of control of actuator* is a situation where the actuator (control surface) is not responding to any control command. Faults in this category include locking actuators in whatever position, known as lock-in-place, locking actuators at their extreme maximum or minimum position or hard-over, and floating actuators independent of control commands, known as floating around trim or floating surface.
- *Partial loss of control of actuator* or loss of effectiveness or bias is a situation in which the actuator does respond to the control command but with some non-zero difference from the control command.

The UAV has four actuators, which include three control surfaces and the UAV engine speed. The controller sends out commands to adjust the angle of the control surfaces such as aileron (δ_a), rudder (δ_r), elevator (δ_e), and speed of the motor (ω). Therefore, the control inputs are as follows:

$$u_c(k) = [\delta_a(k) \quad \delta_r(k) \quad \delta_e(k) \quad \omega(k)]^T \tag{2.20}$$

The lock-in-place faults are modeled as additive faults by a constant value representing the angle at which the actuators are locked. The loss-of-functionality faults are modeled as multiplicative faults by reducing the range of control surface deflection. Hence, the mathematically model of faults are as follows:

$$u_f(k) = Eu(k) + f \tag{2.21}$$

where $u(k)$ is the control inputs issued by the controller, $u_f(k)$ is the actual deflection of faulty actuators, E is a diagonal square matrix with diagonal elements between 0 and 1 as an indication of the magnitude of fault, and f is a vector with elements specifying the angle that control surfaces locked at. The loss-of-functionality of actuators is modeled by setting the relevant rows of f vector to 0 and a value between 1 to 0 for the diagonal elements of E matrix. The actuators lock-in-place, on the other hand, is modeled by setting the relevant diagonal elements of the E matrix to 0, meaning no response to control input, and a value for the corresponding row of the f vector, specifying the angle that the actuators locked at. Hence, the faulty system mathematical model is as follows:

$$\begin{aligned} x(k+1) &= Ax(k) + B[Eu(k) + f(k)] + w(k) \\ y(k) &= Cx(k) + \Delta y_f(k) + \nu(k) \end{aligned} \tag{2.22}$$

where $\Delta y_f(k)$ is additive sensor faults.

2.3.2 False Data Injection Cyber-attack Model

False Data Injection cyber-Attack (FDIA) is a type of cyber-attack where the attacker has access to the communication channel of the controller with the plant and can manipulate real-time measurements and control outputs. When the attackers know the system dynamics, they can inject cyber-attack signals into the measurements to remove the impact of their cyber-attack on control input from measurement and remain undetected. We have modeled the faulty system under cyber-attack as follows:

$$\begin{aligned} x(k+1) &= Ax(k) + B[E(u(k) + u^a(k)) + f(k)] + w(k) \\ y &= Cx(k) + \Delta y_f(k) + y^a(k) + \nu(k) \end{aligned} \tag{2.23}$$

where $u(k)$ is the control inputs, E and f are the faults signature defined in equation (2.21), $\Delta y_f(k)$ is additive sensors faults, $u^a(k)$ is a vector of additive cyber-attack signals on the control inputs and $y^a(k)$ is a vector of additive cyber-attack on measurements. The cyber-attack signals, $u^a(k)$ and $y^a(k)$ can take any value.

2.3.3 Replay Cyber-attack Model

Replay cyber-attack is another notable type of cyber-attack where the attackers with access to system inputs and outputs can launch. In a replay cyber-attack, attackers first record healthy system measurements, and then, at the time of the cyber-attack, they replay previously recorded measurements instead of the actual system measurements. Meanwhile, the attackers modify control input to launch the cyber-attack. The model of the system under replay cyber-attack and fault used is as follows:

$$\begin{aligned} x(k+1) &= Ax(k) + B[E(u(k) + u^a(k)) + f(k)] + w(k) \\ y &= Cx(k) + y^a(k) + v(k) \end{aligned}$$

where $u^a(k)$ and $y^a(k)$ are cyber-attack signals. The attacker recorded measurement from time k_1 to $k_1 + T$ and then removes current measurement and adds recorder measurements [71]

$$y_a = y(k_1) - Cx(k) - v(k) \tag{2.24}$$

At the same time, the attackers inject malicious input $u_a(k)$ to perform the cyber-attack.

2.4 Support Vector Machines Theory

Support vector machines (SVM) were developed in the 1990s and are a generalization of a simple classifier known as *maximum margin classifier*. Maximum margin classifier requires classes to be separable by a line and cannot be applied to most datasets. *Support vector classifier* is an extension of this classifier that has broader applications, and *support vector machines* is an extension of support vector classifier to handle non-linear decision boundaries between classes [97]. SVM is a multivariate statistical method for binary classification of two classes, however, it can be extended to classify multiple classes of data. SVM shows good performance in a variety of problems.

SVM uses the notion of *hyperplane* as its decision boundary to separate data into two classes. A hyperplane is an affine subspace of p dimensions. A hyperplane in a two-dimensional space is a line. An optimal separating hyperplane is one that has a maximum distance from the points known as a support vector. SVM can construct nonlinear decision boundaries by transforming and enlarging the feature space of data and constructing linear boundaries (hyperplane) in the new feature space. These properties make SVM an ideal candidate for fault and cyber-attack detection.

The training data set X is a $n \times p$ matrix with n observations of p parameters, where x_i denotes columns of this matrix. Each observation is assumed to belong to one of two positive or negative classes. Vectors of training data are a pair of data in the form of (x_i, y_i), with $x_i \in \mathbb{R}^p$ and $y \in \{-1.1\}$ and the hyperplane will be defined as:

$$\{x : f(x) = x^T w + b = 0\} \tag{2.25}$$

In the above equation, w is a unit vector. Two parameters (w, b) define the shape and orientation of the hyperplane. The hyperplane with a maximum distance from the nearest data points of any class is obtained by solving an optimization problem. Only the closest points to the hyperplane affect the calculation of the hyperplane, and these points are known as support vectors. The classifier is formulated as follows:

$$G(x) = sign[x^T w + b] \tag{2.26}$$

Considering that we assumed the classes are separable, a solution in the form of $f(x) = x^T w + b$ to this problem can be found with $y_i f(x) > 0 \quad \forall i$. This will result in a hyperplane with maximum *margin* between the training points. Hence, a convex optimization problem can be formulated as follows:

$$\begin{aligned} \underset{w,\,b}{\text{minimize}} \quad & ||w|| \\ \text{subject to} \quad & y_i(x_i^T w + b) \geq 1, \quad i = 1, \ldots, n \end{aligned} \tag{2.27}$$

This is the most common formulation for SVM which results in a convex optimization problem. It should be noted that maximum distance of hyperplane from support vectors or margin is $M = 1/\|w\|$

If there are overlap between two classes a slack variable $\xi = (\xi_1, \ldots, \xi_n)$ is used to allow some points lay in wrong side of margin. One method of modifying the constraint to include the misclassification is:

$$\begin{aligned} \xi_i \geq 0, \quad \forall i \\ \sum_{i=1}^{n} \xi_i \leq c \\ y_i(x_i^T w + b) \geq (1 - \xi_i) \end{aligned} \tag{2.28}$$

where c is a constant. The slack variable ξ_i is proportional to amount of misclassification for a value of $\xi_i > 0$, and the value c limits the total number of misclassifications. The optimization problem became.

$$\begin{aligned} \text{minimize} \quad & \|w\| \\ \text{subject to} \quad & y_i(x_i^T w + b) \geq 1 - \xi_i, \quad \forall i, \\ & \xi_i \geq 0, \\ & \sum \xi_i \leq c \end{aligned} \tag{2.29}$$

A more computationally efficient alternative to (2.29) is,

$$\begin{aligned} \underset{w,\,b}{\text{minimize}} \quad & 1/2\|w\|^2 + C\sum_{i=1}^{n} \xi_i \\ \text{subject to} \quad & y_i(x_i^T w + b) \geq 1 - \xi_i, \quad \forall i, \\ & \xi_i \geq 0 \end{aligned} \tag{2.30}$$

Here C is a cost function which replaces the constant value c in equation (2.29). The optimal value for C can be estimated by cross validation. We need to define the Lagrange multipliers to solve the above equation. The Lagrange function used is [46]:

$$L(w,b,\xi,\alpha,\mu) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i[y_i(x_i^T w + b) - (1-\xi_i)] - \sum_{i=1}^{n}\mu_i\xi_i \tag{2.31}$$

Vector representation of the Lagrange dual function in the form of inner product of input vectors results computationally more appropriate form for kernel function [98]:

$$L(w,b,\xi,\alpha,\mu) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i[y_i(\langle x_i, w\rangle + b) - (1-\xi_i)] - \sum_{i=1}^{n}\mu_i\xi_i \tag{2.32}$$

The parameters α and μ are Lagrangian multipliers. To minimize this equation, it is needed to set the derivative respecting w, b and ξ_i to zero, which results in:

$$w = \sum_{i=1}^{n}\alpha_i y_i x_i, \tag{2.33}$$

$$\sum_{i=1}^{n}\alpha_i y_i = 0, \tag{2.34}$$

$$\alpha_i = C - \mu_i, \quad \forall i \tag{2.35}$$

Replacing equations (2.33), (2.38) and (2.39) into equation (2.31) and taking into account that $\alpha_i, \mu_i, \xi_i > 0$ the dual Lagrangian objective function also known as Wolfe function is obtained, that will be formulated as an optimization problem to obtain α as follows:

$$\max_{\alpha} W(\alpha) = \max_{\alpha}\left[\sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j\langle x_i, x_j\rangle\right] \tag{2.36}$$

Th constrains for this maximization are

$$\begin{aligned} 0 \le \alpha_i \le C, &\quad \forall i \\ \sum_{i=1}^{n}\alpha_i y_i = 0 & \end{aligned} \tag{2.37}$$

This optimization problem when satisfies the additional Karush–Kuhn–Tucker conditions that include the constraints:

$$\alpha_i[y_i(\langle x_i, w\rangle + b) - (1 - \xi_i)] = 0 \tag{2.38}$$

$$\mu_i \xi_i = 0 \tag{2.39}$$

$$y_i(\langle x_i, w\rangle + b) - (1 - \xi_i) \geq 0 \tag{2.40}$$

will uniquely solve this dual quadratic optimization problem. Considering equation (2.33), it can be seen that α_i is nonzero only when the constraint (2.40) satisfies, and hence only these observations affect the separating hyperplane. These observations are known as *support vectors*. By solving this optimization problem, the decision hyperplane will be described by the following equation.

$$G(x) = sign[\langle x.w\rangle + b] \tag{2.41}$$

All of this, results in a linear hyperplane. To handle nonlinear boundaries of classes, like other linear methods, one approaches is to enlarge feature space by employing basis function like polynomial functions. However, this approach can result in unmanageable amount of computation and overfitting. The kernel function is a solution to this problem that enables SVM to employ an infinite number of dimensions. If we show enlarged input features with h(x) then the Lagrange dual function can be presented as:

$$\max_{\alpha} W(\alpha) = \max_{\alpha} \left[\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle h(x_i), h(x_j)\rangle \right] \tag{2.42}$$

Hence, the hyperplane can be obtained by solving $y_i f(x_i) = 1$ for $0 < \alpha_i < C, \quad \forall x_i$ where $f(x_i)$ is:

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i \langle h(x), h(x_i)\rangle + b \tag{2.43}$$

This means for classification in enlarged feature space, only the inner product of the enlarged features, that is kernel function $K(x_i, x_j) = \langle x_i, x_j\rangle$, is required, and the transformation of features $h(x_i)$, itself is not needed. The most common kernel function choices are:

$$K = (1 + \langle x_i, x_j \rangle)^d \tag{2.44}$$
$$K = \tanh(\kappa_1 \langle x_i, x_j \rangle + \kappa_2) \tag{2.45}$$
$$K = exp(-\gamma \|x_i - x_j\|^2) \tag{2.46}$$

2.5 Kalman Filter for Fault and Cyber-attack Detection

For the estimation of faults and cyber-attacks on the plant side and control side, we use a Kalman filter-based estimator. For the system described by equations (2.19), we assumed that measurements noises and modeling uncertainties can be expressed by uncorrelated white noises with known covariance of R and Q respectively with zero-mean, hence:

$$\begin{aligned} E[w_k w_j^T] &= Q\delta_{k-j} \\ E[v_k v_j^T] &= R\delta_{k-j} \\ E[v_k w_j^T] &= 0 \end{aligned} \tag{2.47}$$

where $E[x]$ is the expected value of a random number x, superscript ^{T} means transpose of a matrix or vector, and δ_{k-j} is Kronecker delta function, which means the value of the function is 1 if $k = j$ and zero for all other values. The initial condition for the Kalman filter is shown by $\hat{x}_0$ and is set equal to the expected value of the initial state x_0 that is:

$$\hat{x}_{0|0} = E(x_0)$$

Covariance of estimation error is denoted by P and is given by:

$$\begin{aligned} P_{k|k-1} &= E[(x_k - \hat{x}_{k|k-1})(x_k - \hat{x}_{k|k-1}^T)] \\ P_{k|k} &= E[(x_k - \hat{x}_{k|k})(x_k - \hat{x}_{k|k}^T)] \end{aligned}$$

If the initial states are unknown, we can start with $x_0 = 0$ and then calculate the estimates. Time updates of the state estimation and covariance of estimation error are values of those parameters before new data is available. For time update, we need to use the model of the system, and for an LTI system presented by equation (2.19), time update of state x and covariance P is as follows:

$$\hat{x}_{k|k-1} = A\hat{x}_{k-1|k-1} + Bu_k \tag{2.48}$$

$$P_{k|k-1} = AP_{k-1|k-1}A^T + Q \tag{2.49}$$

When new measurements y_k become available, x and P are updated with that new measurements. Measurement updates of Kalman filter is as follows:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(y_k - C\hat{x}_{k|k-1}) \tag{2.50}$$

$$K_k = P_{k|k-1}C^T(CP_{k|k-1}C^T + R)^{-1} \tag{2.51}$$

$$\begin{aligned} P_{k|k} &= P_{k|k-1} - K_kCP_{k|k-1} \\ &= (I - K_kC)P_{k|k}(I - K_kC)^T + K_kRK_K^T \end{aligned} \tag{2.52}$$

K_k in the above equation is known as Kalman gain. In the above equations, updated values of $P_{k|k-1}$,$P_{k|k}$, and K are only functions of system parameters, which for the LTI systems are constant values. Therefore, we can calculate Kalman gain in advance, which is an advantage, particularly for a system with limited computation power.

2.5.1 Kalman Filter Based Fault and Cyber-attack Detection

The most common method of fault and cyber-attack detection in the literature is residual generation based on the states or outputs of the system. Evaluations of these residuals can help to find out about reasons for deviations. One of the most common evaluation methods of residuals is the χ^2 detector. Other approaches include ML-based classification, such as SVM and ANN.

In general, Kalman gain converges exponentially, and if (A, B) is stabilizable and (A, C) is detectable, the Kalman filter can be assumed to be in a steady state before the cyber-attack or fault occurrence, that is:

$$P \triangleq \lim_{k \to \text{inf}} P_{k|k-1} \tag{2.53}$$

$$K \triangleq \lim_{k \to inf} K_k = PC^T(CPC^T + R)^{-1} \tag{2.54}$$

The estimation residual of measurement for Kalman filter can be calculated as:

$$\begin{aligned} z_k &= y_k - \hat{y}_k = y_{k+1} - C\hat{x}_k \\ &= y_{k+1} - C(A\hat{x}_k + Bu_k) \end{aligned} \tag{2.55}$$

Fault or Cyber-attack Detection Using χ^2

Chi-square distributions are continuous probability distributions used for hypothesis testing. The shape of the χ^2 distribution is determined by the degrees of freedom (k). The χ^2 is defined as follows:

$$\chi^2 = \Sigma_i \frac{(y_i - \hat{y}_i)^2}{\sigma_i^2} \tag{2.56}$$

where y_i is the measured value, $\hat{y}_i$ is expected value and σ_i is uncertainty of measurement. From equation (2.56), we can see that χ^2 is a special type of weight least square where weights are uncertainty of measurement [16].

The procedure for testing hypothesis H_0, is known as Pearson's test where the computed χ^2 is compared with the critical vaues c_α as follows [99, 100]:

$$\begin{aligned} &\chi^2 \geq c_\alpha\text{: } H_0 \text{ is rejected} \\ &\chi^2 < c_\alpha\text{: } H_0 \text{ is not rejected} \end{aligned}$$

Researchers in [101] proved that the χ^2 test can be effectively applied to residuals of the Kalman filter to detect deviation from normal operating conditions. It should be noted that χ^2, is a detector, and does isolates faults or cyber-attack signals [101]. The procedure for detection of deviation of residuals from normal operating condition due to either fault or cyber-attack used in the Chapter 4 is based on evaluation of residuals using χ^2.

The estimated residual z_k in equation (2.55) without cyber-attack has a white Gaussian distribution of $N \sim (0, CPC^T + R)$, so to detect cyber-attack using χ^2 detector, in each time step the quadratic equation of residual is calculated as follows:

$$g_k = z_k^T \Sigma_k^{-1} z_k \tag{2.57}$$

where Σ_k is covariance of residuals, given by:

$$\Sigma_k = CP_{k|k}C + R \tag{2.58}$$

The test results in a scalar value of g_k, which then needs to be compared against a threshold to see if cyber-attack or fault has happened [101]. The chi-square distribution table consists of critical values that are necessary for conducting hypothesis tests using the chi-square distribution. The critical values help determine whether the observed data falls within the expected range or not. Additionally, the table is used to establish confidence intervals for various parameters. It serves as a refernce table to define thresholds for testing hypothesis making decisions based on statistical data [102].

Chapter 3

Support Vector Machines for Detection of Faults and Cyber-attacks

3.1 Introduction

THE ubiquitous of the Internet and emergence of new technologies such as IoT, 5G, and Internet-connected controllers and devices affects how industrial units handle their data. Isolated units are now connected to the Internet and can generate and store enormous amounts of data. These phenomena result in industrial big data, which allows industrial systems to easily integrate and access commercially available analytics tools such as machine learning algorithms. Careful analysis of these data can gain insights into the performance of those industrial processes and later can help identify areas for improvement or discover problems. Industrial big data has numerous applications, including quality control, predictive maintenance, fault diagnosis, and more. Another notable application of big data is for cyber-attack detection on CPSs. However, cyber-attack detection using machine learning has some challenges. One challenge is poorly mined data that can be ineffective for threat analysis. Another challenge is the continuous and fast evolution of cyber-attacks, which makes it challenging for detection models to catch up. Additionally, machine learning methods for cyber-attack detection require training data to go through a series of preprocesses before being used in model training.

Among the most popular data-driven approaches to fault diagnosis and cyber-attack detection are machine learning algorithms such as support vector machines (SVM) and neural networks. Support vector machines (SVM) are most suitable for classification, regression, and outlier detection. As explained in section 2.4, SVM works by finding an optimal boundary, named hyperplane, between data points of two predefined classes. To properly train an SVM model, high-quality label data is required. SVM is specifically effective in high dimensional spaces, even in cases where the number of dimensions exceeds the number of samples. By analyzing large amounts of data, SVM can help detect system vulnerabilities and cyber-attacks that are becoming prevalent. SVM analysis can discover information in real time, which can help detect cyber threats and develop security solutions accordingly. In this chapter, we studied data drive fault and cyber-attack detection using combination of Kalman filtering and SVM. We will compare the performance of the developed method with a linear Kalman filter-based fault detector as a reference.

This chapter is about the problem of data-driven fault and cyber-attack detection on UAVs as an example of a cyber-physical system. Therefore, we need to generate quality labeled data to train our models. We mathematically defined and modeled actuator faults and cyber-attacks and implemented them in a UAV simulator. In the following sections, we explained the mathematical models of the faults, cyber-attacks, the linear model of the system, and the procedure of generating data.

3.2 Mathematical Model of Faults and Cyber-attacks

We produced the labeled training data with a high-fidelity simulator developed by the University of Minnesota lab [103]. We simulated faults and cyber-attacks in simulation and designed a random flight scenario to generate the required data. The details of the UAV model, sensors, and governing equations are available in section 2.2.1.

We implemented a fault detector on the plant side and a cyber-attack detector on the controller side. Apart from the SVM detector, we also designed a linear Kalman filter-based fault detector as a reference to evaluate the performance of the SVM models. We obtained a linearized model for the UAV using Matlab linearization functions applied to the nonlinear UAV model. Since the full state linear model of UAV consists of thirteen states, $\phi, \theta, \psi, p, q, r, u, v, w, X_e, Y_e, Z_e, \omega$, the use of this full state model is not suitable. Therefore, we used two decoupled linear models, one for the longitudinal and the other for the latitudinal states of the UAV. The latitudinal model has five states, side velocity ($v(m/s)$), roll rate ($p(rad/s)$), yaw rate ($r(rad/s)$), roll angle ($\phi(rad)$), and yaw angle ($\psi(rad)$), two inputs, aileron (δ_a) and rudder (δ_r) angle and five outputs, side-slip angle ($\beta(rad/s)$), roll rate ($p(rad/s)$), yaw rate ($r(rad/s)$), roll angle ($\phi(rad)$), and yaw angle ($\psi(rad)$). The linearized latitudinal state matrices of UAV are:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) \end{aligned}$$

$$x = \begin{bmatrix} v \\ p \\ r \\ \phi \\ \psi \end{bmatrix}, y = \begin{bmatrix} \beta \\ p \\ r \\ \phi \\ \psi \end{bmatrix}, u = \begin{bmatrix} \delta_a \\ \delta_r \end{bmatrix}$$

$$A = \begin{bmatrix} 0.8124 & 0.0988 & -1.1420 & 0.8940 & 0 \\ -0.0687 & 0.2716 & 0.5187 & -0.0475 & 0 \\ 0.0523 & -0.0417 & 0.5115 & 0.0285 & 0 \\ -0.0048 & 0.0568 & 0.0354 & 0.9982 & 0 \\ 0.0029 & -0.0030 & 0.0745 & 0.0010 & 1 \end{bmatrix}$$

$$B = \begin{bmatrix} 0.4551 & 1.4430 \\ -3.4080 & 0.1656 \\ -0.6142 & -1.0280 \\ -0.1988 & 0.0198 \\ -0.0367 & -0.0568 \end{bmatrix} \quad and \quad C = \begin{bmatrix} 0.059 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 1.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 1.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 1.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 1.00 \end{bmatrix} \tag{3.1}$$

The longitudinal model has six states, forward velocity ($u(m/s)$), vertical velocity ($w(m/s)$), pitch rate ($q(rad/s)$), pitch angle ($\theta(rad)$), vertical position ($Z_e(m)$) and motor speed ($\omega(rad/s)$), two inputs, elevator angle (δ_e) and throttle (δ_t) and five outputs airspeed ($V_s(m/s)$), angle of attack $\alpha(rad)$, pitch rate ($q(rad/s)$), pitch angle ($\theta(rad)$), and altitude ($h(m)$). The linearized longitudinal state matrices of UAV are:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) \end{aligned}$$

y

$$x = \begin{bmatrix} u \\ w \\ q \\ \theta \\ Z_e \\ \omega \end{bmatrix}, y = \begin{bmatrix} V_s \\ \alpha \\ q \\ \theta \\ h \end{bmatrix}, u = \begin{bmatrix} \delta_e \\ \delta_t \end{bmatrix}$$

$$A = \begin{bmatrix} 0.9442 & 0.0453 & -0.0142 & -0.9526 & -4.744e-06 & 0.0009522 \\ -0.0558 & 0.3278 & 0.1972 & 0.0199 & -5.889e-05 & -3.597e-05 \\ 0.0154 & -0.1064 & -0.0228 & -0.0059 & 1.105e-05 & 9.227e-06 \\ 8.619e-4 & -0.0118 & 0.0239 & 0.9998 & 5.073e-07 & 3.382e-07 \\ -0.0060 & 0.0718 & -0.0110 & -1.6980 & 1 & -2.695e-06 \\ 9.9250 & 0.4112 & -0.0467 & -5.4031 & -0.006279 & 0.5591 \end{bmatrix}$$

$$B = \begin{bmatrix} 0.04747 & 0.1326 \\ -2.537 & -0.003494 \\ -2.491 & 0.0008482 \\ -0.2021 & 2.229e-05 \\ 0.02417 & -0.0002043 \\ -0.1282 & 189.8 \end{bmatrix} \quad and \quad C = \begin{bmatrix} 0.9998 & 0.02175 & 0.00 & 0.00 & 0.00 & 0.00 \\ -0.001279 & 0.05881 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & -1.00 & 0.00 \end{bmatrix} \tag{3.2}$$

Actuator Faults Model

Faults of the aileron and rudder are detectable using the latitudinal model, and the longitudinal model can help to detect elevator faults. As presented in section 2.3, modeled faults include both additive faults for lock-in-place and multiplicative faults for loss-of-functionality of control surfaces. Hence the system model with faults can be represented as:

$$\begin{aligned} x(k+1) &= Ax(k) + B(Eu(k) + f) + w(k) \\ y &= Cx(k) + v(k) \end{aligned}$$

In the above equation, E and f are fault matrices for multiplicative and additive faults, respectively, explained in section 2.3.1. Each type of actuator fault is simulated separately in some randomly chosen time stamps. We simulated the loss-of-functionality by reducing the range of deflation of the actuator by 80%. We simulated the lock-in-place by setting the actuator position to a constant value. Elements of matrix E are between 1 and 0, and elements of vector f are a constant value representing the angle where that control surface is locked. Hence, the control command $u(k) = Eu(k) + f$ is implemented in the simulator to reflect these faults. For instance, in the loss-of-functionality of the aileron in the latitudinal model, we have:

$$u(k) = \begin{bmatrix} 0.2 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \delta_a \\ \delta_r \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

And the lock-in-place fault for the rudder is simulated as:

$$u(k) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \delta_a \\ \delta_r \end{bmatrix} + \begin{bmatrix} 0 \\ c \end{bmatrix}$$

where c is a constant that defines the position of the rudder. Our fault detector objective is to detect and isolate these faults in any of the actuators with either longitudinal or latitudinal models.

Model of FDI Cyber-attack on Actuator

Cyber-attack detection is done on the controller side since we assumed the plant does not decide the legitimacy of received control inputs. Also, we assume all required inputs and states are available to the controller. In this chapter, only cyber-attacks on actuators are studied. On the controller side, both cyber-attacks and faults impact the system. Based on the model of FDI cyber-attack presented in Section 2.3.2, by removing the cyber-attack signals from measurements, the model of the system for FDI cyber-attack in the presence of faults is as follows:

$$\begin{aligned} x(k+1) &= Ax(k) + B_2(E_a u(k) + u_a(k)) + F(k) + w(k) \\ y &= Cx(k) + v(k) \end{aligned}$$

For additive cyber-attack E_a is identity matrix and $u_a(k)$ is:

$$u_a(k) = \begin{bmatrix} u_{a1}(k) \\ u_{a2}(k) \end{bmatrix} \tag{3.3}$$

where $u_{a1}(k)$ and $u_{a2}(k)$ are cyber-attack signals for latitudinal and longitudinal commands sent from the controller. For multiplicative cyber-attack $u_a(k)$ is zero, and E_a is:

$$E_a = \begin{bmatrix} c_1 & 0 \\ 0 & c_2 \end{bmatrix} \tag{3.4}$$

where c_1 and c_2 are constants. where E_a and $u_a(k)$ can take any values. For a healthy system with no faults, matrix E is identity, and $u_a(k)$ and F are zero.

Replay Cyber-attack Model

The replay cyber-attack is simulated by removing the actual outputs of the system and replacing them with the previous time outputs. Meanwhile, we injected a cyber-attack on the system inputs. Hence, a replay cyber-attack on the measurements of the system, as explained in section 2.3.3, can be presented as:

$$y_a = y(k_1) - Cx(k) - v(k)$$

where k_1 is the time stamp of a previous state of the system. To simulate this cyber-attack, we replaced the previous outputs of the system with the current outputs.

3.2.1 Kalman Filter and χ^2 Detector

Model-based fault detection methods generally are associated with residual generation, where states of system are compared with estimation of healthy system states and residuals generated are used for detection and isolation of faults. The Kalman filter with χ^2 detector is among the most popular methods, and in this chapter is considered as a baseline for evaluation of performance of proposed methods. Two pair of Kalman filter-based estimators and associated χ^2 detector are designed for evaluation of residuals of both latitudinal and longitudinal models. The covariance matrix Q for χ^2 detector for latitudinal and longitudinal residuals are calculated based on healthy conditions of the UAV and are presented below:

$$Q_{lat} = \begin{bmatrix} 7.98e-05 & -1.39e-05 & -3.12e-05 & -6.51e-07 & -1.51e-06 \\ -1.39e-05 & 2.86e-04 & 3.35e-05 & 1.47e-05 & 1.22e-05 \\ -3.12e-05 & 3.35e-05 & 6.72e-05 & 1.96e-06 & 2.31e-06 \\ -6.51e-07 & 1.47e-05 & 1.96e-06 & 9.99e-07 & 7.55e-07 \\ -1.51e-06 & 1.22e-05 & 2.31e-06 & 7.55e-07 & 1.80e-06 \end{bmatrix} \tag{3.5}$$

$$Q_{long} = \begin{bmatrix} 9.06e-02 & -1.07e-03 & 5.95e-04 & 2.79e-05 & 2.32e-03 & 2.15e-04 \\ -1.07e-03 & 1.62e-04 & -3.23e-05 & -1.40e-06 & -2.06e-04 & -2.51e-06 \\ 5.95e-04 & -3.23e-05 & 6.10e-05 & 5.40e-07 & 5.11e-05 & 1.39e-06 \\ 2.79e-05 & -1.40e-06 & 5.40e-07 & 6.60e-08 & 1.65e-06 & 6.56e-08 \\ 2.32e-03 & -2.06e-04 & 5.11e-05 & 1.65e-06 & 2.02e+00 & 2.64e-03 \\ 2.15e-04 & -2.51e-06 & 1.39e-06 & 6.56e-08 & 2.64e-03 & 3.96e-06 \end{bmatrix} \tag{3.6}$$

The threshold is calculated based on critical value of χ^2 explained in section 2.5.1.

3.2.2 SVM classification

The fault and cyber-attack detection studied in this chapter are primarily based on the support vector machines (SVM) with Radial Basis Function (RBF) kernel, also known as Gaussian kernel. We trained a bank of SVMs for detection and isolation of faults for each actuator. We trained SVMs with datasets generated by the simulator for different fault scenarios. Tuning of SVM involves choosing a cost parameter (c), which is a non-negative value, and γ, a positive constant used with the RBF kernel only. Parameter c is a measure of violation of margins by support vectors. A small value of the cost parameter means a wide margin with many support vectors on the margin or violating the margin. A large value of the cost parameter indicates a narrower margin with few support vectors violating the margin or on the margin. Increasing the cost can reduce the error in training, but it may lead to overfitting. Parameter γ controls the smoothness of the decision boundary and the variance of the model. A small value of the γ will create a decision boundary more like a line, while a high value of γ will create a more complex decision boundary that can better fit the training data. Increasing the cost will result in more irregular boundaries. Hence, a set of cost and γ values are tested when training an SVM model, and a k-fold cross-validation finds the best matches for parameters. The error takes the form of [98]:

$$CV_{error} = \frac{1}{n}\sum_{i=1}^{n} Err_i \tag{3.7}$$

where Err_i is misclassified observation, n in number of observation and CV_{error} is variance of cross-validations.

We used R statistical software for all the results in this section [104]. We saved the data generated in the Matlab/Simulink simulator and used them in R to train SVMs. We followed the best practice for training machine learning methods for our models by randomly dividing the dataset into 75% of training and 25% of testing data. After training the SVMs, the model is tested with the test dataset not included in the training. SVM is a binary classifier that classifies data into two classes. Therefore, to isolate faults from each other, we used a bank of SVMs and trained three SVM for each actuator fault against the healthy data, then trained another three SVMs to isolate faults of any of the actuators. We also trained an SVM to isolate the fault of the rudder from the aileron since the two actuators share the same features.

3.2.3 Metric for Evaluation of Results

To summarize the prediction performance of a machine learning method a common method is to use a confusion matrix. A confusion matrix displays the number of correct and incorrect predictions made by the model on a test set. For a binary classification such as SVM, the confusion matrix includes the following four parameters.

- True positives (TP): correct prediction of a positive observation
- True negatives (TN): correct prediction of a negative observation.
- False positives (FP): Incorrect prediction of a negative observation.
- False negatives (FN): Incorrect prediction of a positive observation.

A confusion matrix shows the number of each of the above metrics in a table. The most used metric for the accuracy of classifications is the F1-score, which is the harmonic mean of precision and recall. Precision is the fraction of relevant instances among all retrieved instances and recall is the fraction of retrieved instances among all relevant instances. The formula for precision and recall is as follows:

$$Precision = \frac{TP}{TP + FP} \tag{3.8}$$
$$Recall = \frac{TP}{TP + FN} \tag{3.9}$$

where TP is true positive, TN is true negative, FP is false positive, and FN is false negative. For binary classification F1-score can be calculated as:

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{3.10}$$

3.3 Simulation of UAV Flight for Data Generation for SVM Training

To train SVMs, we need to have labeled training data. For the subject of this chapter, the training dataset should contain labeled observations of healthy, faulty, and under cyber-attack states of the system. We used a high-fidelity UAV simulator developed by the University of Minnesota in Matlab/Simulink [103]. We updated some custom blocks of the simulator to make it work with the current versions of Matlab/Simulink. To generate the required data to train SVMs, we added several components to simulate actuators' faults and cyber-attacks. We also implemented Kalman filters-based detectors for latitudinal and longitudinal models for fault and cyber-attack detection. We then designed a flight scenario described later in this section and saved all control inputs, measurements, and state estimates of the Kalman filter to files to train SVMs in R.

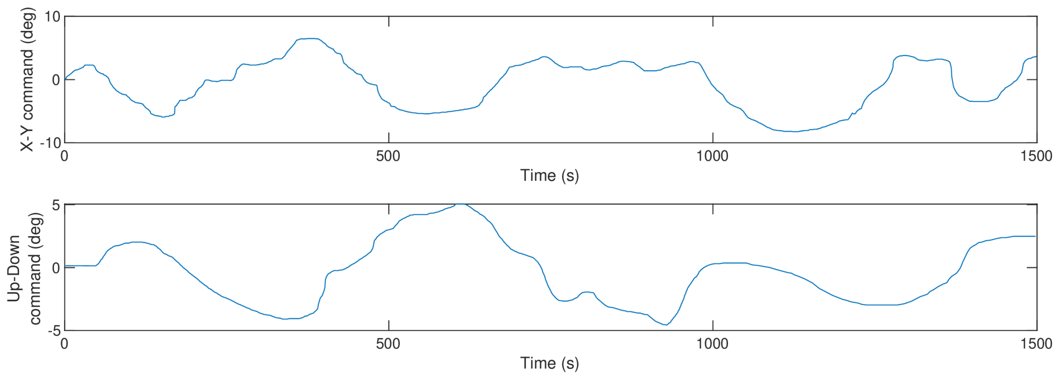


Figure 3.1: UAV ground control system commands.

To generate proper training data for actuator faults, we designed a flight scenario in which the faulty actuator fails to follow control inputs, resulting in a different response expected from a healthy system. The designed flight scenario continuously issues random time-varying control inputs that change the actuator's position for all time steps. These control inputs are shown in Figure 3.1. The applied commands are in the form of latitudinal and longitudinal directions that result in an arbitrary path shown in Figure 3.2.

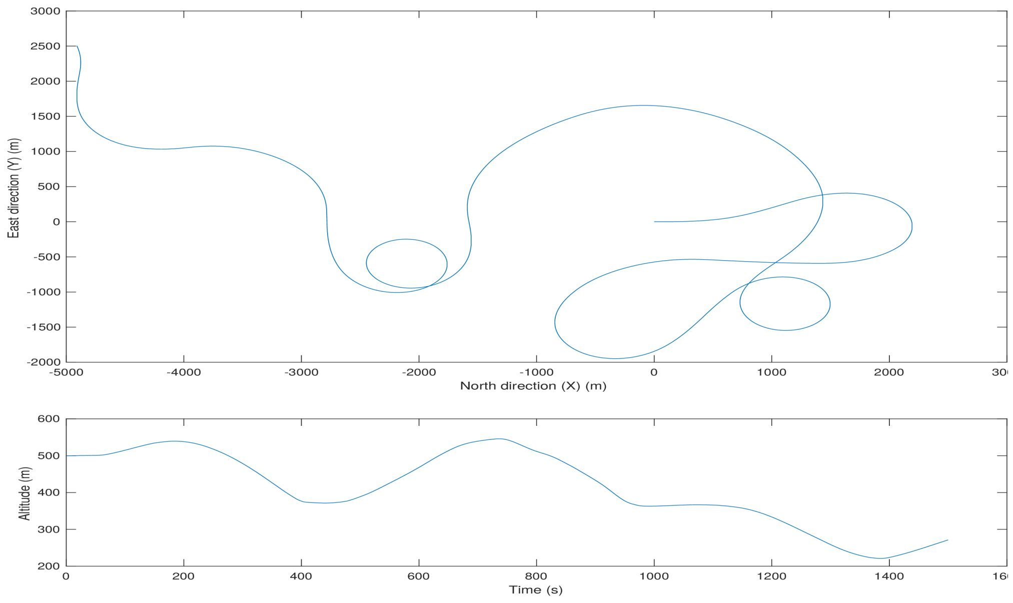


Figure 3.2: UAV flying path.

We simulated cyber-attacks with the same simulator of the UAV described for faults. We applied cyber-attack signals to the ground control commands. The replay cyber-attacks, in general, are launched when the system is in steady-state conditions. For a flying UAV, where some states are continuously changing, the notion of steady-state does not fit well. Therefore, we used an equilibrium condition known as the trim condition to launch a replay cyber-attack. The trim condition in a UAV refers to a state where the UAV is in stable flight, meaning all forces and moments are balanced, and the UAV maintains a steady flight path without requiring continuous control inputs. That involves setting the control surfaces to specific angles to achieve equilibrium. Therefore, to simulate a trim condition, all actuators are set to a constant value such that the UAV flies at a fixed altitude and in a straight line.

In this chapter, we used physical watermarking to detect replay cyber-attacks. Watermarking is a popular technique used by many researchers for replay cyber-attack detection, where in addition to the control commands, a zero mean noise, as a detection mechanism, is added to the control commands. We generated watermarked signals with a zero-mean random signal added to the commands. To simulate a replay cyber-attack, previously recorded outputs are replaced with the current outputs. All datasets generated by the simulator for fault detection include all raw sensor readings, Kalman filter estimates, and control inputs. For FDI and replay cyber-attacks, datasets contain all those data except raw sensor readings. We explained all the sensor readings and states of the UAV in section 3.4.1.

The generated data for the actuator's faults include six different faults simulated in two different flight scenarios, which account for about 50 minutes of simulated flight. Simulated faults are actuators lock-in-place and loss-of-functionality that we applied to the aileron, rudder, and elevator. We applied faults one by one for each actuator in some time stamps and then removed the fault and applied another fault. The duration of each fault is different, but we simulated each faulty state for at least a hundred seconds. Data generated for FDI cyber-attacks include different additive and multiplicative cyber-attacks for about 75 minutes of simulated flight and the replay cyber-attack simulated with about 5 minutes of recorded outputs.

3.4 Fault Detection on UAV Using Kalman Filtering and SVM

For fault detection and isolation, in this chapter, we combined the Kalman filter with the SVM to detect and isolate faults. We also implemented two alternative approaches to compare the results. Those approaches are a model-based linear Kalman filter with χ^2 detector and an SVM model trained solely on raw data. A χ^2 detector triggers an alarm for the existence of faults when residuals violate the baseline defined for detection. The χ^2 is a detector that we use to test the presence of any anomaly, including faults, and it can not isolate different faults from each other or faults from cyber-attacks [101]. While SVM, as an example of a powerful classification method, can detect and isolate faults, in the presence of a cyber-attack, the performance of SVM will degrade, or even it becomes impossible to isolate faults from cyber-attacks. Another issue is the measurement noises, which exist in all industrial systems and can substantially deteriorate the performance of any detection approach, including machine learning methods. Therefore, we will use the Kalman filter to reduce noise and obtain the best estimation of the states of the system. Then, we will train SVM models for the detection and isolation of the faults and the isolation of cyber-attacks from faults. The schematic of the system we used for fault detection is shown in Figure 3.3.

