# Detection and Isolation of Faults and Cyberattacks in Nonlinear Cyber-Physical Systems using Neural Networks

Bita Afshar

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 (Electrical and Computer Engineering) at

Concordia University

Montréal, Québec, Canada

May, 2023

## ABSTRACT

Detection and Isolation of Faults and Cyberattacks in Nonlinear Cyber-Physical Systems using Neural Networks

Bita Afshar

The theory of Cyber-physical systems (CPSs) has applications in critical infrastructures such as smart grids, manufacturing systems, transportation systems, and autonomous systems such as Unmanned Aerial Vehicles(UAVs). In the CPS, there is a coordination between communication, computation, and control. The communication link in CPS can be subjected to malicious cyberattacks. On the other hand, the physical system in CPS can be faced with different faults such as sensor, actuator, and component faults. Therefore, two significant and challenging problems in CPS can be the detection of faults and cyberattacks. These two threats are intrinsically distinctive and need different strategies to deal with when they occur. This research mainly focuses on providing a methodology to detect and isolate faults and cyberattacks.

This work considers false data injection and replay attacks as security threats. Two different adaptive neural network-based detection methods are proposed in this thesis. These adaptive neural networks are able to detect, isolate, and estimate false data injection, and replay attacks.

Another contribution of this thesis is to provide a scheme for isolating faults and cyberattacks (false data injection and replay attacks) by using virtual sensors on the plant side, which makes the simultaneous detection of faults and false data injection cyberattacks possible.

A nonlinear model of a quadrotor is considered the case study, and the performance of the neural network-based schemes is evaluated through various numerical simulation scenarios.

*To my love*

*Mousa*

## ACKNOWLEDGMENT

I would like first to thank my supervisor Prof. Khashayar Khorasani for his endless patience, support, and motivation.

I want to express my gratitude to my friends and colleagues at Concordia University: Mohammadreza, Mehdi, Parisa, Reza, Rezvan, and Kimia. I am so proud to work with such a hardworking and talented group.

Finally, I express my profound gratitude to my husband, Mousa, for his love and understanding, who helped me through tough times. My gratitude extends to my beloved parents, Shahla and Mahmoud, for always believing in me and being there. My special thanks to my sister, Negar, for her unconditional love and support.

# Table of Contents

# List of Tables

# List of Figures

# List of Abbreviations

| | |
|---|---|
| **CPS** | Cyber-Physical System |
| **UAV** | Unmanned Aerial Vehicle |
| **GCS** | Ground Control Station |
| **CIA** | Confidentiality, Integrity, and Availability |
| **DoS** | Denial of Service |
| **FDI** | Fault Detection, and Isolation |
| **WLS** | Weighted Least Square |
| **NN** | Neural Networks |
| **EKF** | Embedded Kalman Filter |
| **SVM** | Support Vector Machine |
| **FDIR** | Fault Detection, Isolation, and Reconfiguration |
| **UIO** | Unknown Input Observer |
| **IMU** | Inertial Measurement Units |
| **LPV** | Linear Parameter Varying |
| **ANFIS** | Adaptive Neuro-Fuzzy Inference System |
| **AASKF** | Adaptive Augmented State Kalman Filter |
| **CNN** | Convolutional Neural Network |
| **LSTM** | Long Short-Term Memory |

| | |
|---|---|
| **DDoS** | Distributed Denial of Service |
| **LQG** | Linear Quadratic Gaussian |
| **ZDA** | Zero Dynamic Attacks |
| **LOE** | Loss Of Effectiveness |
| **LIP** | Lock-in-Place |
| **HOF** | Hard-Over Fault |
| **CCM** | Cross Channel Monitoring |
| **LQR** | Linear Quadratic Regulator |
| **FBL** | FeedBack Linearization |
| **TP** | True Positive |
| **TN** | True Negative |
| **FP** | False Positive |
| **FN** | False Negative |

# Chapter 1

# INTRODUCTION

Cyber-physical systems (CPS) have three main parts: communication, computation, and control units. A communication network is needed among several components, such as sensors, actuators, and controllers in a standard control system. Communication networks in several modern control systems are wireless, making them more accessible to the cyber world [2]. The communication network can face different issues, such as data packet dropouts, network-induced delays, and cyberattacks, to name a few. The security of CPS depends on the integration of vulnerable cyberspace, and complex physical dynamics [3].

The cyber part of the CPS is vulnerable to malicious attacks due to ease of accessibility and inadequate protection. For instance, adversaries can falsify data packets, redirect the transmission path of data packets, block the communication network, degrade the system performance, destabilize the system, or lead to the loss of control in CPSs [4].

Cyberattacks can harmfully lead to substantial economic loss or endanger human lives. The damage to the system can be under control if the cyberattacks are detected, isolated, and mitigated in time. Therefore, attack detection is the first and vital step to avoiding extensive damage to critical industrial infrastructures in the CPS framework [5]. Consequently, studying cyberattack detection, isolation and mitigation are essential for protecting the CPS against cyberattacks.

## 1.1 Problem Statement

The general objective of this work is to study the security of nonlinear autonomous unmanned systems such as quadrotors under cyberattacks (false data injection and replay attacks) and the isolation of faults and cyberattacks. The quadrotor connecting to the ground control station (GCS) is studied as a case study for this research.

The GCS unit monitors and supervises the operation of the quadrotor and sends the desired trajectory to the quadrotor. In this case, the attacker injects false data into the communication link between the GCS and quadrotor and targets data integrity by altering sensors' data which are sent to the command and control center. For instance, the GCS may have wrong information about the quadrotor's location and may issue incorrect commands.

This work provides a framework to simultaneously detect the faults in the sensors and false data-injected attacks in the communication network using neural network-based diagnosis techniques. Since the dynamic model of the quadrotor is nonlinear, neural networks have been considered as a proper choice.

Two scenarios have been investigated in this work. The first scenario is to detect bias sensor faults and multiplicative false data injection attacks and propose a scheme to isolate the faults and false data injection attacks. Moreover, a recovery methodology is presented based on the estimation of the sensor faults and multiplicative false data injection attacks driven by neural networks.

The second scenario studies the detection of replay attacks in the unmanned aerial vehicle (UAV). In this case, the quadrotor has sensor faults while the adversary applies replay attacks to the CPS system. The isolation of sensor faults and replay attacks is a great challenge in this scenario.

## 1.2 Literature Review

### 1.2.1 Security of CPS

Three main properties in data and IT services are defined for security, including confidentiality, integrity, and availability (CIA) [6]. Confidentiality refers to the non-disclosure of data by unauthorized persons; integrity focuses on the trustworthiness of data which means that unauthorized ones do not change data contents and properties. Availability guarantees timely access to data or system functionalities for tracking the occurring events in the system. The two crucial security objectives are data availability and data integrity in system monitoring and control [6; 7]. Therefore, cyberattacks from the control perspective can be categorized into two main categories:

- Data Availability Attacks: The Denial-of-Service (DoS) refers to a lack of data availability in the system. This attack interrupts the transmission of information from sensors and actuators to their destination. Generally, the DoS attack does not need any model knowledge of the system. However, in some sophisticated cases, the adversary uses model knowledge and additional resources. Therefore, this type of attack leads to data absence and a lack of timely access to data or system functionalities [8; 9].

- Data Integrity Attacks: Deception attacks are a class of attacks that affect data integrity in the system. In most cases, data integrity attacks are more tricky than data availability attacks since the manipulated data is spread through the network. Different types of deception attacks are considered in the literature, including replay attacks, false data injection attacks, covert attacks, and zero dynamics attacks [8; 10; 11]

#### 1.2.1.1 False Data Injection Attack Scenario

False data injection attacks are one class of deception attacks where the adversary manipulates the input or output data by injecting false data through the communication network. This type of attack targets data integrity by altering sensors' and actuators' data. Based on the ability of the attacker to compromise the system, three different scenarios can be considered for this type of attack: the attacker injects false data into the actuators, injects false data into the sensors, or injects false data into the sensors and actuators [12; 8].

There are two essential techniques for securing the CPS against cyber attacks in recent literature, including protecting crucial system components in advance and the next step, identifying the false data injection attack in the system [5].

Several methods can be employed for the first approach, such as redundant components or pathways, cryptography, firewalls, etc. It should be pointed out that above approaches are not adequate against cyberattacks. Thus, the second approach is needed to detect cyberattacks and find a way to remove or modify the manipulated data [5].

These false data-injected attack detection approaches can be categorized into the model- or learning-based methods [5]. Some well-known model-based techniques for cyberattack detections are Bayesian detection considering binary hypothesis [13], weighted least square (WLS) approaches [14], $\chi^2$ -detector based on Kalman filters [15], and quasi-FDI (fault detection and isolation) techniques [16]. Learning-based approaches such as neural networks and machine learning techniques can also detect false data-injected attacks [17].

Each of the model-based and learning-based approaches has advantages and disadvantages. For instance, in model-based approaches, real-time anomaly detection is possible with low computational complexity; however, this detection method is strongly dependent on the system's accurate mathematical model [17; 18].

The learning-based methods effectively detect false data-injected attacks without great knowledge of the physical process model and are appropriate for complex dynamical systems. However, there is no need to formulate complex nonlinear dynamics, but it should be noted that the learning-based algorithms rely on the training process, such as supervised learning or reinforcement learning, and accurate and healthy data are needed to avoid mistraining the system [17; 18].

Although learning-based techniques are utilized for complex systems, they require computationally heavy tasks run online and might not scale well. Moreover, it is more complicated to analyze the system's stability using learning-based techniques. The learning-based algorithms will be more accurate by merging with model-based algorithms, while the computational burden and complexity will decrease [17; 18].

Neural networks (NN) have recently been proposed as a practical approach for detecting false data-injected attacks. NN can learn the mathematical connections between inputs and corresponding outputs [19]. Several approaches, including supervised, semisupervised, classifier and feature space fusion, and online learning algorithms, are applied to detect cyberattacks.

Kravchik et al. in [20] apply convolutional neural networks for detecting cyberattacks on industrial control systems. This work detects anomalies by measuring the difference between the predicted and observed values. In [21], Chen et al. studied reinforcement learning techniques for studying false data injected attacks on power systems with automatic voltage controls assuming that the attacker conducts cyberattacks with data-driven methods.

Foroutan et al. in [17] present a framework for false data injected attacks detection based on machine learning techniques, and a semi-supervised approach is introduced based on a mixture of Gaussian distribution.

One of the problems in the above works is that a proper training data set should be chosen. It means that by using the small training data set, the fault or cyberattacks may not be recognized. On the other hand, the big data set led to significant computation time that may lead to detection delays. Therefore, neural networks have significantly been considered for online anomaly detection [19].

For instance, Abbaspour et al. in [19] proposed a neural network-based fault detection design for quadrotor which is subjected to faults and cyberattacks. In this work, false data injection attacks and faults can be detected in the sensors of the quadrotor. However, this work cannot isolate faults and attacks in the sensor measurements.

Sargolzaei et al. in [18] proposed a mitigation and detection scheme for false data injection attacks in the measurement data. The isolation of faults and cyberattacks is not considered, and the false data injection attack is just considered in the measurement channel.

Farivar et al. in [22] proposed a detection, estimation, and mitigation scheme for cyberattacks launched in the input channel of nonlinear CPS. In this work, attack estimation is done with a neural network and is used for mitigation purposes in the control law.

Abbaspour et al. in [23] proposed a detection method for a false data injection attack in the senor of UAV. The online tunning of the neural network is designed based on an embedded Kalman filter (EKF).

#### 1.2.1.2 Replay Attack Scenario

One kind of deception attack is the replay attack, in which the attacker replaces the real-time measurement with the former recorded data. Two steps are mentioned for this attack: firstly, data is collected, and the system is not disturbed in this phase, and secondly, the recorded data will be replayed in the system by the attacker while the exogenous signal is injected into the system. In this type of attack, the adversary does not need to know the system's model. However, if the attacker has adequate knowledge of the system more damaging attack can be designed [8; 10].

There are several studies considering the security of the CPS against replay attacks. One of the primary approaches to detecting replay attacks is adding an authentication signal to the control signal [24; 25; 26; 27]. However, this approach can lead to the degradation of the control performance.

Dan et al. in [28] proposed a detection strategy against cyberattacks based on the sensor coding information. This paper employs stochastic coding to add the coding signal to the sensor information communication link. This approach can detect replay attacks without degradation of the nominal system performance.

Miao et al. in [29] present noncooperative stochastic games to provide an optimal control policy for balancing the control system performance and replay attack detection rate.

A data-driven approach for replay attack detection is presented by Ma et al. in [30]. This work uses the Support Vector Machine (SVM) as a classifier and a set of self-correlation coefficient data feature space for replay attack detection.

Taheri et al. in [31] present a simultaneous detection method for isolating faults and cyberattacks . In this work, an unknown input observer at the plant side is used for fault detection, and two filters on the plant side and the command and control side are designed for detecting deception attacks, such as covert attacks, zero dynamics attacks, and replay attacks.

Ramadan et al. in [32] proposed an approach for isolating sensor faults and replay attacks by designing two groups of observers, including the Luenberger observer and Unknown Input Observer (UIO), and adding watermarking signals to the system inputs.

### 1.2.2 Faults in CPS

One of the critical and challenging problems in controlling many industrial systems is fault detection, isolation, and reconfiguration (FDIR) [33]. Kanev [34] defined fault as "an unpermitted deviation of at least one characteristic property or parameter of the system from the acceptable/usual/standard condition." Faults can be categorized based on physical locations and resources, like actuator faults, sensor faults, and plant component faults. Due to the massive economic and human losses, efficiently and timely detecting and identifying faults in control systems is essential [35; 34].

The control methodology of the FDIR is to guarantee the safe and satisfactory operation of the system when a fault occurs. In this approach, fault detection and isolation (FDI) distinguishes the fault's occurrence and location. Then the FDI technique will support the controller reconfiguration design [35; 33].

### 1.2.3 Case Study (Unmanned Aerial Vehicle (UAV) in the CPS framework )

Unmanned aerial vehicles (UAVs) are flying entities commonly known for the ability to be remotely controlled or autonomously without a human involved [36]. UAVs have been increasingly utilized in civilian and military applications, and it is crucial to consider the impact of cyberattacks and faults on their desired performance. In UAV design, the integration of computation, control, and communication components exists like other cyber-physical systems (CPS).