The Kalman filter-based method involves residuals generation by comparing the best

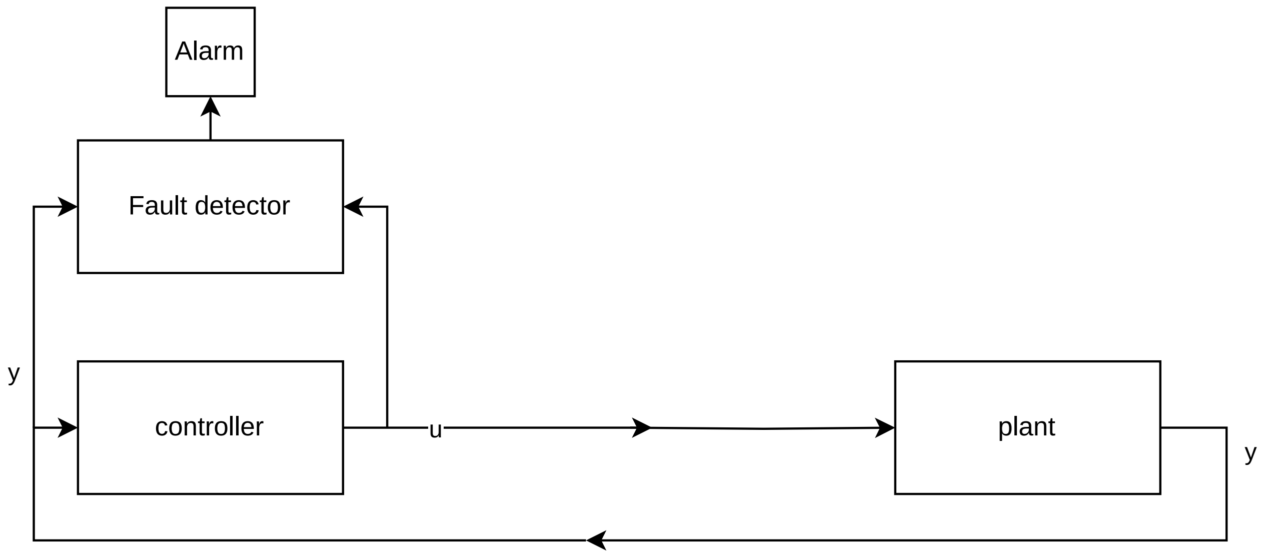


Figure 3.3: Schematic of the system for fault and cyber-attack detection.

estimate of the states using a Kalman filter with the expected estimation of states of the healthy system using the model of the system updated with a priopi information only [101]. We then use these residuals to detect and isolate faults. Based on the Kalman filter equations explained in 2.5, the time updates or a priopi equations are as follows:

$$\hat{x}_{k|k-1} = A\hat{x}_{k-1|k-1} + Bu_k \tag{2.48}$$
$$P_{k|k-1} = AP_{k-1|k-1}A^T + Q \tag{2.49}$$

When new measurements become available, the estimated states and covariance of estimation are updated with that new measurements. Measurement updates of the Kalman filter are as follows:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(y_k - C\hat{x}_{k|k-1}) \tag{2.50}$$
$$K_k = P_{k|k-1}C^T(CP_{k|k-1}C^T + R)^{-1} \tag{2.51}$$
$$\begin{aligned} P_{k|k} &= P_{k|k-1} - K_kCP_{k|k-1} \\ &= (I - K_kC)P_{k|k}(I - K_kC)^T + K_kRK_K^T \end{aligned} \tag{2.52}$$

We may consider two extreme cases for the Kalman filter. One case is when we have little or no trust in the system model, which means $Q \to \infty$, therefore from equation (2.49), $P_{k|k-1} \to Q$, which means R is negligible compared to $P_{k|k-1}$ in equation (2.51), hence:

$$\begin{aligned} K_k &= P_{k|k-1}C^T(CP_{k|k-1}C^T + R)^{-1} \\ &= P_{k|k-1}C^T(CP_{k|k-1}C^T)^{-1} \\ &= P_{k|k-1}C^T(C^T)^{-1}P_{k|k-1}^{-1}C^{-1} \\ &= C^{-1} \end{aligned}$$

Therefore equation (2.50) becomes:

$$\begin{aligned} \hat{x}_{k|k} &= \hat{x}_{k|k-1} + C^{-1}(y_k - C\hat{x}_{k|k-1}) \\ &= \hat{x}_{k|k-1} + C^{-1}y_k - \hat{x}_{k|k-1} \\ &= C^{-1}y_k \end{aligned}$$

Hence, when there is little or no information about the model of the system, the Kalman filter will only use the measurements to update the states. However, if we have an accurate model of the system, the Q will be small, and the states will get the best estimate based on both measurements and the model. The other case is when we have no trust in the measurements, for instance, when we suspect that sensors or actuators may be faulty or tampered with. In this case we may assume that $R \to \infty$, and hence, in the equation (2.51) we have:

$$\begin{aligned} K_k &= P_{k|k-1}C^T(CP_{k|k-1}C^T + R)^{-1} \\ &= P_{k|k-1}C^T R^{-1} \\ &= 0 \end{aligned}$$

Therefore:

$$\begin{aligned} \hat{x}_{k|k} &= \hat{x}_{k|k-1} + 0.(y_k - C\hat{x}_{k|k-1}) \\ &= \hat{x}_{k|k-1} \\ &= A\hat{x}_{k-1|k-1} + Bu_k \end{aligned}$$

Therefore, the Kalman filter will get updates solely on the a priori information, meaning that the estimation will be based on the model of the system only. To distinguish a priori estimate, we will note the a priori estimation as $\tilde{x}_{k|k}$, and for notation convenience, we will note $\tilde{x}_{k|k}$ by $\tilde{x}(k)$. The estimated states of a Kalman filter that updates with both the model and measurements are as follows:

$$\begin{aligned}
\hat{x}(k+1|k) &= A\hat{x}(k) + Bu(k) \\
\hat{x}(k+1) &= \hat{x}(k+1|k) + K[y(k+1) - C\hat{x}(k+1|k)] \\
&= A\hat{x}(k) + Bu(k) + K[Cx(k+1) + \nu(k+1) - C(A\hat{x}(k) + Bu(k))] \\
&= (A - KCA)\hat{x}(k) + Bu(k) - KCBu(k) + KC(Ax(k) + Bu(k) + w(k)) + K\nu(k+1) \\
&= (A - KCA)\hat{x}(k) + Bu(k) + KCAx(k) + KCw(k) + K\nu(k+1) \qquad (3.11)
\end{aligned}$$

The estimation error of the Kalman filter, then is calculated as $e(k+1) = x(k+1) - \hat{x}(k+1)$ as follows:

$$\begin{aligned}
e(k+1) &= x(k+1) - \hat{x}(k+1) \\
&= Ax(k) + Bu(k) + w(k) - [(A - KCA)\hat{x}(k) + Bu(k) + KCAx(k) + KCw(k) + K\nu(k+1)] \\
&= (A - KCA)x(k) - (A - KCA)\hat{x}(k) - K\nu(k+1) + (I - KC)w(k) \\
&= (A - KCA)e(k) - K\nu(k+1) + (I - KC)w(k)
\end{aligned}$$

For the Gaussian process with measurement noises, the expected values of noises are 0, that is $\mathbb{E}[w(k)] = 0$ and $\mathbb{E}[\nu(k+1)] = 0$. For a properly designed Kalman filter $A - KCA$ is stable, and therefore as times progress estimation error $e(k+1)$ approaches zero, that is:

$$\lim_{k\to\infty} \mathbb{E}[e(k+1)] = (A - KCA)\mathbb{E}[e(k)] = 0 \qquad (3.12)$$

For the two estimator configurations, we will calculate the error dynamics of estimation as $\tilde{e}(k+1) = \hat{x}(k+1) - \tilde{x}(k+1)$, and when there is no fault or cyber-attack using equation (3.11) we have:

$$\begin{aligned}\tilde{e}(k+1) &= \hat{x}(k+1) - \tilde{x}(k+1)\\ &= (A-KCA)\hat{x}(k) + Bu(k) + KCAx(k) + KCw(k) + K\nu(k+1) - A\tilde{x}(k) - Bu(k)\\ &= A(\hat{x}(k) - \tilde{x}(k)) + KCA(x(k) - \hat{x}(k)) + K\nu(k+1) + KCw(k)\\ &= A\tilde{e}(k) + KCAe(k) + K\nu(k+1) + KCw(k)\end{aligned}$$

And with the same justifications as for the equation 3.12, for a properly designed Kalman filter the expected value of estimation error will be:

$$\lim_{k\to\infty} \mathbb{E}[\tilde{e}(k+1)] = A\mathbb{E}[\tilde{e}(k)] - KCA\mathbb{E}[e(k)] = 0 \tag{3.13}$$

The disadvantage of this approach is that for a long-running process with disturbances, the a priori model of the system will deviate from the actual states of the system. Therefore, we need to periodically reset the a priori model by the actual states of the system. The researchers in [105] suggested an optimal time interval for resetting the model with the actual states of the system. The residuals of a Kalman filter detector are calculated as $z(k) = y(k) - \hat{y}(k)$ as follows:

$$\begin{aligned}z(k) &= y(k) - \hat{y}(k)\\ &= y(k) - C[A\hat{x}(k-1) + Bu(k-1)]\end{aligned} \tag{3.14}$$

In the same way, we can define residuals of the two-estimator detector as $\tilde{z}(k) = \hat{y}(k) - \tilde{y}(k)$ as follows:

$$\begin{aligned}\tilde{z}(k+1) &= \hat{y}(k+1) - \tilde{y}(k+1)\\ &= C\hat{x}(k+1) - C\tilde{x}(k+1)\\ &= C[\hat{x}(k+1) - \tilde{x}(k+1)]\\ &= Ce(k+1)\end{aligned} \tag{3.15}$$

The residuals in this approach are proportional to state estimation error. Therefore, the residuals are generated based on the Kalman filter best-estimated states of the system and the expected states obtained from the model of the system. An advantage of this approach is that we can generate residuals based on the states that are not measured. This property of residuals is crucial for fault diagnosis of complex systems like gas turbines where some essential states of the engine for fault detection, such as efficiency and capacity, are not measurable and need to be estimated. To show how these estimates change with faults, we will derive the estimation error of the two estimators for a faulty system. For a faulty system, form section 2.3.1, additive and multiplicative actuator faults can be modeled as:

$$u_f(k) = Eu(k) + f$$

where $u_f(k)$ is the actual response of faulty actuators to the control input, and $u(k)$ is the control input issued by the controller. The parameters E and f are the effects of multiplicative and additive actuator faults. Additive sensor faults are modeled as follows:

$$y_f(k) = y(k) + \Delta y_f(k)$$

The estimated states of a faulty system are:

$$\begin{aligned}
\hat{x}_f(k+1|k) &= A\hat{x}_f(k) + Bu(k) \\
\hat{x}_f(k+1) &= \hat{x}_f(k+1|k) + K[y_f(k+1) - C\hat{x}_f(k+1|k)] \\
&= \hat{x}_f(k+1|k) + K[y(k+1) + \Delta y_f(k+1) - C\hat{x}_f(k+1|k)] \\
&= A\hat{x}_f(k) + Bu(k) + K[Cx_f(k+1) + \nu(k+1) + \Delta y_f(k+1) - C(A\hat{x}_f(k) + Bu(k))] \\
&= A\hat{x}_f(k) + Bu(k) + K[C(Ax_f(k) + Bu_f(k) + w(k)) + \nu(k+1) + \Delta y_f(k+1) \\
&\quad - C(A\hat{x}_f(k) + Bu(k))] \\
&= (A - KCA)\hat{x}_f(k) + Bu(k) - KCBu(k) + KC[Ax(k) + B(Eu(k) + f) + w(k)] \\
&\quad + K\Delta y_f(k+1) + K\nu(k+1) \\
&= (A - KCA)\hat{x}_f(k) + KCAx_f(k) + Bu(k) - KCB(I - E)u(k) \\
&\quad + KCBf + K\Delta y_f(k+1) + KCw(k) + K\nu(k+1)
\end{aligned} \tag{3.16}$$

The error dynamics of the two estimators in the presence of both additive and multiplicative actuator faults and additive sensor faults are:

$$\begin{aligned}
\tilde{e}_f(k+1) &= \hat{x}_f(k+1) - \tilde{x}(k+1) \\
&= (A - KCA)\hat{x}_f(k) + KCAx_f(k) + Bu(k) - KCB(I-E)u(k) \\
&\quad + KCBf + K\Delta y_f(k+1) + KCw(k) + K\nu(k+1) - A\tilde{x}_f(k) - Bu(k) \\
&= A[\hat{x}_f(k) - \tilde{x}_f(k)] + KCA[x_f(k) - \hat{x}_f(k)] + KCBf - KCB(I-E)u(k) \\
&\quad + K\Delta y_f(k+1) + KCw(k) + K\nu(k+1) \\
&= A\tilde{e}_f(k) + KCAe_f(k) + KCBf - KCB(I-E)u(k) \\
&\quad + K\Delta y_f(k+1) + KCw(k) + K\nu(k+1) \qquad (3.17)
\end{aligned}$$

and the residuals are calculated as follows:

$$\begin{aligned}
\tilde{z}_f(k+1) &= \hat{y}_f(k+1) - \tilde{y}(k+1) \\
&= C\hat{x}_f(k+1) - C\tilde{x}(k+1) \\
&= C[\hat{x}_f(k+1) - \tilde{x}(k+1)] \\
&= Ce_f(k+1) \qquad (3.18)
\end{aligned}$$

The residuals for the faulty system are also proportional to state estimation error, and from equation (3.17), we can see if there are no multiplicative actuator faults, then $E = I$ and $KCB(I-E)u(k) = 0$, and if there is no additive fault, then $f = 0$ and therefore $KCDf = 0$. On the other hand, if there are no sensor faults, the term $K\Delta y_f(k+1)$ vanishes. Therefore, equations (3.17) and (3.18) for a healthy system will have an expected value of zero, as shown in the below equation:

$$\begin{aligned}
\mathbb{E}[e](k+1) &= A\mathbb{E}[\tilde{e}(k)] + KCA\mathbb{E}[\hat{e}(k)] + KC\mathbb{E}[w(k)] + K\mathbb{E}[\nu(k+1)] \\
&= A\mathbb{E}[\tilde{e}(k)] + KCA\mathbb{E}[\hat{e}(k) = 0
\end{aligned}$$

To detect faults we can use a classical χ^2 on residuals, where violation of a linear threshold, as described in section 2.5, is a sign of a fault. The χ^2 can handle Gaussian noises and can be considered as a type of weight list-square, where the weights are uncertainty of measurement as shown by equation (2.57) below:

$$g_k = z_k^T \Sigma_k^{-1} z_k \tag{2.57}$$

where Σ_k is the covariance of residuals that normalizes residuals. It is worth mentioning that the χ^2 can only detect deviation of residuals from the healthy system and cannot isolate faults. We will use SVM with the Kalman filter as an alternative to χ^2 and residual generation to detect and isolate faults. To detect anomalies and outliers, a popular method is the use of one-class SVM. The one-class SVM is a variant of SVM used for the detection of observations that deviate from normal behavior. Compared with χ^2, SVM can handle non-Gaussian noises with nonlinear decision boundaries. To handle complex nonlinear classification problems with SVM, kernel function is used. The kernel function plays a crucial role in SVMs by transforming the training data into a higher-dimensional feature space. This transformation allows SVMs to find hyperplane, even when the data points are not linearly separable in the original input space. Commonly used kernels include Linear Kernel, Radial Basis Function (RBF) Kernel, Polynomial Kernel, and Sigmoid Kernel. However, similar to χ^2, the normalization of training data is crucial for SVM. There are several approaches to the normalization of data. Considering an SVM with liner kernel as $\kappa(x, x^T) = x.x^T$, where x is the feature vector and x^T is its transpose. A normalization approach for linear kernel is the use of the generalized linear kernel presented as:

$$\kappa(x, x^T) = x^T R x$$

where x is features vector and R is a positive semidefinitive parameter matrix. An approach to train R involves setting R to C^{-1}, where C is covariance matrix of the training data [106]. An improved approach called within-class sets R to W^{-1} where W is the expected within-class covariance matrix over all classes [107]. In [108], researchers used one-class SVM with RBF kernel for cyber-attack detection. They normalized the data by the covariance of data as $x_n = \Sigma^{-\frac{1}{2}} x$, where x_n is normalized features, x is original features, and Σ is the covariance of data points. Another popular approach is normalization and standardization, where the data is scaled into the $[0, 1]$ and shifted to zero-mean. We used the last method for normalization since we can scale normal operating condition data with different measurement units around zero and train SVMs to classify faults based on their deviation from base data. In this chapter, we use the radial basis kernel, a.k.a. the Gaussian kernel, to handle nonlinear boundaries. Assuming that we are training SVM with state matrix $x_{n \times p}$ from Kalman filter innovations, a kernel function κ maps input features, which in this case are normalized states, to higher dimension space f that is $x(k) \to f(k)$ as shown below:

$$\Phi(x(k)) = \kappa(x_k, x_i) = \exp(-\gamma \sum_{j=1}^{p} (x_{kj} - x_{ij})^2)$$

where x_k and x_i are two observations, which in the case of UAV, are some subset of measurements, states, and control inputs chosen as features of two different time stamps k and i, and γ is a hyperparameter that controls the smoothness of the decision boundary and the variance of the model. The term $\sum_{j=1}^{p}(x_{kj} - x_{ij})^2$ represents the squared Euclidean distance between observations x_k and x_i. The parameter γ determines the influence of each observation on the kernel function. Smaller γ values lead to a smoother decision boundary, while larger values result in a more complex boundary. Then from equation (3.19), SVM solves the following optimization problem:

$$\begin{aligned} \underset{w, b, \xi}{\text{minimize}} \quad & \frac{1}{2} w^T w + C \sum_{i=1}^{n} \xi_i \\ \text{subject to} \quad & y_i(w^T . \Phi(x_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0 \quad i = 1, \ldots, n \end{aligned} \tag{3.19}$$

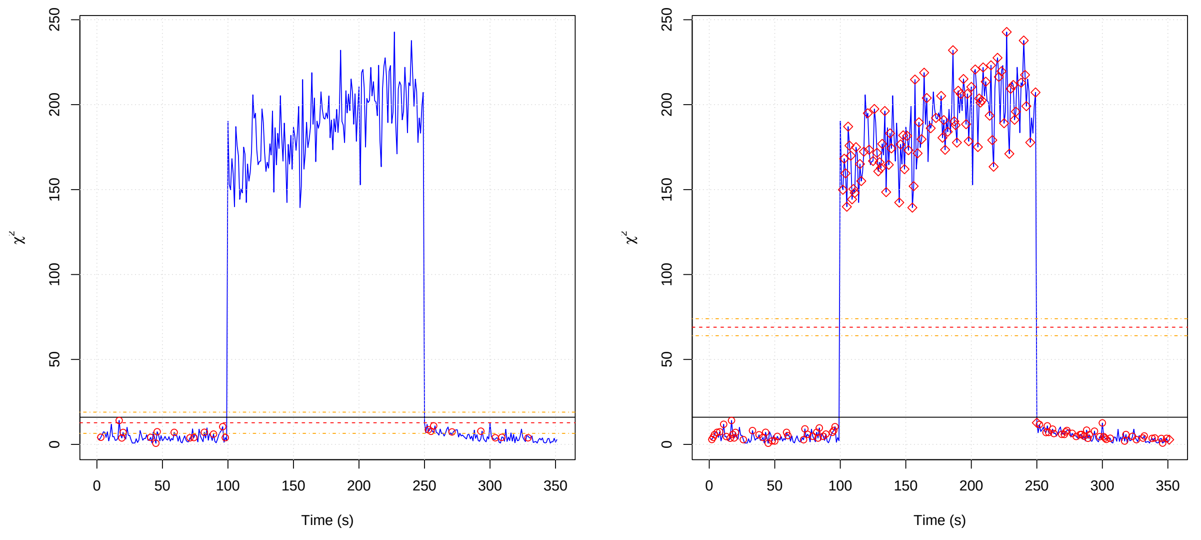
where w is a parameter that defines the linear decision boundary, the slack variable ξ_i is proportional to the amount of misclassification for a value of $\xi_i > 0$, and C is a cost function that limits the total number of misclassifications. The optimal value of C is estimated by cross-validation. The linear boundary in the infinite-dimensional space of the radial basis kernel defines a nonlinear boundary in the original feature space that can separate faults with nonlinear behavior from the healthy system. In the above optimization formulation, the parameter C can take any value, which can be problematic for some training processes. An alternative optimization formulation uses a parameter $\nu \in (0, 1]$ to control the number of support vectors [109] as shown below:

$$\begin{aligned} \underset{w, b, \xi, \rho}{\text{minimize}} \quad & \frac{1}{2} w^T w - \nu\rho + \frac{1}{n} \sum_{i=1}^{n} \xi_i \\ \text{subject to} \quad & y_i(w^T . \Phi(x_i) + b) \geq \rho - \xi_i, \\ & \xi_i \geq 0, \quad i = 1, \ldots, n, \quad \rho \geq 0 \end{aligned} \tag{3.20}$$

When there are few outliers in the dataset and the objective is to detect those outliers, as mentioned above, one-class SVM proposed by [109] can be a good approach. In this approach, SVM is trained only by the normal data points, and hence a boundary is defined around the training set, and any violation of the boundary is considered an outlier. The one-class SVM solves the following optimization problem:

$$\begin{aligned} \underset{w,\xi,\rho}{\text{minimize}} \quad & \frac{1}{2}w^T w - \rho + \frac{1}{\nu n}\sum_{i=1}^{n} \xi_i \\ \text{subject to} \quad & w^T.\Phi(x_i) \geq \rho - \xi_i, , \\ & \xi_i \geq 0, \quad i = 1, \ldots, n, \end{aligned} \tag{3.21}$$

To demonstrate how the SVM works, we compared the χ^2 threshold with SVM generated boundary. We used a simulated flight scenario with a single fault with the latitudinal model of the Kalman filter as explained in section 3.3. We used the one-dimensional matrix of features with n samples, that is $\mathbf{g} \in \mathbb{R}^{1\times n}$, calculated by equation (2.57) as the training set. The results are shown in Figure 3.4.



(a) One-class SVM classification compared with χ^2 threshold

(b) Two-class SVM classification compared with χ^2 threshold

Figure 3.4: One-class SVM shown in the left-hand side plot (a) is trained with healthy system residuals, while the two-class SVM, in the right-hand side plot (b), is trained with both the healthy and the faulty data. The red circles in the plots are support vectors for the healthy class. The support vectors of the faulty class are shown by the diamond symbol in the right-hand side plot (b).

The blue line in the plots in Figure 3.4 is χ^2 data that we used to train SVMs. Figure 3.4a shows one-class SVM, where red circles are the support vectors, the red dashed line is the decision boundary of one-class SVM, and the black line is χ^2 threshold. The area between two orange dash-doted lines is the margin of the SVM boundary, where misclassification is accepted. One-class SVM is trained only with healthy system data, so support vectors are picked up from healthy data only. One-class SVM draws a boundary around the training data points and marks any value that violates the boundary as an outlier. It is best for anomaly and outlier detection when the model of anomalies is unknown. It also works best when the number of anomalies is much less than the normal data. However, training the one-class SVM with noisy data is challenging and could cause many false negative responses. We can see this from the decision boundary margin shown by two orange dash-dotted lines. Figure 3.4b shows a two-class SVM trained with both healthy and faulty data. The red circles are support vectors of the healthy class, and the red diamonds are support vectors of the faulty class. The decision boundary, shown with a red dashed line is positioned at a maximum distance from the two classes and the margin is shown by two orange dash-dotted lines. The decision boundary is well above the χ^2 threshold shown in the black line and separates healthy and faulty data points.

Researchers in [108] trained several one-class SVM to detect and isolate cyber-attacks on a sensors of connected vehicle system. A problem with one-class SVM for fault detection is the narrow boundary that one-class SVM draws around any class. For instance, a one-class SVM trained to detect a specific type of actuator fault will show poor performance when the magnitude of the fault even changes slightly. Moreover, for UAVs, several factors can reduce the performance of one-class SVM, including disturbances due to wind and measurement noise. The input commands and transition states of the system are another issue. On the other hand, we have an accurate model of the system and mathematically defined fault models. Hence, we can effectively train two-class SVM with Kalman filter states to detect and isolate faults with few misclassifications.

The SVM can detect patterns in data without any model and performs well with raw sensor readings. In [110], researchers trained an SVM with accelerometer and inertia system measurement and combined them with control inputs as features for fault detection and isolation. However, they get inconsistent results with different datasets of the same system [110], which is most likely the result of the noisy measurements, which is an issue specifically with low-end sensors used with small UAVs. Another problem with this approach is the choice of features or the best set of measurements. Industrial systems, including UAVs, have several correlated measurements, and manually choosing the best set of features can be time-consuming and inaccurate. On the other hand, the model of the plant is generally available for the industrial system, and the use of Kalman filtering for removing measurement noises is a de facto in industry. These observations led us to combine a model-based Kalman filter with two-class SVM as a powerful fault detection mechanism. Apart from reducing noises, the Kalman filter can estimate the best set of states of the system, eliminating the feature selection step.

It is worth noting that, the SVM is a binary classifier and cannot classify more than two classes. Since SVM does not naturally extend to more than two classes, to solve multi-class classification problems, several proposals were made. The two most popular ones are one-versus-one and one-versus-all. In the One-versus-one approach, $k(k-1)/2$ binary classifiers are trained, and the appropriate class is found by a voting scheme. In this chapter, we did not use multi-class classification for fault detection. This is because including all the data in the training set increases the size of the dataset, and SVM training is slow on large datasets. This makes the training process too long and ineffective. On the other hand, adding any new fault not included in the training set requires retraining of SVM, which makes adding new faults much more difficult. Instead, we used a bank of SVMs, much like the concept of the bank of observers in classical model-based fault detection, and trained fault with the relevant set of features against a healthy system. Therefore, we trained three SVMs with three different feature sets each for one of the actuators to detect and isolate actuator faults. We also trained another three SVMs to isolate two different faults of each actuator.

To summarize, we used the Kalman filter to reduce the noise of the measurement system and used the Kalman filter estimate states as the best set of features of the UAV. To account for the control inputs on the system's dynamics, in a model-based approach, we incorporate the control inputs into the Kalman filter to obtain the most accurate estimates of the current states. We also incorporate the control inputs in the a priori model to drive the expected states of the healthy system. We have used those states as features to train SVMs in three different setups. In one setup, we generated residuals of the Kalman filter and a priori model estimated states based on the equation (3.18) and used them as features. For the other setup, we used estimated states of the Kalman filter and a priori model as features without generating any residual. For the third setup, we used the Kalman filter estimated states and control inputs as features. We trained a bank of SVMs for any of our proposed setups except for raw data and PCA to compare the results of different approaches.

To compare the performance of our models with one-class SVM used in [110], we also trained a multi-class SVM with accelerometer and inertial system measurements and control inputs as features. To improve and automate the process of feature selection, we used Principal Component Analysis (PCA). Since industrial systems, including UAVs, have many correlated measurements that make manual selection of the best set of features challenging and time-consuming, an automated feature selection is crucial. Therefore, we used PCA to extract features from the UAV sensor readings and then trained as SVM with PCA data as another setup. The procedure to extract features using PCA is explained in the following subsection.

3.4.1 Feature Engineering and Feature Extraction

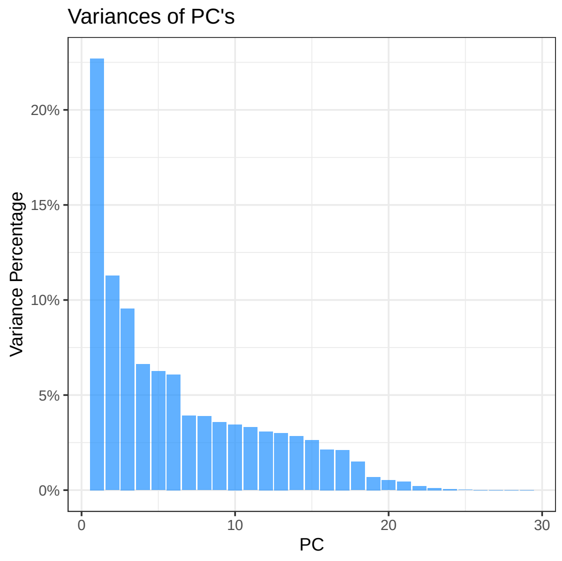
Feature engineering involves selecting and creating new features through methods like feature crossing to obtain the best feature set from the data. Feature crossing is suitable when fitting a linear model like regression to nonlinear data. Then, the nonlinearity can be modeled, for instance, by multiplying some features. SVM uses kernel tricks to handle nonlinearity, which makes feature crossing redundant. However, feature selection and feature extraction are still important. Residuals in a model-based approach may be considered a type of feature engineering, where residuals are generated based on the differences between expected and actual states. Machine learning methods, such as SVM, can recognize patterns in data and classify them accordingly, which makes residual generation less beneficial.

Feature extraction methods like Principal Component Analysis (PCA) can significantly improve the feature selection procedure and help extract the best feature set from correlated data. Several feature extraction methods are available for machine learning, and PCA is one of the best methods [111]. PCA performs well, even compared with methods such as CNN [62].

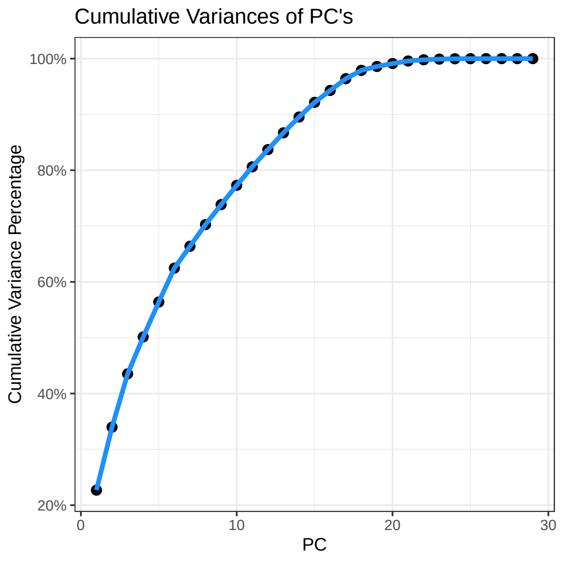
Therefore, we used PCA to extract the best feature set from UAV measurements. The UAV simulator used in this research have several sensors readings, including accelerometer (A_x, A_y, A_z) in m/s^2, Inertial accelerations $(a_{X_e}, a_{Y_e}, a_{Z_e})$ in m/s^2, Angular rates (p, q, r) in rad/s, angular accelerations $(\dot{p}, \dot{p}, \dot{r})$ in rad/s^2, UAV speed (u, v, w) in m/s, air speed (V_a) in m/s, position (X, Y, Z) in m, linear acceleration $(\dot{X}, \dot{Y}, \dot{Z})$ in m/s^2, barometric altitude (h) in m, Euler angles (ϕ, θ, ψ) in rad, Euler angles rate of changes $(\dot{\phi}, \dot{\theta}, \dot{\psi})$ in rad/s, wind axis parameters including angle of attack (α) in rad, angle of sideslip (β) in rad, and rate of changes of those, $\dot{\alpha}$ in rad/s and $\dot{\beta}$ in m/s. UAV position and their derivative in three dimensions are excluded from the dataset primarily because they offer irrelevant information and can deteriorate fault detection performance.

PCA is an unsupervised learning method that helps summarize a large set of correlated data into a smaller feature set that can explain most of the variability in the original data [98]. The main idea is that not all the features have equally important information about any observation, and PCA finds the smaller dimensions of features that can describe each observation. Before applying PCA, it is crucial to scale and center variances of each feature, or else features with lower magnitude, such as roll angle in radian, will not contribute to the resulting principal components.

To demonstrate how PCA works, we constructed a dataset with all sensor readings of about 29 parameters listed above, for the healthy and faulty operating points of the UAV. PCA analysis of this dataset results in another dataset, where features are sorted based on their variances as shown in Figure 3.5a and Figure 3.5b.



(a) Percentage of variances of each principal component.



(b) Cumulative percentage of variances of principal components.

Figure 3.5: Percentage of variance of principal components applied to UAV sensor readings and control inputs.

Figure 3.5a shows the share of variances of each principal component, and Figure 3.5b shows the cumulative share of principal components. From Figure 3.5a, we can see that the majority of features have less than 5% variances, and Figure 3.5b shows that the first six principal components represent about 64% of all variances in the data. Therefore, we may choose the first few principal components to fit a model that captures the majority of variances in the data. When we use principal components as a training dataset of machine learning models, we should apply the same scaling factors to the new observations and extract principal components before classifying them with the ML model.

3.4.2 Results of Simulation of Detection and Isolation of Faults

We used the simulator explained in section 3.3 with random control inputs shown in Figure 3.1. We simulated two different faults for three different actuators. Overall, we simulated six different faulty states. The simulated faults are Lock-in-place and loss-of-functionality of the aileron, rudder, and elevator. We simulated Lock-in-place as a step change where the control surface is locked in a predefined position, and loss-of-functionality is simulated by limiting any control surface actuation to 20% of the default actuation range. Simulated faults are started in a time stamp and finished a few minutes later. Figure 3.6 shows the residuals generated by the Kalman filter, and Figure 3.7 shows the calculated χ^2 based on the residuals for lock-in-place fault.

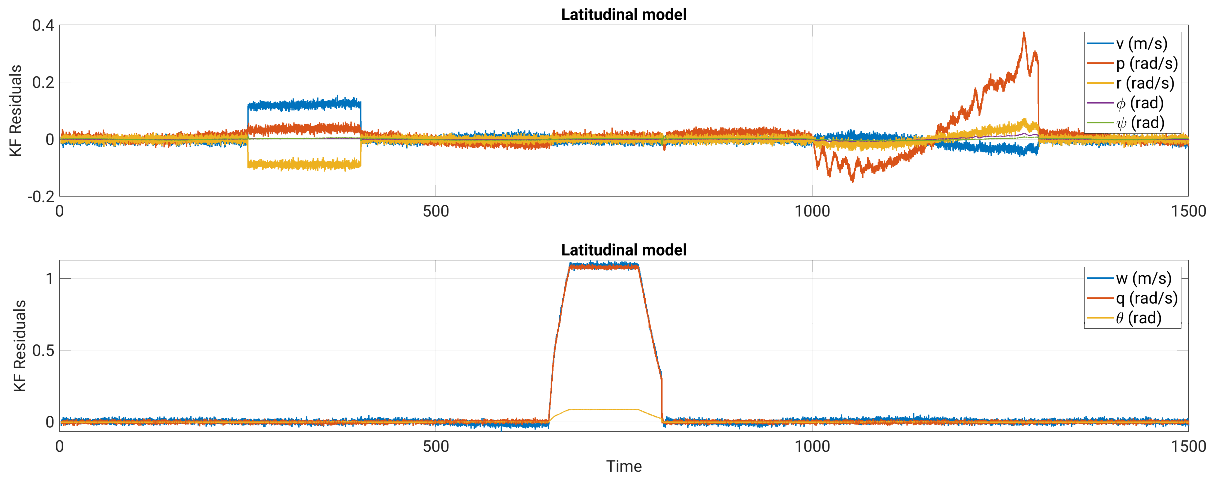


Figure 3.6: Residual generate by Kalman filter for lock-in-place faults. The latitudinal model captures the rudder fault starting at time stamp 250 and ending at time stamp 400, and the aileron fault started at time stamp 1000 and ended at time stamp 1300. The longitudinal model captures elevator faults starting at time stamp 650 and ending at time stamp 400.

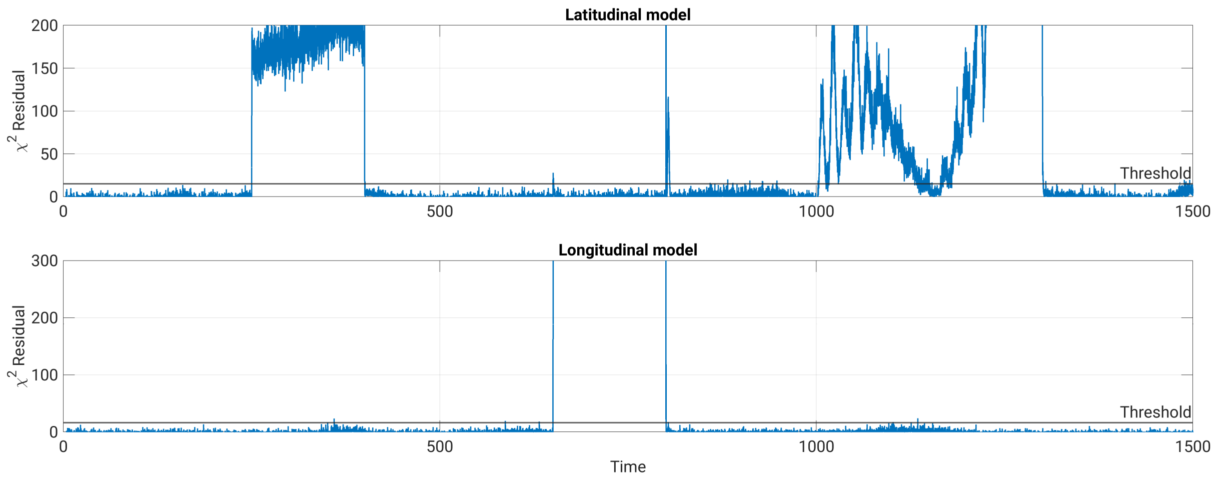


Figure 3.7: χ^2 test for lock-in-place faults of rudder and aileron shown in latitudinal model and elevator faults shown in longitudinal model.

The rudder and aileron fault impacts are visible in the latitudinal model residuals in Figure 3.6, where the rudder fault started at time stamp 250 and ended at time stamp 400. The aileron fault started at time stamp 1000 and ended at time stamp 1300. The longitudinal model shows the elevator faults starting at time stamp 650 and ending at time stamp 400.

For model-based fault detection, we used a χ^2 detector. The covariance matrix of residuals in equations 3.5 and 3.6 are used to calculate χ^2 from latitudinal and longitudinal residuals, respectively. The threshold is calculated based on the critical value of χ^2 explained in section 2.5.1. Figure 3.7 shows that based on this threshold, the χ^2 detector can detect all faults. However, the aileron fault in some time stamps is under the threshold because of the specified maneuver of the UAV. That is, the control input at that operating point is close to the locked position of the aileron.

Simulation of the loss-of-functionality fault is shown in Figure 3.8. For the loss-of-functionality, the latitudinal model residuals are due to the rudder fault starting at time stamp 950 and ending at time stamp 1200, and the aileron fault starting at time stamp 500 and ending at time stamp 650. The longitudinal model is impacted by the elevator fault starting at time stamp 50 and ending at time stamp 200. The χ^2 calculated based on the residuals of loss-of-functionality is shown in Figures 3.8 and 3.9, respectively. With the specified threshold based on the critical values, the χ^2 can also clearly detect this fault.

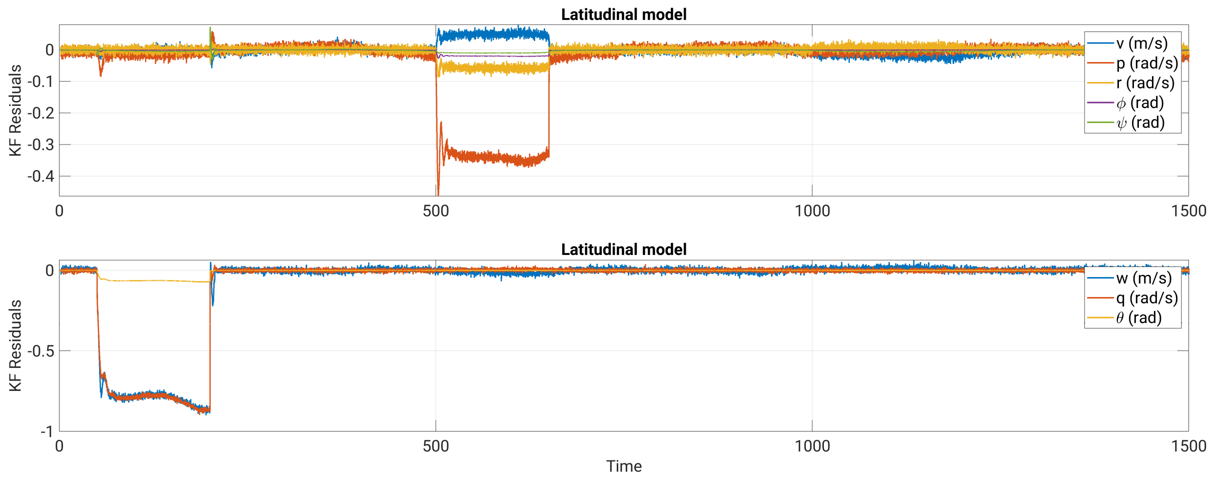


Figure 3.8: Residual generate by Kalman filter for loss-of-functionality faults. Latitudinal model captures rudder fault started at time stamp 950 and ended at time stamp 1200, and aileron fault started at time stamp 500 and ended at time stamp 650. Longitudinal model captures elevator fault started at time stamp 50 and ended at time stamp 200.

The calculated χ^2 of the loss-of-functionality of the rudder is smaller than the other faults and is not visible in Figure 3.9. The zoomed plot of χ^2 is presented in Figure 3.10, and we can see that the residuals violate the threshold of χ^2.

To compare the performance of χ^2 with our proposed method based on SVMs, We trained a bank of SVMs for five different cases by constructing five different datasets with distinct sets of features as follows:

1. *Measurement Data*: This dataset contains accelerometer data in three dimensions (A_x, A_y, and A_z) in m/s^2 and Inertial accelerations in three dimensions (a_{X_e}, a_{Y_e}, and a_{Z_e}), and the three control inputs for three actuators (δ_a, δ_r, and δ_e).

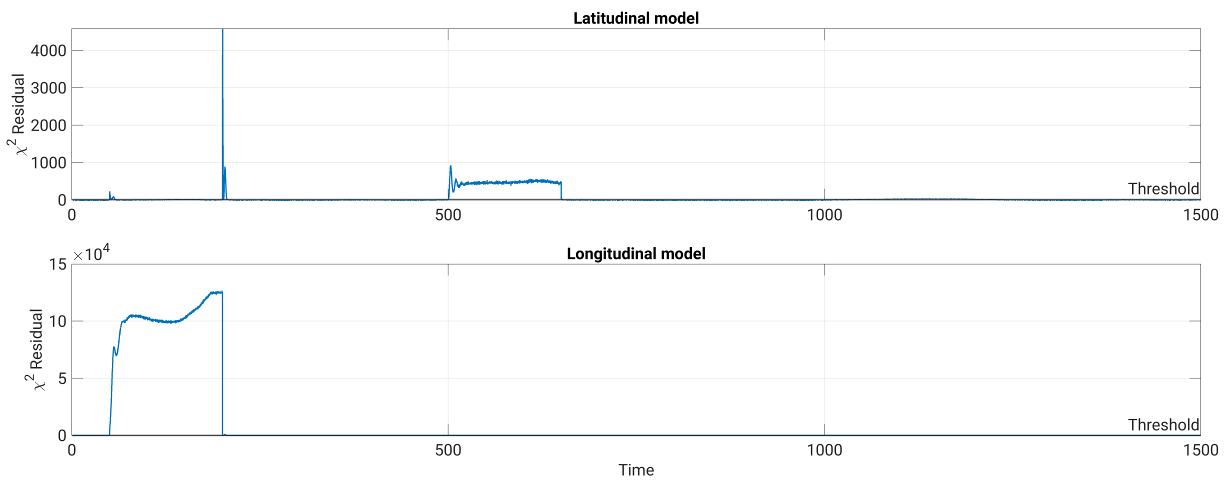


Figure 3.9: χ^2 graph for loss-of-functionality faults.

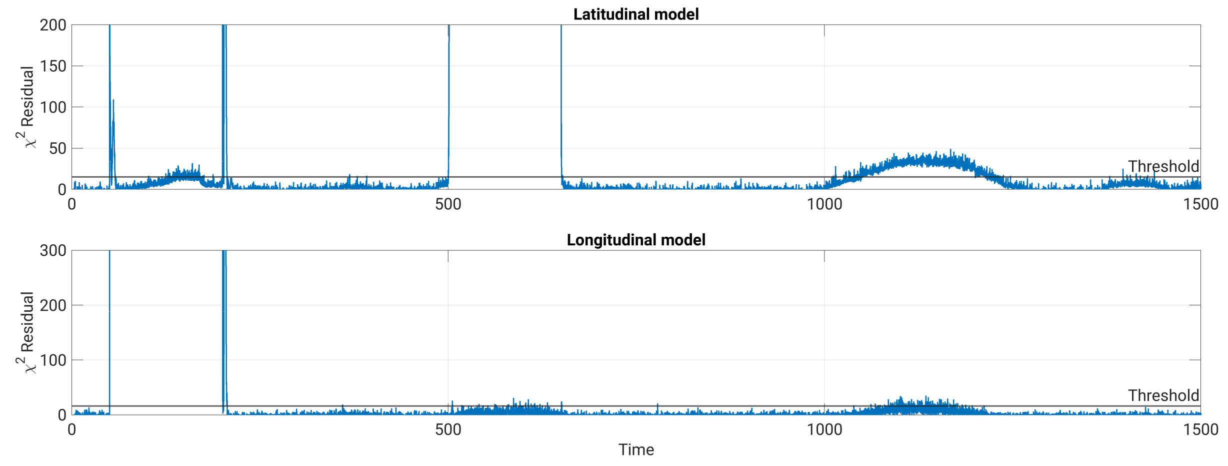


Figure 3.10: Zoomed χ^2 graph for loss-of-functionality faults.

2. *Six PCs & control inputs*: We constructed this dataset in two steps. First, we include all 29 measurements indicated in section 3.4.1, except for the control inputs in a dataset, and extracted the Principal Components. Then, we generated a new dataset by including the first six PCs and three control input commands for three actuators (δ_a, δ_r, and δ_e) and used it for training of SVMs.
3. *Residuals*: This dataset is constructed by generating residuals based on the Kalman filter and a priori model. From equation (3.18), the residuals in this equation are equal to the estimation error of equation (3.17) multiplied by a constant C. We generated the residuals based on this equation (without multiplication by matrix C). We constructed two separated datasets, one for the latitudinal model based on the residuals of five estimated states of side-slip angle ($\beta(rad/s)$), roll rate ($p(rad/s)$), yaw rate ($r(rad/s)$), roll angle ($\phi(rad)$), and yaw angle ($\psi(rad)$). For the longitudinal

model, we included the residuals of five estimated states of airspeed ($V_s(m/s)$), angle of attack $\alpha(rad)$, pitch rate ($q(rad/s)$), pitch angle ($\theta(rad)$), and altitude ($h(m)$).

4. *Kalman filter & a priori model*: We constructed two separate datasets, one for a latitudinal model and one for the longitudinal model, each included Kalman filter estimated states and a proper model estimated states without generating any residual. The dataset for the latitudinal model includes estimated states by two estimators of side-slip angle ($\beta(rad/s)$), roll rate ($p(rad/s)$), yaw rate ($r(rad/s)$), roll angle ($\phi(rad)$), and yaw angle ($\psi(rad)$). We constructed a second dataset for the longitudinal model based on the estimated states of the two estimators of airspeed ($V_s(m/s)$), angle of attack $\alpha(rad)$, pitch rate ($q(rad/s)$), pitch angle ($\theta(rad)$), and altitude ($h(m)$).
5. *Kalman filter & control inputs*: We constructed two separated datasets, one for latitudinal model with Kalman filter estimated states of side-slip angle ($\beta(rad/s)$), roll rate ($p(rad/s)$), yaw rate ($r(rad/s)$), roll angle ($\phi(rad)$), and yaw angle ($\psi(rad)$) and control inputs related to latitudinal model, that is δ_a and δ_r. We constructed the second dataset for the longitudinal model by including the Kalman filter estimated states of airspeed ($V_s(m/s)$), angle of attack $\alpha(rad)$, pitch rate ($q(rad/s)$), pitch angle ($\theta(rad)$), and altitude ($h(m)$) and control input of δ_e.

Table 3.1 lists all faults studied in this chapter.

Abbreviation	System status
Hlty	Healthy system
ALP	Lock-in-place of aileron
ELP	Lock-in-place of elevator
RLP	Lock-in-place of rudder
ALF	Loss-of-functionality aileron
ELF	Loss-of-functionality elevator
RLF	Loss-of-functionality rudder

Table 3.1: List of faults modeled in the simulator.

Bank of SVMs for Isolation of Faults

We used the bank of observers with our proposed Kalman filter features and trained the other two datasets with multi-class SVM. Because when we use the sensor data or PCA, it is impossible to separate latitudinal and longitudinal models; hence, we must include all features in a single large dataset. To train the bank of SVMs, we constructed three datasets, one for each actuator, by including healthy states and each actuator's faulty states. We constructed this dataset for the three setups of the Kalman filter states and residuals.

Therefore, we trained SVMs with much smaller datasets. To isolate any actuator's faults from each other, we trained another SVM between two different faults of an actuator to isolate lock-in-place from loss-of-functionality for each actuator. In our tests, there were misclassifications between faults in rubber and aileron that share the same features. To further isolate these two actuator faults, we trained another SVM only with the faults of the two actuators. Therefore, the detection and isolation procedure of faults with our bank of observers is as follows.

1. We run three SVMs to check if an actuator is faulty.
2. If both aileron and rubber SVMs report a fault, we run another SVM to isolate the actuator.
3. When the faulty actuator is detected, we run another SVM for that actuator to detect the type of fault in the actuator, that is, to distinguish loss-of-functionality from lock-in-place.

Our bank of SVMs consists of seven SVMs, and we executed a maximum of five SVMs to isolate seven different classes. In contrast, the one-versus-one approach creates $n(n-1)/2$ classes and runs all of them to classify. Therefore, if we include all seven classes in a single training dataset, which we had to do for the first two setups, there will be 21 classifications for any observation. Separating latitudinal and longitudinal datasets with multi-class SVMs results in ten classifications for five classes of latitudinal models and three classifications for three classes of longitudinal models, totaling thirteen classifications. In comparison, our bank of SVMs has fewer classifications and performs faster. Additionally, since the training datasets are smaller, tuning of SVMs takes much less time. After tuning our seven SVMs, we tested our procedure with the datasets for all faults and healthy states. The F1-scores are summarized in Table 3.2

		F1-score						
No.	Name of Setup	Hlty	ALP	ELP	RLP	ALF	ELF	RLF
1	Chi-squared (χ^2)	0.95	0.94	0.99	0.99	0.99	0.99	0.55
2	Measurement data	0.95	0.92	0.96	0.97	0.96	0.98	0.72
3	6 PCs & control inputs	0.97	0.98	0.99	0.99	0.99	0.99	0.96
4	Residuals	0.96	0.95	0.99	0.99	0.99	0.99	0.50
5	Kalman filter & a priori model	0.97	0.97	0.99	0.99	0.99	0.99	0.97
6	Kalman filter & control inputs	0.97	0.97	0.99	0.99	0.99	0.99	0.97

Table 3.2: Summary of results of fitting SVM model for different setups.

It seems that generation of residual removes some information from data, since the performance of the SVM is similar to the χ^2 when trained with residuals, while all of the models without generation of residual perform better than the χ^2 and SVM of residual. It is worth noting that the SVM model trained with PCs in the third row performs as well as the SVMs trained with Kalman filter estimated states. This is not surprising since we included all of the measurements and states of UAV in the dataset, and PCA could effectively extract the features from data. However, the process of training and prediction of new observation is more complicated and the training takes longer. Primarily because it is a two step procedure where first we need normalize data and extract PCs, and then add control inputs and normalize and train SVMs. For machine learning methods, data preparation and feature engineering is a crucial step, and generally takes more than half of the time of the all project [112]. Therefore, The use of Kalman filter is a huge improvement, since we already picked up the best features, and Kalman filter prepares data by removing noises. As we can see from Table 3.2, the both models with Kalman filter, have the best performance and since we already prepared two different datasets with distinct features, the datasets are smaller and training process and much shorter. Moreover the SVM models are in general faster with out approach.

3.5 Cyber-attack Detection on UAV Actuators Using SVM

In this section, we will use SVMs to detect cyber-attacks on ground control inputs of a UAV. The ground control system of the UAV simulator that we used for the generation of data has two outputs, one for latitudinal movement in the north-east plain and the other for longitudinal or elevation. We assume that the UAV sends the best estimate of its states by two liner Kalman filters, one for latitudinal and the other for longitudinal states, to the ground control system. We presented the schematic of the system in Figure 3.11

We assume that the attackers inject the FDI cyber-attack signals into the communication channel of the ground control system of the UAV and the replay cyber-attack into the measurements sent from the UAV to the ground control system. First, we will study the problem of cyber-attack detection with SVMs for a healthy system and later will investigate simultaneous fault and cyber-attack detection.

3.5.1 False Data Injection Cyber-attack Detection Using SVM

Researchers employed Support Vector Machines in numerous cyber-attack detection cases in literature. A popular approach to cyber-attack detection is one-class SVM, where an SVM is trained only with healthy system data, and any deviation from a healthy behavior is considered an anomaly. As explained in the previous section, in our test with noisy UAV

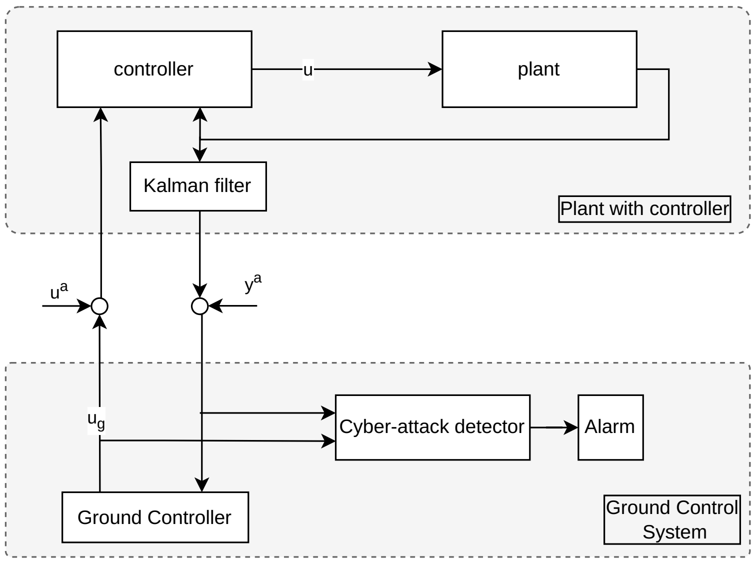


Figure 3.11: Model of the UAV system under cyber-attack

data, one-class SVM had many false positive responses with healthy data. Therefore, we opted out of one-class SVM in favor of two-class SVM for cyber-attack detection. An FDI cyber-attack on a control input is defined as $u'(k) = u(k) + u^a(k)$, where $u'(k)$ is under cyber-attack control input, $u(k)$ is the control input and $u^a(k)$ is a time-varying cyber-attack signal. Therefore, FDI cyber-attacks do not have a fixed definition, so if we train SVM with several predefined scenarios, it will learn the patterns of those specific cyber-attacks and detect them. However, it will not necessarily be able to detect a different cyber-attack scenario. Considering the complex nature of the FDI cyber-attacks and the nonlinear model of the plant, we choose to use the Gaussian Radial Basis Function Kernel (RBF) for SVM. The RBF kernel is presented as follows:

$$\Phi(x(k)) = \kappa(x_k, x_i) = \exp(-\gamma \sum_{j=1}^{p} (x_{kj} - x_{ij})^2) \tag{3.22}$$

where x_k and x_i are two observations that we seek to find the square Euclidean distance between them by term $\sum_{j=1}^{p}(x_{kj} - x_{ij})^2$. From this equation, we can see that when observations are close to each other, the summation term is close to zero, and the exponential term approaches one that is:

$$\Phi(x(k)) = \kappa(x_k, x_i) = \exp(-\gamma \sum_{j=1}^{p} (x_{kj} - x_{ij})^2) \approxeq \exp(0) = 1$$

On the other hand, when the points are far from each other, the Euclidean distance term will grow and the exponential term will approach zero, that is:

$$\Phi(x(k)) = \kappa(x_k, x_i) = \exp(-\gamma \sum_{j=1}^{p} (x_{kj} - x_{ij})^2) \approxeq \exp(-\infty) = 0$$

In other words, the RBF kernel measures the similarity between vectors by using a decaying function of the distance between the vectors (i.e. the squared norm of their distance). When two vectors are close together, their distance ($||x_k - x_i||$) will be small. With $\gamma > 0$, it follows that $-\gamma||x_k - x_i||^2$ will be larger. Thus, closer vectors have a larger RBF kernel value than farther vectors.

Therefore, if we define a minimum value for FDI cyber-attack signals $u^a(k)$, decided by our system noises and disturbances, and assume values smaller than this minimum have a negligible impact on the system, then we can use this minimum to define a boundary around the healthy data. Taking into account that RBF calculates the exponential Euclidean distance, any observation greater than this minimum distance will make the exponential term approach zero and be classified as a cyber-attack. This approach will result in a classification similar to one-class SVM, but we can fine-tune our SVM based on a margin specific to our system to handle noises. For an industrial system such as a UAV, we may calculate a boundary by considering the accuracy of the sensor and actuator. For instance, a general-purpose accelerometer has a typical accuracy of about 0.1 degrees, and hysteresis or bias of electro-mechanical actuators restricts them from accurately following the control commands. With some trial and error on the UAV simulator, we find that changes smaller than 0.3 degrees have an illegible effect on the UAV performance. Therefore, we consider signals smaller than 0.3 degrees as noise. Since the SVM draws the hyperplane with maximum distance from the support vectors of two classes in the training data. This means to classify deviations larger than 0.3 as a cyber-attack on actuators, a 0.6-degree margin needs to be defined for arbitrary cyber-attack signals. Hence, the SVM hyperplane will be 0.3 degrees farther from the healthy data.

For our FDI cyber-attack detection, we will use the Kalman filter states similar to fault detection. However, to design a Kalman filter-based detector for the ground control system, we need to obtain the model of the UAV dynamics and UAV controller as a plant and ground control system commands as control inputs. However, in the previous section, we showed that the Kalman filter estimated states with control inputs as features without a priori model states or other measurements are enough for fault detection and isolation. We also showed that this set of features performs equally well compared with the other setups we tested, and can detect and isolate faults with a very high F1-score. Therefore, to eliminate the need to obtain a new model for the system, we will use Kalman filter estimated states for cyber-attack detection, but this time, we included the ground control system commands as a feature. We constructed two separated datasets, one for a latitudinal model with Kalman filter estimated states of side-slip angle ($\beta(rad/s)$), roll rate ($p(rad/s)$), yaw rate ($r(rad/s)$), roll angle ($\phi(rad)$), and yaw angle ($\psi(rad)$) and latitudinal ground control inputs. We did not include the UAV control inputs, that is, δ_a and δ_r in the training dataset. We constructed the second dataset for the longitudinal model by including the Kalman filter estimated states of airspeed ($V_s(m/s)$), angle of attack $\alpha(rad)$, pitch rate ($q(rad/s)$), pitch angle ($\theta(rad)$), and altitude ($h(m)$) and longitudinal ground control signal and did not include the control input of δ_e.

To generate the training data for SVM, we used about 30 minutes of flight scenario presented in Figure 3.1 of any of the two datasets. Then, we divided each dataset into three randomly selected parts. We injected our artificial cyber-attack signal with a magnitude of 0.6 degrees to a control command for the first slice of the dataset. We injected another cyber-attack signal with a magnitude of -0.6 degrees to the same command for the second slice and kept the third slice as is. A sample plot of the control command of training data is shown in Figure 3.12.

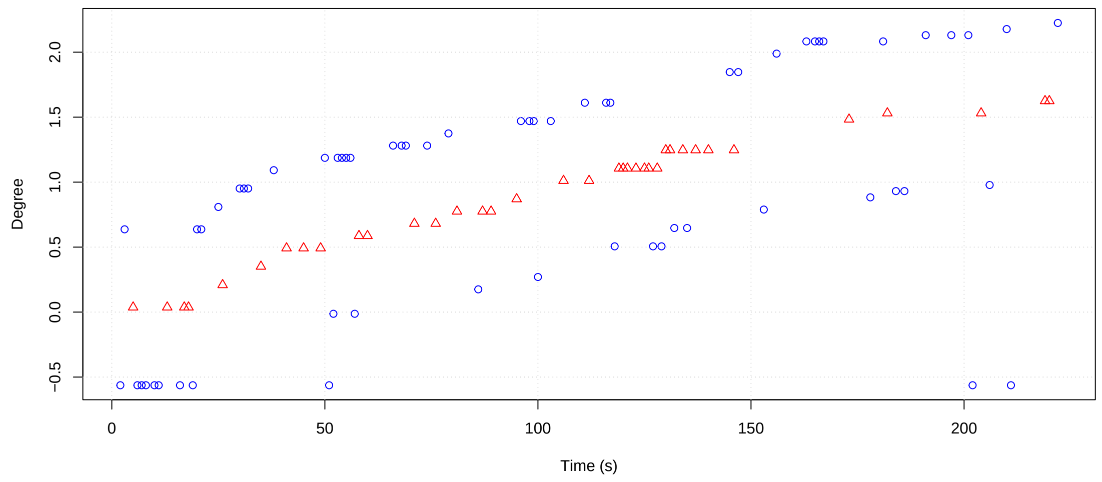


Figure 3.12: A sample of training data for FDI cyber-attack on ground control command. The red triangles are healthy commands, and the blue circles are artificial FDI cyber-attacks on the control input of the ground control system with a magnitude of 0.6 and -0.6 degrees.

We trained a two-class SVM for any of these datasets to detect the FDI cyber-attack on the control input. We then simulated different cyber-attack scenarios and tested the SVMs with the simulated cyber-attacks. We designed two types of FDI cyber-attacks. A step cyber-attack (additive) with different magnitudes, shown in Figure 3.13, which we injected as an additive FDI cyber-attack on a ground control command. Another time-varying additive and multiplicative cyber-attack are shown in Figure 3.14. The blue shaded area in Figure 3.14 refers to additive time-varying cyber-attacks, and red shades refer to multiplicative cyber-attacks. We injected multiplicative cyber-attacks by multiplying the ground control command with 1.5 or 0.5.

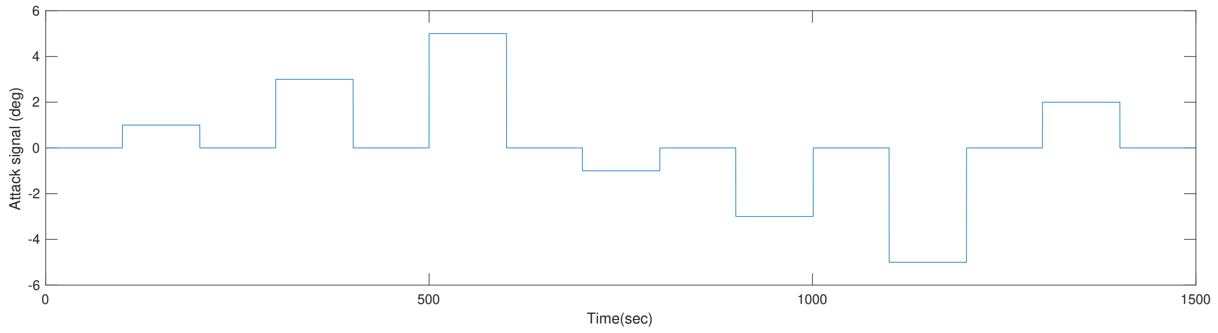


Figure 3.13: FDI cyber-attack signals in the form of steps changes in degree.

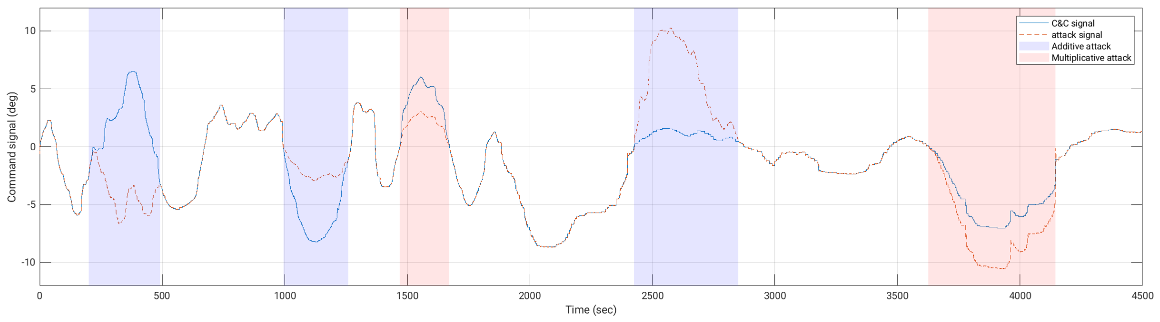


Figure 3.14: FDI cyber-attack signals that gradually changes. Blue area is variable step additive signal and red area is multiplicative cyber-attack.

We also trained one-class SVM with healthy data for comparison. We summarize the results of tests with the datasets based on those two cyber-attack scenarios in Table 3.3. In [108], the researchers used one-class SVM for cyber-attack detection on information exchanged between networks of cars, where there was no control input. However, a challenge for the control and operation of small UAVs is flying in windy conditions. In contrast to autonomous cars or robotics problems, due to variations caused by the unknown and changing effects of the wind, UAV states can not be determined accurately [93]. This results in several false positive classifications of healthy states by one-class SVM in our tests. We handle this problem with our proposed training method.

		F1-score		
No.	Name of Setup	Healthy	Step cyber-attack	Mixed cyber-attack
1	One-Class SVM with Kalman filter	0.91	0.99	0.97
2	Two-class SVM with Kalman filter	0.98	0.99	0.96

Table 3.3: Summary of results of fitting SVM model for different setups.

A fundamental problem with these approaches is that actuator faults most likely will be classified as cyber-attacks. This is due to the similarity of the impacts of faults and cyber-attacks on the same actuator. To handle this problem, we constructed another dataset by adding the faulty data from actuators generated in the previous section as a no-attack to our dataset, then trained the two-class SVM with this new dataset. We tuned SVM to include faulty states as no-attack. We tested the latitudinal SVM model with four faults related to the latitudinal model, under cyber-attack on the latitudinal control command. However, the SVM did not detect small FDI cyber-attacks in the presence of a fault in the same actuator. Increasing cyber-attack signal results in isolation of fault from cyber-attack. Another problem was that there were many misclassifications of healthy observations. We presented the results of this test in Table 3.4, the acronyms of faults are explained in the table 3.1.

	F1-score				
Cyber-attack signal magnitude	Healthy	ALP	RLP	ALF	RLF
$u^a(k) = 0\,deg$	0.91	0.97	0.66	0.66	0.52
$u^a(k) = 1\,deg$	0.86	0.65	0.98	0.98	0.47
$u^a(k) = 2\,deg$	1	0.99	1	1	0.99
$u^a(k) > 2\,deg$	1	1	1	1	1

Table 3.4: Summary of results of simultaneous fault and cyber-attack detection of SVM.

This is an inherent problem with the approach since there is no information about faults in the ground control system. Overall, this approach is fast and effective for healthy systems, but does not have acceptable detection of small cyber-attack signals. To address this problem, in the following subsection, we proposed a method that is more complicated than this approach but can properly isolate faults from cyber-attacks.

3.5.2 Detection of Cyber-attack in Presence of Faults

In the previous section, we showed that if the faulty actuator is under cyber-attack, it is not always possible to detect the cyber-attack by SVM. To solve this problem, we need to have some information about the fault state of the actuator to use it on the ground control system to isolate faults from cyber-attacks. One option can be to use the SVM classification decision value. The SVM classifies two classes based on a decision value between 1 and -1. If an observation is close to support vectors, the decision value will be close to 1 for one class and -1 for the other class, and if it is far from the support vectors, the decision value will be smaller than 1 or greater than -1. Based on equation (3.22), we can observe that the RBF kernel has an exponential equation. Hence, if an observation is farther from the support vectors for a specific value, we will consistently obtain the same decision value. Therefore, the SVM decision value for a faulty actuator that at the same time is under a cyber-attack can be the same for a faulty actuator without a cyber-attack. We can conclude that the decision value of SVM is not an appropriate choice for comparison among two different SVM on the plant side and controller side to isolate faults from cyber-attacks. To solve this problem, we considered the system presented in Figure 3.15 for the isolation of faults on the plant side from cyber-attacks on the communication channel of the ground control system.

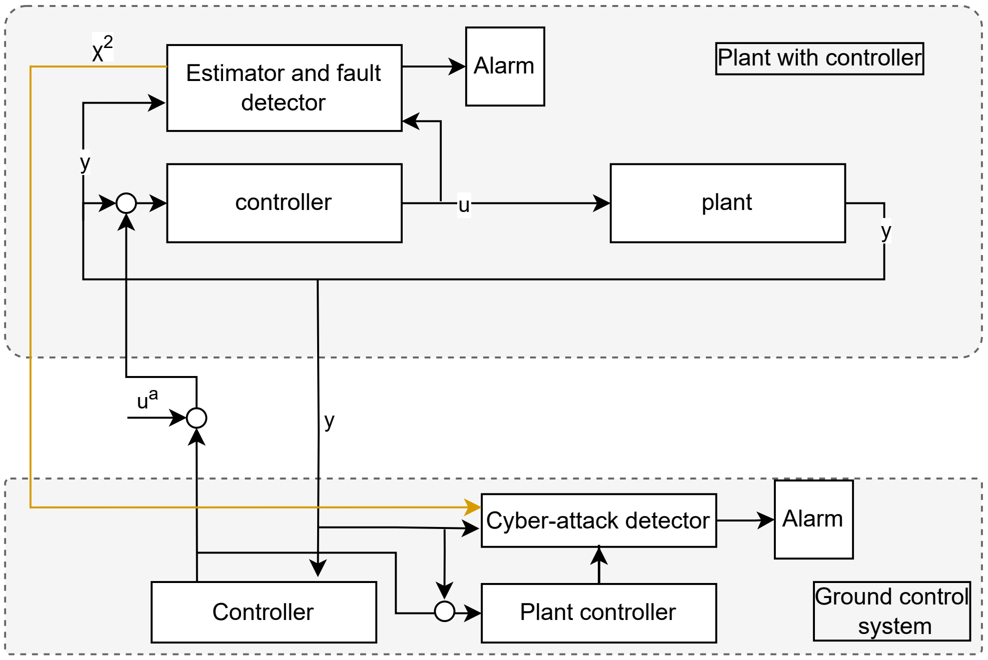


Figure 3.15: Schematic of system for detection and isolation of faults from cyber-attacks

To isolate faults from cyber-attacks, we assume that the plant has a χ^2 based fault detector explained in section 3.4, and send its measurements and the calculated χ^2 to the ground control system. Further, we assume that the ground control system has a replica of the control system of the UAV and can re-generate all the control inputs of the UAV based on the measurements sent from the UAV and ground control system inputs. Then, based on the measurements and control input, it can calculate the best estimates of the states of the system by a Kalman filter and also calculate expected states based on the a priori model explained in the previous section. We will use this information to calculate the ground controller side χ^2 based on the estimated states of the Kalman filter and estimated states of the a priori model to compare it with the received plant side χ^2. Therefore, in the presence of both fault and cyber-attacks, estimated states of the Kalman filter on the ground control side are calculated as:

$$\begin{aligned}
\hat{x}_f^a(k+1|k) &= A\hat{x}_f^a(k) + Bu(k) \\
\hat{x}_f^a(k+1) &= \hat{x}_f^a(k+1|k) + K[y_f^a(k+1) - C\hat{x}_f^a(k+1|k)] \\
&= A\hat{x}_f^a(k) + Bu(k) + K[y(k+1) + \Delta y_f(k+1) + y^a(k+1) + \nu(k+1) \\
&\quad - C(A\hat{x}_f^a(k) + Bu(k))] \\
&= (A - KCA)\hat{x}_f^a(k) + Bu(k) - KCBu(k) + KC[Ax_f^a(k) + B(E(u(k) + u^a(k)) \\
&\quad + f) + w(k)] + K\nu(k+1) + Ky^a(k+1) + K\Delta y_f(k+1) \\
&= (A - KCA)\hat{x}_f^a(k) + KCAx_f^a(k) + Bu(k) - KCB(I - E)u(k) + K\nu(k+1) \\
&\quad + KCw(k) + KCBEu^a(k) + Ky^a(k+1) + KCBf + K\Delta y_f(k+1) \qquad (3.23)
\end{aligned}$$

where $\{.\}_f^a$ means faulty states under cyber-attack, $u^a(k)$ is a cyber-attack signal on the control inputs, $y^a(k)$ is a cyber-attack signal on the measurements, and all of the other symbols used in this equation have the same definition as explained in section 3.4. The estimated states of the a priori model are:

$$\tilde{x}(k+1) = A\tilde{x}(k) + Bu(k)$$

And the error dynamics for a faulty system under cyber-attack is as follows:

$$
\begin{aligned}
\tilde{e}^a_f(k+1) &= \hat{x}^a_f(k+1) - \tilde{x}(k+1) \\
&= (A - KCA)\hat{x}^a_f(k) + KCAx^a_f(k) + Bu(k) - KCB(I-E)u(k) + K\nu(k+1) \\
&\quad + KCw(k) + KCBEu^a(k) + Ky^a(k+1) + KCBf + K\Delta y_f(k+1) \\
&\quad - A\tilde{x}(k) - Bu(k) \\
&= A(\hat{x}^a_f(k) - \tilde{x}(k)) + KCA(x^a_f(k) - \hat{x}^a_f(k)) + KCB(I-E)u(k) + K\nu(k+1) \\
&\quad + KCw(k) + KCBEu^a(k) + Ky^a(k+1) + KCBf + K\Delta y_f(k+1) \\
&= A\tilde{e}^a_f(k) + KCAe^a_f(k) + KCB(I-E)u(k) + K\nu(k+1) \\
&\quad + KCw(k) + KCBEu^a(k) + Ky^a(k+1) + KCBf + K\Delta y_f(k+1) \qquad (3.24)
\end{aligned}
$$

The residuals of the faulty system under cyber-attack become:

$$
\begin{aligned}
\tilde{z}^a_f(k+1) &= \hat{y}^a_f(k+1) - \tilde{y}(k+1) \\
&= C\hat{x}^a_f(k+1) - C\tilde{x}(k+1) \\
&= C[\hat{x}(k+1) - C\tilde{x}(k+1)] \\
&= C\tilde{e}^a_f(k+1) \qquad (3.25)
\end{aligned}
$$

In the above equations, we assumed that FDI cyber-attack signals are time-varying additive signals on both inputs and outputs. When there is no cyber-attack both $u^a(k) = 0$ and $y^a(k+1) = 0$ and the equation (3.24) becomes equation (3.17), and when there is no fault on actuators then $E = I$, $f = 0$, which means $KCBf = 0$ and $KCB(I - E)u(k)$. And if there is no fault in measurement system $\Delta y_f(k+1) = 0$ and equation (3.24) becomes equation (3.13). For a faulty system, when there is no cyber-attack the expected value of residuals on both the plant side and controller side will become:

$$
\lim_{k\to\infty} \mathbb{E}[\tilde{e}_f(k+1)] = A\mathbb{E}[\tilde{e}_f(k)] - KCA\mathbb{E}[e_f(k)] + KCB(I-E)u(k) + KCBf + K\Delta y_f(k+1) \qquad (3.26)
$$

However, for a faulty system under cyber-attack, the expected value of residual on the controller side is:

$$\begin{aligned}\lim_{k\to\infty}\mathbb{E}[\tilde{e}_f^a(k+1)] &= A\mathbb{E}[\tilde{e}_f^a(k)] - KCA\mathbb{E}[e_f(k)] + KCB(I-E)u(k) + KCBf + K\Delta y_f(k+1) \\ &= KCBEu^a(k) + Ky^a(k+1) \qquad (3.27)\end{aligned}$$

We used differences between states of the Kalman filter and a priori estimator as residuals. Hence, the residuals are equal to the estimation error. Comparing the residuals in equations (3.27) and (3.26), we can see that there are two extra parameters for injected cyber-attack signals in the equation (3.27). Since we calculated the residuals with the same data for the same system, they have the same covariance. Therefore from equation (2.57) the χ^2 of these residual are equal as shown below:

$$g_f(k) = z_f(k)\Sigma^{-1}(k)z_f(k) \qquad (3.28)$$

$$g_f^a(k) = z_f^a(k)\Sigma^{-1}(k)z_f^a(k) \qquad (3.29)$$

We showed that SVM can learn the patterns in the data without generating any residuals. Therefore, we will use the square root of the two χ^2, one on the plant side and the other on the controller side. The two χ^2 used for the dataset are for the healthy and faulty states of actuators in the absence of any cyber-attack. From equation (3.26) and (3.27), we can see that in the absence of cyber-attacks, the two χ^2 are equal, but in the presence of a cyber-attack, they will be different.

Therefore, we need to train an SVM to detect the similarities of the $\sqrt{\chi^2}$. Following the previous subsection, we will use our proposed training method to prepare the dataset and train an SVM. To do so, we generated a dataset based on the two $\sqrt{\chi^2}$ and then generated a margin with a width of ± 20% on the plant side $\sqrt{\chi^2}$. We reached the ±20 margin based on the numerical values of the $\sqrt{\chi^2}$ in the dataset and with the trial and error. We used a percentage that is a multiplicative margin instead of adding a constant because the covariance of two $\sqrt{\chi^2}$ increase when the system is faulty and a wider margin is required for proper classification. Therefore, we divided the healthy and faulty plant side $\sqrt{\chi^2}$ data randomly into three parts and added a margin of 20% to the first slice, a -20% margin to the second slice, and kept the third slice as is. We also included $\sqrt{\chi^2}$ of the ground controller as a feature of the dataset. A part of the dataset is plotted in Figure 3.16. The blue circles are the original data, and the red triangles are generated based on adding 20% and -20% margins to the original data. We simulated simultaneous fault and cyber-attacks for the latitudinal model with the four faults for the rudder and aileron listed in Table 3.5. We injected cyber-

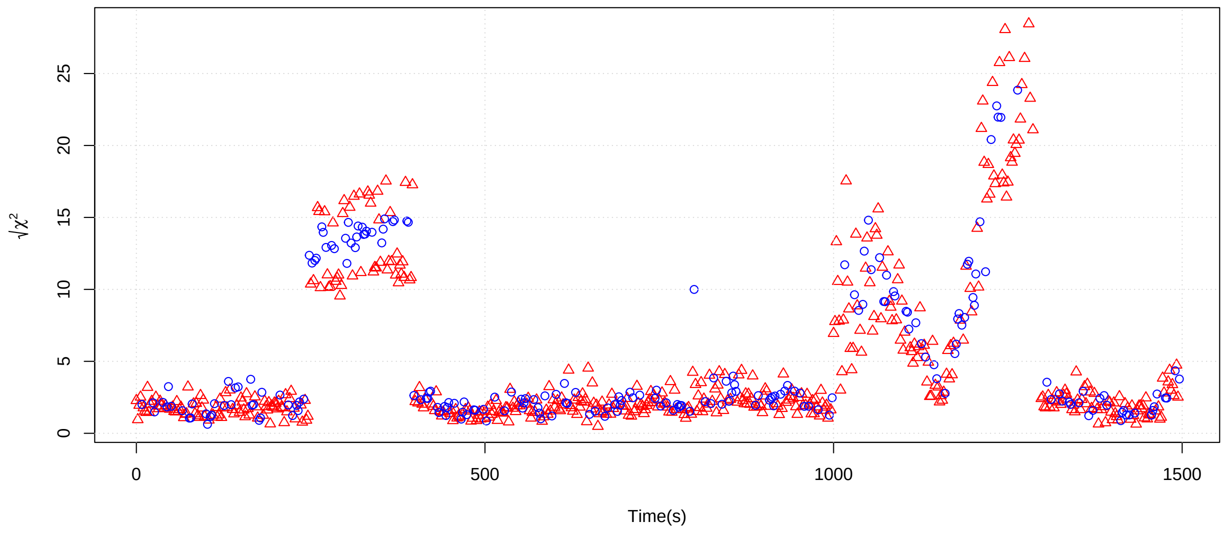


Figure 3.16: A sample of the dataset used for training of SVM with two $\sqrt{\chi^2}$

attack signals in the latitudinal ground control system command while the actuators were faulty and saved the data to a file. Then we calculated the two $\sqrt{\chi^2}$ and tested with the SVM predictor. The results are listed in Table 3.1.

	F1-score				
Cyber-attack signal magnitude	Healthy	ALP	RLP	ALF	RLF
$u^a < 0.35$	0.74	0.64	0.79	0.67	0.74
$u^a > 0.35$	1	0.99	1	1	0.98

Table 3.5: Summary of results of isolation of cyber-attacks from faults

Our SVM model could isolate cyber-attacks with magnitude of greater than 0.35 degree from faults with an F1-score of 0.98 or higher. The false alarms were due to noises. For cyber-attacks smaller the 0.35 degrees the detection was relatively poor with F1-score of between 0.64 to 0.87. However, the small cyber-attack signals have a negligible effect on the UAV operation. We had negligible false positives on the data with no cyber-attacks. Our proposed approach needs more resources to calculate the states of the system, but it has the advantage of accurately isolating faults from cyber-attacks.

3.5.3 Application of SVM for Replay Cyber-attack Detection

A replay cyber-attack is done by recording the outputs of a plant for some time and then replaying the recorded measurements back, while another cyber-attack is manipulating the control inputs of the system. Generally, a replay cyber-attack is possible when the plant is at steady state conditions, so the operators can not distinguish the replied measurements from the actual operating condition measurements. One of the most widely used detection methods for replay cyber-attacks is physical watermarking, where the defender adds a random noise, also known as a watermark or authentication signal, to control input and check the system response for the injected signal. A watermarked control input is as follows [71, 113]:

$$u(k) = u^*(k) + \Delta u(k)$$

where $u^*(k)$ is controller input and $\Delta u(k)$ is a white zero mean Gaussian noise with covariance of s [71, 114]. Adding an authentication signal will degrade the controller performance, but it is a necessary compromise to detect the cyber-attack [114]. Model-based cyber-attack detection is designed based on the χ^2 for the systems with Kalman filter and LQG controller to detect the replay cyber-attacks [71, 113, 114]. These methods were designed for the LQG controller using a Kalman filter; therefore, having the system model available is necessary. In contrast, researchers in [115] use a model-free machine learning method to detect replay cyber-attacks by training an SVM with a Gaussian kernel. They could detect the replay cyber-attack with about 80% accuracy when tested with the simulated data of an IEEE 14 bus power grid model.

In this subsection, we trained an SVM with a radial kernel with preprocessed data and used a search algorithm to detect replay cyber-attacks. The advantage of a machine learning-based method for replay cyber-attack detection is that the model of the system is not necessary, and the detection is independent of the controller type. We simulated a replay cyber-attack by allowing the UAV to fly in a trim condition. We recorded all the measurements and replayed them back during the cyber-attack. For the simulation of the watermarking defense, we added a zero mean random Gaussian signal with a magnitude of 0.7 degrees to the UAV ground control input of the latitudinal command. The watermarked input command and resulting latitudinal Kalman filter outputs are presented in Figures 3.17 and 3.18.

The watermarked control input will generate similar patterns in the measurements of the UAV. Figure 3.19 shows latitudinal commands of the ground control system and the roll angle of UAV ϕ in degrees that are scaled and centered for comparison. Despite some similarities, these signals can hardly match. There is also a tiny lag between the two signals. An SVM trained with these signals showed poor performance.

We applied a moving average over one second or ten samples on the command and did some tuning to compensate for the lag between command and measurement. The result of the cleaning up after scaling and centering the signals is shown in Figure 3.20.

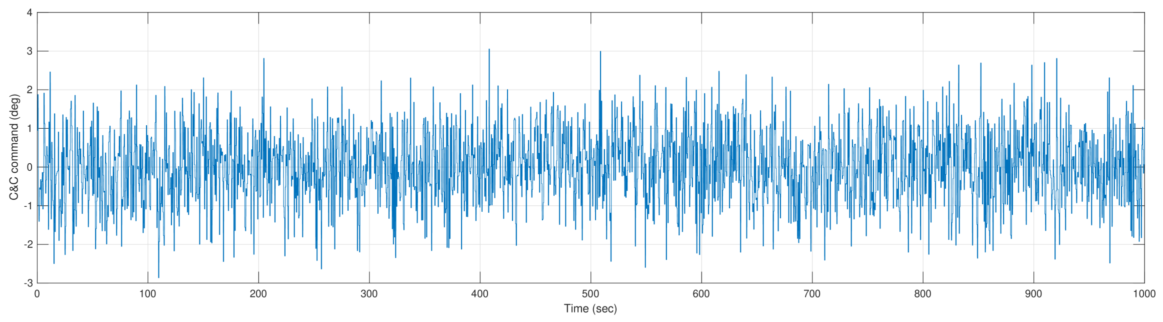


Figure 3.17: Watermarking with IID signal applied to latitudinal UAV control input.

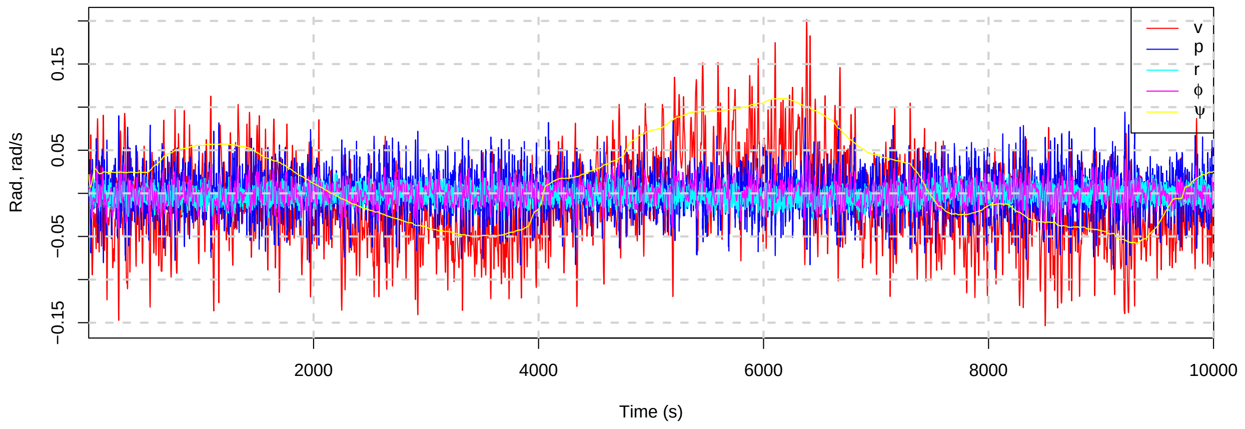


Figure 3.18: Latitudinal Kalman filter states of UAV with watermarking.

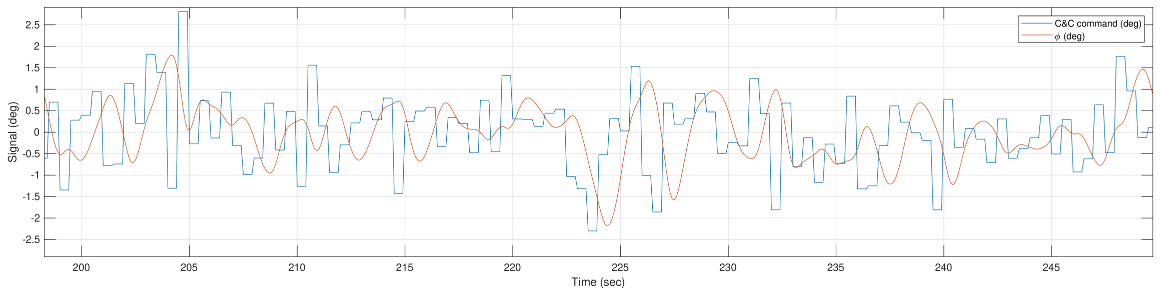


Figure 3.19: Scaled latitudinal Kalman filter roll angle and latitudinal command.

We can see that the signals are now almost perfectly matched. We used this approach on the signals to train an SVM for replay cyber-attacks and then automated the procedure for replay cyber-attack detection. To simulate a replay cyber-attack, we designed a flight scenario where UAV control signals are constant; Therefore, all actuators are at a fixed position. At the time stamp 6500, we started replaying the measurements of the UAV from

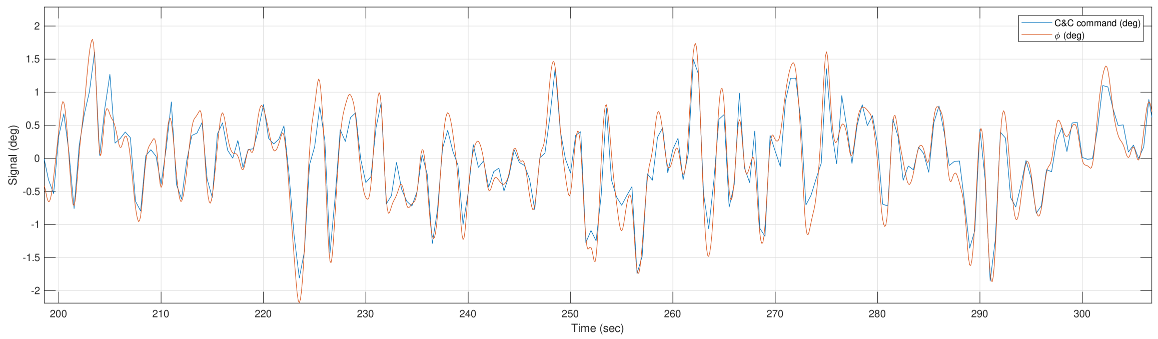


Figure 3.20: Smoothed and scaled latitudinal Kalman filter roll angle and latitudinal command.