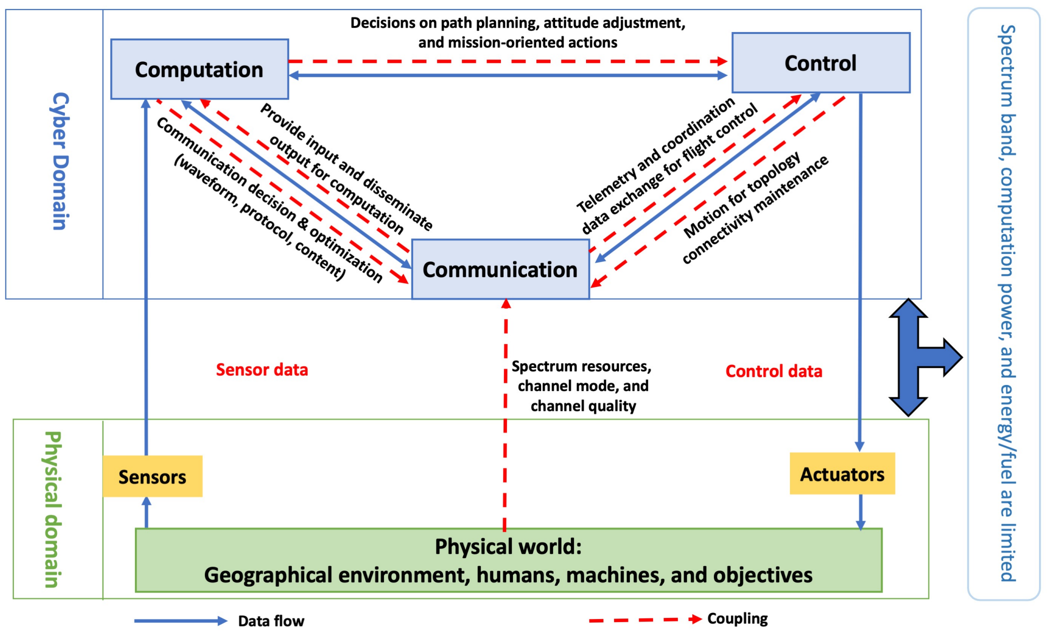
Cyber Domain
Computation
Decisions on path planning, attitude adjustment, and mission-oriented actions
Control
Provide input and disseminate output for computation
Communication decision & optimization (waveform, protocol, content)
Telemetry and coordination data exchange for flight control
Motion for topology connectivity maintenance
Communication
Sensor data
Spectrum resources, channel mode, and channel quality
Control data
Physical domain
Sensors
Actuators
Physical world: Geographical environment, humans, machines, and objectives
Data flow
Coupling
Spectrum band, computation power, and energy/fuel are limited

Figure 1.1: The architecture of UAVs in a cyber-physical system perspective [37] .

In Figure 1.1, the cyber and physical domains of a typical UAV is illustrated. It is shown that the cyber layer consists of computation, communication, control modules, and the physical layer elements, including onboard sensors and actuators. The sensors transfer the data from the physical world to the cyber domain through the communication channel. The computation module analyzes the data and decides based on the gathered information. Finally, the control units send the decision commands to the actuator, which impacts the physical world [36; 38; 37].

Figure 1.1 shows the data flows and coupling effects within the components. The stated coupling effects show the bilateral influences and dependency between the cyber and physical layer components. Single UAVs are generally employed in simple and low-scale missions, and the communication network is considered between the UAV and the ground control station (GCS) through uplinks and downlinks. This framework's computation modules are responsible for task allocation, path planning, and navigation based on optimization and intelligent algorithms. The control module is accountable for flight control, and ensuring the mission's fulfillment [36; 38; 37].

The swarm of UAVs, a coordinated unit to complete the desired task, is utilized for more significant, complex, and systematic tasks. Each agent should cooperate reasonably with other agents to complete the task with efficient coordination in this architecture. The communication links here are UAV-GCS and UAV-UAV. UAVs can directly communicate with each other or through multi-hop links. The computation modules are utilized for joint path planning and task allocation. The control modules should handle cooperative control, collision avoidance, and formation control [36; 38; 37].

As a cyber-physical system, a UAV includes several components, such as sensors, communications, computation, and control modules. These elements may be subject to cyberattacks, resulting in undesired states, system performance degradation, and significant failures [36; 38; 37].

#### 1.2.3.1 Fault Diagnosis of Unmanned Aerial Vehicles

UAVs have been widely used in various critical applications over the last few years. Therefore, they should operate reliably with minimum faults. Unfortunately, the presence of faults in the most advanced technologies is unavoidable. UAVs have different components, such as sensors and actuators, which are vulnerable to malfunction. Moreover, unexpected circumstances and operational environments can lead to failure in physical components. Thus, it is vital to design and employ fault diagnosis methods to detect and isolate fault occurrences in the UAVs' sensors and actuators [39].

- UAV Sensor Faults:

  UAV sensor fault occurres while receiving incorrect measurement data from the UAVs' sensors. For instance, inertial measurement units (IMU) consisting of three-axis gyroscopes, an accelerometer, and a magnetometer have a crucial role in controlling the quadcopters. Therefore, detecting and isolating sensor faults is essential to safe flight operations, accomplishing scheduled missions, and avoiding vehicle crashes [39; 40; 41].

Different approaches, such as hardware redundancy, model-based, and knowledge-based methods, are applied for FDI in UAVs' sensors. Drak et al. in [42] and Shi et al. in [43] use a hardware redundancy approach for the altitude sensors' measurements.

Lopez-Estrada et al. [44] modeled the quadrotor as a Linear Parameter Varying (LPV) system. The bank of robust LPV observers detects and isolates sensor faults. In [45], a sliding-mode observer is used to estimate roll and pitch angles in order to detect, isolate, and estimate sensor bias faults in the gyroscope and accelerometer measurements.

In [46], Hansen et al. present parameter adaptive estimators for fault diagnosis of UAV airspeed sensors. This work employs an extended Kalman filter to estimate the wind velocity vector. One of the knowledge-based approaches is applied in the work of Guo et al. [47], which is an optimized one-class support vector machine approach regulated by local density for UAV sensor fault detection. In the work of Sun et al. [48], a novel data-driven Adaptive Neuro-Fuzzy Inference System (ANFIS)-based approach is introduced to detect onboard navigation sensor faults in UAVs, which is a fast and accurate fault detection method for real-time applications.

- UAV Actuator Faults:

  Actuators have the responsibility for converting control signals into mechanical motions in UAVs. Faults and malfunctions in the actuator of the UAV lead to flight problems and crashes. Thus, it is crucial to detect and isolate faults in actuators [39].

Different types of FDI approaches, such as hardware redundancy, model-based, and knowledge-based, are utilized for fault detection in UAV actuators. For instance, a robust actuator fault detection and diagnosis architecture is presented in the work of Zhong et al. [49]. In this work, an adaptive augmented state Kalman filter (AASKF) is presented to detect and estimate actuator faults, while the dynamics model of actuator faults and disturbances are not completely available.

In the work of Avram et al. [50], a bank of nonlinear adaptive estimators is presented for fault detection and isolation in the quadrotors' actuators. In order to improve fault detection and isolation, nonlinear adaptive thresholds are designed. A deep-learning-based method for fault diagnosis in the UAV actuators is proposed in [51]. This work applies a hybrid Convolutional Neural Network and Long Short-Term Memory (CNN-LSTM) technique for actuator fault diagnosis in the six-rotor UAVs by extracting features and finding internal connections.

#### 1.2.3.2 Cybersecurity of Unmanned Aerial Vehicles

UAVs as cyber-physical systems (CPS) consist of several components such as sensors, communications, computation, and control elements. Each component is vulnerable to cyber-attack, leading to physical system malfunctions and undesired states. Since the communication links in the UAVs are wireless, the adversary can easily falsify and manipulate control and sensor data through cyberattacks, leading to the loss of UAVs'control [37; 52].

Typically, two kinds of cyberattacks can be considered for UAVs, including data availability attacks such as Denial of Service (DoS) attacks and deception attacks, which impact data integrity such as false data injection attacks, replay attacks, etc [53].

- UAVs under Denial of Service (DoS) Attacks:

  In this type of attack, the adversary tries to stop legitimate users from accessing the desired resources. As shown in Figure 1.2, the adversary prevents users from connecting with the access points [54; 55]. Distributed Denial of Service attack (DDoS) is an upgraded version of a DoS attack in which the attacker utilizes multiple sources to flood packets. A DDoS attack on the UAV transceiver or the control center that works with the UAV communication channel can lead to loss and damaged sensitive information packets [56].

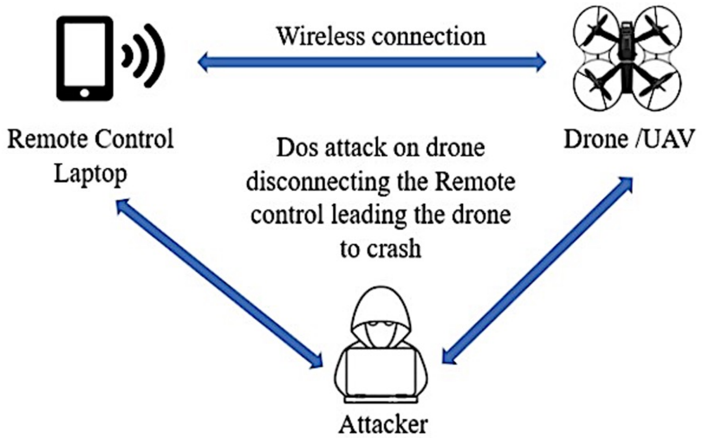
Wireless connection
Remote Control Laptop
Drone /UAV
Dos attack on drone disconnecting the Remote control leading the drone to crash
Attacker

Figure 1.2: DoS attack on the UAV [55] .

Chen et al. in [57] present a defense mechanism against DoS attacks for three crucial system components: CPU, memory, and communication channel. In this work, a software framework provided DoS attack resilient control for real-time UAVs. Ouiazzane et al. in [58] designed a model of DoS intrusion detection systems for UAV networks by using a multiagent system and applying the machine learning decision tree algorithm. It is shown that the proposed method effectively detects known and unknown DoS attacks in the UAV network. In the work of Garg et al. in [59], a tree-based attack-defense model and a game-theoretic approach are proposed for the multi-UAV network with the case study of DDoS attacks.

- UAVs under Deception Attacks:

  In deception attacks, the adversary tampers with the information intentionally. In Gu et al. [53] work, a UAV safety problem under the false data injection attack is investigated. In this paper, the attack characteristics and influencing factors are calculated, and with this regard, the false data injected attack model is obtained. In the next step, the attack detection mechanism is proposed, and finally, the effect of false data injected attack on the control system will be compensated.

  Lin et al. [60] designed a framework to detect false data-injected attacks in UAVs. A Linear Quadratic Gaussian (LQG) controller has been designed in this framework to provide sufficient conditions to guarantee the security of the closed-loop system.

  Abbaspour et al.[23] proposed an adaptive neural network to detect false data-injected attacks in UAV sensors. This work uses a neural network observer to detect possible attacks on UAV IMU sensors online. Abbaspour et al. [19] studied false data injected attacks and natural faults on the sensors of the UAV. This paper uses the neural network-based fault detection method to estimate the real-time faults and false data injection attacks on the quadrotors' sensors.

  Bahrami et al. in [61] proposed a model-based centralized and decentralized observer approach to detect stealthy attacks such as zero-dynamics and covert attacks on the network of UAVs through switching communication topology.

## 1.3 Thesis Contributions

In this section, the main thesis contributions are presented and briefly explained. To the best of the author's knowledge, the isolation of faults and cyberattacks is an open area in the cyber security research domain, particularly in nonlinear control systems. A fault detection strategy for sensors of a nonlinear underactuated quadrotor is presented. A neural network-based detection strategy is employed to detect sensor faults in the nonlinear underactuated quadrotor. The detection approach is based on Talebi et al. [62], which is also used in Abbaspour et al.[19] for sensor fault and cyberattack detection. The main contribution of this work is considering the multiplicative false data injection attacks as well as the isolation of faults and false data injection attacks in the quadrotor.

This observer can detect faults in one or more sensors of the quadrotors, and it is placed on the plant side. Meanwhile, another neural network-based detector is designed to detect multiplicative false data injection attacks in the communication link. This detector is placed in the command and control center and can detect different types of false data injection attacks, such as multiplicative constant cyberattacks or multiplicative time-variant nonlinear cyberattacks with periodic behavior. The false data injection attack can impact sensor measurement data and command reference signal sent from command and control to the quadrotor through a communication channel. Considering the multiplicative false data injection attack and presenting a neural network-based detector is one of the contributions of this work. The significant improvement in this work is providing an isolation scheme for faults and false data injection attacks for the nonlinear CPS. Moreover, the effects of faults and false data injection attacks are mitigated by using virtual sensors. Therefore, simultaneous detection of faults and false data injection attacks on the communication channels is possible in this work.

Although the above two filters are well designed for detecting sensor faults and false data injection attacks in the case of replay attacks, these neural network-based filters are inefficient since they cannot detect, isolate and estimate the injected attack into the control signal. Therefore, a new neural network-based detection strategy is presented, which can detect replay attacks in the nonlinear quadrotor. The idea of designing a neural network-based detector for detecting false data injection attacks on the CPS input comes from the work of Talebi et al. [63]. In that work, a neural network-based detector for the detection of multiplicative actuator faults is proposed. In this work, a new detection scheme is proposed to detect additive false data injection attacks on the control input of the CPS. This new detection scheme can also detect replay attacks. More importantly, the replay attack and sensor faults will be isolated by using virtual sensors.

## 1.4 Thesis Outline

This dissertation consists of five chapters, as described as follows. Chapter one briefly introduces the cyberattacks and fault detection problems in CPSs. The UAV as a case study is further investigated. The UAV in the CPS framework is studied from the point of view of cybersecurity and fault detection. The problem statement and contributions have also been addressed in this section.

The second chapter introduces different types of cyberattacks in the CPS, as well as detection methods. Some background information on different fault detection approaches has been provided. Finally, the mathematical model of the nonlinear quadrotor is considered. In this chapter, the kinematic and dynamic equations, as well as controller design, are presented.

Chapter three first introduces a new neural network-based approach to detect multiplicative false data injection attacks and is placed on the command and control center. The neural network can also estimate false data injection attacks and can be used in the virtual sensor for mitigation purposes. Moreover, a fault detection approach based on neural networks-based detectors is considered for quadrotor sensor fault detection. This approach also estimates the faults that can be used in virtual sensors to isolate sensor faults.

Chapter four presents a neural network-based strategy that can detect false data injection attacks on the input control of CPS and replay attacks. Two neural network-based detectors are designed and considered to isolate faults and replay cyberattacks.