2000 to time stamp 3000, where the Kalman filter-based detector residuals also increased as shown in Figure 3.21. In this subsection, our objective is to detect this replay cyber-attack without any model, just by training an SVM with the data.

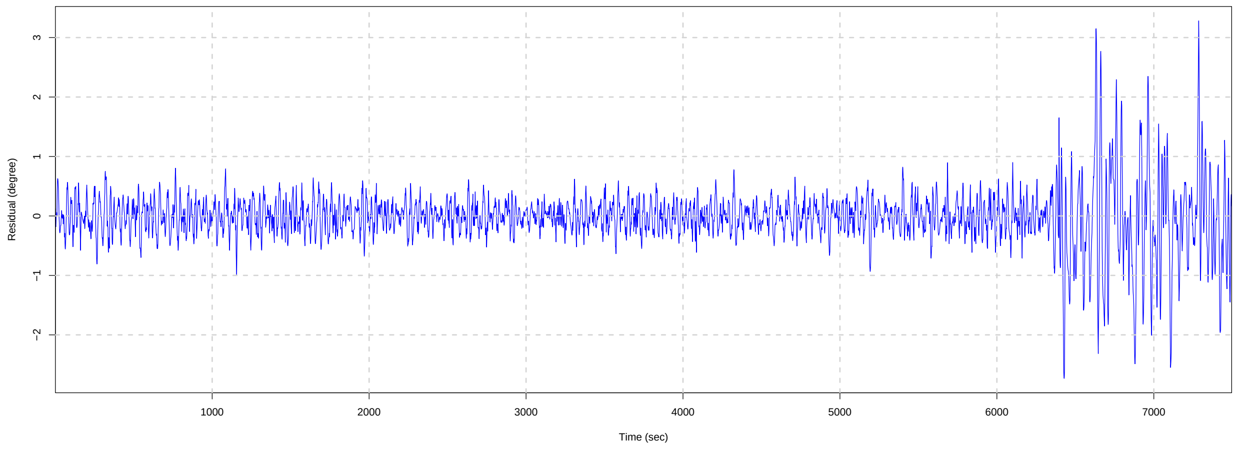


Figure 3.21: Residual of watermarked signals, where replay cyber-attack starts at time stamp of 6500.

We trained a two-class SVM model with the watermarked signals of the ground control system and the UAV roll angle ϕ. The dataset contains about 110 seconds of replayed signals and 640 seconds of healthy signals. The two classes are matched signals and unmatched signals. It is noteworthy that due to the randomness of signals unmatched signals have some similarity. For instance, in our tests, the matching signals have an F1-score of higher than 0.95, whereas unmatched signals of random watermarked input have an F1-score of about 0.65. Due to the lag between the control command and the output, testing new signals with our SVM model will not result in a high F1-score. Therefore, to find the matched signals, we run a search in a window of about 1.5 seconds (with 15 points) with the SVM.

In our proposed procedure, finding an F1-score higher than 0.95 is considered a matching signal and an F1-score close to 0.7 is interpreted as a replay cyber-attack. In the search algorithm, we shift data one time stamp at a time and test the shifted signals with the SVM. We repeat the search for about 50 times stamps and report the highest score as a result of the search F1-score. A crucial factor for detecting replay cyber-attacks is the number of data points in the test procedure. For the fault and FDI cyber-attack detection, we test every observation and report the result. However, for replay cyber-attack detection, we should test a range of consecutive observations together and report the overall F1-score. In our tests, we used about 100 seconds of data for each test to verify the results. We implemented the following procedure for detecting replay cyber-attacks.

1. Acquire the last 100 seconds of data.
2. Smooth control input of command-and-control with a moving average of 10 observations (1 second) window.
3. Scale the control input and combine it with the roll angle received from the UAV to form a data frame.
4. Feed the data to the SVM model and calculate the F1-score.
5. Shift the command data one step and repeat from step 4 15 times (depending on the lag plant, this value might change).
6. Find and save the maximum F1-score.
7. An F1-score of higher than 0.95 is considered a matching signal with no cyber-attack, and an F1-score of lower than 0.95 is a sign of a replay cyber-attack.

We tested a sample with a 15-step loop and recalculated scores for matching signals. The result for a matching signal (no replay cyber-attack) and a mismatching signal (replay cyber-attack) is presented in Figure 3.22. We can see that matching signals reach an F1-score of up to 1 for perfect similarity while mismatching random signals during replay cyber-attacks always have an F1-score of less than 0.70. Compared with the 80% detection rate of [115], we showed more accurate results for replay cyber-attack detection with an F1-score of about 1.

3.6 Conclusion

In this chapter, we studied fault and replay and FDI cyber-attack detection using Kalman filtering and SVM. We showed that SVMs can effectively detect faults and cyber-attacks with high accuracy. However, effective data preparation, including noise removal and feature engineering, is crucial for the successful application of SVMs. For example, in the case of UAVs, PCA can be used to extract features from correlated measurement data, but including

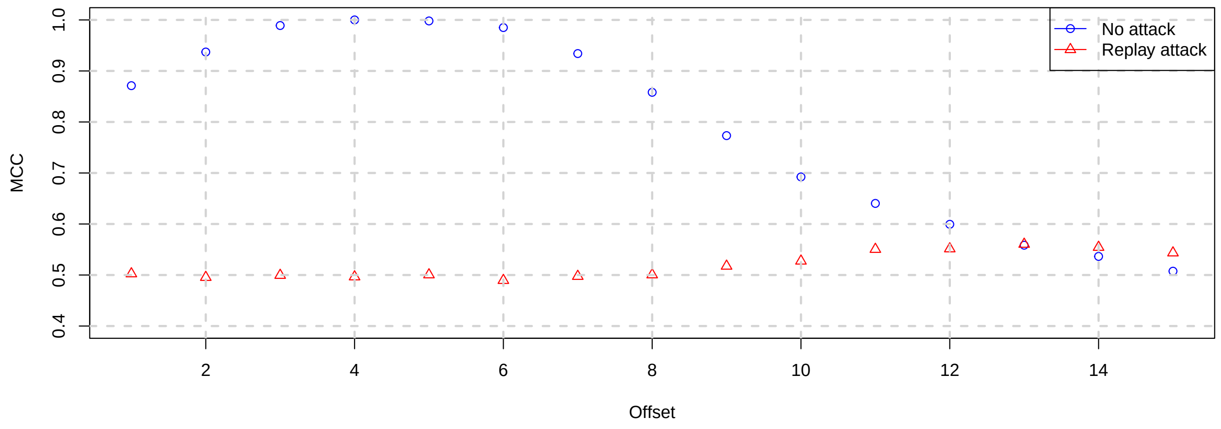


Figure 3.22: Residual of watermarked signals.

specific actuator control input in the dataset before performing PCA will eliminate the variability of the actuator compared with the other measurements, resulting in poor detection with SVM. We also showed that employment of the Kalman filter addresses noise and feature extraction issues and eliminates additional steps of feature extraction with PCA. Typically, the Kalman filter is designed to estimate the most critical states of a system, which can serve as optimal features for machine learning methods. Furthermore, the Kalman filter reduces noise from the measurement system utilizing knowledge of the system model. Additionally, employing a priori estimation with the Kalman filter can convert output residuals into state residuals, thereby improving fault detection for systems with unmeasurable states, such as efficiency. In the case of UAVs, directly including the control inputs as features is as effective as using a priori estimated states. Moreover, when using Support Vector Machines (SVMs) for fault detection, residual generation is unnecessary and can even degrade the performance of SVMs. For the isolation of faults, we used a bank of SVMs trained for each actuator's fault against a healthy system. We also trained SVMs to isolate different types of faults in each actuator. Overall, our proposed method for applying PCA for UAV or directly using the Kalman filtering states with the SVM classification has superb performance, and compared with the [110], our approach has a much better F1-score of about 0.98 with way faster and smaller model.

SVMs work well for FDI and replay cyber-attack detection on ground control system inputs. One-class SVM is best for detecting anomalies when there are few anomalies and the anomaly model is unknown. It defines a boundary around healthy data and classifies observations outside the boundary as anomalies. However, tuning one-class SVM with noisy data is challenging and can lead to false positives. UAVs face challenges from noises and disturbances, including the unknown effects of wind. We used a model to fine-tune a two-class SVM for detecting FDI cyber-attacks on noisy actuators of a UAV. The goal is to have eligible false positives in healthy data. By considering UAV performance and sensor/actuator noise, we can define a margin. cyber-attacks with a magnitude smaller than this margin can not be detected, but will have an eligible effect on the UAV. We created a training dataset based on this idea with the healthy UAV states estimated by a Kalman filter. FDI cyber-attack signals with the minimum magnitude were then injected into randomly selected portions of actuator commands. Compared with the one-class SVM used by [108] to detect cyber-attacks on sensors of autonomous car networks, our two-class SVM trained with this dataset had much lower false positive on healthy data and could detect FDI cyber-attacks equally well with an F1-score of about 0.98 on actuators. A primary problem with both SVM models for cyber-attack detection is the presence of faults in the system, which also will be classified as a cyber-attack. To handle this problem, the availability of some information about faults in the system is necessary. We proposed to use two χ^2, one on the plant side and the other on the controller side. We trained a two-class SVM based on our proposed method with healthy on faulty states χ^2 of both the plant and controller side. Our proposed method could detect cyber-attacks and at the same time isolate faults from cyber-attacks with an F1-score of more than 0.98.

Replay cyber-attack is another important type of cyber-attack, which grabs the attention of many researchers. Watermarking is the most common detection method for relay cyber-attack detection, which is used with the Kalman filter and χ^2 to detect the cyber-attack. We also studied the capabilities of SVM for replay cyber-attack detection without any model of the system and proposed a search method based on the shifting data and consecutively tested with SVM to find the random patterns of input in outputs. Compared with [114], we do not need the model of the system to detect the replay cyber-attack, and compared with [115] which had a performance of about 80%, we could detect replay cyber-attacks with much higher accuracy of F1-score of about 1.

Chapter 4

Coding Schemes for Detection of Cyber-attacks and Isolation of Cyber-attacks from Faults

4.1 Introduction

CYBER-attacks aim to compromise computer systems or networks, compromising confidentiality, integrity, and availability. Modern control systems extensively use computer and communication systems to acquire, analyze, and store data, connect to other systems, and optimize resources, which results in vulnerabilities in control systems. Control systems are often targets of cyber-attacks because they can cause significant damage or disruption to critical infrastructure and services.

Cyber-attack detection on control systems is a challenging and complex subject, specifically detecting stealthy cyber-attacks such as covert and replay cyber-attacks. In a covert cyber-attack, an attacker can manipulate sensor measurements and actuator commands to remove the effects of the cyber-attack from the measurements. This type of cyber-attack can go undetected by most of the existing security mechanisms. In a replay cyber-attack, on the other hand, the attacker records and later plays back legitimate sensor measurements to deceive the controller or the operator. Both types of cyber-attacks can cause serious consequences, such as loss of control, system instability, or physical damage.

Several model-based and data-driven approaches have been proposed in the literature to detect cyber-attacks. Since both of these approaches need a baseline to define the healthy system, both suffer from the same limitation. For instance, the detection limitations of data-driven systems for detecting cyber-attacks are the same as model-based approaches for discrete-time linear time-invariant systems [83]. Detection of stealthy cyber-attacks such as covert cyber-attacks, replay cyber-attacks, and zero-dynamic cyber-attacks is not straightforward and needs extra measures. Most research uses some information unknown to attackers to detect these types of cyber-attacks, such as physical watermarking, moving targets, and coding. In this chapter coding schemes are proposed for covert and replay cyber-attack detection.

Coding is one of the most effective methods for detecting stealthy cyber-attacks, which offers a computationally inexpensive mechanism that can help detect covert cyber-attacks, and with some modifications to detect replay cyber-attacks without degrading controller performance. Coding theory deals with encoding, transmitting, and decoding information. It involves creating and analyzing error-correcting codes, which help improve the reliability of data transmission and storage. The control community uses fast and lightweight coding techniques to handle cyber-attacks while keeping the existing systems fast enough for real-time application. In [77], a one-way coding schema is proposed for covert cyber-attacks detection, and in [79], two-way coding is studied for scalar analog systems to detect zero-dynamics cyber-attacks. In this thesis, two-way coding is extended for MIMO systems and applied to discrete-time control systems to detect covert and replay cyber-attacks.

4.2 Defense Against Covert Cyber-attack

Model-based and data-driven fault detection approaches have been widely studied for cyber-attack detection scenarios on CPSs [9]. Despite that, these methods can detect cyber-attacks in the form of false data injection cyber-attacks but fall short of stealthy cyber-attack detection, such as covert and replay cyber-attacks, designed to fool monitoring systems. Active monitoring methods such as physical watermarking can be effective for replay cyber-attack detection, but degradation of controller performance is inevitable in this method [71, 114, 116].

Coding is another effective method studied for stealthy cyber-attack detection, such as covert cyber-attacks and zero-dynamics cyber-attacks [77, 79]. The principal concept of methods based on coding is to change the inputs and outputs of a system in a way that it became difficult or even impossible for an attacker to identify correct signals [77, 78]. The coding schemes studied for control systems are diverse, including one-way and two-way codings. One-way coding refers to the coding scheme where the information in a channel flows in one direction, and two-way coding, on the other hand, requires a simultaneous flow of information in both directions. The two-way coding schema uses both sides of information for coding. One-way coding, shown in Figure 4.1, uses a matrix unknown to the attackers to code the inputs and outputs of a system. The cloud shape in this figure and other figures is representative of a network of communication channels that can be targeted by attackers. For an attacker to launch a successful cyber-attack, it is necessary to decode the information, which requires knowledge of the coding matrix. Researchers in [77] showed that for a system with unstable eigenvectors in A matrix, where coding is applied only to the unstable poles of the system, an attacker needs $N \geq \max\{n, p\} - 1$ measurements to calculate coding matrix, where n is the number of states and p is the number of outputs. Hence, the coding matrix should change frequently to protect the system. On the other hand, two-way coding for scalar systems introduced by [79] offers stiffer coding by combining inputs and outputs information.

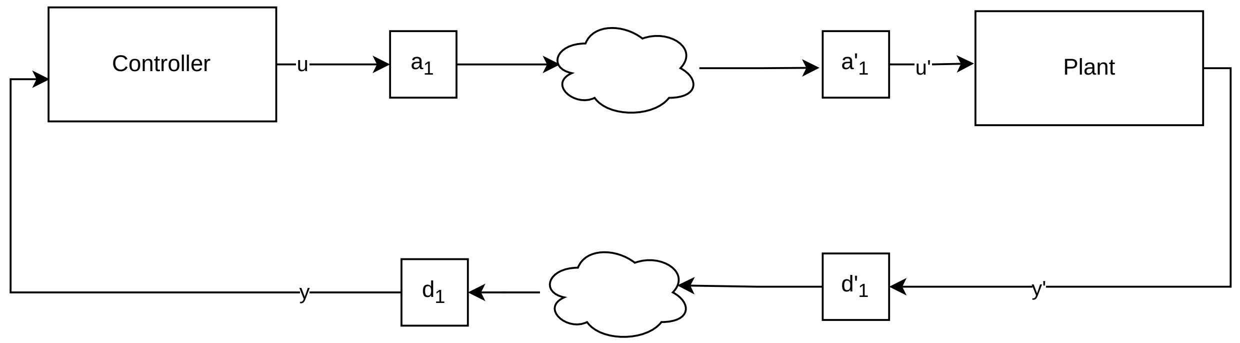


Figure 4.1: One way coding [79]. The cloud shape is representative of a network of communication channels that can be targeted by attackers.

Two-way coding was first introduced by Shannon for communication channels [117]. It is adapted for scalar continuous-time control systems by [79]. An advantage of two-way coding for cyber-attack detection is that since changes in the controller inputs $u(t)$ are reflected on the plant outputs $y(t)$ and vice-versa, it can be used to detect zero-dynamics cyber-attacks.

Two-way coding described in [79], is shown in Figure 4.2, is developed for the scalar system, with Single-Input Single-Output (SISO) continuous-time systems.

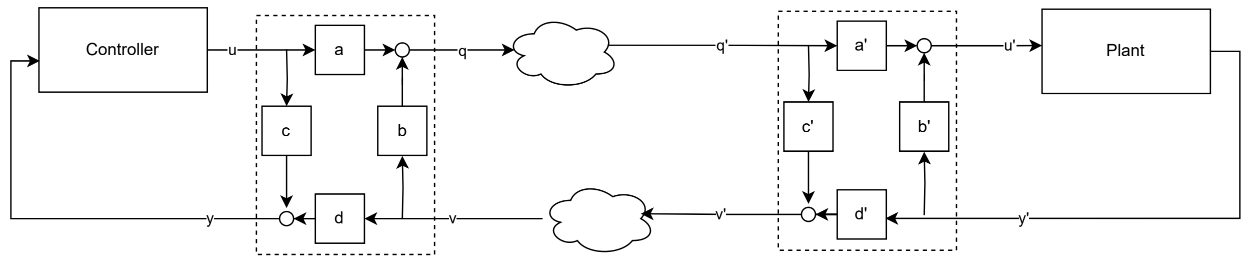


Figure 4.2: Two way coding [79]. The cloud shape is representative of a network of communication channels that can be targeted by attackers.

Since an attacker might manage to manipulate signals via the communication channel, the parameters on the plant side are shown with $\{.\}'$ to distinguish them from the controller side parameters. Hence, on the controller side, u is the controller input, y plant output, q is the coded controller input, and v is the coded plant output. On the plant side, u' is the controller input, y' is the plant output, q' is the coded controller input, and v' is the coded plant output. The coding matrix elements are constants a, b, c, and d. The inverse coding matrix elements are a', b', c', and d'. The mathematical representation for information flow from controller to plant based on the scalar two-way coding developed by [79] is presented here. However, since we are studying discrete-time systems, we will present the coding for discrete-time system notations. In controller side the output $u(k)$ is coded to generate coded input $q(k)$ and received coded output $v(k)$ is decoded to get $y(k)$ as follows:

$$\begin{bmatrix} q(k) \\ y(k) \end{bmatrix} = M \begin{bmatrix} u(k) \\ v(k) \end{bmatrix}, \qquad M = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \tag{4.1}$$

This can be written as:

$$q(k) = au(k) + bv(k) \tag{4.2}$$

$$y(k) = cu(k) + dv(k) \tag{4.3}$$

In the same way on the plant side, to decode coded input $q'(k)$ and generate coded output $v'(k)$, the following equations are used:

$$\begin{bmatrix} u'(k) \\ v'(k) \end{bmatrix} = M^{-1} \begin{bmatrix} q'(k) \\ y'(k) \end{bmatrix}, \qquad M^{-1} = \begin{bmatrix} a' & b' \\ c' & d' \end{bmatrix} = \frac{1}{ad-bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \tag{4.4}$$

Alternatively, this can be written as:

$$u'(k) = a'q'(k) + b'y'(k) \tag{4.5}$$

$$v'(k) = c'q'(k) + d'y'(k) \tag{4.6}$$

If both b and c are zero, the two-way coding will become one-way coding, but if a is zero, then information of control input u, will be lost, and if d is zero, information of measurement, y, will be lost. Therefore, neither a nor d should be zero, which leads to condition $ad \neq 0$. Hence, the required condition for matrix M is

$$\begin{aligned} ad &\neq 0 \\ ad - bc &\neq 0 \end{aligned} \tag{4.7}$$

The decoded input and output in the absence of cyber-attack must be the same as the original input and output, based on [79], in the absence of cyber-attack, we have $q'(k) = q(k)$, and $v'(k) = v(k)$ and from (4.6) and (4.2) we have:

$$\begin{aligned} v(k) &= v'(k) = c'q'(k) + d'y'(k) \\ &= c'[au(k) + bv(k)] + d'y'(k) \\ &= \frac{-ac}{ad - bc}u(k) + \frac{-bc}{ad - bc}v(k) + \frac{a}{ad - bc}y'(k) \\ \left[1 - \frac{-bc}{ad - bc}\right] v(k) &= \frac{-ac}{ad - bc}u(k) + \frac{a}{ad - bc}y'(k) \\ v(k) &= -\frac{c}{d}u(k) + \frac{1}{d}y'(k) \end{aligned} \tag{4.8}$$

And replacing (4.8) in (4.3) we have:

$$\begin{aligned} y(k) &= cu(k) + d\left[-\frac{c}{d}u(k) + \frac{1}{d}y'(k)\right] \\ &= cu(k) - cu(k) + y'(k) \\ y(k) &= y'(k) \end{aligned}$$

Hence, in the absence of cyber-attacks, plant output decoding on the controller side works correctly. To show that decoding of coded controller input on the plant side is working, from (4.2) and (4.6), and considering that when there is no cyber-attack $q(k) = q'(k)$ we can write:

$$\begin{aligned}
q'(k) &= au(k) + b[c'q'(k) + d'y'(k)] \\
&= au(k) + \frac{-bc}{ad-bc}q'(k) + \frac{ab}{ad-bc}y'(k) \\
\left[1 - \frac{-bc}{ad-bc}\right] q'(k) &= au(k) + \frac{ab}{ad-bc}y'(k) \\
q'(k) = q(k) &= \frac{ad-bc}{d}u(k) + \frac{b}{d}y'(k) \\
&= \frac{1}{a'}u(k) + \frac{b}{d}y'(k) \qquad (4.9)
\end{aligned}$$

Replacing (4.9) into (4.5) we have:

$$\begin{aligned}
u'(k) &= a'\left[\frac{1}{a'}u(k) + \frac{b}{d}y'(k)\right] + b'y'(k) \\
&= u(k) + \frac{b}{ad-bc}y'(k) + \frac{-b}{ad-bc}y'(k) \\
u'(k) &= u(k)
\end{aligned}$$

Therefore, decoding of inputs in plant side is also working correctly.

4.3 Two-way Coding for MIMO Systems

In this section, two-way coding proposed by [79] for SISO systems is extended to the multi-input multi-output systems (MIMO). To develop this scheme, first, it is assumed that there are no cyber-attacks, hence $q = q'$ and $v = v'$. In later sections, we will examine the cyber-attacks on the developed coding scheme. There are two alternative coding schemes for MIMO systems. The first case is when the number of inputs and outputs of a system are equal so that coding is straightforward. The second case is when the number of inputs and outputs are unequal, and some modifications are required to develop coding schemes.

4.3.1 MIMO System with Equal Number of Inputs and Outputs

The two-way coding scheme for the case of equal inputs and outputs is straightforward. For instance, the coding scheme of a system with two inputs and two outputs is shown in Figure 4.3. This pattern can be repeated for any number of input-output pairs. The coding matrix on the controller side are constants a_i, b_i, c_i and d_i, and on the plant side are a'_i, b'_i, c'_i and d'_i, where subscript i represent the $i-th$ input-output pair.

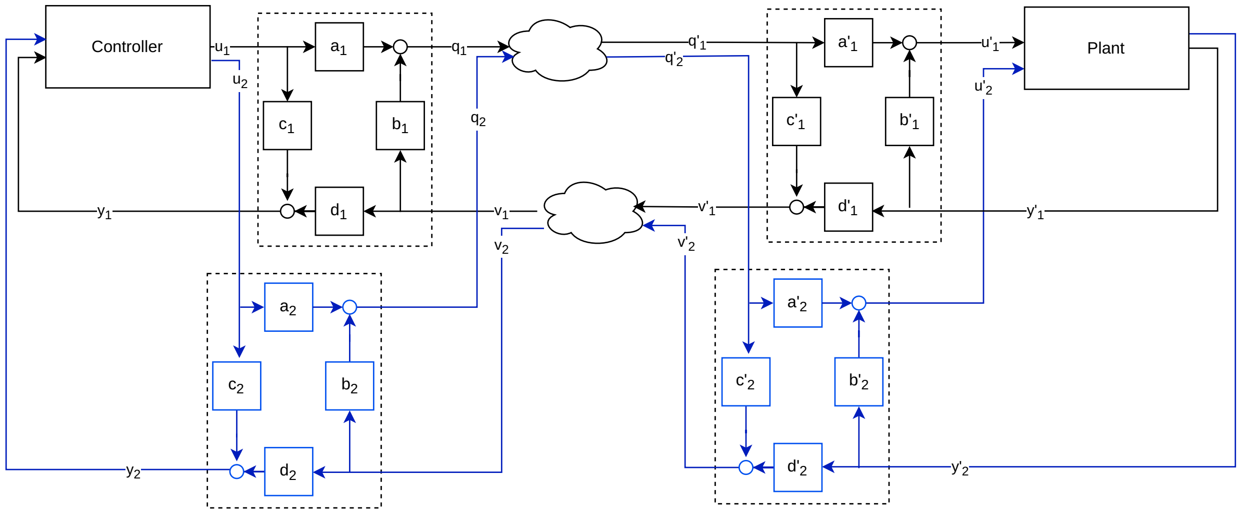


Figure 4.3: Two way coding MIMO.

This scheme can be described as matrix operation for two inputs and two outputs as shown below:

$$\begin{bmatrix} q_1(k) \\ y_1(k) \\ q_2(k) \\ y_2(k) \end{bmatrix} = M \begin{bmatrix} u_1(k) \\ v_1(k) \\ u_2(k) \\ v_2(k) \end{bmatrix}, \qquad M = \begin{bmatrix} a_1 & b_1 & 0 & 0 \\ c_1 & d_1 & 0 & 0 \\ 0 & 0 & a_2 & b_2 \\ 0 & 0 & c_2 & d_2 \end{bmatrix} \tag{4.10}$$

$$\begin{bmatrix} u'_1(k) \\ v'_1(k) \\ u'_2(k) \\ v'_2(k) \end{bmatrix} = M^{-1} \begin{bmatrix} q'_1(k) \\ y'_1(k) \\ q'_2(k) \\ y'_2(k) \end{bmatrix}, \qquad M^{-1} = \begin{bmatrix} a'_1 & b'_1 & 0 & 0 \\ c'_1 & d'_1 & 0 & 0 \\ 0 & 0 & a'_2 & b'_2 \\ 0 & 0 & c'_2 & d'_2 \end{bmatrix} \tag{4.11}$$

If we assign $h_1 = a_1d_1 - b_1c_1$, and $h_2 = a_2d_2 - b_2c_2$ then, we can write M^{-1} as:

$$M^{-1} = \begin{bmatrix} d_1/h_1 & -b_1/h_1 & 0 & 0 \\ -c_1/h_1 & a_1/h_1 & 0 & 0 \\ 0 & 0 & d_2/h_2 & -b_2/h_2 \\ 0 & 0 & -c_2/h_2 & a_2/h_2 \end{bmatrix} \tag{4.12}$$

And same as scalar two-way coding, the following should hold:

$$\begin{aligned} a_i d_i &\neq 0, \quad i = 1,2 \\ a_i d_i - b_i c_i &\neq 0, \quad i = 1,2 \end{aligned} \tag{4.13}$$

If number of inputs and outputs are not equal, some modifications are required to use two-way coding for MIMO systems, which will be discussed in next subsection.

4.3.2 MIMO System with Unequal Number of Inputs and Outputs

When the number of inputs and outputs are not equal, one possible solution would be to reuse some signals to code extra incoming signals. Therefore, for instance, for a system with one input and two outputs, both outputs will be coded with the same input, leading to redundant decoded input on the plant side. For a healthy system, redundant signals would be identical and could be combined to get the desired signal. But for a system under cyber-attack, there is no guarantee that decoded redundant signals are the same. One solution to handle redundant signals would be an extra analytical step to evaluate the signals' health. This approach will add overhead to coding and decoding but will offer some recovery for a system under cyber-attack. In this chapter, redundant coding will not be used, and only for completeness of discussion is presented here. The schematic for such a coding scheme with one input and two outputs is presented in Figure 4.4, and the box labeled detector is assumed to be an analytical logic to make decisions about the health of redundant signals.

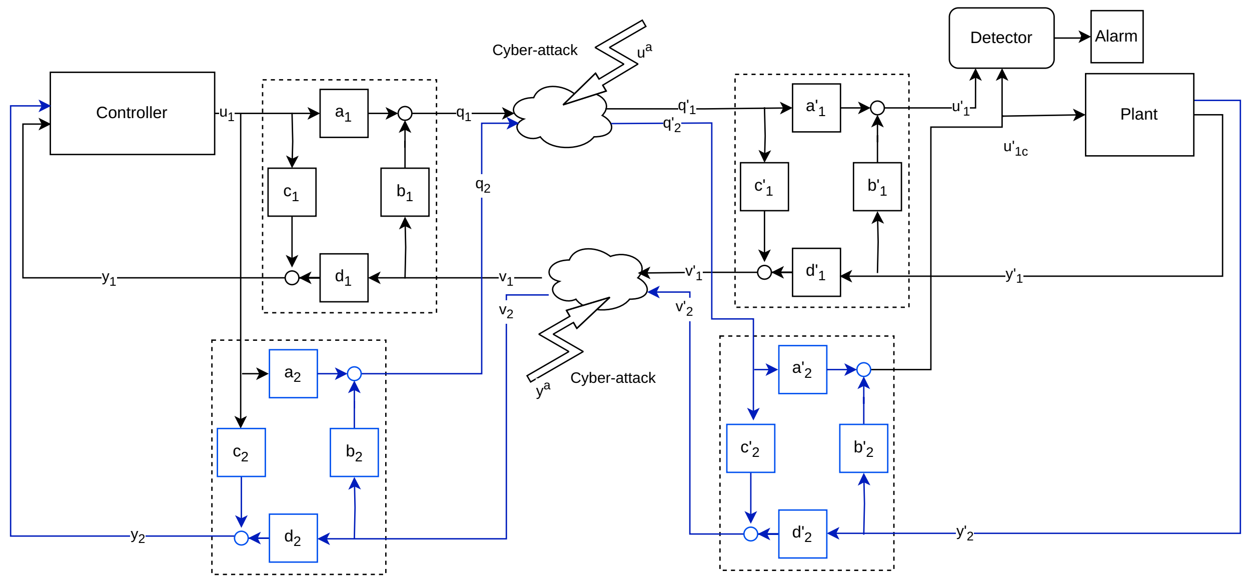


Figure 4.4: Two-way coding with one input and two outputs.

In this scheme, on the plant side, there are two redundant inputs, and only one is required. A consequence of this coding scheme is that an extra signal needs to be transmitted, but, on the other hand, there are opportunities to detect and recover cyber-attacked signals on the plant side, which is impossible otherwise. In this coding scheme, only cyber-attacks on redundant signals are detectable on the plant side. Any number of inputs and outputs could be coded with this scheme. The mathematical representation for one input and two outputs, shown in Figure 4.4, would be the same as the one discussed in equations (4.10) to (4.13), just $u_2(k)$ should be replaced with $u_1(k)$ in all equations.

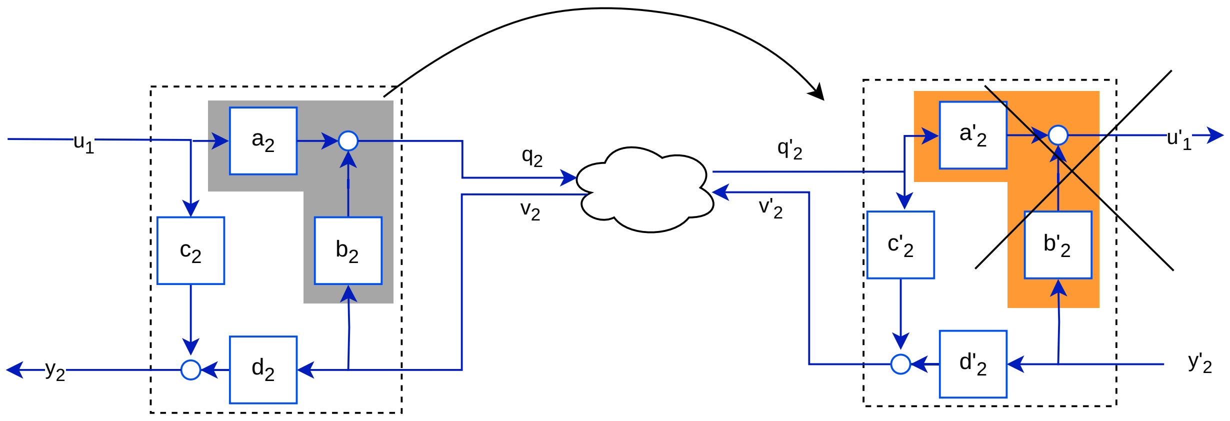
An alternative approach is to code the single input on the controller side with one of the outputs. On the plant side, after decoding this input, it is again used to code the other output. However, we need to generate a new coded input (q_2) on the plant side using the decoded control input (u_1) for coding the second output. Therefore, the coded input q_1', on the plant side should first decoded to get u_1', then the coding scheme for the second coded input on the controller side (q_2) should be implemented on the plant side using decoded input u_1'. Since the input u_1' is already available on the plant side via the first input-output pair, the decoding part of the second output can be removed. We illustrated this procedure in Figures 4.5a and 4.5b.

Figure 4.5a is the coding scheme for the second output y_2 and first input u_1. The gray area on the controller side is the coding part that should transfer to the plant side, and the orange area on the plant side is the decoding part that should be removed from coding since it will produce redundant input. This operation will result in the coding scheme presented in Figure 4.5b. The overall coding scheme developed for one input and two outputs without redundancy is shown in Figure 4.6.

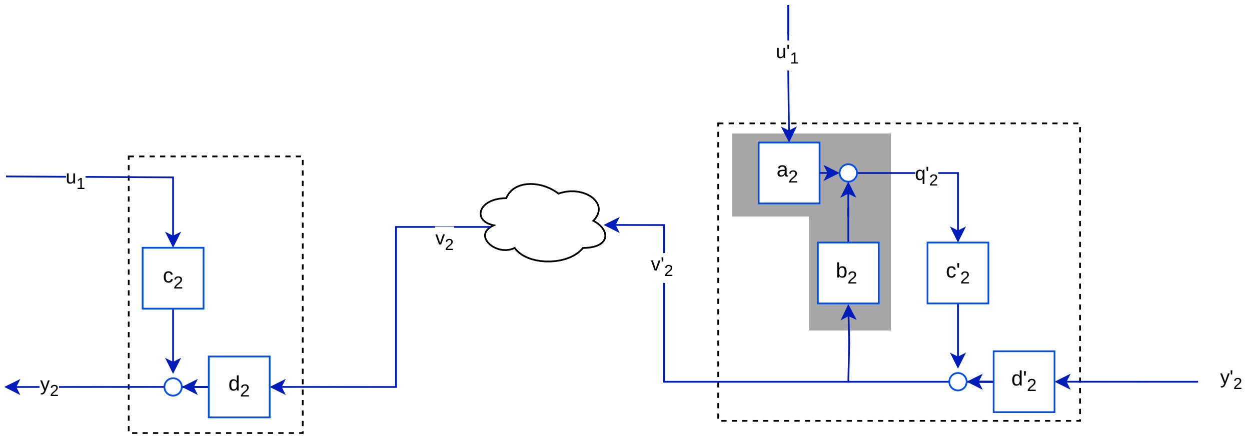
The mathematical representation for the coding scheme in Figure 4.6 on controller side is as follows:

$$\begin{bmatrix} q_1(k) \\ y_1(k) \\ y_2(k) \end{bmatrix} = M_1 \begin{bmatrix} u_1(k) \\ v_1(k) \\ v_2(k) \end{bmatrix}, \qquad M_1 = \begin{bmatrix} a_1 & b_1 & 0 \\ c_1 & d_1 & 0 \\ c_2 & 0 & d_2 \end{bmatrix} \tag{4.14}$$

The coding scheme in Figure 4.6 generates one coded input and decodes two outputs on the controller side. On the plant side, the coding generates two coded outputs and decodes one input. On the plants side, $q_2'(k)$ is constructed using decoded $u_1'(k)$ as shown in the following equation:



(a) Two-way coding of the first input with the second output.



(b) Generation of second coded input $q'_2(k)$ by first input $u'_1(k)$ on the plant side.

Figure 4.5: Two-way coding for one input and two outputs without redundancy.

$$q'_2(k) = \begin{bmatrix} b_2 & a_2 \end{bmatrix} \begin{bmatrix} v'_2(k) \\ u'_1(k) \end{bmatrix} \tag{4.15}$$

And then the coded output $v'_2(k)$ is generated using the $q'_2(k)$ as shown below:

$$\begin{bmatrix} v'_1(k) \\ u'_1(k) \\ v'_2(k) \end{bmatrix} = M_2 \begin{bmatrix} y'_1(k) \\ q'_1(k) \\ y'_2(k) \\ q'_2(k) \end{bmatrix}, \qquad M_2 = \begin{bmatrix} a'_1 & b'_1 & 0 & 0 \\ c'_1 & d'_1 & 0 & 0 \\ 0 & 0 & d'_2 & c'_2 \end{bmatrix} \tag{4.16}$$

Combining this two operations in one step results in calculate $v'_2(k)$, directly based on decoded $u'_1(k)$ and $y'_2(k)$, as shown below:

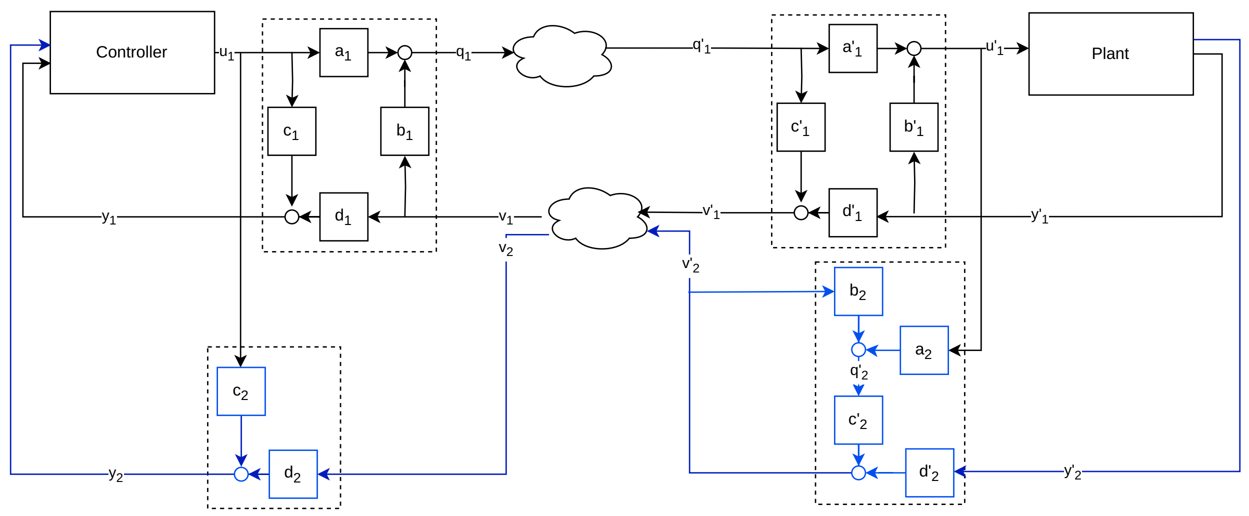


Figure 4.6: Two-way coding for a MIMO system when number of inputs are less than number of outputs with no redundant data transfer.

$$\begin{aligned} q_2'(k) &= a_2 u_1'(k) + b_2 v_2'(k) \\ v_2'(k) &= c_2' q_2'(k) + d_2' y_2'(k) = c_2' a_2 u_1'(k) + c_2' b_2 v_2'(k) + d_2' y_2'(k) \\ (1 - c_2' b_2) v_2'(k) &= c_2' a_2 u_1'(k) + d_2' y_2'(k) \end{aligned}$$

replacing values of c' and d' from equation (4.4), and simplification results in:

$$v_2'(k) = -\frac{c_2}{d_2} u_1'(k) + \frac{1}{d_2} y_2'(k) \tag{4.17}$$

Hence, the one step coding scheme on the plant side is represented as follows:

$$\begin{bmatrix} v_1'(k) \\ u_1'(k) \\ v_2'(k) \end{bmatrix} = M_{1r} \begin{bmatrix} y_1'(k) \\ q_1'(k) \\ y_2'(k) \\ u_1'(k) \end{bmatrix}, \qquad M_{1r} = \begin{bmatrix} a_1' & b_1' & 0 & 0 \\ c_1' & d_1' & 0 & 0 \\ 0 & 0 & \frac{1}{d_2} & -\frac{c_2}{d_2} \end{bmatrix} \tag{4.18}$$

The graphical representation of this approach is presented in Figure 4.7, which is simplified compared to original representation.

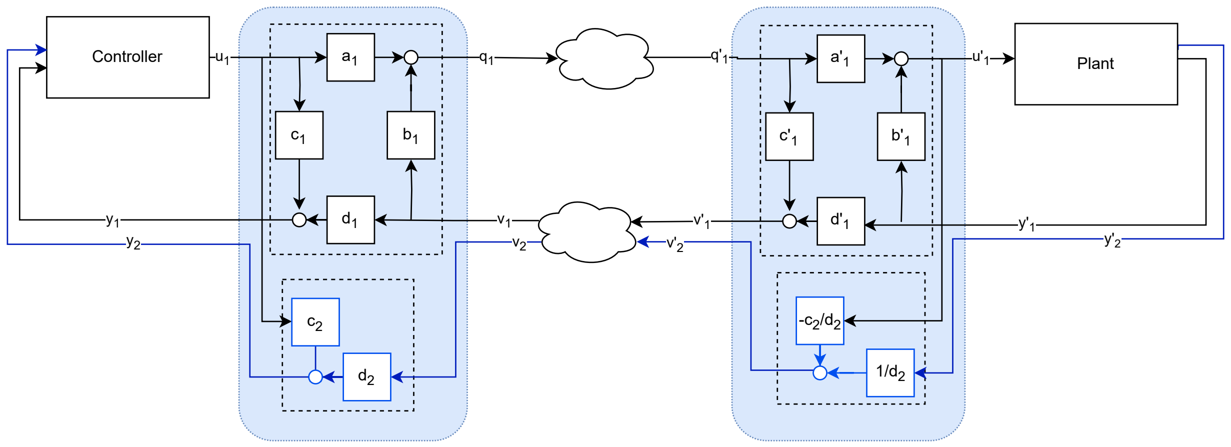


Figure 4.7: Simplified two-way coding for one input are two outputs.

With a similar analysis, when there is one output and two inputs, the second input $u_2(k)$ is coded with the first output $y_1(k)$ on the controller side to generate coded output $v_2(k)$. And then this coded output $v_2(k)$ and $u_2(k)$ are used to generate coded input $q_2(k)$, as shown below:

$$v_2(k) = c_2' q_2(k) + d_2' y_1(k)$$
$$q_2(k) = a_2 u_2(k) + b_2 v_2(k)$$

Combining these two equations will result in one-step coding of $u_2(k)$ as follows:

$$q_2(k) = \frac{1}{a_2'} u_2(k) + \frac{b_2}{d_2} y_1(k) \tag{4.19}$$

The schematic diagram of this coding scheme is presented in the Figure 4.8

Hence, this coding scheme on the plant side is presented with matrix operation as shown below:

$$\begin{bmatrix} u_1'(k) \\ v_1'(k) \\ u_2'(k) \end{bmatrix} = M_3 \begin{bmatrix} q_1'(k) \\ y_1'(k) \\ q_2'(k) \end{bmatrix}, \qquad M_3 = \begin{bmatrix} a_1' & b_1' & 0 \\ c_1' & d_1' & 0 \\ 0 & b_2' & a_2' \end{bmatrix} \tag{4.20}$$

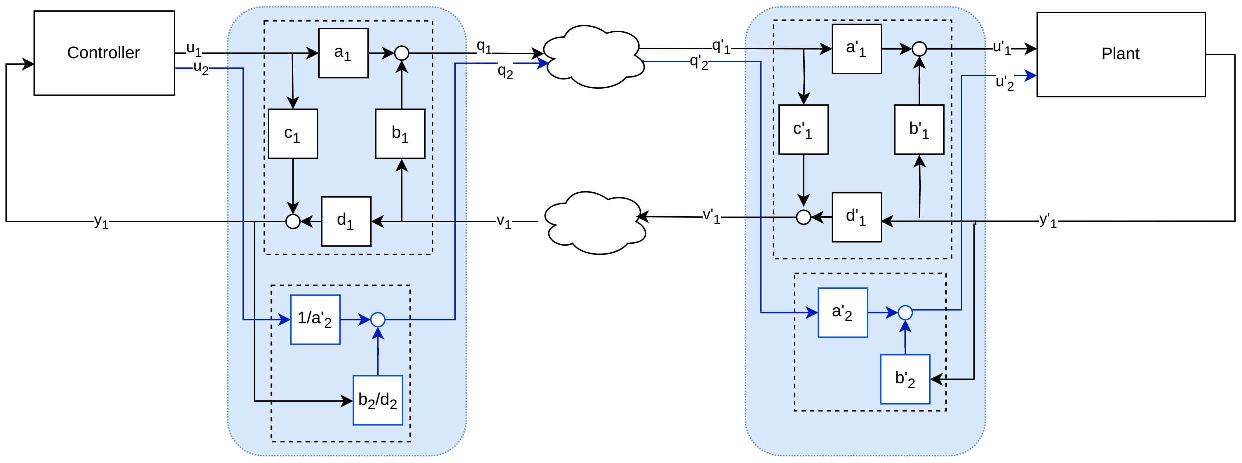


Figure 4.8: Simplified two-way coding for two inputs and one output.

$$\begin{bmatrix} q_1(k) \\ y_1(k) \\ q_2(k) \end{bmatrix} = M_{3r} \begin{bmatrix} u_1(k) \\ y_1(k) \\ u_2(k) \end{bmatrix}, \qquad M_{3r} = \begin{bmatrix} a_1 & b_1 & 0 \\ c_1 & d_1 & 0 \\ 0 & \frac{b_2}{a_2} & \frac{1}{a_2'} \end{bmatrix} \tag{4.21}$$

These two coding schemes can be repeated for any number of inputs and outputs for a MIMO system. It is worth noting that one can develop the coding scheme for continuous-time systems with a similar analysis.

4.4 Synthesis and Analysis of Discrete-Time LTI systems with Two-Way Coding Under Cyber-attack

Coding schemes proposed in literature [77] are designed to be simple with low computational overhead for covert cyber-attack detection. Two-way coding proposed in [79], designed to mitigate cyber-attack effects only when attackers attack zero dynamics of the augmented system consisting of coding and a scalar plat with zero dynamics. Justification is that if an attacker attacks this augmented system zero dynamics, the effect of the cyber-attack does not show on the output with two-way coding implemented, and the cyber-attack will remain undetectable. The drawback of this approach is that it only mitigates a single point of operation, but attackers can always launch other cyber-attacks, including zero dynamics of the scalar plant itself. Another notable drawback is that if the attacker is aware of the coding matrix, for instance, by following the same design procedure presented in the paper, they can launch covert cyber-attacks by removing the effects of the cyber-attack from the output and remaining undetectable making coding useless. It is worth noting that one can design sophisticated coding capable of recovering attacked signals, however, this results in increased computation time, which can be impractical for most control systems with limited computation capabilities.

Another notable limitation of coding schemes is the incapability to isolate faults from cyber-attack since an observer-based detector will trigger an alarm on both faults and cyber-attacks. This section and the following sections are dealing with the detection of cyber-attacks. Faults isolation from cyber-attacks are investigated in the last section of this chapter.

In this section, the effects of cyber-attacks on inputs and outputs of a multi-input multi-output (MIMO) LTI system are examined, and it is shown how cyber-attacks on coded signals change decoded inputs and outputs. For the analysis, it is assumed that the attackers possess complete knowledge of the system dynamics and have full access to all communication channels, but are unaware of the coding scheme constants. Hence, attackers can manipulate coded inputs and outputs as shown in Figure 4.9.

For clarity, the figure illustrates one input and two outputs, however, the analysis is not restricted by the number of inputs and outputs. Assume attackers inject cyber-attack signals $u_i^a(k)$ on coded input $q_i(k)$ and another cyber-attack signal $y_i^a(k)$ on coded output $v_i'(k)$, as shown in Figure 4.9.

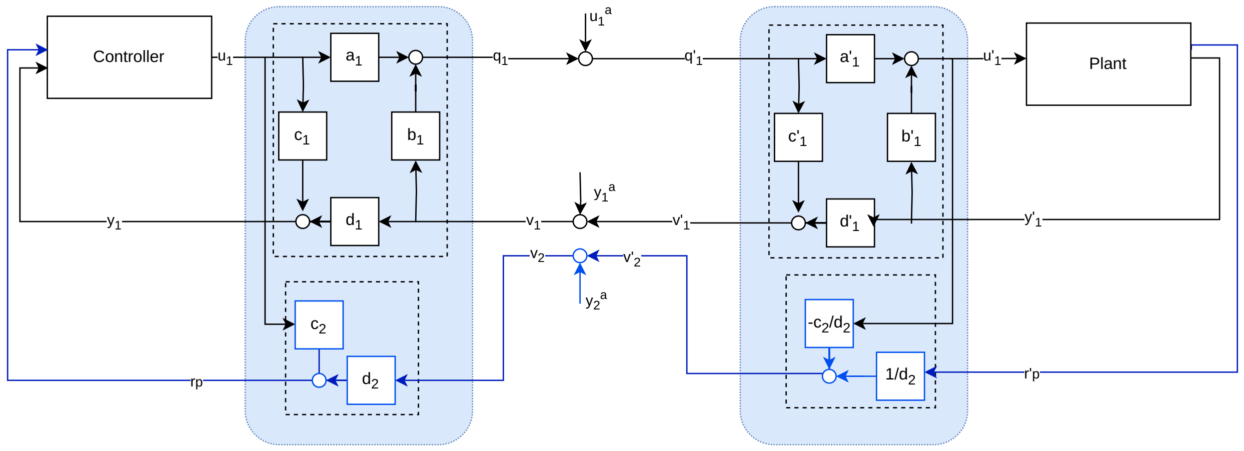


Figure 4.9: Two-way coding of a MIMO system under cyber-attack.

For this system, we have:

$$q_i'(k) = q_i(k) + u_i^a(k) \tag{4.22}$$

$$v_i(k) = v_i'(k) + y_i^a(k) \tag{4.23}$$

For the first pair of input-output, on the plant side the cyber-attack signal will change the value of coded output as follows:

$$\begin{aligned} v_1'(k) &= c_1'q_1'(k) + d_1'y_1'(k) \\ &= c_1'[q_1(k) + u_1^a(k)] + d_1'y_1'(k) \\ &= c_1'q_1(k) + d_1'y_1'(k) + c_1'u_1^a(k) \end{aligned} \tag{4.24}$$

On the controller side, from equation (4.23) and (4.24) we have:

$$v_1(k) = c_1'q_1(k) + d_1'y_1'(k) + c_1'u_1^a(k) + y_1^a(k) \tag{4.25}$$

Under cyber-attack coded input $q_1(k)$, can be calculated by replacing $v_1(k)$ from equation (4.25):

$$\begin{aligned} q_1(k) &= a_1u_1(k) + b_1v_1(k) \\ &= a_1u_1(k) + b_1[c_1'q_1(k) + d_1'y_1'(k) + c_1'u_1^a(k) + y_1^a(k)] \\ &= a_1u_1(k) + b_1c_1'q_1(k) + b_1d_1'y_1'(k) + b_1c_1'u_1^a(k) + b_1y_1^a(k) \\ (1 - b_1c_1')q_1(k) &= a_1u_1(k) + b_1d_1'y_1'(k) + b_1c_1'u_1^a(k) + b_1y_1^a(k) \\ q_1(k) &= \frac{1}{a_1'}u_1(k) + \frac{b_1}{d_1}y_1'(k) - \frac{b_1c_1}{a_1d_1}u_1^a(k) + \frac{b_1}{d_1d_1'}y_1^a(k) \end{aligned} \tag{4.26}$$

Equation (4.26) shows the changes in coded input due to the cyber-attacks on both coded input and output. Replacing for $q_1(k)$ from (4.26) into (4.25), and simplification, the under cyber-attack coded output became:

$$v_1(k) = -\frac{c_1}{d_1}u_1(k) + \frac{1}{d_1}y_1'(k) - \frac{c_1}{a_1d_1}u_1^a(k) + \frac{1}{d_1d_1'}y_1^a(k) \tag{4.27}$$

Now we can calculate under-attack decoded output $y_1(k)$ on the controller side as shown below:

$$\begin{aligned} y_1(k) &= c_1u_1(k) + d_1v_1(k) \\ &= c_1u_1(k) + d_1[-\frac{c_1}{d_1}u_1(k) + \frac{1}{d_1}y'_1(k) - \frac{c_1}{a_1d_1}u_1^a(k) + \frac{1}{d_1d'_1}y_1^a(k)] \\ &= y'_1(k) - \frac{c_1}{a_1}u_1^a(k) + \frac{1}{d'_1}y_1^a(k) \end{aligned} \tag{4.28}$$

Next, it is needed to find under-attack control input on the plant side. From equation (4.26), on the plant side we have:

$$\begin{aligned} q'_1(k) &= q_1(k) + u_1^a(k) \\ &= \frac{1}{a'_1}u_1(k) + \frac{b_1}{d_1}y'_1(k) - \frac{b_1c_1}{a_1d_1}u_1^a(k) + \frac{b_1}{d_1d'_1}y_1^a(k) + u_1^a(k) \\ &= \frac{1}{a'_1}u_1(k) + \frac{b_1}{d_1}y'_1(k) + \frac{1}{d_1d'_1}u_1^a(k) + \frac{b_1}{d_1d'_1}y_1^a(k) \end{aligned} \tag{4.29}$$

Hence, the decoded under cyber-attack control input $u'_1(k)$ on the plant side is:

$$\begin{aligned} u'_1(k) &= a'_1q'_1(k) + b'_1y'_1(k) \\ &= a'_1[\frac{1}{a'_1}u_1(k) + \frac{b_1}{d_1}y'_1(k) + \frac{1}{d_1d'_1}u_1^a(k) + \frac{b_1}{d_1d'_1}y_1^a(k)] + b'_1y'_1(k) \\ &= u_1(k) + \frac{1}{a_1}u_1^a(k) + \frac{b_1}{a_1}y_1^a(k) \end{aligned} \tag{4.30}$$

The effect of the cyber-attack signal on the second coded output is different since it is coded with a different coding matrix than the one used for coding the first control input. To calculate the effect of cyber-attack on the second coded output $v'_2(k)$ coded by the first decoded input $u'_1(k)$, the under-attack value of $u'_1(k)$ is replaced from equation (4.30) into equation (4.17) as follows:

$$\begin{aligned} v'_2(k) &= -\frac{c_2}{d_2}u'_1(k) + \frac{1}{d_2}y'_2(k) \\ &= -\frac{c_2}{d_2}\left[u_1(k) + \frac{1}{a_1}u_1^a(k) + \frac{b_1}{a_1}y_1^a(k)\right] + \frac{1}{d_2}y'_2(k) \\ &= -\frac{c_2}{d_2}u_1(k) - \frac{c_2}{a_1d_2}u_1^a(k) - \frac{c_2b_1}{a_1d_2}y_1^a(k) + \frac{1}{d_2}y'_2(k) \end{aligned} \tag{4.31}$$

Decoding of second under cyber-attack coded output $v_2'(k)$ on the controller side, we can calculate the effect of cyber-attack on decoded output $y_2(k)$ on the controller side:

$$\begin{aligned}
y_2(k) &= c_2u_1(k) + d_2v_2(k) \\
&= c_2u_1(k) + d_2\left[-\frac{c_2}{d_2}u_1(k) - \frac{c_2}{a_1d_2}u_1^a(k) - \frac{c_2b_1}{a_1d_2}y_1^a(k) + \frac{1}{d_2}y_2'(k) + y_2^a(k)\right] \\
&= y_2'(k) - \frac{c_2}{a_1}u_1^a(k) - \frac{c_2b_1}{a_1}y_1^a(k) + d_2y_2^a(k) \qquad (4.32)
\end{aligned}$$

With a similar analysis for a system that the number of inputs is more than the number of outputs and hence, the second control input is coded with the first plant output as shown in Figure 4.8, using equation (4.19) and (4.28), we have:

$$\begin{aligned}
q_2(k) &= \frac{b_2}{d_2}y_1(k) + \frac{1}{a_2'}u_2(k) \\
&= \frac{b_2}{d_2}\left[y_1'(k) - \frac{c_1}{a_1}u_1^a(k) + \frac{1}{d_1'}y_1^a(k)\right] + \frac{1}{a_2'}u_2(k) \\
&= \frac{b_2}{d_2}y_1' - \frac{b_2c_1}{d_2a_1}u_1^a(k) + \frac{b_2}{d_2d_1'}y_1^a(k) + \frac{1}{a_2'}u_2(k) \qquad (4.33)
\end{aligned}$$

Using this coded input and equation (4.22), the second under-attack decoded input $u_2'(k)$ on the plant side can be written as:

$$\begin{aligned}
u_2'(k) &= a_2'q_2'(k) + b_2'y_1'(k) \\
&= a_2'\left[\frac{b_2}{d_2}y_1' - \frac{b_2c_1}{d_2a_1}u_1^a(k) + \frac{b_2}{d_2d_1'}y_1^a(k) + \frac{1}{a_2'}u_2(k) + u_2^a(k)\right] + b_2'y_1'(k) \\
&= u_2(k) + \frac{b_2'c_1}{a_1}u_1^a(k) - \frac{b_2'}{d_1'}y_1^a(k) + a_2'u_2^a(k) \qquad (4.34)
\end{aligned}$$

The effect of cyber-attacks on decoded inputs and outputs of a MIMO system equipped with the proposed coding schemes can be summarized as follows:

$$y_1(k) = y'_1(k) - \frac{c_1}{a_1} u_1^a(k) + \frac{1}{d'_1} y_1^a(k) \tag{4.28}$$

$$u'_1(k) = u_1(k) + \frac{1}{a_1} u_1^a(k) + \frac{b_1}{a_1} y_1^a(k) \tag{4.30}$$

$$y_2(k) = y'_2(k) - \frac{c_2}{a_1} u_1^a(k) - \frac{c_2 b_1}{a_1} y_1^a(k) + d_2 y_2^a(k) \tag{4.32}$$

$$u'_2(k) = u_2(k) + \frac{b'_2 c_1}{a_1} u_1^a(k) - \frac{b'_2}{d'_1} y_1^a(k) + a'_2 u_2^a(k) \tag{4.34}$$

For analysis of the systems and notation convenience, the terms introduced by the cyber-attack could be indicated by σ_{yi} for an cyber-attack on output $y_i(k)$, and as σ_{ui} for an cyber-attack on input $u'_i(k)$. Hence, the compact form of the above equations is presented as follows:

$$u'_i(k) = u_i(k) + \sigma_{ui}(k) \tag{4.35}$$

$$y_i(k) = y'_i(k) + \sigma_{yi}(k) \tag{4.36}$$

And under cyber-attack inputs and outputs in the vector form is represented as:

$$u'(k) = u(k) + \Sigma_u(k) \tag{4.37}$$

$$y(k) = y'(k) + \Sigma_y(k) \tag{4.38}$$

where $\Sigma_u(k)$ and $\Sigma_y(k)$ for the above system with one input and two outputs is as follows:

$$\Sigma_u(k) = \underbrace{\begin{bmatrix} \frac{1}{a_1} & 0 \\ \frac{b'_2 c_1}{a_1} & a'_2 \end{bmatrix}}_{C_u} \begin{bmatrix} u_1^a(k) \\ u_2^a(k) \end{bmatrix} + y_1^a(k) \underbrace{\begin{bmatrix} \frac{b_1}{a_1} \\ \frac{-b'_2}{d'_1} \end{bmatrix}}_{C_y} \tag{4.39}$$

$$\Sigma_y(k) = u_1^a(k) \underbrace{\begin{bmatrix} \frac{-c_1}{a_1} \\ \frac{-c_2}{a_1} \end{bmatrix}}_{D_u} + \underbrace{\begin{bmatrix} \frac{1}{d'} & 0 \\ \frac{c_2 b_1}{a_1} & d_2 \end{bmatrix}}_{D_y} \begin{bmatrix} y_1^a(k) \\ y_2^a(k) \end{bmatrix} \tag{4.40}$$

For the derivation of this equation, a system with one input and two outputs is considered, however, it can be extended for any number of inputs and outputs. The effect of cyber-attacks on inputs and outputs, shown in equations (4.39) and (4.40), is a combination of a constant matrix constructed with coding matrix elements multiplied by the injected cyber-attacks. Hence, it can be rearranged in a more compact form as follows:

$$\Sigma_u(k) = C_u u^a(k) + C_y y^a(k) \tag{4.41}$$

$$\Sigma_y(k) = D_u u^a(k) + D_y y^a(k) \tag{4.42}$$

where u^a and y^a are injected cyber-attack signals on inputs and outputs, and C_u, C_y, D_u and D_y are constant vectors or matrices constructed only by coding matrix elements. These equations show that the decoded inputs and outputs under cyber-attacks are functions of the coding matrix. Therefore, the condition for the attackers to remove the effects of the cyber-attacks on inputs from outputs is the knowledge of the coding matrix, which is assumed unknown to the attackers.

4.4.1 Detection of Cyber-attack Using Kalman Filter

This subsection deals with the problem of covert cyber-attack detection, which is undetectable with traditional observer-based detectors, such as the Kalman filter. The two-way coding scheme developed in the previous sections is proposed as a solution, to make the cyber-attacks detectable with a Kalman filter-based detector for a multi-input multi-output (MIMO) discrete-time system. The Kalman filter-based detector as described in Section 2.5, equations (2.48) to (2.52) is implemented on the controller sides as shown in Figure 4.10.

Kalman filter has a proven history in state estimation and fault detection and is widely used for cyber-attack detection. In this section the Kalman filter is used to analyze the effects of coding on the estimation and hence, detection of cyber-attacks. In the presence of a Kalman filter detector, the stealthy cyber-attack objective would be to maximize estimator error while keeping residual $z(k)$ small enough to prevent triggering of an alarm [77]. Therefore, to analyze the effect of the cyber-attack, it is required to calculate residuals and estimator errors under cyber-attack. In the following discussion.

Kalman gain $K(k)$, for the LTI systems is a constant value, and it will be shown as K. Also, for notation convenience, $\hat{x}_{k|k}$ will be noted as $\hat{x}(k)$. The estimation error dynamics for a healthy system is calculated using equations (2.48) and (2.50) as shown below:

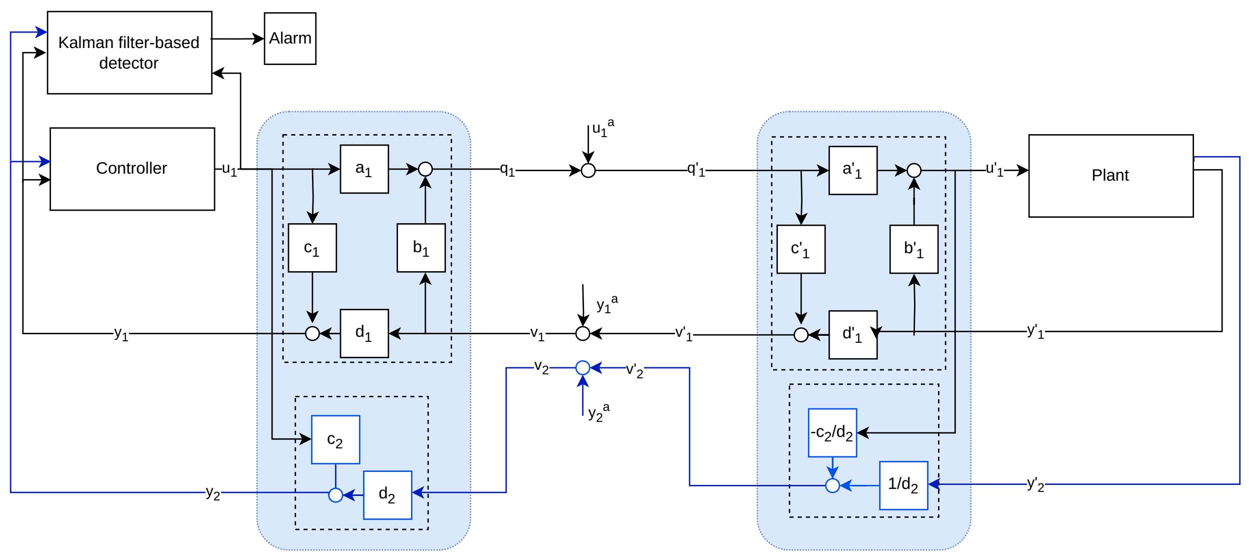


Figure 4.10: Two-way coding of a MIMO system under cyber-attack equipped with a Kalman filter-based detector on the controller side.

$$
\begin{aligned}
\hat{x}(k+1|k) &= A\hat{x}(k) + Bu(k)\\
\hat{x}(k+1) &= \hat{x}(k+1|k) + K[y(k+1) - C\hat{x}(k+1|k)]\\
&= A\hat{x}(k) + Bu(k) + K[Cx(k+1) + \nu(k+1) - C(A\hat{x}(k) + Bu(k))]\\
&= (A - KCA)\hat{x}(k) + Bu(k) - KCBu(k) + KC(Ax(k) + Bu(k) + w(k)) + K\nu(k+1)\\
&= (A - KCA)\hat{x}(k) + Bu(k) + KCAx(k) + KCw(k) + K\nu(k+1) \qquad (4.43)
\end{aligned}
$$

The error dynamics of estimation, when there is no cyber-attack is $e(k+1) = x(k+1) - \hat{x}(k+1)$ hence, using equation (4.43) we have:

$$
\begin{aligned}
e(k+1) &= x(k+1) - \hat{x}(k+1)\\
&= Ax(k) + Bu(k) + w(k) - [(A - KCA)\hat{x}(k) + Bu(k) + KCAx(k) + KCw(k) + K\nu(k+1)]\\
&= (A - KCA)x(k) - (A - KCA)\hat{x}(k) - K\nu(k+1) + (I - KC)w(k)\\
&= (A - KCA)e(k) - K\nu(k+1) + (I - KC)w(k) \qquad (4.44)
\end{aligned}
$$

Since the process noise $w(k)$ and measurement noises $\nu(k+1)$ are assumed Gaussian, the expected values of noises are 0, that is $\mathbb{E}[w(k)] = 0$ and $\mathbb{E}[\nu(k+1)] = 0$. Hence, the expected estimation error $e(k+1)$ of properly designed Kalman filter, that is for stable $A - KCA$, as times progress approaches zero.

$$\lim_{k\to\infty} \mathbb{E}[e(k+1)] = (A - KCA)\mathbb{E}[e(k)] = 0 \tag{4.45}$$

For normal operating conditions when there are no cyber-attacks the residual is as $z(k) = y(k) - \hat{y}(k)$ and is calculated as follows:

$$\begin{aligned} z(k) &= y(k) - \hat{y}(k) \\ &= y(k) - C[A\hat{x}(k-1) + Bu(k-1)] \end{aligned} \tag{4.46}$$

From equation (2.57), as explained in the Subsection 2.5.1, to detect a cyber-attacks, the $g(k) = z(k)^T \Sigma(k)^{-1} z(k)$ should exceed a threshold τ. For a successful cyber-attack on the system described in this chapter, the attacker's objective would be to increase estimation error while keeping the residual small enough to prevent the triggering of an alarm.

To analyze the system under cyber-attack, assume the attackers inject sequences of cyber-attack signals $u^a(k)$ and $y^a(k)$ on coded inputs and outputs. The estimated states $\hat{x}$ on the controller side are calculated by $u(k)$, and the outputs are calculated by the equation (4.38). Meantime, the actual states of the system on the plant side are affected by compromised input $u'(k)$, Hence, the under-attack estimated states $\hat{x}^a$ are calculated as follows:

$$\begin{aligned} \hat{x}^a(k+1|k) &= A\hat{x}^a(k) + Bu(k) \\ \hat{x}^a(k+1) &= \hat{x}^a(k+1|k) + K[y(k+1) - C\hat{x}^a(k+1|k)] \\ &= A\hat{x}^a(k) + Bu(k) + K[y'(k+1) + \Sigma_y(k+1) - C\hat{x}^a(k+1|k)] \\ &= A\hat{x}^a(k) + Bu(k) + K\Sigma_y(k+1) + K[Cx'(k+1) + \nu(k+1) - C(A\hat{x}^a(k) + Bu(k))] \\ &= (A - KCA)\hat{x}^a(k) + Bu(k) - KCBu(k) + KC[Ax'(k) + Bu'(k) + w(k)] \\ &\quad + K\nu(k+1) + K\Sigma_y(k+1) \\ &= (A - KCA)\hat{x}^a(k) + Bu(k) - KCBu(k) + KCAx'(k) + KCB(u(k) + \Sigma_u(k)) \\ &\quad + KCw(k) + K\nu(k+1) + K\Sigma_y(k+1) \\ &= (A - KCA)\hat{x}^a(k) + Bu(k) + KCAx'(k) + KCw(k) + K\nu(k+1) + \\ &\quad K\Sigma_y(k+1) + KCB\Sigma_u(k) \end{aligned} \tag{4.47}$$

where $\hat{x}^a$, is under cyber-attack estimated states, and x' is the actual states of the system on the plant side affected by cyber-attack signals. Hence, under cyber-attack estimation error,

$e^a(k+1) = x'(k+1) - \hat{x}^a(k+1)$ is calculated as follows:

$$\begin{aligned}
e^a(k+1) &= x'(k+1) - \hat{x}^a(k+1) \\
&= Ax'(k) + Bu'(k) + w(k) - [(A-KCA)\hat{x}^a(k) + Bu(k) + KCAx'(k) \\
&\quad + KCw(k) + K\nu(k+1) + K\Sigma_y(k+1) + KCB\Sigma_u(k)] \\
&= (A-KCA)x'(k) - (A-KCA)\hat{x}^a(k) + Bu(k) + B\Sigma_u(k) - Bu(k) \\
&\quad - K\nu(k+1) + (I-KC)w(k) - K\Sigma_y(k+1) - KCB\Sigma_u(k) \\
&= (A-KCA)e^a(k) - K\nu(k+1) + (I-KC)w(k) - K\Sigma_y(k+1) + (B-KCB)\Sigma_u(k)
\end{aligned} \tag{4.48}$$

Therefore, the under cyber-attack expected error of estimation, as time progresses will become:

$$\lim_{k\to\infty} \mathbb{E}[e^a(k+1)] = (A-KCA)\mathbb{E}[e^a(k)] - K\Sigma_y(k+1) + (B-KCB)\Sigma_u(k) \tag{4.49}$$

Since the inputs on the controller side $u(k)$ are different from inputs on the plant side $u'(k)$, actual states are deviating from estimated states, and so the estimation error $e^a(k)$ will not approach zero. Moreover, since we assumed that the attackers do not know the coding matrix, they can not cancel out terms containing $\Sigma_y(k+1)$ and $\Sigma_u(k)$, which results in a further increase of the error term $e^a(k+1)$.

The under cyber-attack residuals calculated by the controller side Kalman filter detector are as follows:

$$\begin{aligned}
z^a(k+1) &= y(k+1) - \hat{y}^a(k+1) \\
&= y'(k+1) + \Sigma_y(k+1) - C\hat{x}^a(k+1) \\
&= Cx'(k+1) + \nu(k+1) + \Sigma_y(k+1) - C[A\hat{x}^a(k) + Bu(k)] \\
&= C[Ax'(k) + Bu'(k) + w(k)] + \nu(k+1) + \Sigma_y(k+1) - CA\hat{x}^a(k) - CBu(k) \\
&= CA[x'(k) - \hat{x}^a(k)] + CBu(k) + CB\Sigma_u(k) + Cw(k) + \nu(k+1) \\
&\quad + \Sigma_y(k+1) - CBu(k) \\
&= CAe^a(k) + Cw(k) + \nu(k+1) + CB\Sigma_u(k) + \Sigma_y(k+1)
\end{aligned} \tag{4.50}$$

Rearrange equation (4.48), using equation (4.50) we have:

$$\begin{aligned}e^a(k+1) &= (A-KCA)e^a(k) - K\nu(k+1) + (I-KC)w(k) - K\Sigma_y(k+1) + (B-KCB)\Sigma_u(k)\\ &= [Ae^a(k) + B\Sigma_u(k)] - K[CAe^a(k) + \nu(k+1) + CB\Sigma_u(k) + \Sigma_y(k+1)] + (I-KC)w(k)\\ &= [Ae^a(k) + B\Sigma_u(k)] - Kz^a(k+1) + (I+KC)w(k)\end{aligned}$$

Hence, the expected value of estimation error became:

$$\lim_{k\to\infty} \mathbb{E}[e^a(k+1)] = A\mathbb{E}[e^a(k)] + B\Sigma_u(k) - K\mathbb{E}[z^a(k+1)]$$

To launch a successful cyber-attack on this system, the attackers should ensure that $e^a(k+1)$ will grow while keeping the residual small enough to make the cyber-attack undetectable. That is $z^T(k+1)\Sigma(k+1)z(k+1) < g$, to prevent trigger of alarm. Therefore, the attackers objective would be to grow $A\mathbb{E}[e^a(k)] + B\Sigma_u(k) \to \infty$. Meanwhile, the expected value of residuals $z^a(k+1)$ from equation (4.50) are:

$$\begin{aligned}\mathbb{E}[z^a(k+1)] &= \mathbb{E}[CAe^a(k) + Cw(k) + CB\Sigma_u(k) + \Sigma_y(k+1)] + \mathbb{E}[\nu(k+1)]\\ &= CA\mathbb{E}[e^a(k)] + CB\Sigma_u(k) + \Sigma_y(k+1) \qquad (4.51)\end{aligned}$$

To keep the residuals small enough to prevent triggering the alarm it is required to have $\Sigma_y(k+1) = -CAe^a(k) - CB\Sigma_u(k)$. Hence, the attackers have to construct $\Sigma_y(k+1)$ by injecting an appropriate cyber-attack signal on the coded outputs to remove the overall effect of the cyber-attack signal $u^a(K)$ on coded inputs. To find the required cyber-attack signal, first, we need to replace equations (4.41) and (4.42) into the above equation as follows:

$$D_u u^a(k+1) + D_y y^a(k+1) = -CAe^a(k) - CBC_u u^a(k) - CBC_y y^a(k) \qquad (4.52)$$

Rearranging equation (4.52), the required cyber-attack signal is:

$$y^a(k+1) = D_y^{-1}D_u u^a(k+1) - D_y^{-1}CAe^a(k) - D_y^{-1}CBC_u u^a(k) - D_y^{-1}CBC_y y^a(k) \qquad (4.53)$$

To remove the effect of cyber-attack from residuals, the attacker needs to solve equation (4.53) and calculate the cyber-attack signal. To solve this equation, the attackers need to construct C_u, C_y, D_u, and D_y using coding matrices and have full knowledge of the system dynamics. Since we assumed that the attackers do not know coding matrices, they can not fulfill their objectives. This proves that the proposed coding schemes can detect covert cyber-attacks.

4.4.2 Simulation Results of Cyber-attack on Two-way Coding Scheme

Depending on the design of two-way coding, it is possible to achieve different objectives. For instance, equation (4.30) shows that a large a, relative to control input $u(k)$, can reduce or even remove the effect of noise-like cyber-attack signals $u^a(k)$ on control input on the plant side. In addition, keeping b/a small will reduce the consequences of cyber-attack signal $y^a(k)$ on control input on the plant. It is always possible for the attackers to inject relatively larger signals, but we assumed that attackers do not know the coding schemes. Therefore, it would not be possible for them to precisely calculate the cyber-attack signals' magnitude to achieve a specific objective.

To demonstrate the effect of the cyber-attack on coded inputs and outputs and its impact on the Kalman filter detector, the UAV simulator presented in Section 3.3 is used, with two-way coding implemented between the controller and the plant's actuators and sensors. The system is a linearized latitudinal model of UAV consisting of two inputs of aileron (δ_a) and rudder (δ_r) angles and five outputs of side-slip angle ($\beta(rad/s)$), roll rate ($p(rad/s)$), yaw rate ($r(rad/s)$), roll angle ($\phi(rad)$), and yaw angle ($\psi(rad)$). A linear Kalman filter for the latitudinal model, as explained in section 2.5, is implemented on the controller side. The overall coded system is a replica of Figure 4.10.

Apart from the design of two-way coding for the mitigation of zero-dynamics of the augment system in [79], which is just for one specific cyber-attack signal, there are no general procedures for designing coding for cyber-attack detection in the literature. In this simulation, the coding is designed to alter the original signals as a combination of both sides' signals. For instance, the roll angle ϕ which is coded with aileron command δ_a, has an order of magnitude larger numerical values. Hence, the coding matrix elements are chosen to magnify the effect of δ_a, in coded input and output.

The objective of the simulation is to show how coding can make a system resilient against covert cyber-attacks. Hence, a simple coding was designed and implemented to demonstrate this. Assuming the design objective of two-way coding is to reduce the effect of noise-like cyber-attack signals on control input, magnify the impact of cyber-attack on control input on the plant outputs to make the detection faster and hide the actual dynamics of input-output pair by mixing them.

To achieve these objectives, for instance, we may take the roll angle ϕ and aileron command δ_a as a pare of input-output for coding. Since in the normal operating condition, ϕ has an order of magnitude larger numerical values than δ_a, to scramble the coded input and output, we need to magnify δ_a by order of magnitude to make its effects equivalent to ϕ. Moreover, inspecting equations (4.28) to (4.34) shows that to mitigate the impact of a small noise-like cyber-attack on coded input, we need a relatively large value for a and a small value b/a. On the other hand, to magnify changes in decoded output, a relatively large value of c/a and small d' is required.

Combining these requirements and inspecting numerical values of signals from the UAV simulator, with some try-and-error, we may choose $a = 10$ for scaling up δ_a an order of magnitude, and then from equation (4.30), to keep b/a small we can choose $b = 0.5$. Since δ_a in normal operating conditions in the simulator varies between 0.005 to 0.05 radian, these values of a and b also can mitigate noise-like cyber-attack signals.

To magnify the impact of cyber-attacks on decoded output, inspecting equation (4.28) we have:

$$\begin{aligned} y_1(k) &= y_1'(k) - \frac{c_1}{a_1}u_1^a(k) + \frac{1}{d_1'}y_1^a(k) \\ &= y_1'(k) - \frac{c_1}{a_1}u_1^a(k) + d_1 y_1^a(k) - \frac{b_1 c_1}{a_1}y_1^a(k) \end{aligned}$$

If $c > a$, then the impact of the cyber-attack signal $u^a(k)$ on input will be more severe on decoded output $y(k)$. A small value for d will reduce the effect of the cyber-attack $y^a(k)$ on the outputs $y(k)$. Of course, too large a value of c will magnify the response of the controller and can have a diverse impact on the plant, which is undesirable. Hence, after some try-and-error, these two constants are set to $c = 20$ and $d = 0.5$. The coding matrix for this pair of input-output will become:

$$M = \begin{bmatrix} 10 & 0.5 \\ 20 & 0.5 \end{bmatrix} \qquad M^{-1} = \begin{bmatrix} -0.1 & 0.1 \\ 4 & -2 \end{bmatrix} \tag{4.54}$$

Assume that the attackers injected cyber-attack signals into the pair of coded input-output constructed by aileron command δ_a and roll angle ϕ. From equations (4.28) and (4.30) and the coding matrix shown by equation (4.54), the decoded roll angle ϕ on the controller side and decoded aileron command on the plant side under cyber attack are given by:

$$\phi(k) = \phi'(k) - 2u_1^a(k) - 0.5y_1^a(k) \tag{4.55}$$

$$\delta_a'(k) = \delta_a(k) + 0.1u_1^a(k) + 0.05y_1^a(k) \tag{4.56}$$

where the $u_1^a(k)$ and $y_1^a(k)$ are injected cyber-attack signals into the coded ϕ and coded δ_a signals. Equations (4.55) and (4.56) show the impact of cyber-attacks on decoded signals. It is worth noting that although we coded ϕ and δ_a as a pair signal, the under cyber-attack decoded signals are only impacted by the cyber-attack signals.

Using the above coding scheme, the original signals, that is, δ_a and ϕ, and the resulting coded input and output, that is q_1 and v_1' in the absence of cyber-attacks, are shown in Figure 4.11.

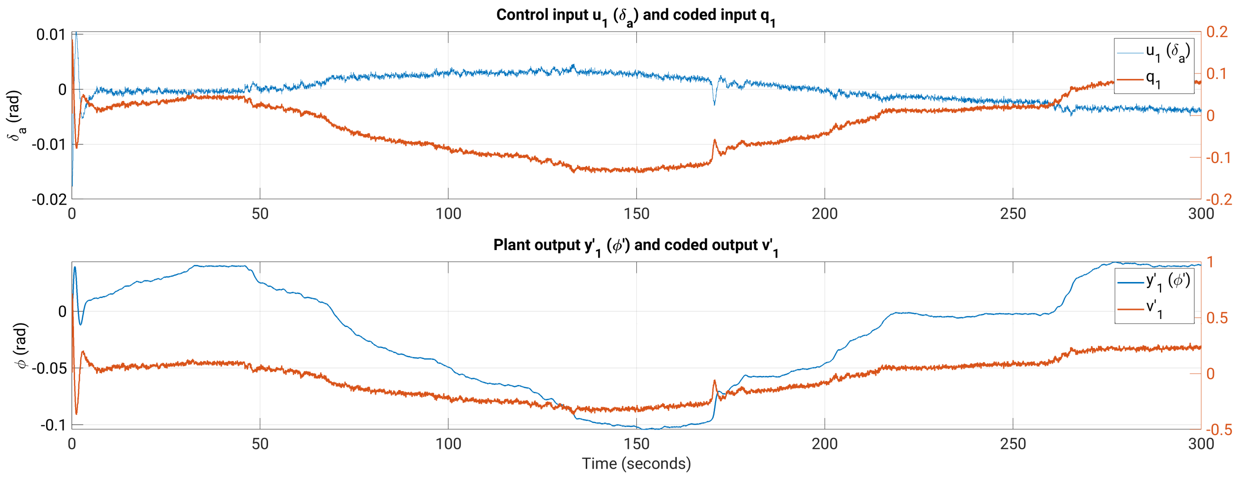


Figure 4.11: Original and coded signals of UAV in the absence of cyber-attack.

It can be seen that coded input and output are mapped to different values concerning the original signals. We emphasize that a system equipped with coding is resilient to covert cyber-attacks. Hence, it is impossible to launch a covert cyber-attack on this system. In the simulation, we assume the attackers inject random signals to probe the system response. If the number of injected signals increases, it would be easier to detect the cyber-attack; hence, we assume that attackers inject a pair of signals with opposite magnitude to a pair of coded input-output.

To demonstrate the impacts of cyber-attacks on a system equipped with the two-way coding scheme, we assume that attackers inject two ramp signals, shown in Figure 4.12, into coded input and output to find out system weaknesses. The cyber-attack starts at time stamp 100, and its magnitude gradually increases up to time stamp 200, where signals remain constant at their maximum values.

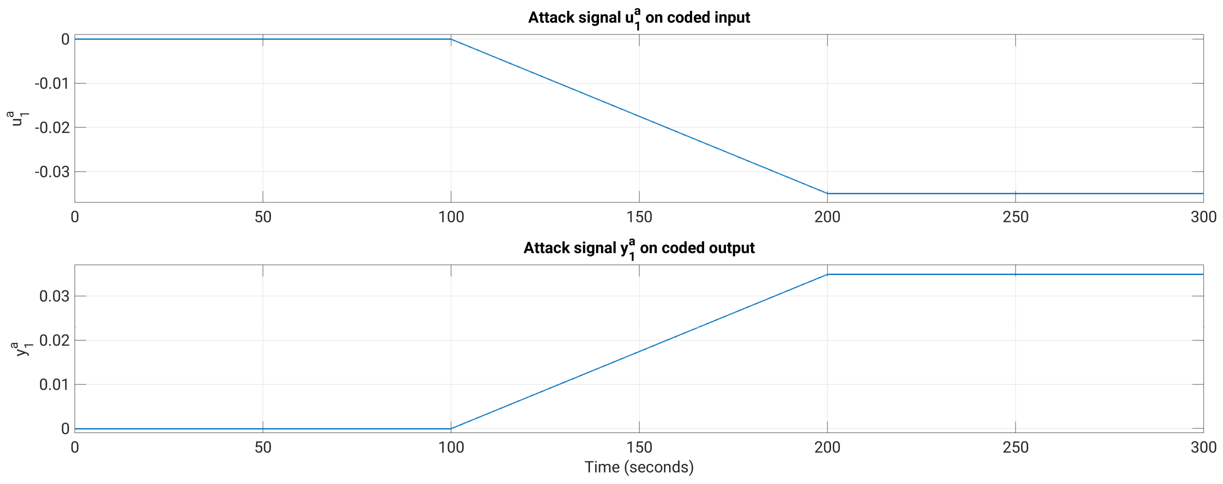


Figure 4.12: Cyber-attack signals injected on coded input and output.

The maximum cyber-attack signal magnitude injected on coded input of δ_a is -0.035 radian, and the maximum cyber-attack signal injected on the coded output of ϕ is 0.035, which is equivalent to +2 degree and -2 degree if there was no coding. The coded and decoded input and output after injection of the cyber-attack signal are shown in Figure 4.13.