Chapter five is assigned to the thesis conclusion and some suggestions for the open problems to be considered for future works.

# Chapter 2

# BACKGROUND INFORMATION

This chapter provides a brief introduction regarding various types of cyberattacks and faults in the control system. Moreover, the derivation of the quadrotor model as the case study is reviewed. Section 2.1 explains the possibility of cyberattacks on the communication channel and describes different types of cyberattacks in CPS from the control perspective. Section 2.2 provides a short review of fault modeling for the control system and introduces fault diagnosis approaches. In Section 2.3, the movement and dynamics of the quadrotor are provided. In order to derive the dynamic equations of quadrotor motion, the Newton-Euler equations are applied. Section 2.4 introduces the nonlinear control approach for quadrotor hovering flight.

## 2.1 Cyberattacks Definitions

Cyber-physical systems are vulnerable to various types of cyberattacks, and Figure 2.1 illustrates different types of attacks. This figure introduces four types of attacks: Physical attack, deception attack, disclosure attack, and disruption attack.

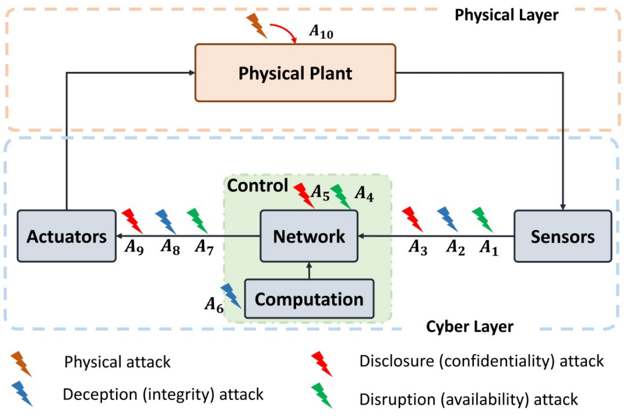

Figure 2.1: A schematic of various attacks in a CPS [64].

Disclosure attacks are trying to access informative information. Deception attacks refer to intentional tampering with information. Disruption attacks describe the lack of sensor or control data availability, making it impossible for the subsystems and their components to communicate and transfer data [64]. Five types of cyberattacks can be categorized in the attack space, as shown in Figure 2.2.

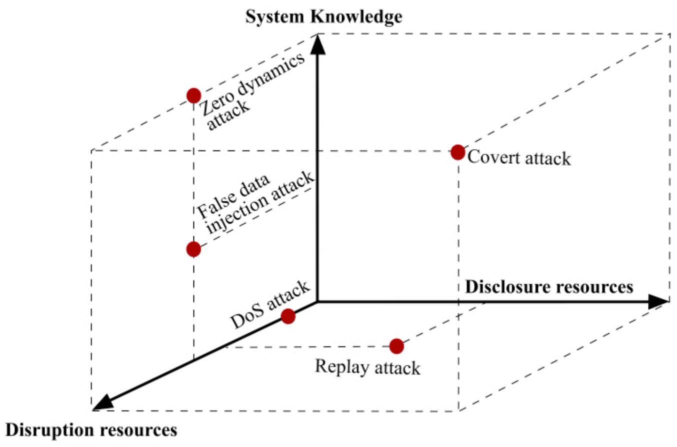

Figure 2.2: Five types of cyberattacks in the attack space [8].

The nonlinear CPS model can be represented as:

$$\begin{cases} \dot{x}(t) = f\left(x(t), u(t)\right), \\ y(t) = Cx(t), \end{cases} \tag{2.1}$$

where $x \in \mathbb{R}^n$ is the state variable, $u \in \mathbb{R}^m$ is the input, $y \in \mathbb{R}^q$ is the output. $C$ denotes a constant measurement matrix.

### 2.1.1 Denial-of-Service (DoS) Attack

The Denial-of-Service (DoS) refers to a lack of data availability in the system. This attack interrupts the transmission of information from sensors and actuators to their destination[8; 9]. Figure 2.3 shows an example of a DoS attack in which the adversary prevents data transmission between controller and plant.

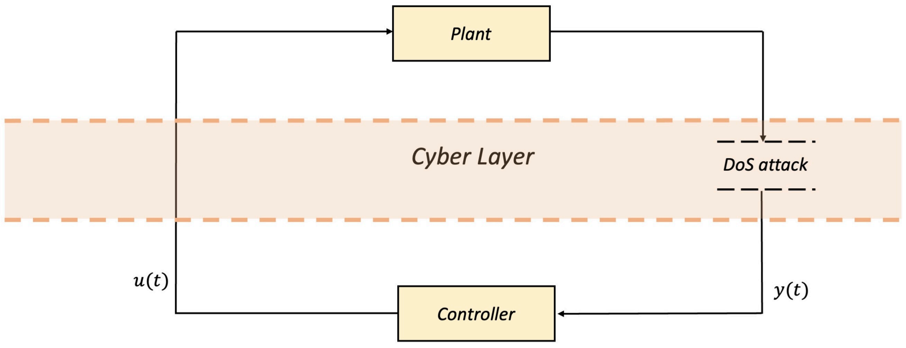

Figure 2.3: DoS attack scenario.

The DoS attack can be modeled in different ways, such as Bernoulli, Markov, and Queuing models [8]. One of the models which are considered in [65] for defining DoS attack is presented as follows:

$$\begin{cases} \dot{x}(t) = f\left(x(t), u\right), \\ y(t) = \beta(t) C x(t), \end{cases} \tag{2.2}$$

where $\beta(t)$ is used to show the impact of DoS attack, when the system is under DoS attack $\beta(t) = 0$, and otherwise $\beta(t) = 1$.

## 2.1.2 False Data Injection Attack

A false data injection attack targets data integrity by injecting false data into the input or output channels of the CPS and altering sensors and actuators data [8]. The false data can be injected into the cyber layer, as shown in Figure 2.4.

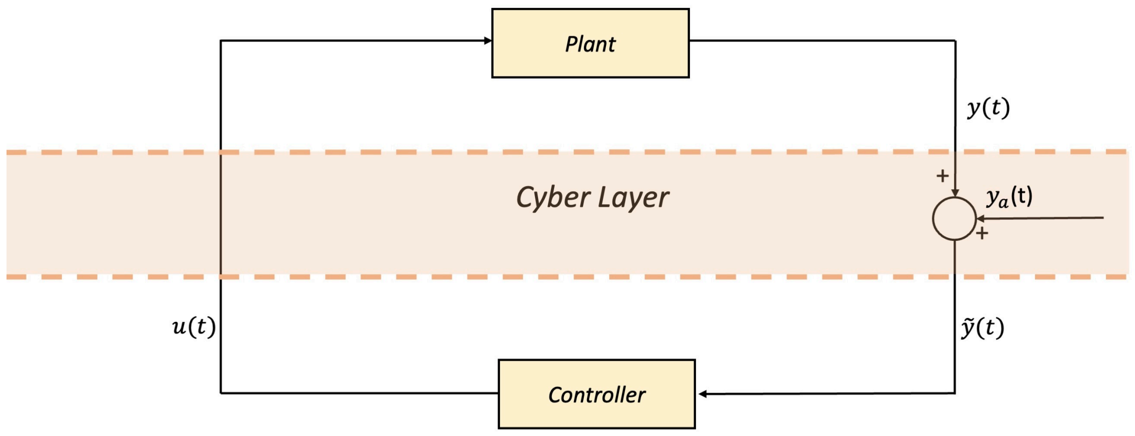

Figure 2.4: False Data Injection attack in CPS.

In the case that sensor measurements are manipulated, the system can be described as [8]:

$$\begin{cases} \dot{x}(t) = f\left(x(t), u(t)\right), \\ \tilde{y}(t) = Cx(t) + y_a(t), \end{cases} \tag{2.3}$$

where $y_a(t)$ is the false data that is added to the original measurement signal by the attacker. $\tilde{y}(t)$ is the manipulated signal received in the command and control center.

## 2.1.3 Covert Attack

A covert attack is another form of deception attack. In this type of attack, an adversary manipulates the control input signal, but its impact on the sensor output signal is compensated. Therefore, the attacker can gain access to the measurement and actuator signals in the CPS.

Since this attack is undetectable by the monitoring system, it can cause damage to the plant [66]. This type of attack requires system knowledge, as shown in Figure 2.2. The covert attack is shown in Figure 2.5.

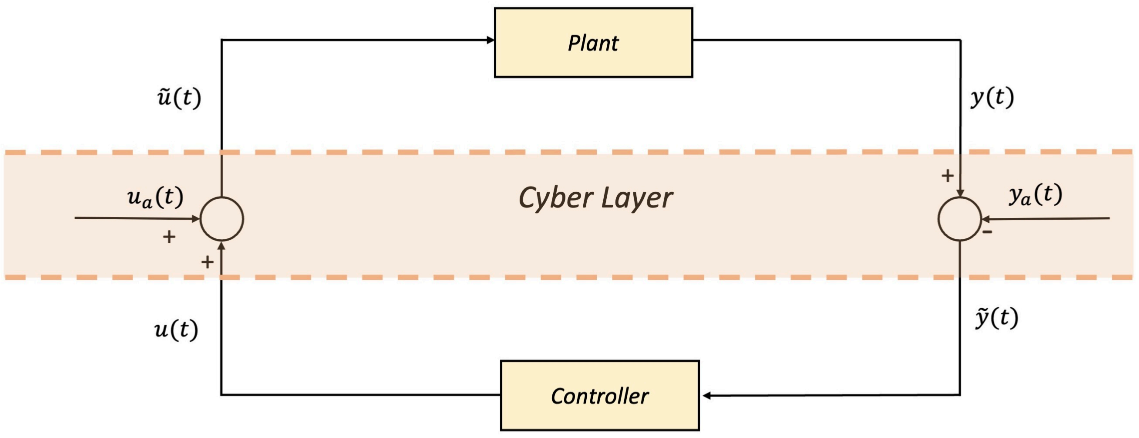

Figure 2.5: Covert attack in CPS.

The CPS under covert attack can be described as [66]:

$$\begin{cases} \dot{x}(t) = f\left(x(t), \tilde{u}(t)\right), \\ y(t) = Cx(t), \\ \tilde{u}(t) = u(t) + u(a), \\ \tilde{y}(t) = y(t) - y_a(t), \end{cases} \tag{2.4}$$

where $u_a(t)$ is the attack signal which is designed by the adversary and injected into the control input while simultaneously its effect on the measurement signal is compensated by the $y_a(t)$ [66].

### 2.1.4 Replay Attack

A replay attack is a kind of deception attack which includes two phases of recording sensor measurement data and replacing it with actual data while tampering with the input channel. In this attack, it is assumed that the system is under a constant or periodic steady state [67].

The replay attack can be described as follows [68; 67]:

• First Stage:

The attacker records all sensor measurement data from time $t_0$ until $t_0 + w$ where $w$ is the time frame that the attack happens. The first step of the replay attack is shown in Figure 2.6.

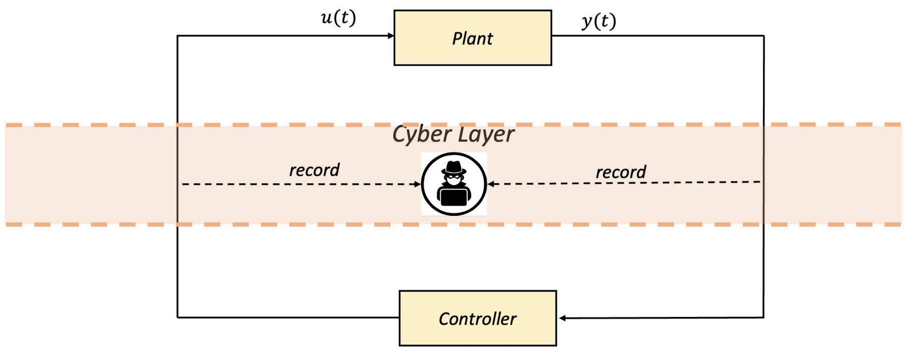

Figure 2.6: Schematic of replay attack- first phase.

The nonlinear CPS in the first phase of replay attack can be described as [68; 67]:

$$\begin{cases} \dot{x}(t) = f\left(x(t), u(t)\right), \\ y(t) = Cx(t), \end{cases} \quad t_0 \le t \le t_0 + w. \tag{2.5}$$

• Second Stage:

At time $t_1$ the adversary replays the recorded sensor output data while simultaneously changing the control inputs.The second step of the replay attack is shown in Figure 2.7.

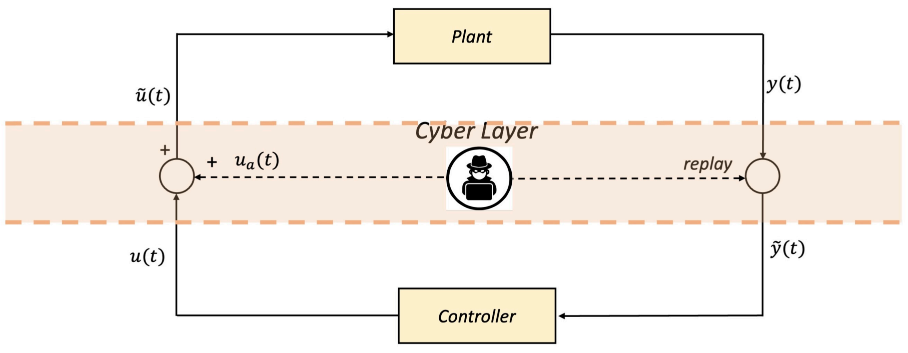

Figure 2.7: Schematic of replay attack- second phase.

The nonlinear CPS in the second phase of replay attack can be described as [68; 67]:

$$\begin{cases} \dot{x}(t) = f\left(x(t), \tilde{u}(t)\right), \\ \tilde{u}(t) = u(t) + u_a, \\ \tilde{y}(t) = y(t + t_0 - t_1 - (N_f - 1)\,w), \end{cases} \quad t_1 + (N_f - 1)\,w \le t \le t_1 + N_f\,w, \quad N_f \in \mathbb{N}, N_f \ge 1. \tag{2.6}$$

Figure 2.8 shows how the replay attack can affect the sensor measurement. The attacker collects the actual sensor data in the time interval of $[t_0, t_0 + w]$ and then replaces this data in $[t_1, t_1 + w]$. While the adversary replays the collected data, the control signal is also manipulated. Since the system output shows the expected steady state, the attack will not be recognizable [69].