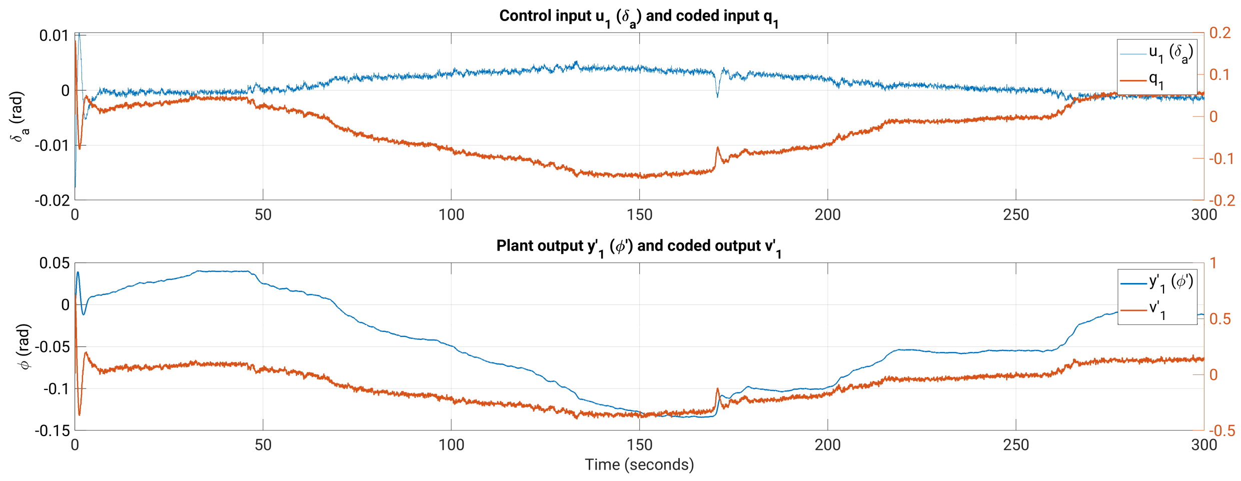


Figure 4.13: Coded and decoded signals under cyber-attack.

Figure 4.14 illustrates original and under cyber-attack signals, that compared with the healthy system, both coded and decoded signals are changed. At time stamp 100 when the cyber-attack started, the deviation of the under cyber-attacked signals from the original one gradually grows and reaches its maximum at time stamp 200, where the cyber-attack signal remains constant. Using equations (4.28) and (4.30), the maximum deviation of plant output and controller input are respectively -0.0525 and 0.00175 radians. Converting radian to degree we have $y_1(k) = y'_1(k) - 5.0$ and $u'(k) = u(k) + 0.1$, which can be visually verified from Figure 4.14. Hence, the impact of the cyber-attack signals with a maximum magnitude of 2 degrees, with the developed coding scheme in place, is 0.1 degrees on the input and 5 degrees on the output. Therefore, the cyber-attack signal impact is mitigated on the plant side and magnified on the controller side as expected.

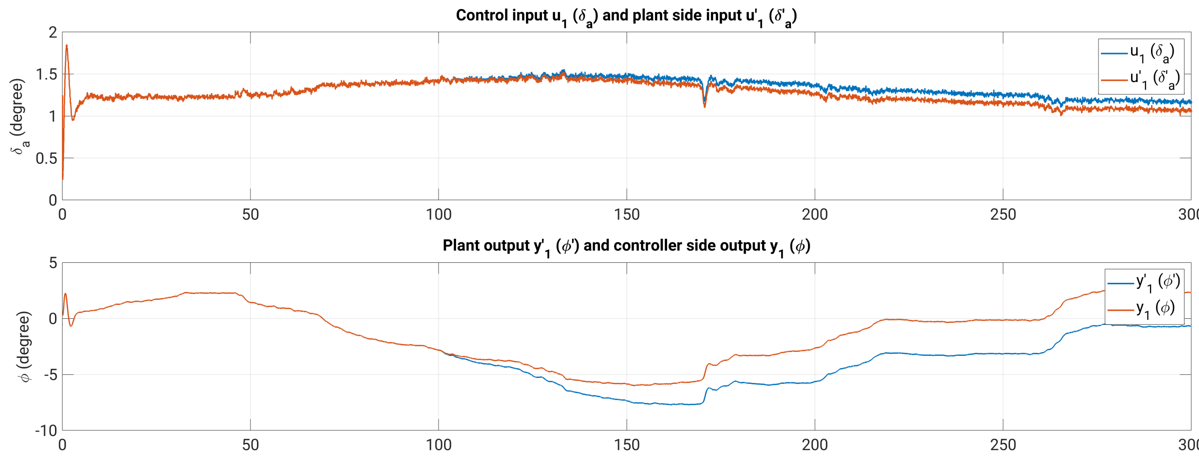


Figure 4.14: Original and under cyber-attack decoded signals. Cyber-attack starts at time stamp 100 and reaches its maximum at time stamp 200.

On the controller side a Kalman filter with χ^2 detector is designed as explained in Section 2.5. Residuals generated by the Kalman filter and χ^2 are shown in Figure 4.15. The threshold for χ^2 is determined from the table of χ^2 distribution [101].

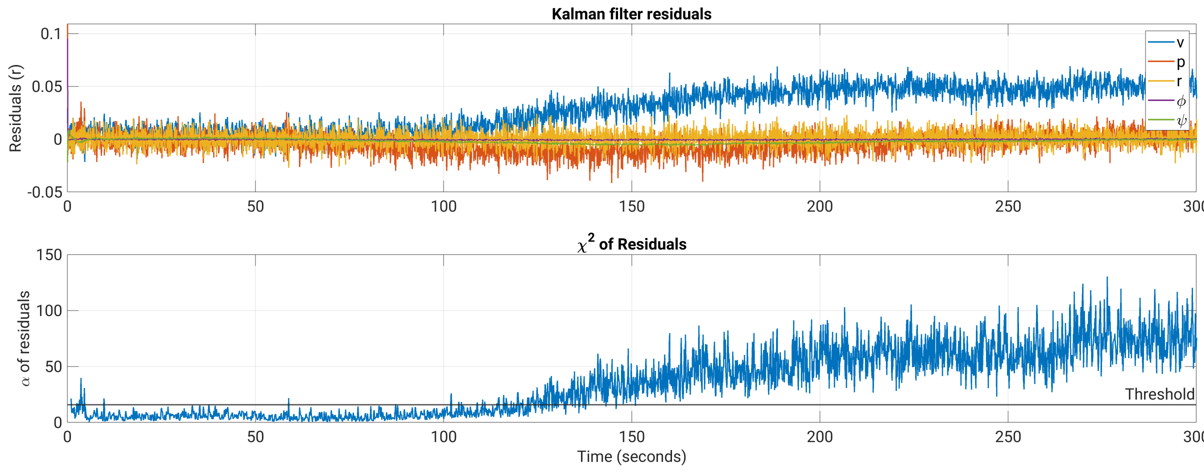


Figure 4.15: Cyber-attack detection with Kalman filer generated residuals and χ^2 detector.

From Figure 4.15, it can be seen that when cyber-attack signals are small, residuals do not show detectable deviation. By growing cyber-attack signals, the Kalman filter generated residuals also increase, which results in the χ^2 value surpassing the threshold and detection of the cyber-attack. To quantify the results of the cyber-attack detection for a system equipped with two-way coding, we run a Monte Carlo simulation for 50 random scenarios. We used a flight scenario of about 1800 seconds based on randomly generated control commands for the ground control system. Some parts of the flight path of this scenario are shown in Figure 3.2, and the relevant actuators' control commands are shown in Figure4.16.

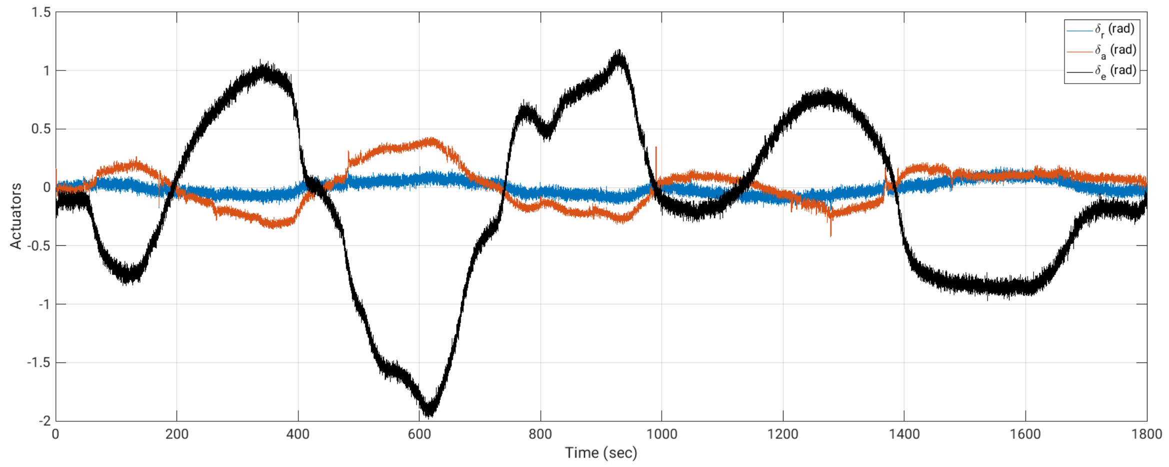


Figure 4.16: Randomly generated control commands for Monte Carlo simulation

We also designed a set of step cyber attack signals that when decoded, have an impact of 0.05 to 5 degrees on actuator control command. We randomly selected five slices of 300-second non-overlapping flight scenarios and injected ten pairs of step cyber-attack signals with randomly chosen magnitude on the coded input and output. Since we have ten samples per second, there are 3000 samples for each 300-second scenario. We calculated χ^2 of each sample and compared it with the threshold to detect cyber-attacks. The χ^2 detector could not detect The χ^2 detector could not detect cyber-attacks with an impact of less than 0.1 degrees on actuators and less than 1 degrees on measurements because these values are in the range of noises and disturbances of those signals. We could marginally detect cyber-attacks with an impact of between 0.1 to 0.2 degrees on actuators and 1 to 2 degrees on measurements. Our detector could detect cyber-attacks that result in greater than 0.2 degrees deviation on the actuators and 2 degrees on measurements. We averaged the results of 50 runs of simulation presented in Table 4.1 to 4.3

Our proposed method can detect cyber-attacks with an F1-score of 0.99 for attack signals greater than $|0.1|$ degrees on the actuator and greater than $|1|$ degree on the measurement. It is worth noting that, The coding scheme instantaneously reflects any injected cyber-attack into the decoded inputs, which results in faster and more effective cyber-attack detection. However, if a cyber-attack impacts the decoded control input but has a negligible effect on the

F1-Score= 0.46		
	Healthy	Cyber-attack
Healthy	1475	1039
Cyber-attack	25	461

Table 4.1: Confusion matrix and F1-Score for cyber-attack with an impact of less than 0.1 degrees on the actuator and less than 1 degree on measurement.

F1-Score=0.76		
	Healthy	Cyber-attack
Healthy	1472	553
Cyber-attack	28	947

Table 4.2: Confusion matrix and F1-Score for cyber-attack with an impact of between 0.1 to 0.2 degrees on the actuator and between 1 to 2 degrees on measurement.

F1-Score=0.99		
	Healthy	Cyber-attack
Healthy	1476	7
Cyber-attack	24	1493

Table 4.3: Confusion matrix and F1-Score for cyber-attack with an impact of greater than 0.2 degrees on the actuator and greater than 1 degree on measurement.

decoded output, the cyber attack will be detected only based on the system dynamics similar to an FDI cyber-attack. This will result in a small delay in detection. In our simulator, the actuation delay is $40ms$, so any changes in control commands will be applied to the UAV after a $40ms$ delay. For our small UAV simulator, without coding, the Kalman filter could detect step changes after $100ms$.

4.5 Two-way Coding with Random Number Generator as a Defense Against Cyber-attack

In previous sections, we showed how coding could defend against covert cyber-attacks. However, coding schemes by itself are not effective against replay cyber-attacks. For an attacker to launch a successful replay cyber-attack, the system should operate in steady-state conditions where control inputs and plant outputs are not changing. The two-way coding maps any input-output pair with a constant matrix to new values. Since inputs and outputs in steady-state conditions are almost unchanged, replaying previously coded signals with the same coding matrix will be undetectable.

Physical watermarking is one of the most used methods to detect replay cyber-attacks, where the defender injects some arbitrary dynamics to control input in the form of $\tilde{u}(k) =$

$u(k)+\Delta u(k)$ [71]. However, these methods degrade the control system performance [114] and are useless for covert cyber-attack detection. Another problem with coding is that an attacker with knowledge of the system dynamics may discover the coding matrix by eavesdropping.

In this section, we introduce a two-way coding scheme that can detect both covert and replay cyber-attacks. This scheme has the added benefit of coding all the measurements with a Gaussian random number, which makes it impossible for attackers to discover the coding matrix by eavesdropping. However, additional resources are required to generate the random numbers and transmit to the plant.

We assume that the controller-side coding uses a random number generator to generate sequences of a random number, which changes at every step. The controller-side coding scheme treats the randomly generated numbers as a random input signal. It codes it with one of the outputs, and the plant-side coding scheme treats this random signal as an input and uses it to code all the outputs with this randomly generated signal.

Figure 4.17 shows the schematic of a system with one input and two outputs with blue-shaded areas representing coding schemes. This scheme is extendable for any number of inputs and outputs. Also worth noting is that we need one extra coded input to transmit the random signal to the plant side. Hence, for instance, the coding scheme for the one-input system in Figure 4.17 transmits two inputs, one for the controller input $u_1(k)$ and the other one for the randomly generated signal $r_c(k)$. In the figure, r_c is a random signal generated by a Gaussian random number generator on the controller side coding scheme, where we coded it with an output y_1. The coded input of random signal r_c is shown by q_c on the controller side and by q'_c on the plant side, and the random signal on the plant side is presented by r'_c. We denoted the output used to code the random signal with subscript 1. This output can be any of the plant's outputs. We denoted all the other outputs with subscript j and inputs with subscript i.

In this thesis, a Kalman filter-based detector is implemented on the controller side for cyber-attack detection, as shown in Figure 4.17, that monitors changes in the plant measurements. Therefore, control input coding is not necessary for the detection of cyber-attacks. But for defense against zero dynamics, it is required to code both inputs and outputs. In addition, input coding can improve cyber-attack detection since any attack signal will be shown up in the next timestep, as shown by equation 4.51. If the attackers inject their cyber-attack signals into the uncoded inputs only, the cyber-attack problem will be the same as a simple FDI attack. Therefore, in such a scenario coding will be useless. Hence one should always code both inputs and outputs.

From Figure 4.17, the first pair of coded input-output, that is $q_c(k)$ and $v'_1(k)$ are:

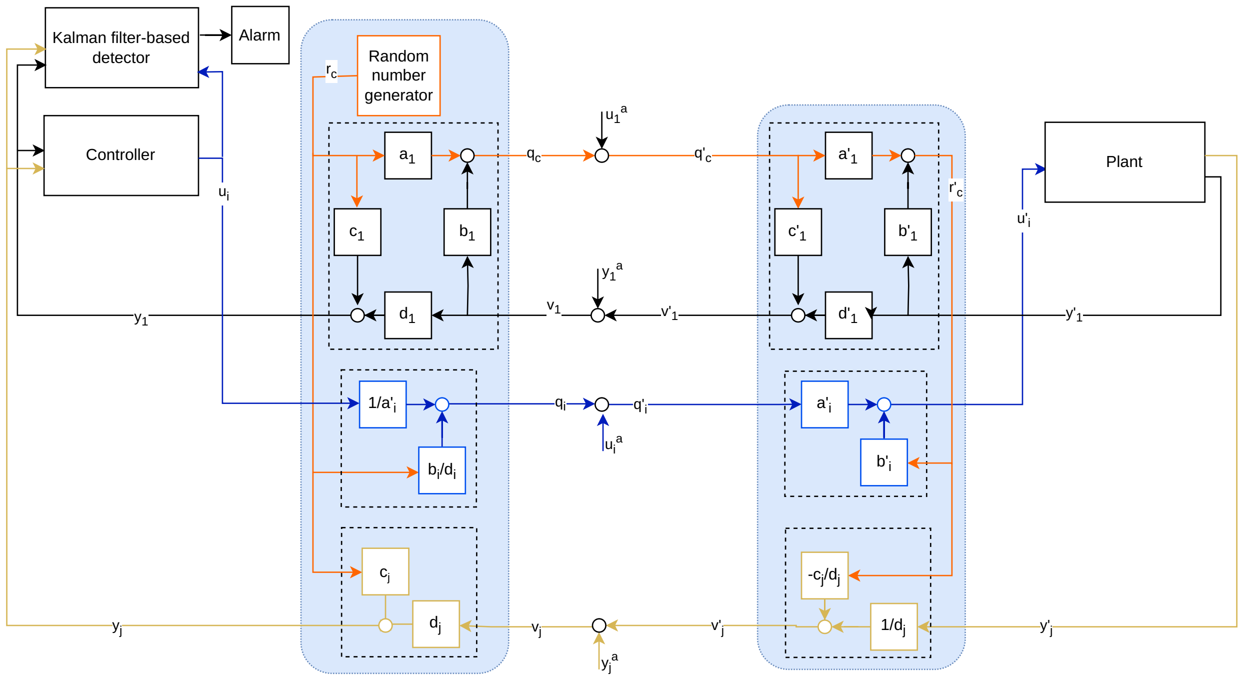


Figure 4.17: The schematic presents a system with one input and two outputs, coded with a random number r_c on the controller side coding scheme. This scheme is transparent to the plant and controller. There is no limit to the number of inputs u_j and outputs y_j coded with this scheme.

$$q_c(k) = a_1 r_c(k) + b_1 v_1(k) \tag{4.57}$$

$$v'_1(k) = c'_1 q'_c(k) + d'_1 y'_1(k) \tag{4.58}$$

we can see that the coded input-output pair of $r_c(k)$ and $y'_1(k)$ is the same as the two-way coding of input-output pair of $u_1(k)$ and $y'_1(k)$, shown in Figure 4.7 and described with equations (4.2) and (4.6), except that $u_1(k)$ is replaced with $r_c(k)$. Therefor, if we follow the same proof we will get the coding equation for the current coding scheme when there is no cyber-attacks as follows:

$$v'_1(k) = -\frac{c_1}{d_1} r_c(k) + \frac{1}{d_1} y'_1(k) \tag{4.59}$$

$$q_c(k) = \frac{1}{a'_1} r_c(k) + \frac{b_1}{d_1} y_1(k) \tag{4.60}$$

All coded inputs are similar to $q_2(k)$ of Figure 4.8 described by equation (4.19), But instead of $y_1(k)$, they are coded with $r_c(k)$, hence we can write the coding equation as follows:

$$q_i(k) = \frac{1}{a'_i} u_i(k) + \frac{b_i}{d_i} r_c(k) \tag{4.61}$$

All coded outputs other than $y'_1(k)$ are similar to $v'_2(k)$ of Figure 4.7 described by equation (4.17), but they are coded with $r'_c(k)$ instead of $u'_1(k)$, hence we have:

$$v'_i(k) = -\frac{c_j}{d_j} r'_c(k) + \frac{1}{d_j} y'_j(k) \tag{4.62}$$

Equations (4.59) to (4.62), which describe coded inputs and outputs, are random signals constructed by the sum of a control input or a plat output with a randomly generated signal. Since the linear transformation of a Gaussian random number results in a new Gaussian random number [118], the coded inputs and outputs with our proposed scheme are also random numbers. Hence, with a proper selection of coding matrix, the coded inputs $q_i(k)$ and coded outputs $v'_i(k)$ will completely hide the system dynamics. Therefore, it would be impossible for attackers to discover the coding matrix by eavesdropping.

Control input coding is not necessary for covert and replay cyber-attack detection, and it can be considered the worst-case scenario for a defender, with the minimum resource consumption. It also makes it possible to launch a meaningful cyber-attack, which makes simulation results more tangible. Setting the coding matrix of control inputs to identity will remove the coding of inputs, and the equation (4.28) to (4.34) will become:

$$y_1(k) = y'_1(k) - \frac{c_1}{a_1} u_1^a(k) + \frac{1}{d'_1} y_1^a(k) \tag{4.63}$$

$$r'_c(k) = r_c(k) + \frac{1}{a_1} u_1^a(k) + \frac{b_1}{a_1} y_1^a(k) \tag{4.64}$$

$$y_j(k) = y'_j(k) - \frac{c_j}{a_1} u_1^a(k) - \frac{c_j b_1}{a_1} y_1^a(k) + d_j y_j^a(k) \tag{4.65}$$

$$u'_i(k) = u_i(k) + u_i^a(k) \tag{4.66}$$

And

$$\Sigma_u(k) = u^a(k) \tag{4.67}$$

which results in $C_u = I$ and $C_y = 0$ and:

$$\Sigma_y(k) = u_1^a(k)\begin{bmatrix}\frac{-c_1}{a_1}\\ \frac{-c_2}{a_1}\\ \vdots\\ \frac{-c_j}{a_1}\end{bmatrix} + \begin{bmatrix}\frac{1}{d'} & 0 & \dots & \dots & 0\\ \frac{-c_2b_1}{a_1} & d_2 & 0 & \dots & 0\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ \frac{-c_pb_1}{a_1} & 0 & \dots & 0 & d_p\end{bmatrix}\begin{bmatrix}y_1^a(k)\\ y_2^a(k)\\ \vdots\\ y_p^a(k)\end{bmatrix} \tag{4.68}$$

where p is then number of outputs. Therefore, equation (4.53) becomes:

$$y^a(k+1) = D_y^{-1}D_u u^a(k+1) - D_y^{-1}CAe^a(k) - D_y^{-1}CBu^a(k) \tag{4.69}$$

This shows that the attackers still need to know the coding matrix in order to remove cyber-attack signals $u^a(k+1)$ impact from measurements. And since it is assumed that attackers do not know the coding matrix, it would be impossible for them to remove the effects of the cyber-attacks from measurements.

4.5.1 Simulation Results of Cyber-attack Detection on a System Equipped with Two-way Coding Scheme with Random Number Generator

In the two-way with a random number generator, we replaced the control input with a random signal. Therefore, all of the analyses in the previous sections are the same. From equation (4.28) to (4.34), we can see that only cyber-attack signals are affecting the decoded signals and the other pair has no effect. Therefore, the detection rate of the cyber-attack in the Monte Carlo simulation for two-way coding with the random signal is identical to the coding with control input. Hence, in this simulation, our objective is to show how this coding scheme can convert all signals to random signals. We used the same UAV system and settings presented in Subsection 4.4.2, with the same coding matrix to simulate a cyber-attack. However, we used the coding scheme with random number generator shown in Figure 4.17. The coding matrix we used is as follows:

$$M = \begin{bmatrix} 10 & 0.5 & 0 & 0 & \dots & 0 & 0 \\ 20 & 0.5 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 0 & 1 \end{bmatrix}$$

$$M_r = \begin{bmatrix} -0.1 & 0.1 & 0 & 0 & \dots & 0 & 0 \\ 4 & -0.1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & -0.3755 & 0.125 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & -0.3755 & 0.125 \end{bmatrix} \tag{4.70}$$

where M is the coding matrix on the controller side, and M_r is the reverse coding matrix on the plant side. The latitudinal and longitudinal Kalman filters we used for cyber-attack detection have four inputs and ten outputs explained in Section 3.2. We also need to add a row and a column for the randomly generated signal to the coding matrices. Therefore, we need a 15×15 coding matrix to code all the signals. In our simulation, we only coded latitudinal Kalman filter signals and the randomly generated signal, so we need 8×8 matrices for M and M_r. For example, we will demonstrate how to code an arbitrary signal y, that changes linearly from 0 to 10 degrees or 0 to 0.17 radians, using the first two rows of coding matrix

M, and a random signal. With the coding constants we have $a = 10, b = 0.5, c = 20, d = 0.5$, and equation (4.60) became $q_c = -10r_c + y$. We listed some samples of the coded input q_c, calculated with this formula in Table 4.4 and plotted the coded output for 100 signals in Figure 4.18.

y (rad)	0.00	0.04	0.05	0.07	0.11	0.12	0.14	0.16	0.17
r_c	0.11	0.44	0.45	0.10	0.60	0.57	0.15	0.84	0.19
q_c	-1.13	-4.39	-4.48	-0.91	-5.91	-5.63	-1.37	-8.19	-1.73

Table 4.4: Some examples of coding of an output $y(rad)$ with a random signal

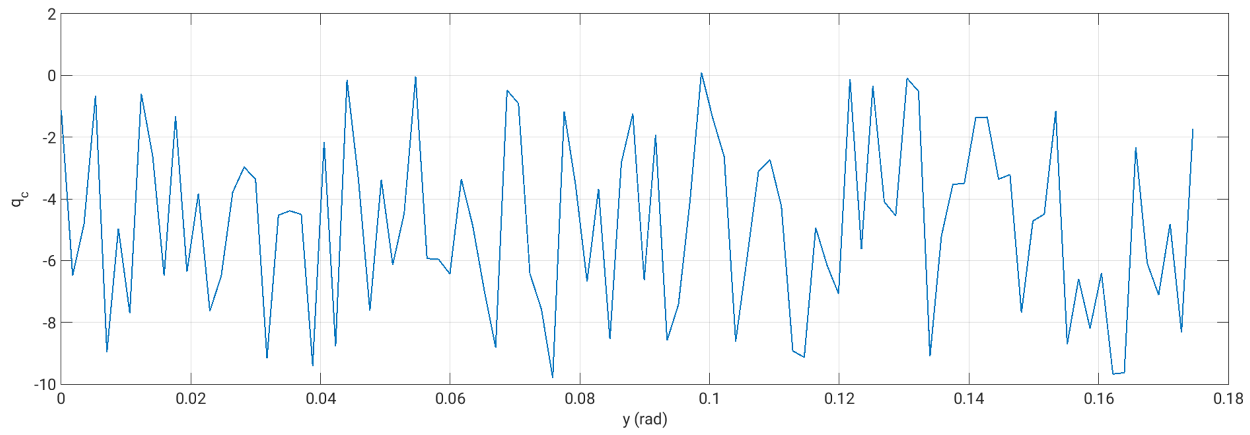


Figure 4.18: Sample plot of coding a output signal linearly changing from zero to ten degrees coded with sequence of randomly generated signal.

For simulation, we used ϕ from the plant to code with the randomly generated signal. We did not code the inputs to simulate the worst-case scenario. Therefore, attackers can inject their cyber-attack signals into inputs. However, we did code all the outputs. The cyber-attack signal on input can be seen in Figure 4.12. Since the outputs are coded, the attackers cannot launch a covert cyber-attack. We assume that the attackers inject the signal shown in Figure 4.12 to probe the system for vulnerabilities. We injected the cyber-attack signal on δ_a and another signal on ϕ to evaluate the system response. We used a random number generator in Simulink to generate Gaussian random number in the range of ± 0.01 and used it with our coding scheme. The random numbers generated in Matlab/Simulink are pseudorandom numbers. The under cyber-attack input and output of the system are shown in Figure 4.19. Since control input is not coded, coded and decoded input in the figure are identical. The under cyber-attack coded signal, as explained, is converted to a random number that is impossible to decode.

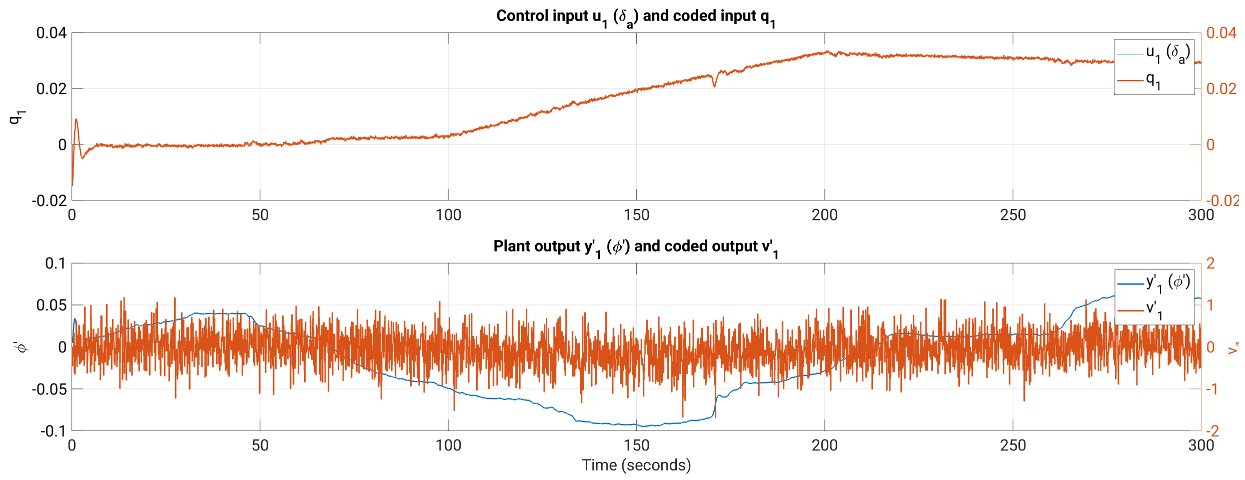


Figure 4.19: Cyber-attack on input u_1 and coded output v_1, where there is no coding on input, but output is coded using random number generated in controller side.

The under cyber-attack coded and decoded input and output on the controller side and plant side are shown in Figure 4.20

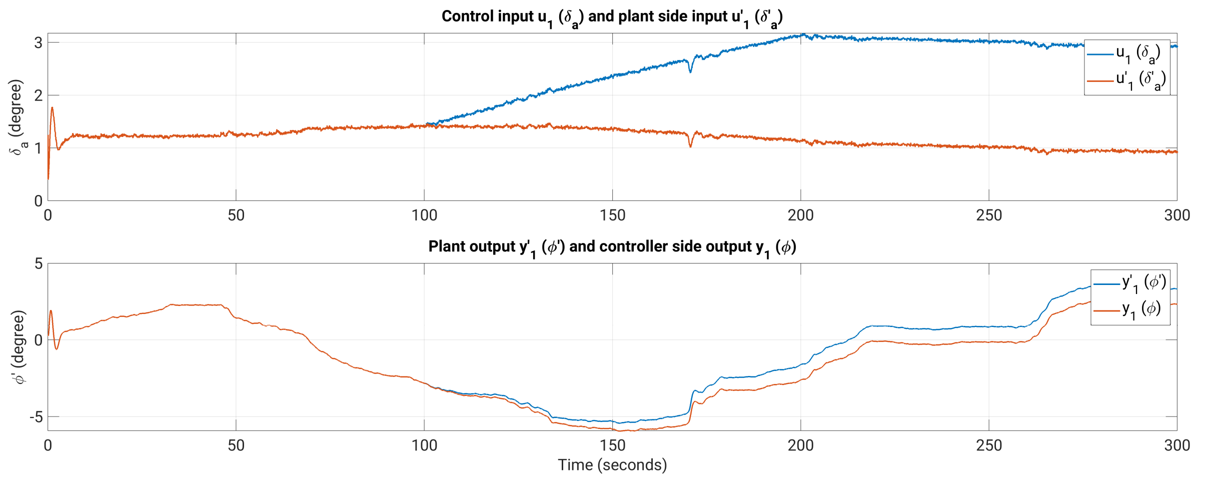


Figure 4.20: Impact of cyber-attack on input (u_1) and output on the controller side (ϕ) and on the plant side (ϕ').

All other coded outputs on the plant side are also converted to random signals, as shown in Figure 4.21. Since ϕ is under cyber-attack, the random number $(r_c(k))$ coded with it, also is impacted by the cyber-attack signal. Hence, the random numbers $(r_c'(k))$ on the plant side are different from the random numbers $(r_c(k))$ on the controller side. Therefore, all the outputs decoded on the controller side have different values from the signals on the plant side as shown in Figure 4.21.

This will result in even more radical changes in Kalman filter residuals and hence the χ^2 detector will be triggered faster as shown in Figure 4.22.

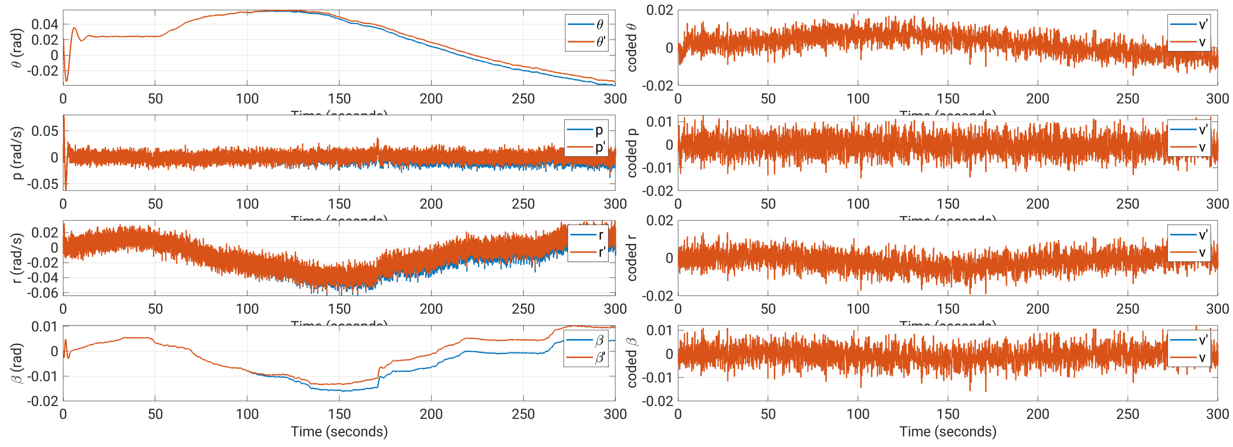


Figure 4.21: Plant side outputs and coded plant side outputs with random numbers.

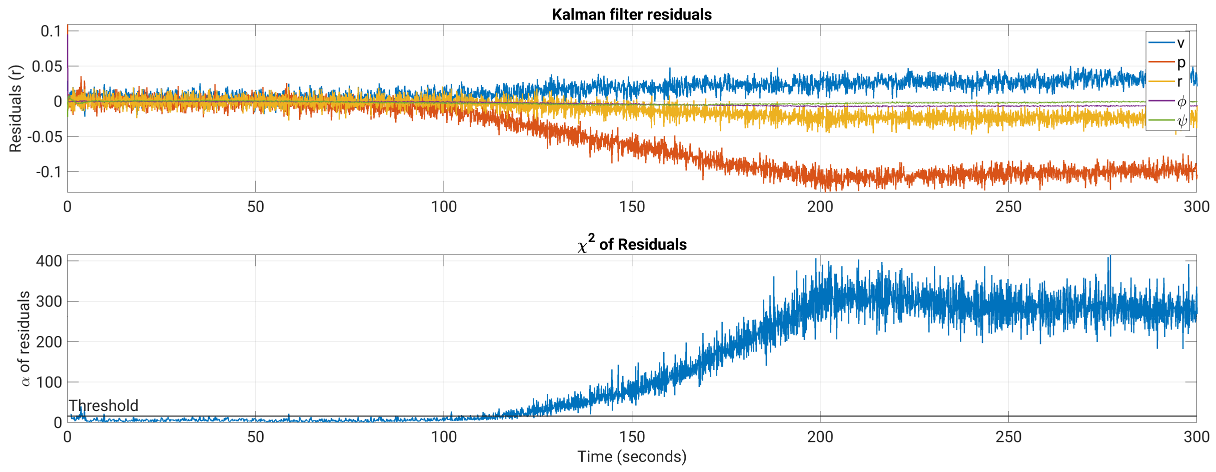


Figure 4.22: Kalman filter residuals and χ^2 detector for system under cyber-attack with random number coding.

It should be noted that the deviation of decoded outputs on the controller side from outputs on the plant side is only a function of the coding matrix and cyber-attack signal magnitude. The random numbers used for coding do not affect residuals. However, coding with random numbers turns all coded outputs into a time-varying signal, which hides plant actual dynamics.

Compared with the method proposed in [77], which requires recalculation of a coding matrix in any few steps on both sides synchronously, this method only uses a single coding matrix and a random number generator on the controller side, and no extra calculation is needed. The detection of a cyber-attack depends on the coding matrix and magnitude of the cyber-attack signal on both coded input and output. Because with two-way coding, the cyber-attack signal on input impacts the output signal and vice versa. The following results are produced based on the coding matrix and attack signal we explained in this section. However, the signals used for coding do not impact the decoded signals, meaning that the random number used only changes the coded inputs and outputs and has no effect on the decoded signal. We used the same coding matrix and attack signal in the previous subsection in this simulation. Since the coding with randomly generated signals does not change the detection rate of the coding schemes, the results of the Monte Carlo simulation of previous sections are also valid for this scheme. However, for completeness of the discussion we present the results of the current simulation in table 4.5.

F1-Score=0.99		
	Healthy	Cyber-attack
Healthy	998	14
Cyber-attack	12	1546

Table 4.5: Confusion matrix and F1-Score for cyber-attack with an impact of greater than 0.2 degrees on the actuator and greater than 1 degree on measurement.

Our proposed method can detect attacks with an F1-score of 0.99 for attack signal magnetite of greater than 0.2 degrees on actuators and greater than 1 degree on measurement. And we could effectively convert all signals to random signals to hide the system dynamics.

4.5.2 Coding for Replay Cyber-attack Detection

The random number coding scheme, developed in this section, can also be used for replay cyber-attack detection. Compared with physical watermarking, an advantage of this method is that it does not degrade the controller's performance. This is an advantage compared to coding methods proposed by [79] that cannot detect replay cyber-attacks, and the watermarking method such as [116] that degrade controller's performance.

Replay cyber-attack on coded outputs can be defined based on equation (2.24) by replacing plant outputs with coded outputs as follows:

$$v^a(k) = v'(k1) - v'(k) \tag{4.71}$$

where $k1$ refers to a time stamp where the attacker starts recording the coded outputs. This signal will be added to current coded output $v'(k)$ and hence:

$$\begin{aligned} v(k) &= v'(k) + v'(k1) - v'(k) \\ &= v'(k1) \end{aligned} \tag{4.72}$$

where $v(k) = v'(k1)$ refers to coded outputs that the attacker has already recorded in time stamp $k1$, in the absence of an cyber-attack that is $u^a(k) = 0$ and $y^a(k) = 0$. Then from equation (4.59), we have:

$$v'(k1) = \frac{-c}{d} r_c(k1) + \frac{1}{d} y'(k1) \tag{4.73}$$

We assume the attacker replays recorded outputs from time stamp $k1$ at time stamp k and simultaneously injects cyber-attack signals $u^a(k)$ on input. The decoding equation based on the coding definition and Figure 4.17 on the controller side is as follows:

$$y_1(k) = d_1 v_1(k) + c_1 r_c(k) \tag{4.74}$$

However, during the replay cyber-attack the coded signals at time stamp k are replaced with coded signals on the plant side at time stamp $k1$. Therefore, from equation (4.72), the coding scheme on the controller side will decode $v'(k1)$ instead of $v(k)$, but will use $r_c(k)$ as follows:

$$\begin{aligned} y_1(k) &= d_1 v_1'(k1) + c_1 r_c(k) \\ &= d_1 \left[\frac{-c_1}{d_1} r_c(k1) + \frac{1}{d_1} y_1'(k1) \right] + c_1 r_c(k) \\ &= y_1'(k1) + c_1 [r_c(k) - r_c(k1)] \\ &= y_1'(k1) + c_1 \Delta r_c(k) \end{aligned} \tag{4.75}$$

where Δr_c is differences between current random number on the controller side $r_c(k)$ and replayed random number at time stamp $k1$, that is $r_c(k1)$. All other replayed coded outputs $v_i(k1)$, that are constructed by coding of outputs $y_i(k1)$ with the random number on the plant side $r_c'(k1)$, will be decoded on the controller side as shown below:

$$\begin{aligned}
v_j(k) &= v'_j(k1) = \frac{-c_j}{d_j} r'_c(k1) + \frac{1}{d_j} y'_j(k1) \\
y_j(k) &= d_j v_j(k1) + c_j r_c(k) \\
&= d_j \left[\frac{-c_j}{d_j} r'_c(k1) + \frac{1}{d_j} y'_1(k1) \right] + c_j r_c(k) \\
&= y'_j(k1) + c_j [r_c(k) - r_c(k1)] \\
&= y'_j(k1) + c_j \Delta r_c(k) \qquad (4.76)
\end{aligned}$$

Under replay cyber-attack, the plant will also receive a random number different from the one generated on the controller side. To calculate differences first, we need to calculate coded input $q_c(k)$ on the controller side based on received replayed coded output $v_1(k1)$ from the plant side. From Figure 4.17 the coded input $q_c(k)$ will become:

$$\begin{aligned}
v_1(k) &= v'_1(k1) = c'_1 q'_c(k1) + d'_1 y'_1(k1) \\
q_c(k) &= a_1 r_c(k) + b_1 v_1(k) \\
&= a_1 r_c(k) + b_1 v'_1(k1) \\
&= a_1 r_c(k) + b_1 c'_1 q'_c(k1) + b_1 d'_1 y'_1(k1) \qquad (4.77)
\end{aligned}$$

From equation (4.60), considering that at the time of recording coded output at time stamp $k1$, there is no cyber-attack we have:

$$q'_c(k1) = q_c(k1) = \frac{1}{a'_1} r_c(k1) + \frac{b_1}{d_1} y'_1(k1) \qquad (4.78)$$

Replacing in equation (4.77) we have:

$$
\begin{aligned}
q_c(k) &= a_1 r_c(k) + b_1 d_1' y_1'(k1) + b_1 c_1' \left[\frac{1}{a_1'} r_c(k1) + \frac{b_1}{d_1} y_1'(k1) \right] \\
&= a_1 r_c(k) + b_1 d_1' y_1'(k1) + \frac{b_1 c_1'}{a_1'} r_c(k1) + \frac{b_1^2 c_1'}{d_1} y_1'(k1) \\
&= a_1 r_c(k) + \frac{b_1 c_1'}{a_1'} r_c(k1) + \frac{d_1 b_1 d_1' + b_1^2 c_1'}{d_1} y_1'(k1) \\
&= a_1 r_c(k) + \frac{b_1 c_1'}{a_1'} r_c(k1) + \frac{d_1 b_1 a_1 - b_1^2 c_1}{d_1(a_1 d_1 - b_1 c_1)} y_1'(k1) \\
&= a_1 r_c(k) + \frac{b_1 c_1'}{a_1'} r_c(k1) + \frac{b_1}{d_1} y_1'(k1) \qquad (4.79)
\end{aligned}
$$

Assuming there are cyber-attack on code input, that is $q_1'(k) = q_1(k) + u_1^a(k)$, the decoded random number on the plant side $r_c'(k)$ will be:

$$
\begin{aligned}
r_c'(k) &= a_1' q_c'(k) + b_1' y_1'(k) \\
&= a_1' \left[a_1 r_c(k) + \frac{b_1 c_1'}{a_1'} r_c(k1) + \frac{b_1}{d_1} y_1'(k1) + u_1^a(k) \right] + b_1' y_1'(k) \\
&= a_1' a_1 r_c(k) + b_1 c_1' r_c(k1) + \frac{a_1' b_1}{d_1} y_1'(k1) + b_1' y_1'(k) + a_1' u_1^a(k) \\
&= \frac{a_1 d_1}{a_1 d_1 - b_1 c_1} r_c(k) - \frac{b_1 c_1}{a_1 d_1 - b_1 c_1} r_c(k1) + \frac{b_1 d_1}{d_1(a_1 d_1 - b_1 c_1)} y_1'(k1) + b_1' y_1'(k) + a_1' u_1^a(k) \\
&= \frac{a_1 d_1 - b_1 c_1}{a_1 d_1 - b_1 c_1} r_c(k) - \frac{b_1 c_1}{a_1 d_1 - b_1 c_1} \Delta r_c(k) - b_1' y_1'(k1) + b_1' y_1'(k) + a_1' u_1^a(k) \\
&= r_c(k) + b_1' c_1 \Delta r_c(k) + b_1' [y_1'(k1) - y_1'(k)] + a_1' u_1^a(k) \qquad (4.80)
\end{aligned}
$$

Hence, coding and decoding control inputs on the plant side from Figure 4.17 can be written as:

$$\begin{aligned}
q_i(k) &= \frac{1}{a'_i}u_i(k) + \frac{b_i}{d_i}r_c(k) \\
u'_i(k) &= a'_i q'_i(k) + b'_i r'(k) \\
&= a'_i\left[\frac{1}{a'_i}u_i(k) + \frac{b_i}{d_i}r_c(k) + u_i^a(k)\right] + b'_i r'(k) \\
&= u_i(k) + b'_i[-r_c(k) + r'_c(k)] + a'_i u_i^a(k) \\
&= u_i(k) + b'_i\{-r_c(k) + r_c(k) + b'_1 c_1 \Delta r_c(k) + b'_1[y'_1(k1) - y'_1(k)] + a'_1 u_1^a(k)\} + a'_i u_i^a(k) \\
&= u_i(k) + b'_i b'_1\{c_1 \Delta r_c(k) + [y'_1(k1) - y'_1(k)]\} + b'_i a'_1 u_1^a(k) + a'_i u_i^a(k) \qquad (4.81)
\end{aligned}$$

Assuming steady state condition where $y_1(k1) \simeq y'_1(k)$, decoded control input $u'_i(k)$ in plant side became:

$$u'_i(k) = u_i(k) + b'_i b'_1 c_1 \Delta r_c(k) + b'_i a'_1 u_1^a(k) + a'_i u_i^a(k) \qquad (4.82)$$

Since we assumed that the attacker replays the outputs, logically, they will not inject cyber-attack signals on the random number itself, and hence, $u_1^a(k) = 0$. Therefore, decoded random number form equation (4.80) at steady state condition became:

$$r'_c(k) = r_c(k) + b'_1 c_1 \Delta r_c(k) \qquad (4.83)$$

and if control input is not coded then we have:

$$u'_i(k) = u_i(k) + u_i^a(k) \qquad (4.84)$$

To summarize, for the system of Figure 4.17, under replay cyber-attack, with coding applied only to outputs, the decoded outputs on the controller side will be:

$$\begin{aligned}
r'_c(k) &= r_c(k) + b'_1 c_1 \Delta r_c(k) && (4.83) \\
u'_i(k) &= u_i(k) + u_i^a(k) && (4.84) \\
y_1(k) &= y'_1(k1) + c_1 \Delta r_c(k) && (4.75) \\
y_j(k) &= y'_j(k1) + c_j \Delta r_c(k) && (4.76)
\end{aligned}$$

From equation (4.83), we can see that, even with no cyber-attack on the random number $r_c(k)$, a replay cyber-attack will change the random number on the plant side $r'_c(k)$. To launch a replay cyber-attack, the attacker sends previously good-looking recorded outputs. Therefore, the impact of cyber-attack signals will not reflected in the outputs. However, since the random number on the plant side is different from the controller side, from equations (4.75) and (4.76), we can see that decoded plant outputs on the controller side are different from actual outputs on the plant side by a fluctuating random value of $c_j \Delta r_c(k)$. Hence the equations (4.67) to (4.69) became:

$$\Sigma_u(k) = u^a(k) \tag{4.85}$$

and

$$\Sigma_y(k) = \begin{bmatrix} c_1 \Delta r_c(k) \\ c_2 \Delta r_c(k) \\ \vdots \\ c_p \Delta r_c(k) \end{bmatrix} \tag{4.86}$$

The equation (4.69) will not change, but the parameters are different:

$$y^a(k+1) = D_y^{-1} D_u u^a(k+1) - D_y^{-1} C A e^a(k) - D_y^{-1} C B u^a(k) \tag{4.69}$$

This proves that replaying previously coded outputs is not enough to hide the cyber-attack impact when random number coding is implemented, and the attackers are required to inject cyber-attack signals $y^a(k+1)$ to remove the cyber-attack impact on outputs. However, since we assumed the attackers do not know the coding matrix and the random numbers change on any time stamp, they cannot remove cyber-attack impacts.