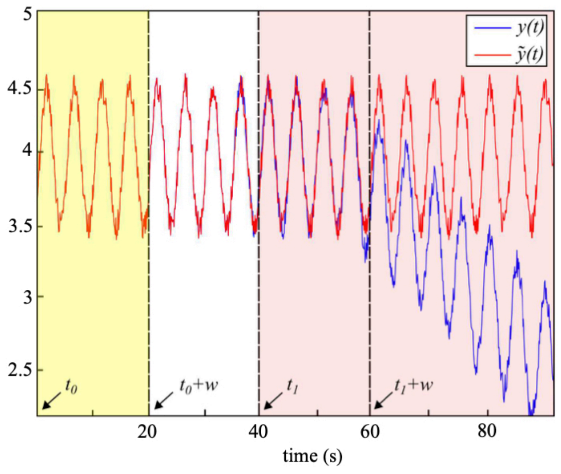

Figure 2.8: Example of replay attack effect on sensor measurement [67].

### 2.1.5 Zero Dynamics Attacks

Zero dynamic attacks (ZDA) employ invariant zeros of the system, making the attack undetectable, and there is no output for the attack signal. This type of attack needs the perfect knowledge of the plant dynamics. If the adversary has comprehensive knowledge of the system model and the plant has non-minimum phase zeros, unstable modes can be launched in the state response without being represented by any detectors [8]. ZDA for nonlinear control-affine systems is defined in [70]. In this work, it assumed that some combination of inputs and outputs are changed. The nonlinear control-affine system under ZDA can be presented as follows [70]:

$$\begin{cases} \dot{x}(t) = f(x(t)) + g(x(t))\tilde{u}(t), \\ y(t) = Cx(t), \\ \dot{x}_a(t) = a_y, \\ \tilde{u}(t) = u(t) + a_u, \\ \tilde{y}(t) = y(t) - x_a(t), \end{cases} \tag{2.7}$$

where $x_a$ is the attack state, $a_y$ represents the output attack signal, and $a_u$ denotes the input attack signal. The block diagram of ZDA attacks is presented in Figure 2.9.

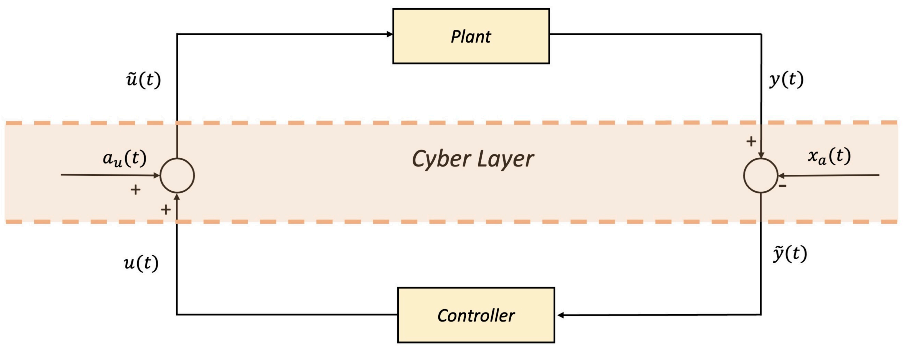

Figure 2.9: Zero Dynamics attacks in CPS.

## 2.2 Fault Modeling

Faults can be classified according to their physical locations and resources. They can be defined as actuator, sensor, and plant component faults as shown in Figure 2.10. In this section, faults in different components of the system will be introduced.

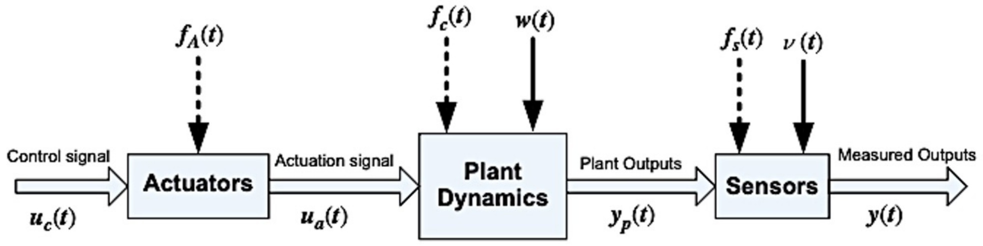

Figure 2.10: Location of potential faults in a control system [35] .

### 2.2.1 Sensor Faults

Sensors transmit information about the system's behaviour and internal states. Sensors malfunctioning can lead to abnormal changes in the measurement process. For that reason, faults in sensors may result in significant performance degradation, especially in decision-making systems or processes such as feedback control systems, safety control systems, quality control systems, etc. Typical sensor faults are (a) bias; (b) drift; (c) performance degradation (or loss of accuracy); (d) sensor freezing; and (e) calibration error [35; 71; 69] which are depicted in Figure 2.11 .

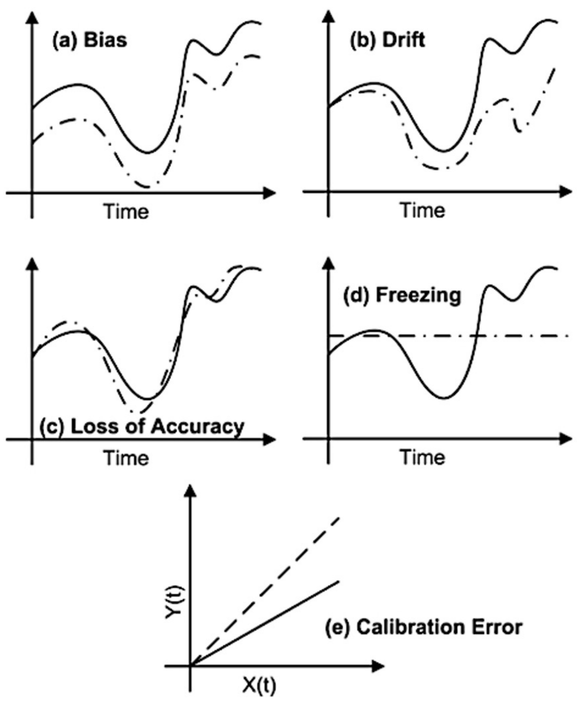

Figure 2.11: The common types of sensor faults. The dotted lines show the desired value of the sensor, and the solid lines show the actual value. a) bias; (b) drift; (c) performance degradation (or loss of accuracy); (d) sensor freezing; and (e) calibration error [35] .

Generally, faults can be described as an additive or multiplicative term regarding the dynamics of the system [72]. The nonlinear continuous-time system with additive faults can be represented in the general form as:

$$\begin{cases} \dot{x}(t) = f\left(x(t), u(t)\right), \\ y(t) = Cx(t) + f_s(t), \end{cases} \tag{2.8}$$

where $x \in \mathbb{R}^n$ is the state variable, $u \in \mathbb{R}^m$ is the input, $y \in \mathbb{R}^q$ is the output , $f$ is the known nonlinear function, $C$ is a constant measurement matrix, and $f_s(t) \in \mathbb{R}^q$ is an additive sensor fault. Different types of additive sensor faults are mathematically represented in the following [35]:

$$f_{si}(t) = \begin{cases} b_i(t) & \dot{b}_i(t) = 0, b_i(t_{Fi}) \neq 0, & & Bias, \\ b_i(t) & |b_i(t)| = c_i(t), 0 \leq c_i \ll 1, & \forall t \geq t_{Fi}, & Drift, \\ b_i(t) & |b_i(t)| \leq \bar{b}_i, \dot{b}_i(t) \in L^{\infty}, & \forall t \geq t_{Fi}, & Loss \quad of \quad accuracy, \end{cases} \tag{2.9}$$

where $t_{Fi}$ represents the time that a fault has occurred on the $i^{th}$ sensor and $b_i$ is the accuracy coefficient such that $b_i \in [-\bar{b}_i, \bar{b}_i]$ where $\bar{b}_i > 0$.

The nonlinear continuous-time system with a sensor freezing fault can be represented as [35]:

$$\begin{cases} \dot{x}(t) = f\left(x(t), u(t)\right), \\ y(t) = Cx(t), \\ y_i(t) = C_i x_i(t_{Fi}), & \forall t \geq t_{Fi}, \quad Sensor \quad freezing, \end{cases} \tag{2.10}$$

where $t_{Fi}$ represents the time that a fault has occurred on the $i^{th}$ sensor, and $C_i$ is the $i^{th}$ row of matrix C.

The nonlinear continuous-time system with a sensor calibration error can be represented as [35]:

$$\begin{cases} \dot{x}(t) = f\left(x(t), u(t)\right), \\ y(t) = Cx(t), \\ y_i(t) = k_i(t) x_i(t), \quad 0 < \bar{k} \leq k_i(t) \leq 1, & \forall t \geq t_{Fi}, \quad Calibration \quad error, \end{cases} \tag{2.11}$$

where $t_{Fi}$ represents the time that a fault has occurred on the $i^{th}$ sensor. The minimum sensor effectiveness is shown by $\bar{k}_i > 0$ , $k_i \in [\bar{k}_i, 1]$.

### 2.2.2 Actuator Faults

Faults in the actuator describe abnormal changes in the actuator. Actuators are applied to convert the control signal into mechanical motions. Actuator faults can adversely affect system performance and may lead to complete system failure. Since actuators are typically expensive, it is challenging to consider extra hardware redundancy. Actuator faults are usually contingent on the type of the actuator [35; 71; 69]; some actuator faults are illustrated in Figure 2.12 .

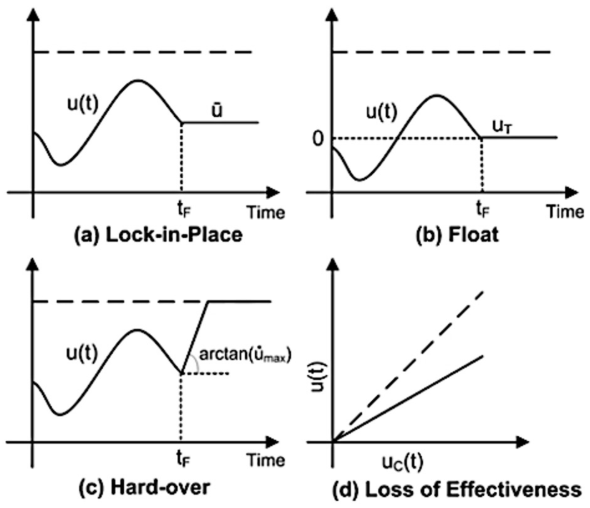

Figure 2.12: The common types of actuator faults. Dotted lines show the desired value of the actuator, and the solid lines show the actual value. (a) Lock-in-place (b) Float (c) Hard-over (d) Loss of effectiveness [35] .

The nonlinear continuous-time system with a actuator fault can be represented as:

$$\begin{cases} \dot{x}(t) = f\left(x(t), u_a(t)\right), \\ y(t) = Cx(t), \end{cases} \tag{2.12}$$

where $x \in \mathbb{R}^n$ is the state variable, $u_a \in \mathbb{R}^m$ is the actuator output, $y \in \mathbb{R}^q$ is the output , $f$ is the known nonlinear function, $C$ is a constant measurement matrix. Different types of actuator faults are mathematically represented in the following [35]:

$$u_a^i(t) = \begin{cases} u_c^i(t), & & & No \quad Faults, \\ k_i(t)u_c^i(t), & 0 < \epsilon_i \le k_i(t) < 1, & \forall t \ge t_{Fi}, & (LOE), \\ 0, & & \forall t \ge t_{Fi}, & (Float), \\ u_c^i(t_{Fi}), & & \forall t \ge t_{Fi}, & (LIP), \\ u_{imin} \vee u_{imax}, & & \forall t \ge t_{Fi}, & (HOF), \end{cases} \tag{2.13}$$

where $u_c^i$ represent the actuator input to the $i^{th}$ actuator, $t_{Fi}$ represents the time that a fault has occurred on the $i^{th}$ actuator, and the actuator effectiveness is shown by $\bar{k}_i > 0$ , $k_i \in [\bar{k}_i, 1]$, $\epsilon_i > 0$. The lower and upper actuation level limits of the $i^{th}$ actuator is shown by $u_{imin}$ and $u_{imax}$.

### 2.2.3 Components Faults

Plant component faults may result in changes in the nominal dynamic relationship between the variables of the system. Some physical parameter changes in the system, such as resistance, inductance, and amplifier gain, may lead to these faults. Since faults in components could lead to severe catastrophic consequences, it is essential to diagnose these faults from the very early stages [35; 71; 69]. The components faults can represented as follows:

$$\begin{cases} \dot{x}(t) = f(x(t), u(t)) + f_c(t), \\ y(t) = Cx(t), \end{cases} \tag{2.14}$$

where $f_c(t) \in \mathbb{R}^n$ is the plant component faults.

### 2.2.4 Fault Diagnosis

Fault diagnosis methods enhance the system's reliability by monitoring, locating, and identifying faults. FDI techniques employ the idea of redundancy, either hardware redundancy or analytical redundancy, as shown in Figure 2.13 [33; 73].

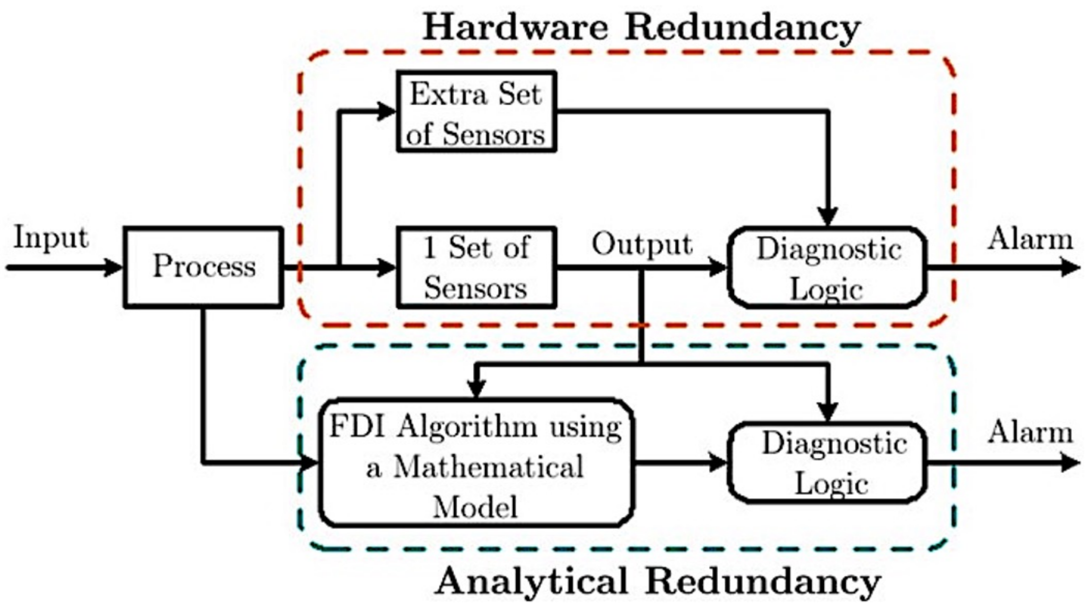

Figure 2.13: The concepts of hardware redundancy and analytical redundancy [33] .

- Hardware Redundancy:

  The main idea of hardware redundancy is to employ the same components with the same input signal. The output of the replicated system will be compared to the plant output signal to distinguish the occurrence of the fault. However, hardware redundancy is a reliable method, but it is an expensive option that needs more resource and space. Therefore, hardware redundancy approach can be applied for critical components in the system, but it is not appropriate for the entire plant because of the cost and constrained weight and space. Some of the standard techniques in the hardware redundancy approach are the cross channel monitoring (CCM) method, residual generation using parity generation, and signal processing methods such as wavelet transformation [33; 71; 73].

- Analytical Redundancy:

  Analytical redundancy is another method for FDI that utilizes the system's mathematical model and estimation approaches. This method is not as expensive as the hardware redundancy approach since additional components are not required. However, this approach can be more complicated by considering model uncertainties, noise, and unknown disturbances. The analytical redundancy method can be categorized into two leading groups: quantitative and qualitative methods [33]. A general framework of the analytical redundancy approach for FDI is depicted in Figure 2.14.

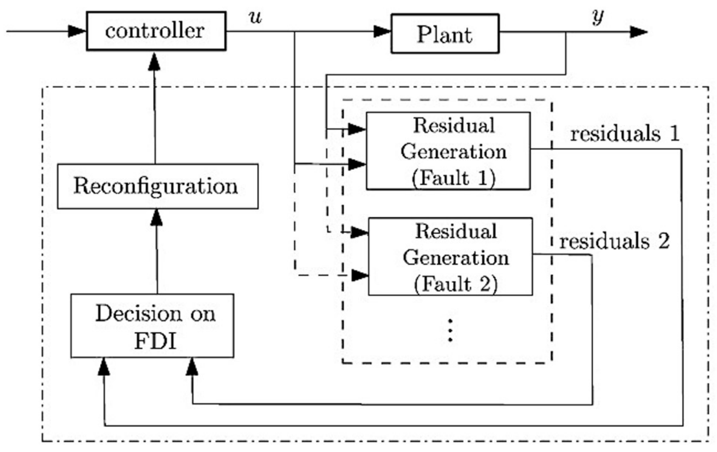

Figure 2.14: A general framework of the analytical redundancy approach for FDI [33] .

Three steps are considered for FDIR, as shown in Figure 2.14. In the first step, certain variables identified as residuals are generated using one or more generation filters. While there is no fault condition in the system, the residual should be zero. In order to do fault isolation in some FDIR frameworks, two or more residual generation filters are used in parallel. Each residual generation filter is responsive only to this framework's particular set of faults. The second step is to decide fault occurrence (fault detection) and the fault types (fault isolation) based on the generated residual. Statistical tools can help test if the generated residual differs considerably from zero. Lastly, in the third step, the controller will be reconfigured online in response to detected faults [33]. The different classifications of FDIR approaches are presented in Figure 2.15.

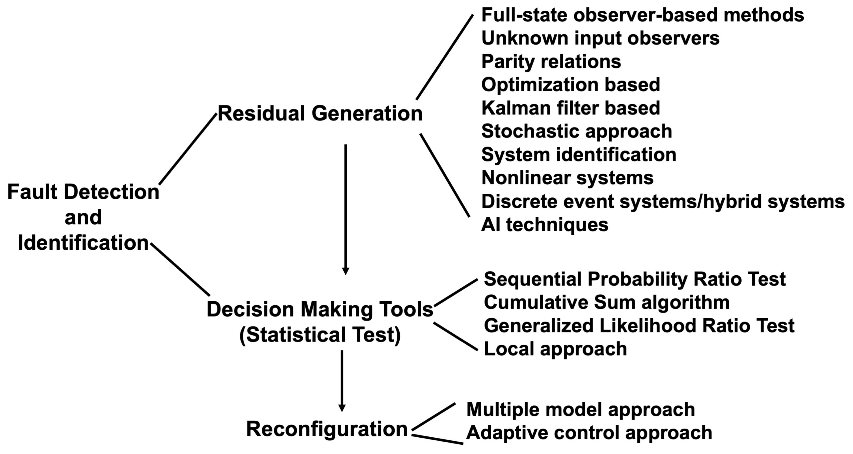

Figure 2.15: Classifications of FDIR methods [33] .

## 2.3 Mathematical Model of Quadrotor

There are two different configurations for the quadrotor, "×" and "+". The "×" configuration is admitted as a more stable configuration than the "+," which is more agile [74]. The two configurations are depicted in Figure 2.16.

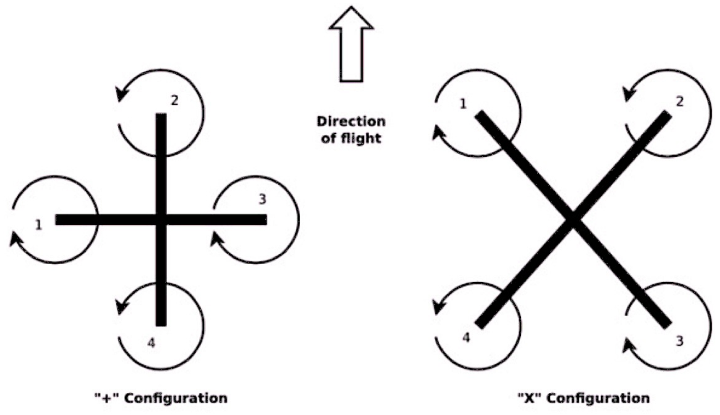

Figure 2.16: Quadrotor configurations [74] .

The quadrotor is an underactuated system with six degrees of freedom and four rotors as four control inputs. Nevertheless, it is possible to control the quadrotor by choosing four proper controllable variables. It is assumed that propellers rotate in the fixed and parallel axes, and the structure of the propellers is relatively rigid. Four dc motors are placed on each propeller. Therefore, by adjusting the speed of each rotor, the required trust and torques for the quadrotor movement and balance are produced [75; 1].

The model of the quadrotor is illustrated in Figure 2.17. The quadrotor thrusts are represented as $T_1, T_2, T_3, T_4$ Moreover, to maintain quadrotor balance, the front and rear propellers' rotation is counter-clockwise while clockwise in the right and left propellers. The propellers' rotation speeds are shown by $\Omega_i (rad/s)$ and the rotation direction is shown by a rotated curved arrow line, and a vertical line shows the thrust generated by each propeller.

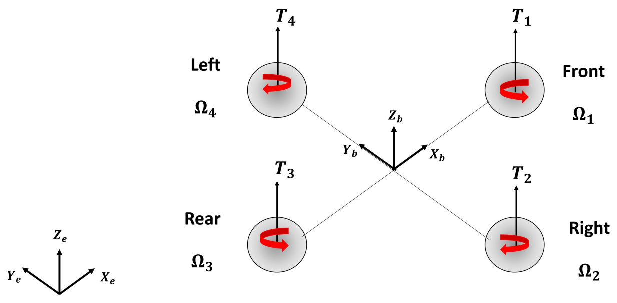

Figure 2.17: The quadrotor schematic .

In order to reach the desired position and move the quadrotor, the propellers' speeds are appropriately changed. For instance, angular speed differences between two sets of rotors lead to vertical rotation. The quadrotor can move upward and downward by simultaneously increasing or decreasing all the rotors' speeds [1]. The dynamic model of the quadrotor is defined with two reference frames, i.e., body-fixed and earth-fixed. In order to define the linear and angular position of the quadrotor, the earth-fixed frame is considered. However, the linear velocity and angular velocity, forces, and torques are determined in the body-fixed frame [75; 1]. The linear position and angular position in the earth-fixed frame are defined as [1]:

$$\begin{aligned} r^E &= [x, y, z]^T, \\ \Theta^E &= [\phi, \theta, \psi]^T. \end{aligned} \tag{2.15}$$

The linear and angular velocities in the body-fixed frame are defined as [1]:

$$\begin{aligned} V^B &= [u, v, \omega]^T, \\ \omega^B &= [p, q, r]^T. \end{aligned} \tag{2.16}$$

The rotation matrix $R$ defines the rotation from the body to the earth frame, that is [1]:

$$R = \begin{bmatrix} \cos\psi\cos\theta & \cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi & \cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi \\ \sin\psi\cos\theta & \sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi & \sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix}. \tag{2.17}$$

The angular transformation for the angular velocity is presented by the matrix $T$ [76],

$$T = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \end{bmatrix}. \tag{2.18}$$

In order to derive the dynamic equations of quadrotor motion, the Newton-Euler formula is applied, that is [76]:

$$\begin{aligned} V^E &= RV^B, \\ \omega^E &= T\omega^B. \end{aligned} \tag{2.19}$$

The quadrotor kinematic model is given by [76]:

$$\begin{cases} \dot{x} = \omega\left(\sin\phi\sin\psi + \cos\phi\cos\psi\sin\theta\right) - v\left(\cos\phi\sin\psi - \cos\psi\sin\phi\sin\theta\right) + u\left(\cos\psi\cos\theta\right), \\ \dot{y} = v\left(\cos\phi\cos\psi + \sin\phi\cos\psi\sin\theta\right) - \omega\left(\cos\psi\cos\phi - \cos\phi\sin\psi\sin\theta\right) + u\left(\cos\theta\sin\psi\right), \\ \dot{z} = \omega\left(\cos\phi\cos\theta\right) - u\left(\sin\theta\right) + v\left(\cos\theta\sin\phi\right), \\ \dot{\phi} = p + r\left(\cos\phi\tan\theta\right) + q\left(\sin\phi\tan\theta\right), \\ \dot{\theta} = q\left(\cos\phi\right) - r\sin\phi, \\ \dot{\psi} = r\frac{\cos\phi}{\cos\theta} + q\frac{\sin\phi}{\cos\theta}. \end{cases} \tag{2.20}$$

The following assumptions are considered for the quadrotor dynamic model [75]:

• The quadrotor structure is fixed.

• The structure of the quadrotor is symmetrical, and its inertia matrix is time-invariant and diagonal.

• The quadrotor center of mass is located on the origin of the body frame.

• The blade deformation of the quadrotor at high velocities is neglected.

Newton's second law is used to derive the translational dynamics of the quadrotor [76],

$$m\left[\omega^B \wedge V^B + \dot{V}^B\right] = F^B, \tag{2.21}$$

where $m$ represents the quadrotor's mass, $\wedge$ represents the cross product and $F^B = [f_x \quad f_y \quad f_z] \in R^3$ is the total force.

Euler's equation derives the total torque that affects the quadrotor [76],

$$I\dot{\omega}^B + \omega^B \wedge \left(I\dot{\omega}^B\right) = \tau^B, \tag{2.22}$$

where the total torque is given by $\tau^B = [\tau_x \quad \tau_y \quad \tau_z]^T$ and $I$ is the inertia matrix that is time-invariant and diagonal, that is [76]:

$$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \in R^3. \tag{2.23}$$

The quadrotor dynamic model in the body-fixed frame is given by [76]:

$$\begin{cases} f_x = m\left(\dot{u} + q\omega - rv\right), \\ f_y = m\left(\dot{v} - p\omega + ru\right), \\ f_z = m\left(\dot{\omega} + pv - qu\right), \\ \tau_x = \dot{p}I_{xx} - qrI_{yy} + qrI_{zz}, \\ \tau_y = \dot{q}I_{yy} + prI_{xx} - prI_{zz}, \\ \tau_z = \dot{r}I_{zz} - pqI_{xx} + pqI_{yy}. \end{cases} \tag{2.24}$$

Several different forces affect the quadrotor while moving in the air, and the resultant force determines the movement direction [77]. Figure 2.18 presents different forces such as thrust, gravitational force, lift, and drag.

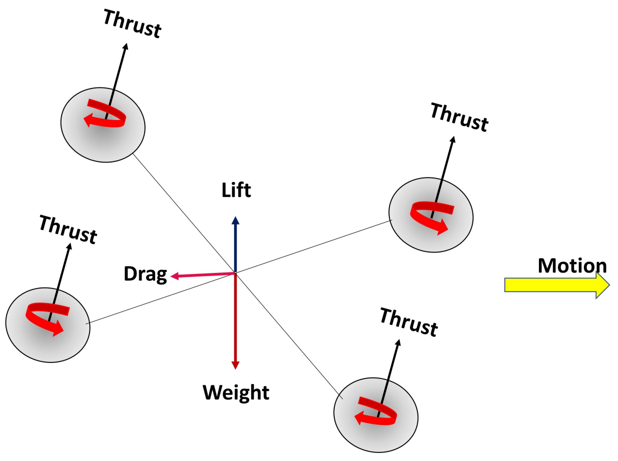

Figure 2.18: Quadrotor Forces.

The external forces in the body-fixed frame $z$ axis are presented as [76]:

$$F^B = mgR^T \hat{e}_z - T_t \hat{e}_3 + f_w, \tag{2.25}$$

where $\hat{e}_z$ is a unit vector in the $z$ axis in the inertial coordinates, $\hat{e}_3$ is a unit vector in the $z$ axis in the body-fixed coordinates, $T_t$ is the total generated thrust by rotors, and $f_w = [f_{wx} \quad f_{wy} \quad f_{wz}]^T \in R^3$ is disturbance that affect quadrotor, such as wind.