4.5.3 Simulation Results of the Detection of Replay Cyber-attack

In this subsection, we simulated a replay cyber-attack on the system presented in Figure 4.17 and with the same coding scheme and UAV simulation designed in Subsection 4.5.1. The replay cyber-attack starts at time stamp 150, where previously recorded outputs from 100 seconds earlier were replayed. Meanwhile, the attacker injects a constant cyber-attack signal close to zero so that the UAV will continue to fly in a slightly different direction.

The top plot of Figure 4.23 shows the controller input signal. Since we did not code the inputs, both inputs ($u(k)$) and coded inputs ($q(k)$) are identical. Therefore, the blue line, representing the input on the controller side, is covered by the red line, representing the coded input. Moreover, since the attackers replace control inputs with their signals, the plant does not receive these control inputs, but the controller side detector uses them to estimate the state of the plant. In the bottom plot of Figure 4.23, the red line is the replayed coded output ($v'(k)$), and the blue line is the output of the plant on the plant side ($y'(k)$). We can see that while the replayed output does not show any meaningful changes, the plant side output starts deviating from the steady state condition by the start of the replay cyber-attack at time stamp 150 due to a cyber-attack signal injected on the control input.

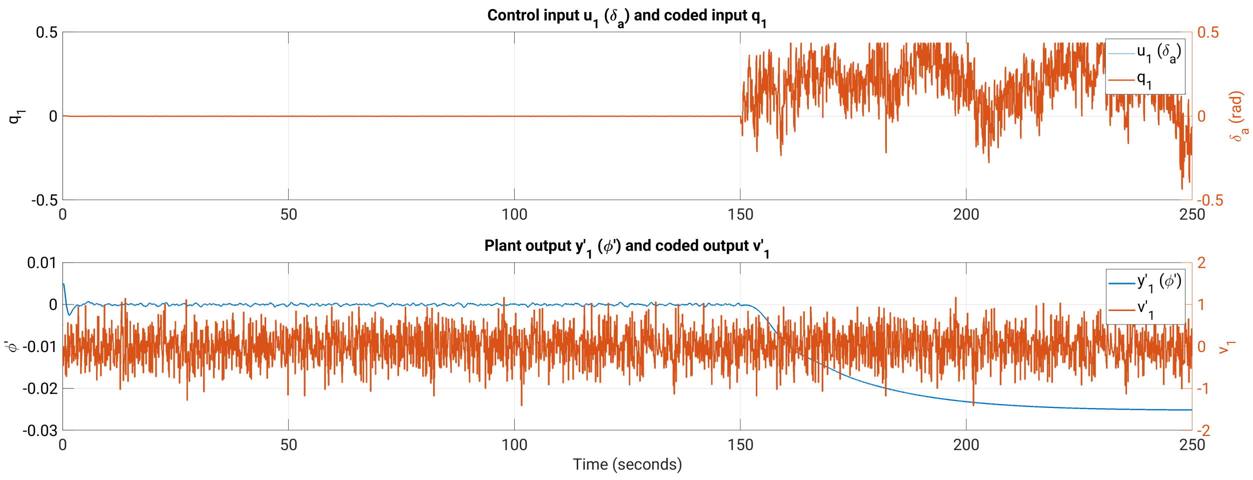


Figure 4.23: Coded and decoded controller side input and plant side output before and after replay cyber-attack.

The top plot of Figure 4.24 shows control input on the controller side $u_1(k)$ by the blue line and manipulated input injected by the attackers on the plant side $u_1'(k)$ by the red line. Since we did not code the inputs, the controller side and plant side inputs were identical before the start of the replay cyber-attack. However, after the replay cyber-attack at time stamp 150, the attackers inject a constant signal without any noise for the input $u_1'(k)$, shown by the red line on the top plot. After the cyber-attack, the control input $u_1(k)$ is affected by the received replayed signals, resulting in a fluctuating control input shown by the blue line on the top plot of Figure 4.24.

The plant output on the plant side $y_1'(k)$ is the blue line, and decoded replayed plant output $y_1(k1)$ on the controller side is the red line in the bottom plot in Figure 4.24.

We can see that while the attackers' replayed outputs do not show any visible changes, since the current random numbers are different from replayed ones, decoding outputs on the controller side show fluctuating outputs ($y(k)$) as shown in the bottom plot of Figure 4.24 in

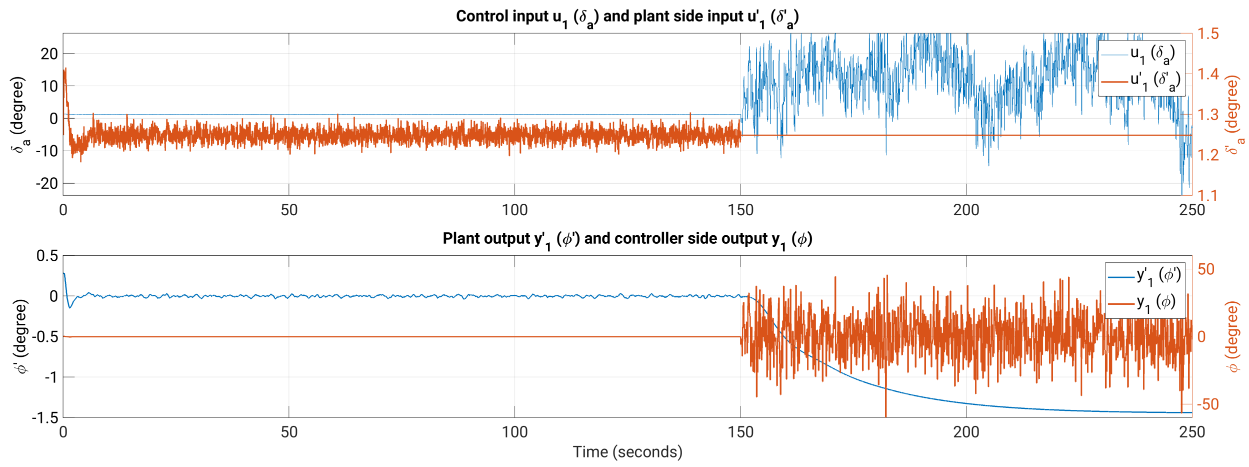


Figure 4.24: Decoded input and output before and after replay cyber-attack.

the red line.

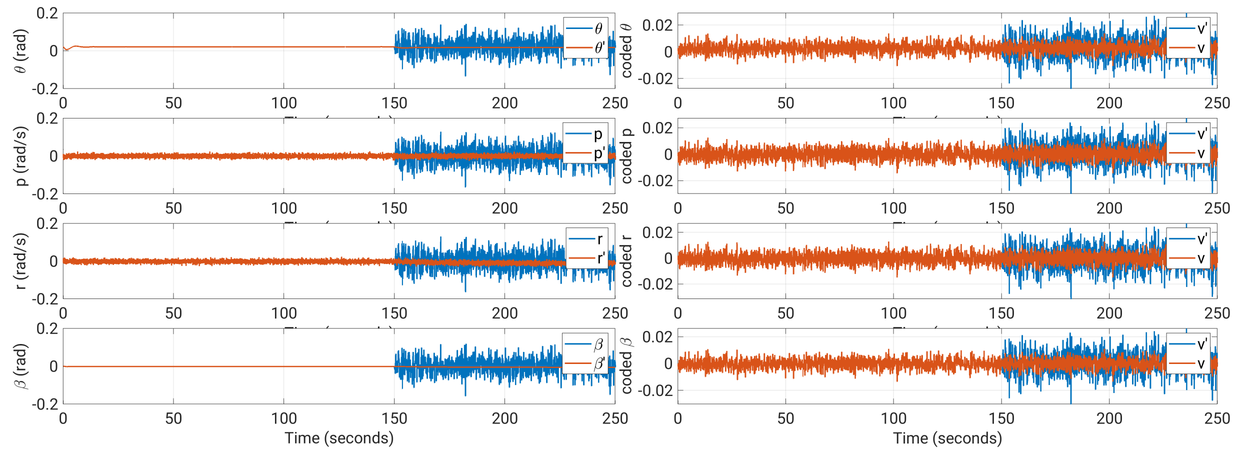


Figure 4.25: Coded and decoded inputs and outputs before and after replay cyber-attack.

The plots on the right side of Figure 4.25 are coded outputs on the plant side in the blue line and coded outputs received by the controller in the red line. Since the system is under replay cyber-attack, the controller only receives replayed output shown with the red lines.

The plots on the left side of Figure 4.25 are the UAV measurements on the plant side in the red line and decoded measurements on the controller side in the blue line. During the replay cyber-attack, these outputs on the plant side are invisible to the controller, and instead, the replayed outputs are received and decoded on the controller side. However, on the controller side, decoded replayed outputs will fluctuate due to the difference between random numbers at the current time stamp and replayed random number shown in equation (4.75), and displayed by the blue line in the left plots of Figure 4.25.

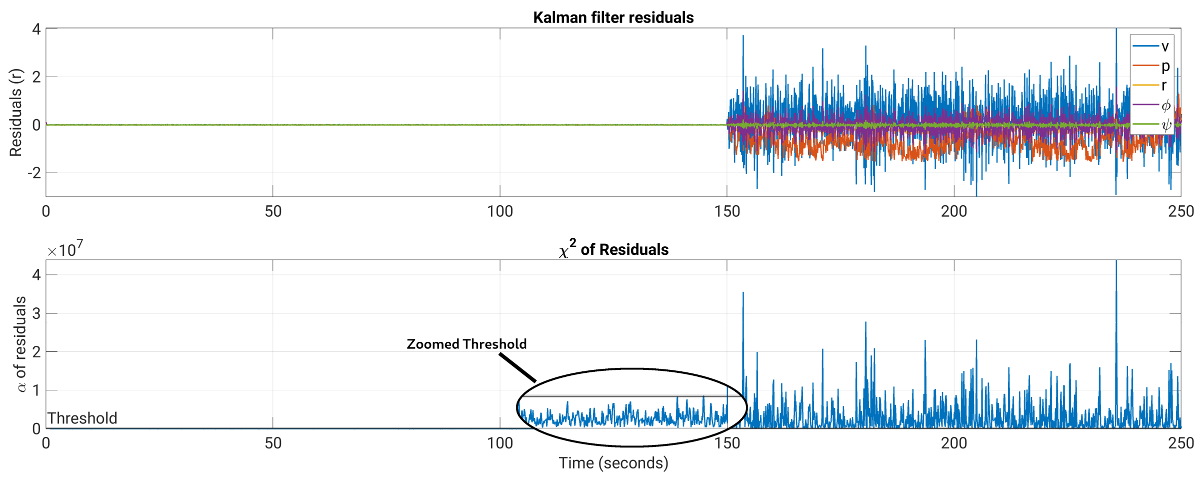


Figure 4.26: Kalman filter residuals and χ^2 detector for replay cyber-attack detection.

By proper selection of coding scheme, as explained in Subsection 4.4.2, and random numbers magnitude, we can magnify the fluctuating residuals generated by the Kalman filter, which leads to fast detection of cyber-attack by a χ^2 detector as shown in Figure 4.26. The top plot of the figure shows the residuals generated by the Kalman filter. These residuals start growing after the replay cyber-attack. The bottom plot shows χ^2 of the residuals after start of the reply cyber-attack exceeds the threshold. We repeated this test with randomly selected control inputs and with different randomly selected replay time stamp.

In replay cyber-attacks, the attacker replays previously recorded signals coded with a different random number used for decoding. Since we choose a relatively large random number to hide the dynamics of inputs and outputs, decoding with different random numbers results in a large difference in the decoded signals. This, in turn, results in a large fluctuation of signals and immediate detection of replay cyber-attacks. Another important factor for detecting replay cyber-attacks is testing The following results are produced based on the coding matrix and attack signal we explained in this section.

In replay cyber-attacks, the attacker replays previously recorded signals coded with a different random number used for decoding. Since we choose a relatively large random number to hide the dynamics of inputs and outputs, decoding with different random numbers results in a large difference in the decoded signals. This, in turn, results in a large fluctuation of signals and immediate detection of replay cyber-attacks. The following results are produced based on the coding matrix and attack signal we explained in this section.

Our proposed coding scheme could detect the replay cyber-attack instantaneously and with a F1-score of 1 for perfect detection of the attack.

F1-score	
Healthy	Replay Cyber-attack
0.98	1

Table 4.6: Summary of results of replay cyber-attacks detection

4.6 Detection of Cyber-attack in The Presence of Faults

The detectors we developed in the previous sections are based on the Kalman filter and χ^2, which can only detect deviations from the normal operating condition and cannot make any distinction between faults and cyber-attacks. Therefore, faults can raise false alarms for a cyber-attack. We presented the Kalman filter base fault detection in Section 2.5 and the UAV fault models in Section 2.3.1. We proposed a solution based on two similar detectors, one on the plant side and the other on the controller side, for isolation of faults from cyber-attacks in the Chapter 3. We will use the same concept and will use the Kalman filter based fault detection method explained in Section 2.5 to proof that this idea also works with our coding scheme. Figure 4.27 shows the schematic of the proposed system that can isolate faults from cyber-attack.

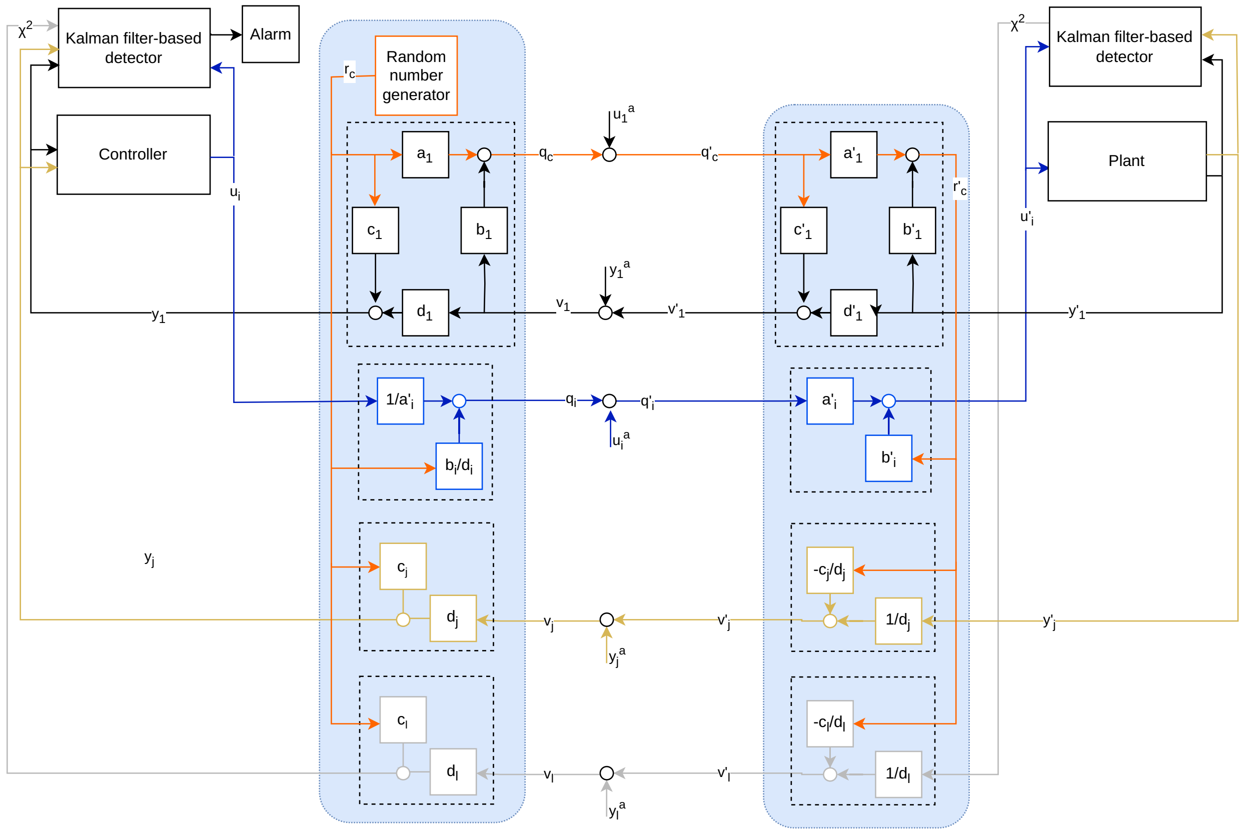


Figure 4.27: System configuration for detection of simultaneous fault and cyber-attack.

As shown in Figure 4.27, a Kalman filter with χ^2 detector which is a replica of the detector on the controller side is also implemented on the plant side. These two detectors will generate exactly the same χ^2 for faulty systems when there is no cyber-attack. However, for a system under cyber-attack, the control inputs and/or measurements will be different on the plant side from those on the controller side, which means that χ^2 on the plant side will be different from the controller side. Therefore, generating a residual based on the two χ^2 will help to isolate faults from cyber-attacks on the controller side.

4.6.1 State Estimates and Residual of Faulty Systems

We assumed that there was a Kalman filter-based fault detector on the plant side. However, the plant-side detector cannot distinguish legitimate control inputs from tampered inputs. Hence, this detector will be triggered only when a fault exists in the system and will not be sensitive to cyber-attacks. Form Section 2.3.1, additive and multiplicative actuator faults can be modeled as:

$$u_f(k) = Eu(k) + f$$

where $u_f(k)$ is the actual response of faulty actuators to the control input, $u(k)$ is the control input issued by the controller, and E and f are the effects of multiplicative and additive actuator faults. Additive sensor faults are modeled as follows:

$$y_f(k) = y(k) + \Delta y_f(k)$$

The estimated states of a faulty system, without any cyber-attack is:

$$
\begin{aligned}
\hat{x}_f(k+1|k) &= A\hat{x}_f(k) + Bu(k) \\
\hat{x}_f(k+1) &= \hat{x}_f(k+1|k) + K[y_f(k+1) - C\hat{x}_f(k+1|k)] \\
&= \hat{x}_f(k+1|k) + K[y(k+1) + \Delta y_f(k+1) - C\hat{x}_f(k+1|k)] \\
&= A\hat{x}_f(k) + Bu(k) + K[Cx_f(k+1) + \nu(k+1) + \Delta y_f(k+1) - C(A\hat{x}_f(k) + Bu(k))] \\
&= A\hat{x}_f(k) + Bu(k) + K[C(Ax_f(k) + Bu_f(k) + w(k)) + \nu(k+1) + \Delta y_f(k+1) \\
&\quad - C(A\hat{x}_f(k) + Bu(k))] \\
&= (A - KCA)\hat{x}_f(k) + Bu(k) - KCBu(k) + KC[Ax(k) + B(Eu(k) + f) + w(k)] \\
&\quad + K\Delta y_f(k+1) + K\nu(k+1) \\
&= (A - KCA)\hat{x}_f(k) + KCAx_f(k) + Bu(k) - KCB(I - E)u(k) \\
&\quad + KCBf + K\Delta y_f(k+1) + KCw(k) + K\nu(k+1) \qquad (4.87)
\end{aligned}
$$

The error dynamics of a Kalman filter estimator in the presence of both additive and multiplicative actuator faults and additive sensor faults are:

$$
\begin{aligned}
e_f(k+1) &= x_f(k+1) - \hat{x}_f(k+1) \\
&= Ax_f(k) + Bu_f(k) + w(k) - (A - KCA)\hat{x}_f(k) + KCAx_f(k) + Bu(k) - KCB(I - E)u(k) \\
&\quad + KCBf + K\Delta y_f(k+1) + KCw(k) + K\nu(k+1) \\
&= (A - KCA)(x_f(k) - \hat{x}_f(k)) + (I - KC)Bf + (I - KC)B(I - E)u(k) \\
&\quad + K\Delta y_f(k+1) + KCw(k) + K\nu(k+1) \\
&= (A - KCA)e_f(k) + (I - KC)Bf + (I - KC)B(I - E)u(k) \\
&\quad + K\Delta y_f(k+1) + KCw(k) + K\nu(k+1) \qquad (4.88)
\end{aligned}
$$

and the residuals are calculated as follows:

$$\begin{aligned}
z_f(k+1) &= y_f(k+1) - \hat{y}_f(k+1) \\
&= y(k+1) + \Delta y_f(k+1) - C\hat{x}_f(k+1) \\
&= Cx(k+1) + \nu(k+1) + \Delta y_f(k+1) - C\hat{x}_f(k+1) \\
&= C[x(k+1) - \hat{x}_f(k+1)] + \nu(k+1) + \Delta y_f(k+1) \\
&= C[Ax(k) + Bu_f(k) + w(k) - A\hat{x}_f(k) - Bu(k)] + \nu(k+1) + \Delta y_f(k+1) \\
&= CA[x_f(k) - \hat{x}_f(k)] + CB(Eu(k) + f) - CBu(k) + Cw(k) + \nu(k+1) + \Delta y_f(k+1) \\
&= CAe_f(k) + Cw(k) + \nu(k+1) - CB(I-E)u(k) + CBf + \Delta y_f(k+1) \qquad (4.89)
\end{aligned}$$

From equation (4.89), we can see if there are no multiplicative actuator faults, then $E = I$ and $CB(I - E)u(k) = 0$, and if there is no additive fault, then $f = 0$ and therefore $CDf = 0$. On the other hand, if there are no sensor faults, the term $\Delta y_f(k+1)$ vanishes. Therefore, equation (4.89) for a healthy system will have an expected value of zero, as shown in the below equation:

$$\begin{aligned}
\mathbb{E}[z](k+1) &= \mathbb{E}[CAe(k)] + \mathbb{E}[Cw(k)] + \mathbb{E}[\nu(k+1)] \\
&= CA\mathbb{E}[e(k)] = 0
\end{aligned}$$

4.6.2 State Estimates and Residual of Faulty Systems Equipped with Coding Schemes Under Cuber-attack

For a system under cyber-attack, the estimated states on the controller side will be affected by the cyber-attack signal. Hence, it will be different from the states on the plant side. The estimated states of faulty systems under cyber-attack will become:

$$
\begin{aligned}
\hat{x}^a_f(k+1|k) &= A\hat{x}^a_f(k) + Bu(k) \\
\hat{x}^a_f(k+1) &= \hat{x}^a_f(k+1|k) + K[y^a_f(k+1) - C\hat{x}^a_f(k+1|k)] \\
&= A\hat{x}^a_f(k) + Bu(k) + K[y'(k+1) + \Delta y'_f(k+1) + \Sigma_y(k+1) - C(A\hat{x}^a_f(k) + Bu(k))] \\
&= (A - KCA)\hat{x}^a_f(k) + Bu(k) - KCBu(k) + K\Sigma_y(k+1) + K\Delta y_f(k+1) + KC[Ax'_f(k) \\
&\quad + B(Eu'_f(k) + f) + w(k)] + K\nu(k+1) \\
&= (A - KCA)\hat{x}^a_f(k) + Bu(k) - KCBu(k) + K\Sigma_y(k+1) + K\Delta y_f(k+1) \\
&\quad KCAx'_f(k) + KCB[E(u(k) + \Sigma_u(k)) + f) + KCw(k) + K\nu(k+1) \\
&= (A - KCA)\hat{x}^a_f(k) + KCAx'_f(k) + Bu(k) - KCB(I-E)u(k) + KCBf + K\Delta y'_f(k+1) \\
&\quad + KCBE\Sigma_u(k) + K\Sigma_y(k+1) + KCw(k) + K\nu(k+1) \qquad (4.90)
\end{aligned}
$$

And the error dynamics for a faulty system under cyber-attack is as follows:

$$
\begin{aligned}
e^a_f(k+1) &= x'_f(k+1) - \hat{x}^a_f(k+1) \\
&= Ax'_f(k) + B[E(u(k) + \Sigma_u(k)) + f] + w(k) - (A - KCA)\hat{x}^a_f(k) - KCAx'_f(k) - Bu(k) \\
&\quad + KCB(I-E)u(k) - KCBf - K\Delta y'_f(k+1) - KCBE\Sigma_u(k) - K\Sigma_y(k+1) \\
&\quad - KCw(k) - K\nu(k+1)] \\
&= (A - KCA)(x'_f(k) - \hat{x}^a_f(k)) + (I - KC)Bf + (I - KC)B(I-E)u(k) + K\Delta y_f(k+1) \\
&\quad + (I - KC)B\Sigma_u(k) + K\Sigma_y(k+1) + KCw(k) + K\nu(k+1) \\
&= (A - KCA)e^a_f(k) - (I - KC)B(I-E)u(k) + (I - KC)Bf + K\Delta y_f(k=1) \\
&\quad + (I - KC)BE\Sigma_u(k) + K\Sigma_y(k+1) + KCw(k) + K\nu(k+1) \qquad (4.91)
\end{aligned}
$$

The residuals of the faulty system under cyber-attack become:

$$\begin{aligned}
z_f^a(k+1) &= y'_f(k+1) - \hat{y}_f^a(k+1) \\
&= y'(k+1) + \Delta y'_f(k+1) + \Sigma_y(k+1) - C\hat{x}_f^a(k+1) \\
&= Cx'(k+1) + \nu(k+1) + \Delta y_f(k+1) + \Sigma(k+1) - C\hat{x}_f^a(k+1) \\
&= C[Ax'_f(k) + Bu'_f(k) + w(k) - A\hat{x}_f^a(k) - Bu(k)] + \nu(k+1) + \Delta y_f(k+1) + \Sigma_y(k+1) \\
&= CA[x'_f(k) - \hat{x}_f^a(k)] + CB[E(u(k) + \Sigma_u(k)] + f] - CBu(k) + Cw(k) \\
&\quad + \nu(k+1) + \Delta y_f(k+1) + \Sigma_y(k+1) \\
&= CAe_f^a(k) - CB(I-E)u(k) + CBf + \Delta y_f(k+1) + CBE\Sigma_u(k) + \Sigma_y(k+1) \\
&\quad + Cw(k) + \nu(k+1) \qquad (4.92)
\end{aligned}$$

Since we assumed the cyber-attack signals are time-varying additive signals on both inputs and outputs, this equation is valid for both coding system, and any FDI cyber-attack signals without coding. When there is no cyber-attack both $\Sigma_u(k+1) = 0$ and $\Sigma_y(k+1) = 0$ and the equation (4.92) will become equation (4.89), and when there is no fault then $E = I$, $f = 0$ and $\Delta y'_f(k+1) = 0$ and equation (4.92) becomes equation (4.50).

For a faulty system, when there is not cyber-attack the expected value of residuals on the both the plant side and controller side will became:

$$\mathbb{E}[z_f(k+1)] = CA\mathbb{E}[e_f(k)] - CB(I-E)u(k) + CBf + \Delta y_f(k+1) \qquad (4.93)$$

However, for a faulty system under cyber-attack the expected value of residual on the controller side is:

$$\begin{aligned}
\mathbb{E}[z_f^a(k+1)] = CA\mathbb{E}[e_f^a(k)] &- CB(I-E)u(k) + CBf + \Delta y_f(k+1) \\
&+ CBE\Sigma_u(k) + \Sigma_y(k+1) \qquad (4.94)
\end{aligned}$$

On the plant side, the control input under cyber-attack is $u'(k) = u(k) + \Sigma_u(k)$, but there is no cyber-attack on the measurements on the plant side. Replacing for $u(k)$ in the equation (4.89) for the residual on the plant side under cyber-attack, the expected value of residual based on control input generated on the plant side is:

$$\begin{aligned}
z_f(k+1) &= CAe_f(k) - CB(I-E)u'(k) + CBf + Cw(k) + \nu(k+1) + \Delta y_f(k+1) \\
&= CAe_f(k) - CB(I-E)u(k) + CBf - CB(I-E)\Sigma_u(k) + \Delta y_f(k+1) \\
&\quad + Cw(k) + \nu(k+1) \\
\mathbb{E}[z_f(k+1)] &= CA\mathbb{E}[e_f(k)] - CB(I-E)u(k) + CBf - CB(I-E)\Sigma_u(k) + \Delta y_f(k+1)
\end{aligned} \tag{4.95}$$

equation (4.95) is the expected value of the residuals of Kalman filter for faulty system under cyber-attack on the plant side. Compared with equation (4.94), we can see that there are two extra parameters for injected cyber-attack signals in the equation (4.95). Since these residuals are for the same system, they have the same covariance. Therefor from equation (2.57) the χ^2 of them are also different as shown below:

$$g_f(k) = z_f(k)\Sigma^{-1}(k)z_f(k) \tag{4.96}$$

$$g_f^a(k) = z_f^a(k)\Sigma^{-1}(k)z_f^a(k) \tag{4.97}$$

A residual of these two values can isolate faults from cyber-attacks. Since residuals are already squared, we generate a new residual by the root square of the two residuals as follows:

$$z_{fa}(k) = \sqrt{g_f^a(k)} - \sqrt{g_f(k)} \tag{4.98}$$

$$g_{fa}(k) = z_{fa}(k)\Sigma_{fa}^{-1}(k)z_{fa}(k) \tag{4.99}$$

Where $\Sigma_{fa}^{-1}(k)$ is covariance for the two χ^2 terms.

4.6.3 Simulation Results of Isolation of Cyber-attack from Faults

In this subsection, we simulated the system presented in Figure 4.27. There are two identical Kalman filter-based detectors shown in that figure. One detector is on the plant side, and the other is on the controller side. In our analysis, aileron and rudder faults result in residuals in the latitudinal model, while elevator impacts residuals in the longitudinal model. Roll angle (p) and roll rate (ϕ) show high sensitivity to aileron faults. While side velocity v and yaw angle r also generate residuals with aileron faults, the magnitude of residuals is relatively smaller. On the other hand, rudder faults result in high residuals of side velocity (v) and yaw angle (r) but have a relatively small roll angle (p) residual. Elevator faults impact the longitudinal model and result in large residuals of vertical velocity (w) and pitch angle (q). This analysis can help isolate faults; however, in this chapter, our objective is to isolate cyber-attacks from faults. Therefore, we use a χ^2 detector on the plant side, which calculates the weighted squared sum of all residuals to get a single value indicating the presence of any residual in the faulty system. We used the same set of inputs and output on the controller side to calculate another χ^2. When there is no cyber-attack, the two χ^2 are calculated based on the same inputs and outputs, which will have the same expected value, and the deviation would be due to the noises of the system. Therefore, a residual calculated based on the equation (4.51) of the two χ^2 will stay below the threshold. On the other hand, in the presence of a cyber attack, the plant-side Kalman filter will estimate the states based on the actual outputs $y'(k)$ and under cyber-attack inputs $u'(k) = u(k) + \Sigma_u$, and at the same time, the controller side Kalman filter will estimate states based on actual control input $u(k)$ and under cyber-attack outputs $y(k) = y'(k) + \Sigma_y$. These two Kalman filters will generate different residuals, resulting in two different χ^2. It is worth noting that there will be an additional deviation between two χ^2 due to the dynamics of the system as shown in the equation (4.51); however, the deviation due to changes in signals that are the result of coding scheme impact the residuals instantaneously.

To graphically demonstrate this approach, we designed a cyber-attack scenario with the same configuration of coding and cyber-attack signals as the simulation presented in Section 4.5.1, except that from time stamp 70, a fault is started and remains on the system. Therefore, up to time stamp 70, there are neither faults nor cyber-attacks. From time stamp 70 to time stamp 100, there is a fault in the system, but no cyber-attack. From time stamp 100 up to the end of the simulation, simultaneous faults and cyber-attacks exist in the simulated system. The residuals generated in plant side and controller side is shown in Figure 4.28 and there are differences in some residuals after injection of cyber-attack.

The top plot of Figure 4.29 shows the plant side χ^2, and the bottom plot of that figure shows the controller side χ^2. Despite the general similarity, there are differences between the magnitude of the two χ^2. A residual based on differences of the square root of these two χ^2 is generated and shown on the top plot of Figure 4.30. After the start of the cyber-attack, when the cyber-attack signal magnitude increases, a deviation from the mean value of zero can be observed in this residual. The bottom plot of Figure 4.30 is a new χ^2, generated based on the equation (4.99), which can detect the start of the cyber-attach of a faulty system when the χ^2 increases from the specified threshold. We choose the threshold of the χ^2 for the residuals shown in Figure 4.15 from the table of critical values of χ^2 [100] by considering two degrees of freedom for two $\sqrt{\chi^2}$ of the plant side and controller side, that we used to calculate the

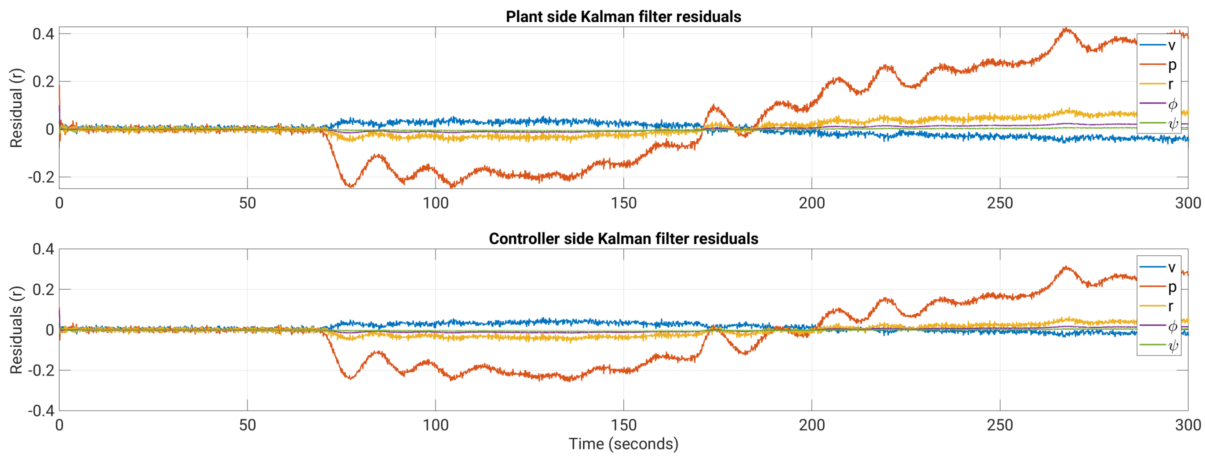


Figure 4.28: Kalman filter generated residuals on the plant side and controller side for a faulty system under cyber-attack.

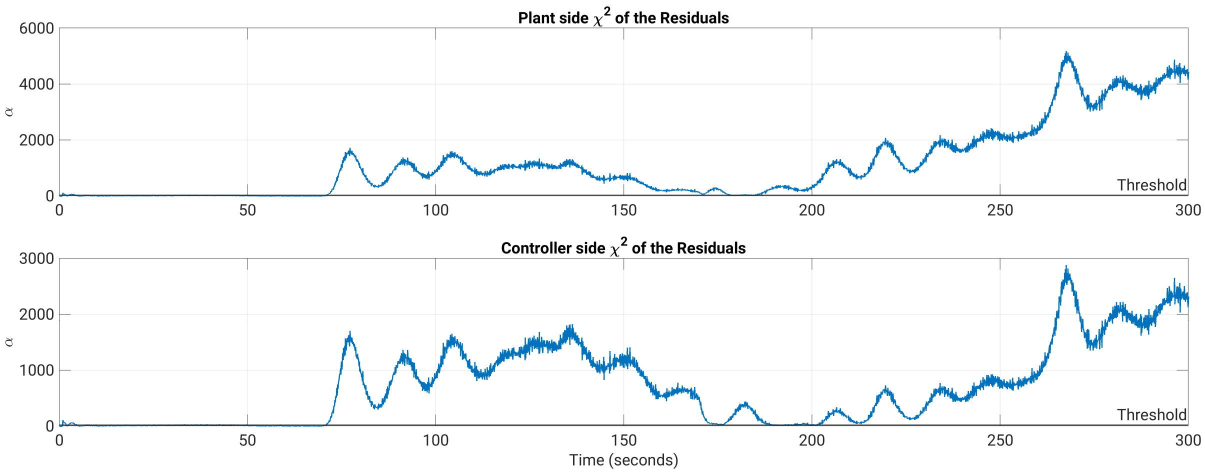


Figure 4.29: χ^2 on the plant side versus controller side for a faulty system under cyber-attack.

new χ^2. With a probability of 99% of χ^2 below the critical value, we get a value of 9.2 for the threshold. A 99% Probability means we expect that 99% of calculated χ^2 of the healthy system have a value of less than 9.2. Therefore, we expect a 1% misclassification of healthy states as faulty.

We also run a Monte Carlo simulation for 50 different scenarios. We used the randomly generated flight scenario as explained in Section 4.4.2 and randomly picked up five nonoverlapping slices of 350 seconds. We simulated lock-in-place and loss-of-functionality faults in random time stamps in the slices of the simulated flight scenario and injected cyber-attack signals at random time stamps after the start of the fault. We used a combination of different step cyber-attack signals that simultaneously injected into coded input and output. We calculated the detection rate of our method for all sample points of those scenarios. The

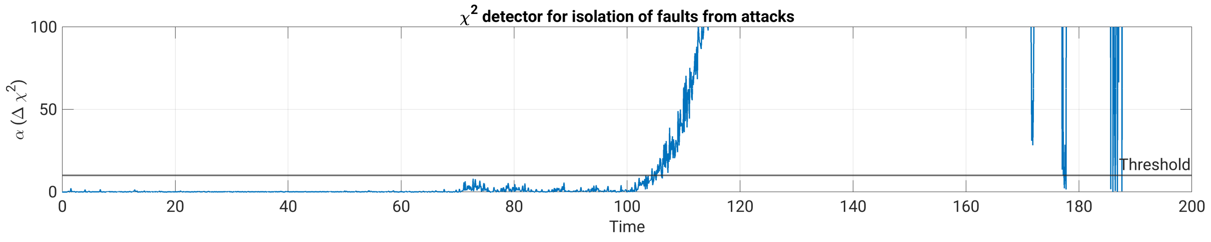


Figure 4.30: χ^2-based fault and cyber-attack isolation.

averaged results of the Monte Carlo simulation are presented in Tables 4.7 to 4.9.

F1-Score= 0.34		
	No cyber-attack	Cyber-attack
No cyber-attack	1653	1402
Cyber-attack	67	378

Table 4.7: Confusion matrix and F1-Score for cyber-attack with an impact of less than 0.15 degrees on the actuator and less than 2 degree on measurement.

F1-Score=0.61		
	No cyber-attack	Cyber-attack
No cyber-attack	1672	976
Cyber-attack	48	807

Table 4.8: Confusion matrix and F1-Score for cyber-attack with an impact of between 0.15 to 0.25 degrees on the actuator and between 2 to 3.5 degrees on measurement.

F1-Score=0.98		
	No cyber-attack	Cyber-attack
No cyber-attack	1669	12
Cyber-attack	51	1768

Table 4.9: Confusion matrix and F1-Score for cyber-attack with an impact of greater than 0.25 degrees on the actuator and greater than 3.5 degree on measurement.

We can see that our proposed method works well with coding and can correctly detect and isolate faults from cyber-attacks for attack signals that have an impact of greater than 0.15 degrees on inputs and 3 degrees on outputs.

4.7 Conclusion

In this chapter, the two-way coding introduced for scalar continuous-time control systems by [79], extended for MIMO systems and applied to discrete-time systems. We analyzed the proposed MIMO coding with the discrete-time Kalman filter and showed that generated residuals can detect stealthy cyber-attacks on inputs and outputs of the system. Further, we showed that for an attacker to remove the cyber-attack impact from coded output, it is necessary to know the coding.