The external moments in body-fixed coordinates are presented as [76]:

$$\tau^B = \tau_B - g_a + \tau_w, \tag{2.26}$$

where $g_a$ denotes the gyroscopic moments, which result from joined rotation of all the four rotors and quadrotor body, $\tau_B = [\tau_x \quad \tau_y \quad \tau_z]^T \in R^3$ is the control torque which is generated by the rotor speeds differences, $\tau_w = [\tau_{wx} \quad \tau_{wy} \quad \tau_{wz}]^T \in R^3$ denotes the torque generated by the disturbance affecting the quadrotor motion, like the wind, and $g_a$ is defined as [76]:

$$g_a = \sum_{i=1}^{4} J_{tp} \left(\omega_B \wedge \hat{e}_3\right) (-1)^{i+1} \Omega_i, \tag{2.27}$$

where $J_{tp}$ represents each rotor inertia, $\Omega_i$ is the angular speed of each rotor. In this work, it is assumed $J_{tp}$ to be small, and therefore, in the controller formulation and design, the gyroscopic moments are eliminated. The complete quadrotor dynamic model is given by

[76]:

$$\begin{cases} -mg\sin\theta + f_{wx} = m\left(\dot{u} + q\omega - rv\right), \\ mg\left(\cos\theta\sin\phi\right) + f_{wy} = m\left(\dot{v} - p\omega + ru\right), \\ mg\left(\cos\theta\cos\phi\right) + f_{wz} - T_t = m\left(\dot{\omega} + pv - qu\right), \\ \tau_x + \tau_{wx} = \dot{p}I_{xx} - qrI_{yy} + qrI_{zz}, \\ \tau_y + \tau_{wy} = \dot{q}I_{yy} + prI_{xx} - prI_{zz}, \\ \tau_z + \tau_{wz} = \dot{r}I_{zz} - pqI_{xx} + pqI_{yy}. \end{cases} \tag{2.28}$$

A voltage is applied to each DC motor, and a net torque occurs in the shaft of the rotor, which leads to a thrust $T_i$. A drag force $D_i$ is caused by forwarding velocity, and it performs on the opposite direction of the motion [1]. The thrust and drag are formulated as follows [1]:

$$\begin{aligned} T_i &= b\Omega_i^2, \\ D_i &= d\Omega_i^2. \end{aligned} \tag{2.29}$$

The thrust and drag coefficients are represented by $b$ and $d$, respectively.

In order to control the quadrotor's behavior, the inputs are applied to the system [1],

$$\begin{cases} T_t = b\left(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2\right), \\ \tau_x = bl\left(\Omega_4^2 - \Omega_2^2\right), \\ \tau_y = bl\left(\Omega_3^2 - \Omega_1^2\right), \\ \tau_z = d\left(\Omega_2^2 + \Omega_4^2 - \Omega_3^2 - \Omega_1^2\right), \end{cases} \tag{2.30}$$

where $l$ is defined as the distance between the quadrotor rotors and center of mass, $b$, and $d$ are the thrust factor and the drag factor, respectively [1]. In order to simplify the model, the effect of disturbance is neglected, that is [1]

$$\begin{cases} \dot{u} = rv - qw + g\sin\theta, \\ \dot{v} = pw - ru - g\left(\sin\phi\cos\theta\right), \\ \dot{\omega} = qu - pv + g\left(\cos\theta\sin\phi + \frac{U_1}{m}\right), \\ \dot{p} = \frac{I_{yy}-I_{zz}}{I_{xx}} rq - \frac{J_{tp}}{I_{xx}} q\Omega_T + \frac{U_2}{I_{xx}}, \\ \dot{q} = \frac{I_{zz}-I_{xx}}{I_{yy}} pr + \frac{J_{tp}}{I_{yy}} p\Omega_T + \frac{U_3}{I_{yy}}, \\ \dot{r} = \frac{I_{xx}-I_{yy}}{I_{zz}} pq + \frac{U_4}{I_{zz}}, \end{cases} \tag{2.31}$$

where the propellers' speed input and total thrust are defined below. The control inputs are dedicated to vertical trust $U_1$ and each angular motion$(U_2 - U_4)$. The overall propellers' speed is shown by $\Omega_T(rad/s)$ [1],

$$\begin{cases} U_1 = b\left(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2\right), \\ U_2 = bl\left(\Omega_4^2 - \Omega_2^2\right), \\ U_3 = bl\left(\Omega_3^2 - \Omega_1^2\right), \\ U_4 = d\left(\Omega_2^2 + \Omega_4^2 - \Omega_3^2 - \Omega_1^2\right), \\ \Omega_T = -\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4. \end{cases} \tag{2.32}$$

The quadrotor dynamic model is represented in the body-fixed frame. For the control objectives and in order to generalize the position vector, it is beneficial to represent quadrotor dynamics in the earth-fixed frame. It can be assumed that angular velocity is equal in both frames because, in the hovering condition, the angular transformation matrix $T$ is an identity matrix. With this assumption, the quadrotor dynamics in the earth-fixed frame are represented as follows [1]:

$$\begin{cases} \ddot{x} = \left(\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi\right)\frac{U_1}{m}, \\ \ddot{y} = \left(\cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi\right)\frac{U_1}{m}, \\ \ddot{z} = -g + \left(\cos\phi\cos\theta\right)\frac{U_1}{m}, \\ \ddot{\phi} = \dot{\theta}\dot{\phi}\left(\frac{I_{yy}-I_{zz}}{I_{xx}}\right) - \frac{J_{tp}}{I_{xx}}\dot{\theta}\Omega_T + \frac{U_2}{I_{xx}}, \\ \ddot{\theta} = \dot{\phi}\dot{\psi}\left(\frac{I_{zz}-I_{xx}}{I_{yy}}\right) + \frac{J_{tp}}{I_{yy}}\dot{\phi}\Omega_T + \frac{U_3}{I_{yy}}, \\ \ddot{\psi} = \dot{\phi}\dot{\theta}\left(\frac{I_{xx}-I_{yy}}{I_{zz}}\right) + \frac{U_4}{I_{zz}}. \end{cases} \tag{2.33}$$

## 2.4 Controller Design for the Quadrotor

In this section, a nonlinear control for the stabilization of the quadrotor is presented. In order to design nonlinear control for the quadrotor, the FeedBack Linearization (FBL) technique and Linear Quadratic Regulator (LQR) approaches can be employed. The fundamental concept of the feedback linearization technique is transforming the nonlinear dynamics of the system into a fully or partially linearized system where linear feedback control can be utilized. The nonlinearity functions can be canceled by applying the feedback linearization technique, while the linear control approaches can be utilized. The LQR control methodology is applied for tracking the desired trajectory based on a linear control system [1]. The schematic of the control methodology is shown in Figure 2.19.

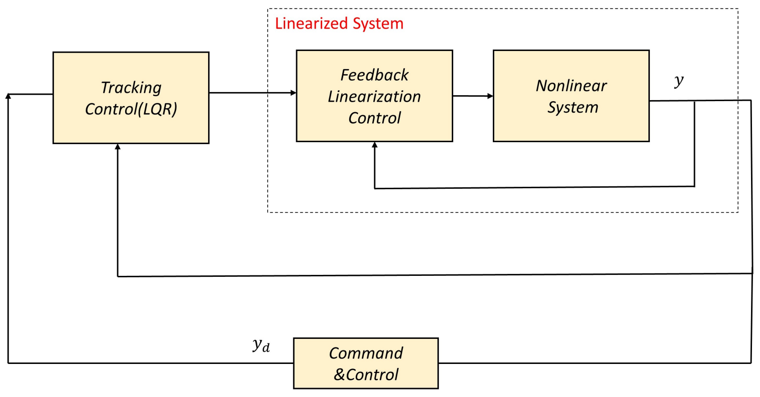

Figure 2.19: Feedback linearization control schematic.

This work assumes that all the system states can be measured and full state feedback is considered. The quadrotor has four inputs $u = [U_1 \quad U_2 \quad U_3 \quad U_4]^T$, and since the number of control inputs is less than six degrees of freedom quadrotor, this system is under-actuated, and four variables can be controlled. The position of the quadrotor can be determined by its linear position $[x \quad y \quad z]$ and yaw angle $\psi$, and these four variables can be controlled by feedback linearization and LQR to stabilize the system and follow a desired trajectory [1].

This work assumes that the quadrotor's motion is near the hovering condition and small angular changes occur. Therefore, the gyroscopic effects in the quadrotor dynamic equations, the term of $\ddot{\phi}$ and $\ddot{\theta}$, are neglected. Since the quadrotor has a symmetric structure, $I_{xx}$ and $I_{yy}$ are equal, and it leads to some simplification in the $\ddot{\psi}$ equation. It should be pointed out that in the hovering condition, the gravitational force is equal to the total thrust, which means that when the height of the quadrotor arrives at its desired value, then $\ddot{z} \rightarrow 0$. Therefore, with this assumption and considering negligible roll and angle, it can be concluded that $U_1 \rightarrow mg$ [1]. By considering these assumptions, the $x$ and $y$ equations can be simplified . The quadrotor dynamics equation considering the movement vector dynamic is represented as follows :

$$\begin{cases} \ddot{x} = g\sin\theta, \\ \ddot{y} = -g\sin\phi, \\ \ddot{z} = -g + \left(\cos\phi\cos\theta\right)\frac{U_1}{m}, \\ \ddot{\phi} = \frac{U_2}{I_{xx}}, \\ \ddot{\theta} = \frac{U_3}{I_{yy}}, \\ \ddot{\psi} = \frac{U_4}{I_{zz}}. \end{cases} \tag{2.34}$$

As mentioned $x, y, z$ and $\psi$ are variables that will be controlled by $U_1, U_2, U_3$, and $U_4$. The quadrotor dynamics equation considering the movement vector dynamic shows that the system can be divided into four semi-decoupled subsystems. The yaw angle can be controlled with tunning $U_4$. An LQR controller is considered for controlling this subsystem. This subsystem's control goal is to reach the desired zero yaw angle in the near hovering condition. For the rest of the variables, three semi-decoupled subsystems are considered as follows [1]:

$$\begin{cases} \ddot{x} = g\sin\theta, \\ \ddot{\theta} = \frac{U_3}{I_{yy}}, \end{cases} \tag{2.35}$$

$$\begin{cases} \ddot{y} = -g\sin\phi, \\ \ddot{\phi} = \frac{U_2}{I_{xx}}, \end{cases} \tag{2.36}$$

$$\ddot{z} = -g + \left(\cos\phi\cos\theta\right)\frac{U_1}{m}, \tag{2.37}$$

These three subsystems will be controlled by applying input-output feedback linearization and LQR approaches. The control approaches are completely explained in the work of Ranjbaran et al. in [1]. The difference is that, in this work, the actuator dynamics is not considered.

The state space representation can be driven as below.

$$\dot{X} = f(X,u) = \begin{cases} X_7 \\ X_8 \\ X_9 \\ X_{10} \\ X_{11} \\ X_{12} \\ g\sin X_5 \\ -g\sin X_4 \\ -g + (\cos X_4 \cos X_5)\frac{U_1}{m} \\ \frac{U_2}{I_{xx}} \\ \frac{U_3}{I_{yy}} \\ \frac{U_4}{I_{zz}} \end{cases}, \tag{2.38}$$

where the states of the systems are:

$$X^T = [X_1\ X_2\ X_3\ X_4\ X_5\ X_6\ X_7\ X_8\ X_9\ X_{10}\ X_{11}\ X_{12}] = [x\ y\ z\ \phi\ \theta\ \psi\ \dot{x}\ \dot{y}\ \dot{z}\ \dot{\phi}\ \dot{\theta}\ \dot{\psi}], \tag{2.39}$$

and the control inputs are:

$$u^T = [U_1\ U_2\ U_3\ U_4]. \tag{2.40}$$

## 2.5 Conclusions

This chapter first presents the concept of cyberattacks from the control viewpoint in CPS, and introduces different types of attacks. Different types of faults in the control system are then introduced, and fault diagnosis approaches are explained. Finally, the fundamental notions of quadrotor motion and dynamics is discussed. The state space representation of the nonlinear system is provided, and the control approach is explained.

The next chapter will be assigned to designing a neural network-based detector for the detection of sensor faults and false data injection attacks and provide a schematic for the isolation of faults and false data injection attack.

# Chapter 3

# Detection and Isolation of Sensor Faults and False Data Injection Attacks

This chapter will discuss the detection and isolation of false data injection attacks and sensor faults in the quadrotor. One of this chapter's great contributions is considering multiplicative false data injection attacks. The second contribution of this chapter is employing virtual sensors at the plant and command and control sides. Using a virtual sensor at the plant side enables the isolation of faults and false data injection attacks in the quadrotor. The virtual sensor at the command and control side recovers the sensor measurements data that are subjected to multiplicative false data injection attacks. The data recovery implemented at the command and control center can mitigate the impact of multiplicative false data injection attacks on the path generator module.

Section 3.1 introduces the false data injection attacks and sensor faults in the quadrotor and presents the data transmission scheme between the quadrotor and the GCS. Section 3.2 deals with the online detection, isolation, and estimation of multiplicative false data injection attacks that impact communication links. Section 3.3 provides the sensor fault detection, isolation, and estimation approach with an online neural network-based detection technique. Section 3.4 presents a typical data recovery approach by introducing the concept of virtual sensors employed at command and control and plant sides. Section 3.5 provides the simulation results by considering different scenarios of sensor faults and false data injection attacks. Section 3.6 investigates the performance of the designed detectors and provides confusion matrices results for the detectors.

## 3.1 False Data Injection Attacks and Sensor Faults in Quadrotors

A false data injection attack can manipulate the input or output channels of the CPS. The false data can be injected into the cyber layer, as shown in Figure 3.1. The sensor faults are considered in this work, as shown in Figure 3.1. In the quadrotor, as a case study, the false data are injected through the communication link between the quadrotor and GCS. The desired trajectory is sent to the quadrotor from the GCS, and in the opposite direction, the quadrotor will send the measurement data. Both directions are vulnerable to cyberattacks, and the adversary may manipulate the transmitted data.

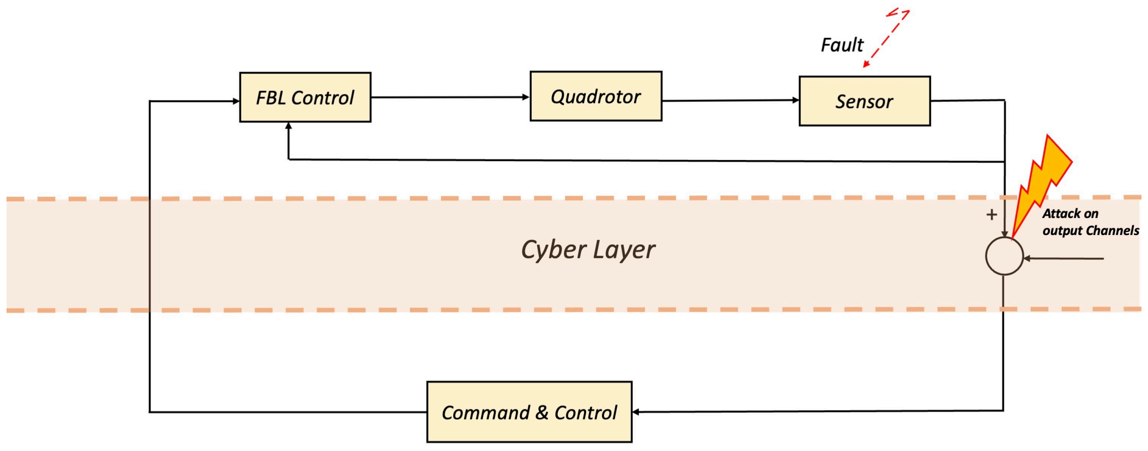

Figure 3.1: CPS under false data injection attack.

In this work, it is assumed that the output communication channel is under multiplicative false data injection attacks. In this scenario, the output communication channel which sends data to the command and control center is under attack. Therefore, the received sensor measurement data is manipulated, and it can be described as:

$$\begin{cases} \dot{x}(t) = f\left(x(t), u(t)\right), \\ \tilde{y}(t) = \alpha * Cx(t), \end{cases} \tag{3.1}$$

where $x \in \mathbb{R}^n$ is the state variable, $u \in \mathbb{R}^m$ is the input, $y \in \mathbb{R}^q$ is the output. $C$ denotes a constant measurement matrix. $\alpha$ represents the attack signal multiplied by the sensor measurement data sent to the command and control center.

The faults in the sensors will be detected, isolated, and estimated on the plant side. False data injection attacks will be detected, isolated, and estimated on the command and control sides, as shown in Figure 3.2.

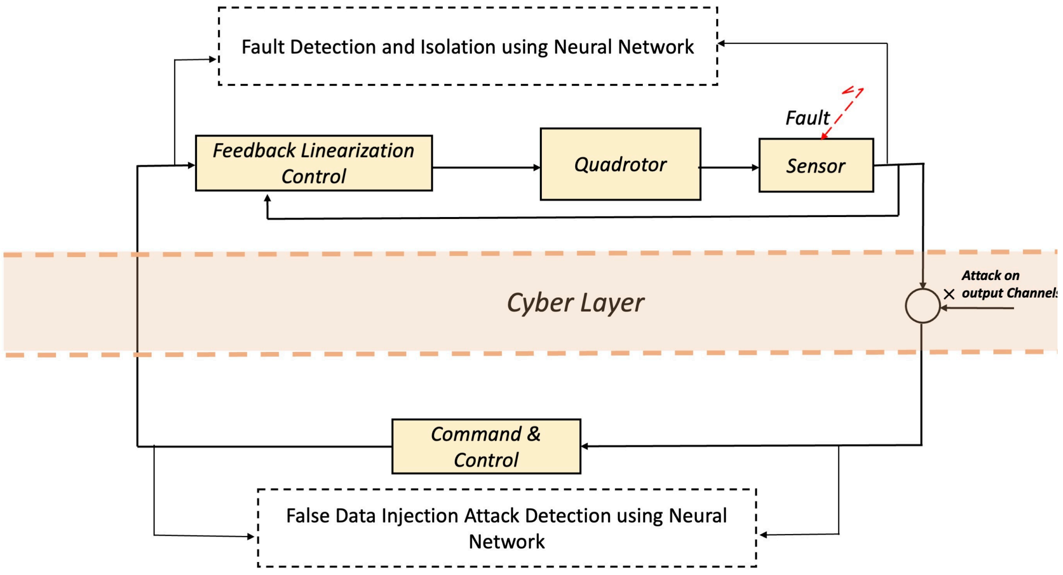
Fault Detection and Isolation using Neural Network
Fault
Feedback Linearization Control
Quadrotor
Sensor
Cyber Layer
Attack on output Channel
Command & Control
False Data Injection Attack Detection using Neural Network

Figure 3.2: Sensor fault and false data injection attacks detection and isolation scheme .

## 3.2 Detection, Isolation, and Estimation of Multiplicative False Data Injection Attacks

In this section, an approach for the detection, isolation, and estimation of multiplicative false data injection attacks is presented. The false data can be injected through the communication channel between the quadrotor and the command and control center. The output channel can be under multiplicative false data injection attacks, as it is shown in Figure 3.2.

The sensor measurement data is transmitted to the command and control center, where an attacker can manipulate it by multiplication of the attack signal in the transmitted data. The manipulated data received at the command and control center can adversely affect the path generation process, misguide the quadrotor, and lead to a collision. The designed neural network-based detector can detect, isolate, and estimate the false data injection attacks on the output communication channel.

The schematic of the NN-based detector in the command and control side is shown in Figure 3.3.

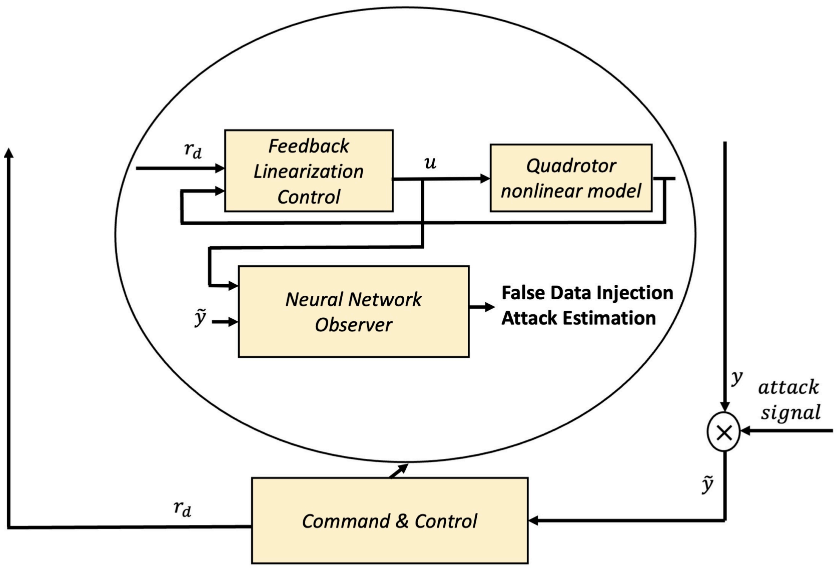
$r_d$
Feedback Linearization Control
$u$
Quadrotor nonlinear model
Neural Network Observer
$\tilde{y}$
False Data Injection Attack Estimation
$y$
attack signal
$\tilde{y}$
$r_d$
Command & Control

Figure 3.3: The schematic of NN observer for False Data Injection attack detection, isolation, and estimation.

The control signal required for the NN observer can be generated at the command and control center, taking into account the nominal model of the system and utilizing feedback linearization control. The control signal $u$ and the manipulated signal $\tilde{y}$ are then transmitted to the adaptive NN observer. The NN's output represents an estimate of the attack signal, and the structure of the NN is described further below.

The nonlinear continuous-time system under multiplicative false data injection attacks can be represented in the general form as:

$$\begin{cases} \dot{x}(t) = f\left(x(t), u(t)\right), \\ y(t) = \alpha\left(x(t), u(t)\right) * Cx(t), \end{cases} \tag{3.2}$$

where $x \in \mathbb{R}^n$ is the state variable, $u \in \mathbb{R}^m$ is the input, $y \in \mathbb{R}^q$ is the output , $f$ is the known nonlinear function and $\alpha$ denotes the unknown multiplicative false data injection attack.

A neural network detection module will be considered to detect, isolate, and estimate the multiplicative false data injection attacks. The false data injection attack detection structure is depicted in Figure 3.4. It is shown that a neural network (NN) will detect, isolate, and estimate the multiplicative false data injection attacks.

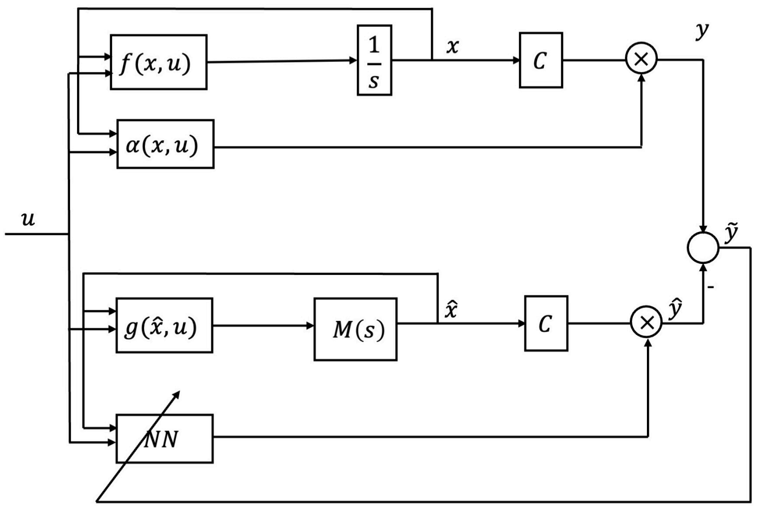

Figure 3.4: Neural-network-based scheme for multiplicative false data injection attacks detection, isolation, and estimation.

As it is shown in Figure 3.4, the nonlinear state space representation of the observer considering the multiplicative false data injection attacks can be rewritten as follows:

$$\begin{cases} \dot{\hat{x}} = A\hat{x} + g(\hat{x}, u), \\ \hat{y} = \hat{\alpha}(\hat{x}(t), u(t)) * C\hat{x}, \end{cases} \tag{3.3}$$

where $A$ is a Hurwitz matrix and $g(\hat{x}, u) = f(\hat{x}, u) - A\hat{x}$. The transfer matrix $M(s)$ is defined as follows [62]:

$$M(s) = (sI - A)^{-1}. \tag{3.4}$$

The NN is going to estimate the multiplicative false data injection attacks $\hat{\alpha}(\hat{x}(t), u(t))$. This detection scheme will detect, isolate and estimate multiplicative false data injection attacks in the nonlinear quadrotor. The NN in this structure has three layers, including input, hidden, and output, as shown in Figure 3.5.

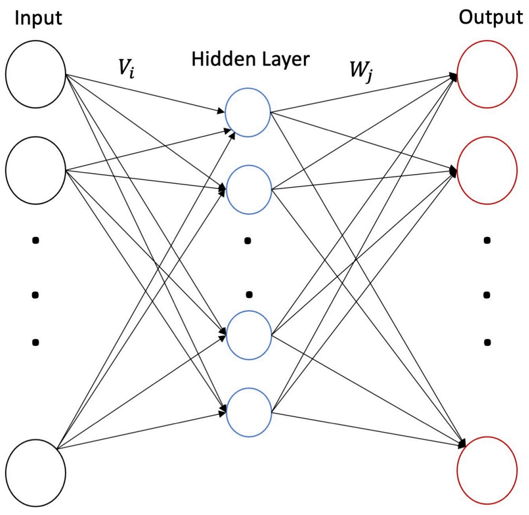

Figure 3.5: Neural network structure.

The NN will obtain the approximation of false data injection attacks with the following structure:

$$\hat{\alpha}\left(\hat{\bar{x}}(t), u(t)\right) = \hat{W}\sigma(\hat{V}\hat{\bar{x}}), \tag{3.5}$$

where $\hat{\bar{x}} = \begin{bmatrix} \hat{x}^T & u^T \end{bmatrix}$, and $\hat{W}$ and $\hat{V}$ are neural network weight matrices.

The activation function is defined as [62]:

$$\sigma_i(V^i\hat{\bar{x}}) = \frac{2}{1 + exp^{-2V^i\hat{\bar{x}}}} - 1, \qquad i = 1, 2, ..., N, \tag{3.6}$$

where $N$ is the number of neurons in the hidden layer.

The weights of this NN are updated based on the theorem stated in the work of Talebi et al.[62].

$$\begin{aligned} \dot{\hat{W}} &= -\eta_1\left(\frac{\partial J}{\partial \hat{W}}\right) - \rho_1 ||\left(y - \hat{y}\right)|| \hat{W}, \\ \dot{\hat{V}} &= -\eta_2\left(\frac{\partial J}{\partial \hat{V}}\right) - \rho_2 ||\left(y - \hat{y}\right)|| \hat{V}, \end{aligned} \tag{3.7}$$

where $\eta_1$ and $\eta_2 > 0$ are learning rates and $\rho_1$ and $\rho_2$ are small positive constants and the objective function of neural network is defined as [62]:

$$J = \frac{1}{2}((y - \hat{y})^T(y - \hat{y})). \tag{3.8}$$

In order to define learning rules, the first two variables are defined as [62]:

$$\begin{aligned} &net_{\hat{v}} = \hat{V}\hat{\bar{x}}, \\ &net_{\hat{w}} = \hat{W}S_1, \\ &S_1 = \sigma(\hat{V}\hat{\bar{x}}), \\ &S_2 = I - \Lambda(\hat{V}\hat{\bar{x}}), \\ &\Lambda(\hat{V}\hat{\bar{x}}) = diag\{\sigma_i^2\left(V^i\hat{\bar{x}}\right)\}. \end{aligned} \tag{3.9}$$

Therefore, $\dfrac{\partial J}{\partial \hat{W}}$ and $\dfrac{\partial J}{\partial \hat{V}}$ can be determined as [63; 78]:

$$\begin{aligned} \frac{\partial J}{\partial \hat{W}} &= \frac{\partial J}{\partial net_{\hat{w}}} \cdot \frac{\partial net_{\hat{w}}}{\partial \hat{W}}, \\ \frac{\partial J}{\partial \hat{V}} &= \frac{\partial J}{\partial net_{\hat{v}}} \cdot \frac{\partial net_{\hat{v}}}{\partial \hat{V}}. \end{aligned} \tag{3.10}$$

Then,

$$\begin{aligned} \frac{\partial J}{\partial net_{\hat{w}}} &= \frac{\partial J}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial net_{\hat{w}}} = -(y-\hat{y})^T \frac{\partial \hat{y}}{\partial net_{\hat{w}}}, \\ \frac{\partial J}{\partial net_{\hat{v}}} &= \frac{\partial J}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial net_{\hat{v}}} = -(y-\hat{y})^T \frac{\partial \hat{y}}{\partial net_{\hat{v}}}, \end{aligned} \tag{3.11}$$

and,

$$\begin{aligned} &\frac{\partial \hat{y}}{\partial net_{\hat{w}}} = C\hat{x}, \\ &\frac{\partial \hat{y}}{\partial net_{\hat{v}}} = \frac{\partial \hat{y}}{\partial net_{\hat{w}}} \cdot \frac{\partial net_{\hat{w}}}{\partial net_{\hat{v}}} = \\ &C\hat{x}.\hat{W}.\frac{\partial \sigma(\hat{V}\hat{\bar{x}})}{\partial net_{\hat{v}}} = C\hat{x}\hat{W}S_2, \end{aligned} \tag{3.12}$$

and,

$$\begin{aligned}
\frac{\partial net_{\hat{w}}}{\partial \hat{W}} &= S_1^T, \\
\frac{\partial net_{\hat{v}}}{\partial \hat{V}} &= \hat{\bar{x}}, \\
\frac{\partial \sigma(\hat{V}\hat{\bar{x}})}{\partial net_{\hat{v}}} &= S_2^T.
\end{aligned} \tag{3.13}$$

The weight matrices will be updated as below:

$$\begin{aligned}
\dot{\hat{W}} &= -\eta_1 \left(\tilde{y}^T A_C^{-1}\, C\; diag(x)\right)^T S_1^T - \rho_1 ||\tilde{y}|| \hat{W}, \\
\dot{\hat{V}} &= -\eta_2 \left(\tilde{y}^T A_C^{-1}\, C\; diag(x) \hat{W} S_2\right)^T \hat{\bar{x}}^T - \rho_2 ||\tilde{y}|| \hat{V},
\end{aligned} \tag{3.14}$$

where $\eta_1$ and $\eta_2 > 0$ are learning rates and $\rho_1$ and $\rho_2$ are small positive constants. $A_C$ is a Hurwitz matrix which can help to speed the convergence.