Moreover, we proposed a new coding scheme using random numbers on the controller side that effectively converts coded inputs and outputs to time-varying random numbers. This coding scheme makes it impossible for an attacker to discover the coding matrix by eavesdropping, which is an improvement over [77] since there is no need for continuous calculation of the coding matrix and synchronization of switching coding matrix. Also, except for the initial coding matrix, there is no need to share any other information between the controller and the plant.

Another improvement compared to [79] is that this schema can detect replay cyber-attacks without degradation of controller performance, which is an improvement over watermarking methods such as [116].

Finally, the problem of isolation of faults from cyber-attacks, which can lead to false alarms, is solved by a new residual based on the plant side and controller side χ^2. Neither of the papers mentioned above studied the isolation of faults from cyber-attacks.

Overall, the advantage of the proposed coding is that a single coding scheme can defend against zero-dynamics, covert and replay cyber-attacks, and isolate faults from cyber-attacks.

Chapter 5

Conclusions and Future Works

UNMANNED Aerial Vehicles (UAVs) have revolutionized aviation operations, both civilian and military. With a projected global market revenue of $102 billion by 2030 [119], their applications are vast and valuable. However, UAVs are vulnerable to security weaknesses that could impact safety and national security. Robust fault and cyber-attack detection mechanisms are crucial for securing UAVs and maintaining their integrity in various applications. To address these problems, we leveraged SVMs as a supervised machine-learning technique and Kalman filtering as a proven model-based method. We showed that these methods can effectively detect and isolate faults and cyber-attacks. This thesis emphasizes the importance of hybrid approaches to fault and cyber-attacks by combining model-based and ML-based methods. In Chapter 3, showed that for fault and cyber-attack detection and isolation, SVMs can perform with exceptionally high accuracy. However, like any other machine learning method, knowledge of the data and data preparation is crucial. The first step of data preparation is removing unwanted noises from data, and the next steps are feature engineering and feature extraction. We showed that in the case of the UAV, PCA can perfectly extract features from correlated measurement data. However, knowing the data is essential for effectively applying the different tools. For instance, when the objective is fault detection of a specific actuator, including that actuator control input in the dataset before extracting features with PCA and picking up the first few principal components results in a dataset of PCs that did not include the variation of the specific actuator, and hence, ended up with a poor SVM model. The reason was the high variability of measurements in response to the actuator deflection. Therefore, the correct feature extraction method for the UAV is to pick the first few PCs from the measurements and states dataset and include the actuator's control input as features in a new dataset. We also showed that using the Kalman filter estimated states as features can solve both noise and feature extraction problems in one step. We generally design the Kalman filter to estimate the most essential states of the system. These states are the best choice of features for the case of machine learning methods. In addition, the Kalman filter removes noises from the measurement system with the knowledge of the system model. In machine learning projects, data preparation and feature extraction commonly take more than half of the time of the project. The choice of Kalman filer estimated states as features can hugely impact the industrial project execution time. We also showed that a priori estimator employment with the Kalman filter converts output residuals to state residuals. The state residuals can improve fault detection of systems that have unmeasurable states, such as efficiency. It can also account for the impact of control inputs on the states of the system. For the case of the UAV, directly including the control inputs as features, works as well as employment of a priori estimated states. In any of the cases, to use SVMs for fault detection, we do not need to generate residuals. Indeed, the use of residuals as features degraded the performance of SVMs. To isolate faults, we used a bank of SVMs trained to detect each actuator's faults. Our bank of SVMs has fewer classifications than one-versus-one multi-class SVMs and performs faster, and since the size of training datasets is smaller, tuning of SVMs is much faster. In conclusion, our proposed method of employing Kalman filtering and a bank of SVMs demonstrates a remarkably high detection rate and swift performance.

SVMs perform equally well for the FDI and replay cyber-attack detection on the control inputs of the ground control system. One-class SVM is a variant of SVM that is best for anomaly detection, specifically in cases where there are few anomalies and the model of the anomaly is unknown. One-class SVM defines a boundary around healthy data and classifies any observations that violate the boundary as anomalies. However, tuning one-class SVM with noisy data is challenging and generally results in false positives with healthy data. However, the noises and disturbances are a problem with UAVs in general since, in addition to measurement noises, the unknown and varying effects of wind impact the states of the UAV. Since we have defined the FDI cyber-attack as an additive signal for the control inputs or sensor outputs, we have a model for FDI. Therefore, we used this model to fine-tune two-class SVM to detect FDI cyber-attacks on noisy actuators of a UAV with eligible false positives in healthy data. Based on the UAV's performance and the noises of sensors and actuators, we can define a margin where a cyber-attack with a magnitude smaller than that margin will have an eligible effect on the UAV operation. Therefore, we generated a training dataset based on the healthy states of the UAV estimated by a Kalman filter and then artificially injected the FDI cyber-attack signals with the minimum magnitude to randomly selected portions of actuator commands. Compared with one-class SVM used by [108] for cyber-attack detection on autonomous car network sensors, our two-class SVM trained with this dataset could detect FDI cyber-attacks with an F1-score of about 0.97 on actuators with eligible false positives on healthy signals. A primary problem with both SVM models for cyber-attack detection is the presence of faults in the system, which also will be classified as a cyber-attack. To handle this problem, the availability of some information about faults in the system is necessary. We proposed to use two χ^2, one on the plant side and the other on the controller side. We trained a two-class SVM based on our proposed method with healthy on faulty states χ^2 of both the plant and controller side. Our proposed method could precisely detect cyber-attacks and reliably isolate faults from cyber-attacks. Another notable cyber-attack that grabs the attention of researchers is the replay cyber-attack. Physical watermarking is the most common method employed for relay cyber-attack detection. It is combined with the Kalman filter and χ^2 to detect the cyber-attack. For replay cyber-attack detection, we studied the capabilities of SVM without any model of the system. We proposed a search method based on the shifting data and consecutively tested it with the SVM to find the random patterns of watermarked inputs in the outputs. Compared with [114], we do not need the model of the system to detect the replay cyber-attack, and compared with [115], we could detect replay cyber-attacks in our tests with a 100% accuracy.

In Chapter 4, we dealt with the problem of stealthy cyber-attacks, where the attackers exploit full knowledge of the system to launch undetectable cyber-attacks. The most widely studied stealthy cyber-attacks in the control community literature are zero dynamics, covert, and replay cyber-attacks. To make stealthy cyber-attacks detectable, we used coding techniques. Coding for error correction introduced by Shannon's had a profound impact on various engineering and scientific fields [117]. In control literature, coding is modified and used for stealthy cyber-attack detection as an alternative to encryption. The two-way coding was used for zero-dynamics cyber-attack detection of the SISO continuous-time control systems by [79]. We extended it for MIMO systems and studied its properties for stealthy cyber-attack detection of discrete-time systems. We analyzed our proposed MIMO coding with the discrete-time Kalman filter and showed that generated residuals can detect stealthy cyber-attacks on inputs and outputs of the system. We showed that for an attacker to remove the cyber-attack impact from coded output, it is necessary to know the coding. To make it impossible for the attackers to reverse-engineer the coding matrix with eavesdropping, we proposed a new coding scheme using random numbers on the controller side that effectively converts coded inputs and outputs to time-varying random numbers. This coding scheme makes it impossible for an attacker to discover the coding matrix by eavesdropping, which is an improvement over [77] since there is no need for continuous calculation of the coding matrix and synchronization of switching coding matrix. Also, except for the coding matrix, there is no need to share any other information between the controller and the plant. Another improvement compared to [79] is that this schema can detect replay cyber-attacks without degradation of controller performance, which is an improvement over watermarking methods such as [116]. In simulations, we showed that a properly designed coding matrix can help mitigate or remove noise-like cyber-attack signals. Therefore, our proposed coding scheme can detect major stealthy cyber-attacks such as zero-dynamics [79], cover and replay cyber-attacks, and mitigate cyber-attacks compatible with the noises of the system. Finally, the problem of isolation of faults from cyber-attacks, which can lead to false alarms, is solved by a new residual based on the plant side and controller side χ^2. Neither of the papers mentioned above studied the isolation of faults from cyber-attacks.

5.1 Directions for Future Research

In this thesis, we investigated fault detection isolation and FDI cyber-attack detection using machine learning and stealthy cyber-attack detection problems of UAVs. We showed that combining SVM with Kalman filtering could detect and isolate faults. SVM by itself is not capable of detecting stealthy cyber-attacks. In Chapter 4, we showed that the two-way coding could help a model-based method such as the Kalman filter to detect stealthy cyber-attacks such as covert and replay cyber-attacks. We used a random number generator to make it difficult for an attacker to find out about the coding matrix by eavesdropping. However, this method is not safe for an internal attacker. Moreover, the coding cannot mitigate a cyber-attack, which means an attacker can harm the physical system even with coding implemented. Therefore, We suggest the following directions for future research:

- Combining coding with machin learning for stealthy cyber-attack detection and isolation
- Employment of random-number coding to modify the constant coding matrix during communication as a defense against internal threat
- Employment of a lightweight and modeless approach for isolation of faults from cyber-attacks
- Effectively mitigation of stealthy cyber-attacks on the plant

Bibliography

[1] National Academy of Science and Engineering, ed. *Cyber-Physical Systems.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. ISBN: 978-3-642-29089-3 978-3-642-29090-9. DOI: 10.1007/978-3-642-29090-9. (Visited on 05/16/2021).

[2] Paulo Leitão et al. "Smart Agents in Industrial Cyber–Physical Systems". In: *Proceedings of the IEEE* 104.5 (May 2016), pp. 1086–1101. ISSN: 1558-2256. DOI: 10.1109/JPROC.2016.2521931.

[3] USA National Science Foundation. *Nsf21551 Cyber-Physical Systems (CPS) — NSF - National Science Foundation.* https://www.nsf.gov/publications/pub_summ.jsp?ods_key=nsf21551&
2021. (Visited on 05/18/2021).

[4] Innovation Government of Canada. *Report from Canada's Economic Strategy Tables: Advanced Manufacturing.* https://www.ic.gc.ca/eic/site/098.nsf/eng/00021.html. Sept. 2018. (Visited on 10/14/2020).

[5] Derui Ding et al. "A Survey on Security Control and Attack Detection for Industrial Cyber-Physical Systems". In: *Neurocomputing* 275 (Jan. 2018), pp. 1674–1683. ISSN: 09252312. DOI: 10.1016/j.neucom.2017.10.009. (Visited on 12/23/2019).

[6] org iiconsortium. *What Is the Industrial Internet? — Industrial Internet Consortium.* https://www.iiconsortium.org/about-industrial-internet.htm. 2021. (Visited on 05/25/2021).

[7] *Commercial Drone Market Size & Share Report, 2021-2028.* https://www.grandviewresearch.com/in
analysis/global-commercial-drones-market. 2022. (Visited on 01/15/2022).

[8] *Map of Global Drone Incidents by Dedrone Anti-Drone / CUAS Solution.* https://www.dedrone.com/
Jan. 2022. (Visited on 01/14/2022).

[9] Fabio Pasqualetti, Florian Dörfler, and Francesco Bullo. "Attack Detection and Identification in Cyber-Physical Systems". In: *IEEE Transactions on Automatic Control* 58.11 (Nov. 2013), pp. 2715–2729. ISSN: 2334-3303. DOI: 10.1109/TAC.2013.2266831.

[10] A. J. Gallo et al. “Distributed Cyber-Attack Detection in the Secondary Control of DC Microgrids”. In: *2018 European Control Conference (ECC)*. June 2018, pp. 344–349. DOI: 10.23919/ECC.2018.8550549.

[11] Yao Liu, Peng Ning, and Michael K Reiter. “False Data Injection Attacks against State Estimation in Electric Power Grids”. In: (2011), p. 16. DOI: 10.1145/1952982.1952995.

[12] André Teixeira et al. “Cyber Security Analysis of State Estimators in Electric Power Systems”. In: *49th IEEE Conference on Decision and Control (CDC)*. Dec. 2010, pp. 5991–5998. DOI: 10.1109/CDC.2010.5717318.

[13] György Dán and Henrik Sandberg. “Stealth Attacks and Protection Schemes for State Estimators in Power Systems”. In: *2010 First IEEE International Conference on Smart Grid Communications*. Oct. 2010, pp. 214–219. DOI: 10.1109/SMARTGRID.2010.5622046.

[14] Y. Shoukry and P. Tabuada. “Event-Triggered State Observers for Sparse Sensor Noise/Attacks”. In: *IEEE Transactions on Automatic Control* 61.8 (Aug. 2016), pp. 2079–2091. ISSN: 1558-2523. DOI: 10.1109/TAC.2015.2492159.

[15] Yilin Mo and Bruno Sinopoli. “False Data Injection Attacks in Control Systems”. In: (2010), p. 7.

[16] Carey Witkov and Keith Zengel. *Chi-Squared Data Analysis and Model Testing for Beginners*. Oxford University Press, 2019.

[17] Kebina Manandhar et al. “Detection of Faults and Attacks Including False Data Injection Attack in Smart Grid Using Kalman Filter”. In: *IEEE Transactions on Control of Network Systems* 1.4 (Dec. 2014), pp. 370–379. ISSN: 2325-5870. DOI: 10.1109/TCNS.2014.2357531.

[18] Cheng-Zong Bai, Fabio Pasqualetti, and Vijay Gupta. “Security in Stochastic Control Systems: Fundamental Limitations and Performance Bounds”. In: *2015 American Control Conference (ACC)*. July 2015, pp. 195–200. DOI: 10.1109/ACC.2015.7170734.

[19] Mohammad Hossein Basiri et al. “Kalman Filter Based Secure State Estimation and Individual Attacked Sensor Detection in Cyber-Physical Systems”. In: *2019 American Control Conference (ACC)*. Philadelphia, PA, USA: IEEE, July 2019, pp. 3841–3848. ISBN: 978-1-5386-7926-5. DOI: 10.23919/ACC.2019.8814963. (Visited on 08/20/2020).

[20] Jie Chen, Ron J. Patton, and Hong-Yue Zhang. “Design of Unknown Input Observers and Robust Fault Detection Filters”. In: *International Journal of Control* 63.1 (Jan. 1996), pp. 85–105. ISSN: 0020-7179, 1366-5820. DOI: 10.1080/00207179608921833. (Visited on 11/30/2020).

[21] Peter Fogh Odgaard and Jakob Stoustrup. “Fault Tolerant Control of Wind Turbines Using Unknown Input Observers”. In: *IFAC Proceedings Volumes*. 8th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes 45.20 (Jan. 2012), pp. 313–318. ISSN: 1474-6670. DOI: 10.3182/20120829-3-MX-2028.00010. (Visited on 05/25/2022).

[22] Zhiwei Gao, Xiaoxu Liu, and Michael Z. Q. Chen. “Unknown Input Observer-Based Robust Fault Estimation for Systems Corrupted by Partially Decoupled Disturbances”. In: *IEEE Transactions on Industrial Electronics* 63.4 (Apr. 2016), pp. 2537–2547. ISSN: 1557-9948. DOI: 10.1109/TIE.2015.2497201.

[23] Hassan Haes Alhelou, Mohamad Esmail Hamedani Golshan, and Nikos D. Hatziargyriou. “Deterministic Dynamic State Estimation-Based Optimal LFC for Interconnected Power Systems Using Unknown Input Observer”. In: *IEEE Transactions on Smart Grid* 11.2 (Mar. 2020), pp. 1582–1592. ISSN: 1949-3061. DOI: 10.1109/TSG.2019.2940199.

[24] Saurabh Amin et al. “Cyber Security of Water SCADA Systems—Part II: Attack Detection Using Enhanced Hydrodynamic Models”. In: *IEEE Transactions on Control Systems Technology* 21.5 (Sept. 2013), pp. 1679–1693. ISSN: 1558-0865. DOI: 10.1109/TCST.2012.2211874.

[25] A. Teixeira, H. Sandberg, and K. H. Johansson. “Networked Control Systems under Cyber Attacks with Applications to Power Networks”. In: *Proceedings of the 2010 American Control Conference*. June 2010, pp. 3690–3696. DOI: 10.1109/ACC.2010.5530638.

[26] Zhiwei Gao, Carlo Cecati, and Steven X. Ding. “A Survey of Fault Diagnosis and Fault-Tolerant Techniques—Part I: Fault Diagnosis With Model-Based and Signal-Based Approaches”. In: *IEEE Transactions on Industrial Electronics* 62.6 (June 2015), pp. 3757–3767. ISSN: 1557-9948. DOI: 10.1109/TIE.2015.2417501.

[27] S. Tan et al. “Brief Survey on Attack Detection Methods for Cyber-Physical Systems”. In: *IEEE Systems Journal* 14.4 (Dec. 2020), pp. 5329–5339. ISSN: 1937-9234. DOI: 10.1109/JSYST.2020.2991258.

[28] Janos J. Gertler. *Fault Detection and Diagnosis in Engineering Systems*. Boca Raton: CRC Press, 1998. ISBN: 978-0-203-75612-6. DOI: 10.1201/9780203756126.

[29] Maiying Zhong et al. “Parity Space-Based Fault Detection by Minimum Error Minimax Probability Machine”. In: *IFAC-PapersOnLine* 51.24 (2018), pp. 1292–1297. ISSN: 24058963. DOI: 10.1016/j.ifacol.2018.09.568. (Visited on 06/28/2022).

[30] Sen Tan et al. “False Data Injection Cyber-Attacks Detection for Multiple DC Microgrid Clusters”. In: *Applied Energy* 310 (Mar. 2022), p. 118425. ISSN: 03062619. DOI: 10.1016/j.apenergy.2021.118425. (Visited on 06/28/2022).

[31] Varun Chandola, Arindam Banerjee, and Vipin Kumar. “Anomaly Detection: A Survey”. In: *ACM Computing Surveys* 41.3 (July 2009), pp. 1–58. ISSN: 0360-0300, 1557-7341. DOI: 10.1145/1541880.1541882. (Visited on 11/02/2021).

[32] Xiuyao Song et al. “Conditional Anomaly Detection”. In: *IEEE Transactions on Knowledge and Data Engineering* 19.5 (May 2007), pp. 631–645. ISSN: 1558-2191. DOI: 10.1109/TKDE.2007.1009.

[33] Ryohei Fujimaki, Takehisa Yairi, and Kazuo Machida. “An Approach to Spacecraft Anomaly Detection Problem Using Kernel Feature Space”. In: *Proceeding of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining - KDD ’05*. Chicago, Illinois, USA: ACM Press, 2005, p. 401. ISBN: 978-1-59593-135-1. DOI: 10.1145/1081870.1081917. (Visited on 11/17/2021).

[34] Zhiwei Gao, Carlo Cecati, and Steven X. Ding. “A Survey of Fault Diagnosis and Fault-Tolerant Techniques—Part II: Fault Diagnosis With Knowledge-Based and Hybrid/Active Approaches”. In: *IEEE Transactions on Industrial Electronics* 62.6 (June 2015), pp. 3768–3774. ISSN: 1557-9948. DOI: 10.1109/TIE.2015.2419013.

[35] Lei Cui et al. “Detecting False Data Attacks Using Machine Learning Techniques in Smart Grid: A Survey”. In: *Journal of Network and Computer Applications* 170 (Nov. 2020), p. 102808. ISSN: 10848045. DOI: 10.1016/j.jnca.2020.102808. (Visited on 06/03/2022).

[36] Rui Zhao et al. “Deep Learning and Its Applications to Machine Health Monitoring”. In: *Mechanical Systems and Signal Processing* 115 (Jan. 2019), pp. 213–237. ISSN: 08883270. DOI: 10.1016/j.ymssp.2018.05.050. (Visited on 08/03/2021).

[37] Jun Zhang et al. “Deep Learning Based Attack Detection for Cyber-Physical System Cybersecurity: A Survey”. In: *IEEE/CAA Journal of Automatica Sinica* 9.3 (Mar. 2022), pp. 377–391. ISSN: 2329-9274. DOI: 10.1109/JAS.2021.1004261.

[38] Alexander N. Gorban et al., eds. *Principal Manifolds for Data Visualization and Dimension Reduction*. Vol. 58. Lecture Notes in Computational Science and Enginee. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. ISBN: 978-3-540-73749-0 978-3-540-73750-6. DOI: 10.1007/978-3-540-73750-6. (Visited on 05/29/2022).

[39] Xun Wang et al. “Nonlinear PCA With the Local Approach for Diesel Engine Fault Detection and Diagnosis”. In: *IEEE Transactions on Control Systems Technology* 16.1 (Jan. 2008), pp. 122–129. ISSN: 1558-0865. DOI: 10.1109/TCST.2007.899744.

[40] Bingzhen Jiang, Jiawei Xiang, and Yanxue Wang. “Rolling Bearing Fault Diagnosis Approach Using Probabilistic Principal Component Analysis Denoising and Cyclic Bispectrum”. In: *Journal of Vibration and Control* 22.10 (June 2016), pp. 2420–2433. ISSN: 1077-5463. DOI: 10.1177/1077546314547533. (Visited on 05/29/2022).

[41] Chen Jing and Jian Hou. “SVM and PCA Based Fault Classification Approaches for Complicated Industrial Process”. In: *Neurocomputing* 167 (Nov. 2015), pp. 636–642. ISSN: 09252312. DOI: 10.1016/j.neucom.2015.03.082. (Visited on 07/26/2021).

[42] Steven X. Ding et al. “A Novel Scheme for Key Performance Indicator Prediction and Diagnosis With Application to an Industrial Hot Strip Mill”. In: *IEEE Transactions on Industrial Informatics* 9.4 (Nov. 2013), pp. 2239–2247. ISSN: 1941-0050. DOI: 10.1109/TII.2012.2214394.

[43] Riccardo Muradore and Paolo Fiorini. “A PLS-Based Statistical Approach for Fault Detection and Isolation of Robotic Manipulators”. In: *IEEE Transactions on Industrial Electronics* 59.8 (Aug. 2012), pp. 3167–3175. ISSN: 1557-9948. DOI: 10.1109/TIE.2011.2167110.

[44] Jun Yi et al. “A Novel Framework for Fault Diagnosis Using Kernel Partial Least Squares Based on an Optimal Preference Matrix”. In: *IEEE Transactions on Industrial Electronics* 64.5 (May 2017), pp. 4315–4324. ISSN: 1557-9948. DOI: 10.1109/TIE.2017.2668986.

[45] Zuyu Yin and Jian Hou. “Recent Advances on SVM Based Fault Diagnosis and Process Monitoring in Complicated Industrial Processes”. In: *Neurocomputing* 174 (Jan. 2016), pp. 643–650. ISSN: 09252312. DOI: 10.1016/j.neucom.2015.09.081. (Visited on 07/25/2021).

[46] Shen Yin et al. “Study on Support Vector Machine-Based Fault Detection in Tennessee Eastman Process”. In: *Abstract and Applied Analysis* 2014 (2014), pp. 1–8. ISSN: 1085-3375, 1687-0409. DOI: 10.1155/2014/836895. (Visited on 07/24/2021).

[47] Mehdi Namdari, Hooshang Jazayeri-Rad, and Seyed-Jalaladdin Hashemi. “Process Fault Diagnosis Using Support Vector Machines with a Genetic Algorithm Based Parameter Tuning”. In: *Journal of Automation and Control* 2.1 (Dec. 2013), pp. 1–7. DOI: 10.12691/automation-2-1-1. (Visited on 07/26/2021).

[48] Mohammad Esmalifalak et al. “Detecting Stealthy False Data Injection Using Machine Learning in Smart Grid”. In: *IEEE Systems Journal* 11.3 (Sept. 2017), pp. 1644–1652. ISSN: 1937-9234. DOI: 10.1109/JSYST.2014.2341597.

[49] Arslan Shafique, Abid Mehmood, and Mourad Elhadef. “Detecting Signal Spoofing Attack in UAVs Using Machine Learning Models”. In: *IEEE Access* 9 (2021), pp. 93803–93815. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3089847.

[50] Kun Yang, Samory Kpotufe, and Nick Feamster. “An Efficient One-Class SVM for Anomaly Detection in the Internet of Things”. In: *arXiv:2104.11146 [cs]* (Apr. 2021). arXiv: 2104.11146 [cs]. (Visited on 08/04/2021).

[51] Pengyuan Wang and Manimaran Govindarasu. “Multi-Agent Based Attack-Resilient System Integrity Protection for Smart Grid”. In: *IEEE Transactions on Smart Grid* 11.4 (July 2020), pp. 3447–3456. ISSN: 1949-3061. DOI: 10.1109/TSG.2020.2970755.

[52] Yaguo Lei et al. “Applications of Machine Learning to Machine Fault Diagnosis: A Review and Roadmap”. In: *Mechanical Systems and Signal Processing* 138 (Apr. 2020), p. 106587. ISSN: 08883270. DOI: 10.1016/j.ymssp.2019.106587. (Visited on 07/24/2021).

[53] Long Wen et al. “A New Convolutional Neural Network-Based Data-Driven Fault Diagnosis Method”. In: *IEEE Transactions on Industrial Electronics* 65.7 (July 2018), pp. 5990–5998. ISSN: 1557-9948. DOI: 10.1109/TIE.2017.2774777.

[54] Peng Wang et al. “Virtualization and Deep Recognition for System Fault Classification”. In: *Journal of Manufacturing Systems* 44 (July 2017), pp. 310–316. ISSN: 02786125. DOI: 10.1016/j.jmsy.2017.04.012. (Visited on 06/05/2022).

[55] Ruonan Liu et al. “Dislocated Time Series Convolutional Neural Architecture: An Intelligent Fault Diagnosis Approach for Electric Machine”. In: *IEEE Transactions on Industrial Informatics* 13.3 (June 2017), pp. 1310–1320. ISSN: 1941-0050. DOI: 10.1109/TII.2016.2645238.

[56] Donghuan Yao et al. “Energy Theft Detection With Energy Privacy Preservation in the Smart Grid”. In: *IEEE Internet of Things Journal* 6.5 (Oct. 2019), pp. 7659–7669. ISSN: 2327-4662. DOI: 10.1109/JIOT.2019.2903312.

[57] Hyun Min Song, Jiyoung Woo, and Huy Kang Kim. “In-Vehicle Network Intrusion Detection Using Deep Convolutional Neural Network”. In: *Vehicular Communications* 21 (Jan. 2020), p. 100198. ISSN: 22142096. DOI: 10.1016/j.vehcom.2019.100198. (Visited on 06/05/2022).

[58] Senthil Murugan Nagarajan et al. "IADF-CPS: Intelligent Anomaly Detection Framework towards Cyber Physical Systems". In: *Computer Communications* 188 (Apr. 2022), pp. 81–89. ISSN: 01403664. DOI: 10.1016/j.comcom.2022.02.022. (Visited on 05/22/2022).

[59] Wenjun Sun et al. "A Sparse Auto-Encoder-Based Deep Neural Network Approach for Induction Motor Faults Classification". In: *Measurement* 89 (July 2016), pp. 171–178. ISSN: 02632241. DOI: 10.1016/j.measurement.2016.04.007. (Visited on 06/05/2022).

[60] Chen Lu et al. "Fault Diagnosis of Rotary Machinery Components Using a Stacked Denoising Autoencoder-Based Health State Identification". In: *Signal Processing* 130 (Jan. 2017), pp. 377–388. ISSN: 01651684. DOI: 10.1016/j.sigpro.2016.07.028. (Visited on 06/05/2022).

[61] Huaizhi Wang et al. "Deep Learning-Based Interval State Estimation of AC Smart Grids Against Sparse Cyber Attacks". In: *IEEE Transactions on Industrial Informatics* 14.11 (Nov. 2018), pp. 4766–4778. ISSN: 1941-0050. DOI: 10.1109/TII.2018.2804669.

[62] Moshe Kravchik and Asaf Shabtai. "Efficient Cyber Attack Detection in Industrial Control Systems Using Lightweight Neural Networks and PCA". In: *IEEE Transactions on Dependable and Secure Computing* (2021), pp. 1–1. ISSN: 1941-0018. DOI: 10.1109/TDSC.2021.3050101.

[63] Javed Ashraf et al. "Novel Deep Learning-Enabled LSTM Autoencoder Architecture for Discovering Anomalous Events From Intelligent Transportation Systems". In: *IEEE Transactions on Intelligent Transportation Systems* 22.7 (July 2021), pp. 4507–4518. ISSN: 1558-0016. DOI: 10.1109/TITS.2020.3017882.

[64] Jürgen Schmidhuber. "Deep Learning in Neural Networks: An Overview". In: *Neural Networks* 61 (Jan. 2015), pp. 85–117. ISSN: 08936080. DOI: 10.1016/j.neunet.2014.09.003. (Visited on 06/03/2022).

[65] Mei Yuan, Yuting Wu, and Li Lin. "Fault Diagnosis and Remaining Useful Life Estimation of Aero Engine Using LSTM Neural Network". In: *2016 IEEE International Conference on Aircraft Utility Systems (AUS)*. Oct. 2016, pp. 135–140. DOI: 10.1109/AUS.2016.7748035.

[66] Rui Zhao et al. "Machine Health Monitoring Using Local Feature-Based Gated Recurrent Unit Networks". In: *IEEE Transactions on Industrial Electronics* 65.2 (Feb. 2018), pp. 1539–1548. ISSN: 1557-9948. DOI: 10.1109/TIE.2017.2733438.

[67] Cheng Feng, Tingting Li, and Deeph Chana. “Multi-Level Anomaly Detection in Industrial Control Systems via Package Signatures and LSTM Networks”. In: *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. June 2017, pp. 261–272. DOI: 10.1109/DSN.2017.34.

[68] Mohamed Amine Ferrag and Leandros Maglaras. “DeepCoin: A Novel Deep Learning and Blockchain-Based Energy Exchange Framework for Smart Grids”. In: *IEEE Transactions on Engineering Management* 67.4 (Nov. 2020), pp. 1285–1297. ISSN: 1558-0040. DOI: 10.1109/TEM.2019.2922936.

[69] Edan Habler and Asaf Shabtai. “Using LSTM Encoder-Decoder Algorithm for Detecting Anomalous ADS-B Messages”. In: *Computers & Security* 78 (Sept. 2018), pp. 155–173. ISSN: 01674048. DOI: 10.1016/j.cose.2018.07.004. (Visited on 06/08/2022).

[70] Paul Griffioen et al. “A Tutorial on Detecting Security Attacks on Cyber-Physical Systems”. In: *2019 18th European Control Conference (ECC)*. June 2019, pp. 979–984. DOI: 10.23919/ECC.2019.8796117.

[71] Riccardo MG Ferrari and André MH Teixeira. *Safety, Security and Privacy for Cyber-Physical Systems*. Springer, 2021.

[72] Y. Mo and B. Sinopoli. “Secure Control against Replay Attacks”. In: *2009 47th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. Sept. 2009, pp. 911–918. DOI: 10.1109/ALLERTON.2009.5394956.

[73] Y. Mo, S. Weerakkody, and B. Sinopoli. “Physical Authentication of Control Systems: Designing Watermarked Control Inputs to Detect Counterfeit Sensor Outputs”. In: *IEEE Control Systems Magazine* 35.1 (Feb. 2015), pp. 93–109. ISSN: 1941-000X. DOI: 10.1109/MCS.2014.2364724.

[74] Sean Weerakkody and Bruno Sinopoli. “Detecting Integrity Attacks on Control Systems Using a Moving Target Approach”. In: *2015 54th IEEE Conference on Decision and Control (CDC)*. Dec. 2015, pp. 5820–5826. DOI: 10.1109/CDC.2015.7403134.

[75] Paul Griffioen, Sean Weerakkody, and Bruno Sinopoli. “An Optimal Design of a Moving Target Defense for Attack Detection in Control Systems”. In: *2019 American Control Conference (ACC)*. July 2019, pp. 4527–4534. DOI: 10.23919/ACC.2019.8814689.

[76] Jue Tian et al. “Moving Target Defense Approach to Detecting Stuxnet-Like Attacks”. In: *IEEE Transactions on Smart Grid* 11.1 (Jan. 2020), pp. 291–300. ISSN: 1949-3061. DOI: 10.1109/TSG.2019.2921245.

[77] Fei Miao et al. “Coding Schemes for Securing Cyber-Physical Systems Against Stealthy Data Injection Attacks”. In: *IEEE Transactions on Control of Network Systems* 4.1 (Mar. 2017), pp. 106–117. ISSN: 2325-5870. DOI: 10.1109/TCNS.2016.2573039.

[78] Anastasios Tsiamis, Konstantinos Gatsis, and George J. Pappas. “State Estimation Codes for Perfect Secrecy”. In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. Dec. 2017, pp. 176–181. DOI: 10.1109/CDC.2017.8263662.

[79] Song Fang et al. *Two-Way Coding in Control Systems Under Injection Attacks: From Attack Detection to Attack Correction*. Jan. 2019. arXiv: `1901.05420 [cs, eess, math]`. (Visited on 10/20/2022).

[80] Hamza Fawzi, Paulo Tabuada, and Suhas Diggavi. “Secure Estimation and Control for Cyber-Physical Systems Under Adversarial Attacks”. In: *IEEE Transactions on Automatic Control* 59.6 (June 2014), pp. 1454–1467. ISSN: 1558-2523. DOI: 10.1109/TAC.2014.2303233.

[81] M. S. Chong, M. Wakaiki, and J. P. Hespanha. “Observability of Linear Systems under Adversarial Attacks”. In: *2015 American Control Conference (ACC)*. July 2015, pp. 2439–2444. DOI: 10.1109/ACC.2015.7171098.

[82] Roy S. Smith. “A Decoupled Feedback Structure for Covertly Appropriating Networked Control Systems”. In: *IFAC Proceedings Volumes* 44.1 (Jan. 2011), pp. 90–95. ISSN: 14746670. DOI: 10.3182/20110828-6-IT-1002.01721. (Visited on 07/23/2020).

[83] Vishaal Krishnan and Fabio Pasqualetti. “Data-Driven Attack Detection for Linear Systems”. In: *IEEE Control Systems Letters* 5.2 (Apr. 2021), pp. 671–676. ISSN: 2475-1456. DOI: 10.1109/LCSYS.2020.3005102. arXiv: 2003.07949.

[84] Aquib Mustafa, Hamidreza Modares, and Rohollah Moghadam. “Resilient Synchronization of Distributed Multi-Agent Systems under Attacks”. In: *Automatica* 115 (May 2020), p. 108869. ISSN: 00051098. DOI: 10.1016/j.automatica.2020.108869. (Visited on 03/21/2021).

[85] Hung-Jen Liao et al. “Intrusion Detection System: A Comprehensive Review”. In: *Journal of Network and Computer Applications* 36.1 (Jan. 2013), pp. 16–24. ISSN: 10848045. DOI: 10.1016/j.jnca.2012.09.004. (Visited on 05/25/2022).

[86] Jairo Giraldo et al. “A Survey of Physics-Based Attack Detection in Cyber-Physical Systems”. In: *ACM Computing Surveys* 51.4 (Sept. 2018), pp. 1–36. ISSN: 0360-0300, 1557-7341. DOI: 10.1145/3203245. (Visited on 07/28/2020).

[87] Douglas M. Hawkins. “The Problem of Overfitting”. In: *Journal of Chemical Information and Computer Sciences* 44.1 (Jan. 2004), pp. 1–12. ISSN: 0095-2338. DOI: 10.1021/ci0342472. (Visited on 06/09/2022).

[88] Haidong Li et al. “Research on Overfitting of Deep Learning”. In: *2019 15th International Conference on Computational Intelligence and Security (CIS)*. Dec. 2019, pp. 78–81. DOI: 10.1109/CIS.2019.00025.

[89] S. Yin et al. “A Review on Basic Data-Driven Approaches for Industrial Process Monitoring”. In: *IEEE Transactions on Industrial Electronics* 61.11 (Nov. 2014), pp. 6418–6428. ISSN: 1557-9948. DOI: 10.1109/TIE.2014.2301773.

[90] Edward A. Lee. “Cyber Physical Systems: Design Challenges”. In: *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*. May 2008, pp. 363–369. DOI: 10.1109/ISORC.2008.25.

[91] Kaveh Paridari et al. “A Framework for Attack-Resilient Industrial Control Systems: Attack Detection and Controller Reconfiguration”. In: *Proceedings of the IEEE* 106.1 (Jan. 2018), pp. 113–128. ISSN: 1558-2256. DOI: 10.1109/JPROC.2017.2725482.

[92] Fabio Pasqualetti, Florian Dorfler, and Francesco Bullo. “Control-Theoretic Methods for Cyberphysical Security: Geometric Principles for Optimal Cross-Layer Resilient Control Systems”. In: *IEEE Control Systems Magazine* 35.1 (Feb. 2015), pp. 110–127. ISSN: 1941-000X. DOI: 10.1109/MCS.2014.2364725.

[93] Randal W. Beard and Timothy W. McLain. *Small Unmanned Aircraft: Theory and Practice*. Princeton University Press, Feb. 2012. ISBN: 978-1-4008-4060-1.

[94] Yew Chai Paw. “Synthesis and Validation of Flight Control for UAV”. PhD thesis. University of Minnesota, 2009.

[95] Elgiz Baskaya, Murat Bronz, and Daniel Delahaye. “Fault Detection & Diagnosis for Small UAVs via Machine Learning”. In: *2017 IEEE/AIAA 36th Digital Avionics Systems Conference (DASC)*. Sept. 2017, pp. 1–6. DOI: 10.1109/DASC.2017.8102037.

[96] Paul Freeman et al. “Model-Based and Data-Driven Fault Detection Performance for a Small UAV”. In: *IEEE/ASME Transactions on Mechatronics* 18.4 (Aug. 2013), pp. 1300–1309. ISSN: 1941-014X. DOI: 10.1109/TMECH.2013.2258678.

[97] Gareth James et al. *An Introduction to Statistical Learning*. Vol. 112. Springer, 2013.

[98] Fariha Sohil, Muhammad Umair Sohali, and Javid Shabbir. “An Introduction to Statistical Learning with Applications in R: By Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani, New York, Springer Science and Business Media, 2013, \$41.98, eISBN: 978-1-4614-7137-7”. In: *Statistical Theory and Related Fields* 6.1 (Jan. 2022), pp. 87–87. ISSN: 2475-4269, 2475-4277. DOI: 10.1080/24754269.2021.1980261. (Visited on 07/21/2023).

[99] P.E. Greenwood and M.S. Nikulin. *A Guide to Chi-Squared Testing*. Wiley Series in Probability and Statistics. Wiley, 1996. ISBN: 978-0-471-55779-1.

[100] *1.3.6.7.4. Critical Values of the Chi-Square Distribution*. https://www.itl.nist.gov/div898/handbook
2024. (Visited on 04/20/2024).

[101] B. Brumback and M. Srinath. “A Chi-square Test for Fault-Detection in Kalman Filters”. In: *IEEE Transactions on Automatic Control* 32.6 (June 1987), pp. 552–554. ISSN: 1558-2523. DOI: 10.1109/TAC.1987.1104658.

[102] Shaun Turney. *Chi-Square (X^2) Table — Examples & Downloadable Table*. https://www.scribbr.com
square-distribution-table/. May 2022. (Visited on 03/24/2024).

[103] *UMN Ultrastick Simulation*. https://dept.aem.umn.edu/~mettler/Courses/AEM%205333%20(spring
(Visited on 06/07/2023).

[104] R Core Team. *R: A Language and Environment for Statistical Computing*. Manual. Vienna, Austria, 2023.

[105] Ren Da. *Failure Detection of Dynamical Systems with the State Chi-Square Test*. https://arc.aiaa.org/doi/epdf/10.2514/3.21193. 1994. DOI: 10.2514/3.21193. (Visited on 10/09/2023).

[106] William M Campbell. “A Sequence Kernel and Its Application to Speaker Recognition”. In: (2001).

[107] Andrew O. Hatch, Sachin Kajarekar, and Andreas Stolcke. “Within-Class Covariance Normalization for SVM-based Speaker Recognition”. In: *Interspeech 2006*. ISCA, Sept. 2006, paper 1874-Wed1A1O.5–. DOI: 10.21437/Interspeech.2006-183. (Visited on 05/26/2024).

[108] Yiyang Wang, Neda Masoud, and Anahita Khojandi. “Real-Time Sensor Anomaly Detection and Recovery in Connected Automated Vehicle Sensors”. In: *IEEE Transactions on Intelligent Transportation Systems* 22.3 (Mar. 2021), pp. 1411–1421. ISSN: 1524-9050, 1558-0016. DOI: 10.1109/TITS.2020.2970295. arXiv: 1911.01531 [eess, stat]. (Visited on 12/24/2023).

[109] Bernhard Schölkopf et al. "Estimating the Support of a High-Dimensional Distribution". In: *Neural Computation* 13.7 (July 2001), pp. 1443–1471. ISSN: 0899-7667. DOI: 10.1162/089976601750264965. eprint: https://direct.mit.edu/neco/article-pdf/13/7/1443/814849/089976601750264965.pdf.

[110] Murat Bronz et al. "Real-Time Fault Detection on Small Fixed-Wing UAVs Using Machine Learning". In: *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*. Oct. 2020, pp. 1–10. DOI: 10.1109/DASC50938.2020.9256800.

[111] Federico Calesella et al. "A Comparison of Feature Extraction Methods for Prediction of Neuropsychological Scores from Functional Connectivity Data of Stroke Patients". In: *Brain Informatics* 8.1 (Apr. 2021), p. 8. ISSN: 2198-4026. DOI: 10.1186/s40708-021-00129-1. (Visited on 07/22/2023).

[112] Gil Press. *Cleaning Big Data: Most Time-Consuming, Least Enjoyable Data Science Task, Survey Says.* https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/. 2016. (Visited on 06/14/2024).

[113] Bahram Yaghooti, Raffaele Romagnoli, and Bruno Sinopoli. "Physical Watermarking for Replay Attack Detection in Continuous-Time Systems". In: *European Journal of Control*. 2021 European Control Conference Special Issue 62 (Nov. 2021), pp. 57–62. ISSN: 0947-3580. DOI: 10.1016/j.ejcon.2021.06.012. (Visited on 07/11/2023).

[114] Yilin Mo, Rohan Chabukswar, and Bruno Sinopoli. "Detecting Integrity Attacks on SCADA Systems". In: *IEEE Transactions on Control Systems Technology* 22.4 (July 2014), pp. 1396–1407. ISSN: 1558-0865. DOI: 10.1109/TCST.2013.2280899.

[115] Mingliang Ma et al. "Detecting Replay Attacks in Power Systems: A Data-Driven Approach". In: *Advanced Computational Methods in Energy, Power, Electric Vehicles, and Their Integration*. Ed. by Kang Li et al. Communications in Computer and Information Science. Singapore: Springer, 2017, pp. 450–457. ISBN: 978-981-10-6364-0. DOI: 10.1007/978-981-10-6364-0_45.

[116] Arunava Naha et al. "Quickest Physical Watermarking-Based Detection of Measurement Replacement Attacks in Networked Control Systems". In: *European Journal of Control* 71 (May 2023), p. 100804. ISSN: 09473580. DOI: 10.1016/j.ejcon.2023.100804. (Visited on 08/11/2024).

[117] Claude E Shannon. "Two-Way Communication Channels". In: *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability* (1961), pp. 611–644.

[118] Dan Simon. *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches.* John Wiley & Sons, 2006.

[119] Andrea A. Molina. “Machine Learning for Intrusion Detection of Cyber-Attacks in Unmanned Aerial Vehicles”. D.Engr. United States – District of Columbia: The George Washington University, 2023. ISBN: 9798380102216. (Visited on 06/20/2024).