By using this method, the false data injection attack that is injected into the output communication channel is estimated. The estimated attack which is the output of the NN is derived by (3.5) and can be represented as:

$$\hat{\alpha} = [\hat{\alpha}_1 \quad \hat{\alpha}_2 \quad \hat{\alpha}_3 \quad \hat{\alpha}_4 \quad \hat{\alpha}_5 \quad \hat{\alpha}_6 \quad \hat{\alpha}_7 \quad \hat{\alpha}_8 \quad \hat{\alpha}_9 \quad \hat{\alpha}_{10} \quad \hat{\alpha}_{11} \quad \hat{\alpha}_{12}], \tag{3.15}$$

where each element of $\hat{\alpha}$ represents the estimated false data injection attacks for each of the sensor measurements data.

The neural network output, $\hat{\alpha}$, is close to one when the system operates under normal conditions. However, when a cyberattack is injected into the system, the neural network estimates the attack signal, and the output is considered as the residual. To reduce false alarms resulting from measurement noise in the system, a Monte Carlo test is performed under different conditions to determine a suitable threshold. If the residual, which represents the estimation of the attack signal, exceeds the threshold determined through the Monte Carlo test, the attack is detected,

$$|\hat{\alpha}_i| > 1 \pm \delta_{\alpha i}, \qquad i = 1, ..., 12, \tag{3.16}$$

where $\delta_{\alpha i}$ represents the threshold for the $i^{th}$ sensor measurement data.

The detection approach employs a neural network-based detector that estimates the multiplicative false data injected into each sensor measurement data, resulting in estimation for each signal. This estimated value is considered as the residual for the cyberattack diagnosis procedure. By analyzing these residuals, the neural network can identify which output sensor measurement data are under attack. Thus, if the estimation of the attack is not close to one, the neural network-based detection approach can achieve attack isolation. Specifically, the neural network can detect and isolate the multiplicative false injection data attack in the output channel if it estimates the cyberattack signal correctly.

## 3.3 Detection, Isolation, and Estimation of Sensors Bias Faults

The fault detection procedure will be done on the plant side. In this work bias faults for sensors are considered and the neural network-based detection strategy for bias fault detection is presented in [62]. The nonlinear continuous-time system with bias sensor faults can be represented in the general form as:

$$\begin{cases} \dot{x}(t) = f\left(x(t), u(t)\right), \\ y(t) = Cx(t) + T_s\left(x(t), u(t)\right), \end{cases} \tag{3.17}$$

where $x \in \mathbb{R}^n$ is the state variable, $u \in \mathbb{R}^m$ is the input, $y \in \mathbb{R}^q$ is the output , $f$ is the known nonlinear function and $T_s$ denotes the unknown sensor fault.

A neural network detection module will be considered to detect the sensors faults. The sensor fault detection structure is depicted in Figure 3.6. It is shown that the neural network will estimate the sensor faults.

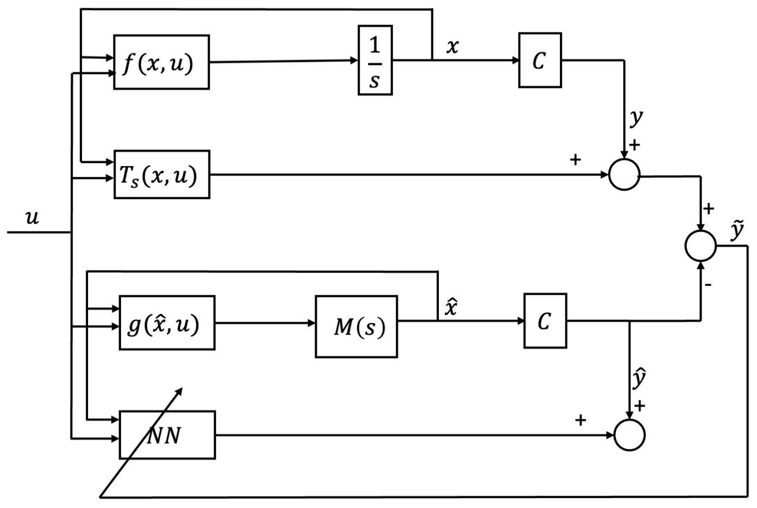

Figure 3.6: Neural-network-based scheme for sensor faults detection and isolation.

As shown in Figure 3.6, the nonlinear state space representation of the observer considering sensor faults can be rewritten as follows [62]:

$$\begin{cases} \dot{\hat{x}} = A\hat{x} + g(\hat{x}, u), \\ \hat{y} = C\hat{x} + \hat{T}_s(\hat{x}, u), \end{cases} \tag{3.18}$$

where $A$ is a Hurwitz matrix and $g(\hat{x}, u) = f(\hat{x}, u) - A\hat{x}$. The transfer matrix $M(s)$ is defined as follows [62]:

$$M(s) = (sI - A)^{-1}. \tag{3.19}$$

The NN is going to estimate the bias bias sensor faults $T_s(x, u)$. This detection scheme will detect, isolate and estimate sensor faults in the nonlinear quadrotor.

The NN will obtain the approximation of fault with the following structure [62]:

$$\hat{T}_s\left(\hat{\hat{x}}, u\right) = \hat{W}\sigma(\hat{V}\hat{\hat{x}}), \tag{3.20}$$

where $\hat{\hat{x}} = \begin{bmatrix}\hat{x}^T & u^T\end{bmatrix}$, and $\hat{W}$ and $\hat{V}$ are neural network weights matrices.

The activation function is defined as below [62]:

$$\sigma_i(V^i\hat{\hat{x}}) = \frac{2}{1+exp^{-2V^i\hat{\hat{x}}}} - 1, \qquad i = 1, 2, ..., N, \tag{3.21}$$

where $N$ is the number of neurons in the hidden layer.

The weight matrices will be updated as follows [62] :

$$\begin{aligned}
\dot{\hat{W}} &= -\eta_1\left(\tilde{y}^T A_C^{-1}\right)^T S_1^T - \rho_1\|\tilde{y}\|\hat{W},\\
\dot{\hat{V}} &= -\eta_2\left(\tilde{y}^T A_C^{-1}\hat{W}S_2\right)^T \hat{\hat{x}}^T - \rho_2\|\tilde{y}\|\hat{V},\\
S_1 &= \sigma(\hat{V}\hat{\hat{x}}),\\
S_2 &= I - \Lambda(\hat{V}\hat{\hat{x}}),\\
\Lambda(\hat{V}\hat{\hat{x}}) &= diag\{\sigma_i^2\left(V^i\hat{\hat{x}}\right)\},
\end{aligned} \tag{3.22}$$

where $\eta_1$ and $\eta_2 > 0$ are learning rates and $\rho_1$ and $\rho_2$ are small positive constants. $A_C$ is a Hurwitz matrix which can help to speed the convergence [62].

By using this method, the bias sensor faults are estimated. The estimated sensor faults which is the output of the NN is derived by (3.20) and can be represented as:

$$\hat{T}_s = [\hat{T}_{s1} \quad \hat{T}_{s2} \quad \hat{T}_{s3} \quad \hat{T}_{s4} \quad \hat{T}_{s5} \quad \hat{T}_{s6} \quad \hat{T}_{s7} \quad \hat{T}_{s8} \quad \hat{T}_{s9} \quad \hat{T}_{s10} \quad \hat{T}_{11} \quad \hat{T}_{12}], \tag{3.23}$$

where each element of $\hat{T}_s$ represents the estimated bias faults for each of the twelve sensors.

During normal operation, the neural network output $\hat{T}_s$ is close to zero. But when there are bias sensor faults in the system, the neural network estimates the faults and takes this estimated value as the residual, which can be used for attack detection. To minimize false alarms caused by measurement noise, a Monte Carlo test is performed under different conditions to identify an appropriate threshold [62]. If the residual, which represents the estimated bias sensor faults, exceeds the selected threshold, then the sensor fault is detected,

$$|\hat{T}_{si}| > \delta_{Tsi}, \qquad i = 1, ..., 12, \tag{3.24}$$

where $\delta_{TAi}$ represents the threshold for the $i^{th}$ sensor.

The detector estimates the bias sensor faults for each sensor, and the estimated value is considered as the residual for each signal. These residuals can also be used for the fault isolation procedure, which can determine which sensors are under bias sensor fault by comparing each residual with a predefined threshold and isolating faulty sensors. Thus, the neural network-based detection approach can achieve fault isolation if the estimation of the fault is nonzero. If the neural network accurately estimates the bias sensor faults signal, it can simultaneously detect and isolate faults in various sensors.

## 3.4 Virtual Sensors for Mitigating Multiplicative False Data Injection Attacks and Sensor Faults

In this section, the concept of virtual sensors is discussed. Generally, faults in the control systems can lead to performance degradation or instability. Therefore it is essential to provide fault-tolerant control techniques which help the desired system performance maintained while the control system is stable [79].

Fault-hiding is an effective and practical active fault-tolerant control approach. This approach involves reconfiguring the faulty system to hide the fault from the controller and observer's perspective, allowing the faulty component to be used without modifications while maintaining stability in the reconfiguration block. In this method, virtual actuators and virtual sensors are introduced into the control loop when actuator and sensor faults are present in the system. These blocks serve to mask the faults and ensure that the control loop operates correctly despite the presence of faulty components [80; 81].

Virtual sensors are mathematical models, not physical components, based on received data from various sensors. They use available sensor measurements and system parameters to estimate faults in the sensors. In the case of sensor faults, virtual sensors employ an observer to estimate and calculate a replacement value [82; 83; 84].

In this work, virtual sensors offer an effective solution for mitigating false data injection attacks and sensor faults by providing replacement data. Placing virtual sensors on the plant side conceals faults from the controller and command and control center and provide a replacement data. Moreover, this scheme makes isolating simultaneous faults and false data injection attacks at the output channel possible. Additionally, virtual sensors can be utilized to recover sensor measurement data that has been compromised by multiplicative false data injection attacks at the command and control center. This ensures that accurate and reliable decisions are made regarding the desired trajectory of the quadrotor.

### 3.4.1 Virtual Sensors for Faulty Sensor Recovery

Figure 3.7 is shown the active fault-tolerant control by using virtual sensor is shown. The figure demonstrates the use of virtual sensors for fault-tolerant control. Virtual sensors conceal faults in sensor data from the controller and command and control center, allowing normal operation. In case of a false data injection attack, the command and control center's detector will identify and isolate it from sensor bias faults.

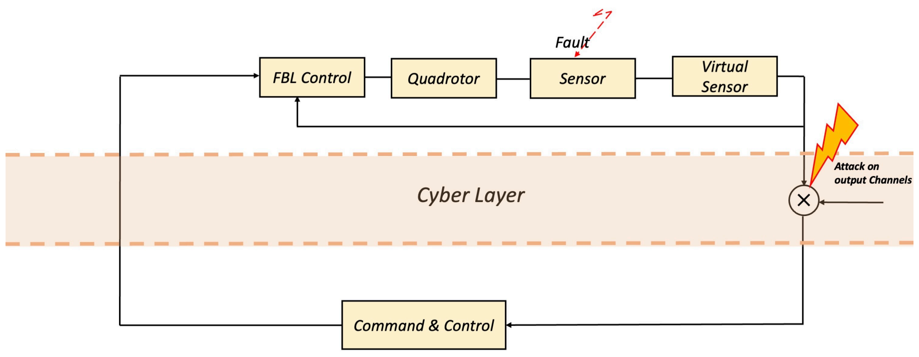

Figure 3.7: Active fault -tolerant control scheme using virtual sensor

The nonlinear continuous-time system with bias sensor faults can be represented in the general form as:

$$\begin{cases} \dot{x}(t) = f\left(x(t), u(t)\right), \\ y(t) = Cx(t) + T_s\left(x(t), u(t)\right), \end{cases} \tag{3.25}$$

where $x \in \mathbb{R}^n$ is the state variable, $u \in \mathbb{R}^m$ is the input, $y \in \mathbb{R}^q$ is the output , $f$ is the known nonlinear function and $T_s$ denotes the unknown sensor fault.

A neural network-based observer is utilized in the virtual sensor block to detect sensor faults. The observer's schematic is depicted in Figure 3.6. The virtual sensor structure for sensor bias faults is defined as follows:

$$\begin{cases} \dot{\hat{x}}_{vs} = A\hat{x}_{vs} + g\left(\hat{x}_{vs}, u\right), \\ \hat{y}_{vs} = y - \hat{T}_s\left(\hat{x}, u\right), \end{cases} \tag{3.26}$$

where $A$ is a Hurwitz matrix and $g\left(\hat{x}, u\right) = f\left(\hat{x}, u\right) - A\hat{x}$ and $M(s)$ is the transfer matrix defined in (3.19). $\hat{T}_s$ represents the estimate of the sensor faults which is obtained by using equations (3.20), (3.21), and (3.22).

By adding a virtual sensor block, the output of a faulty sensor can be replaced with healthy data that mimics the output of a normal, functioning sensor. This allows for the continuation of normal system operations despite the presence of faults in the physical sensor.

### 3.4.2 Virtual Sensors for Multiplicative False Data Injection Attacks Recovery

The proposed neural network-based detection method enables the detection, isolation, and estimation of multiplicative false data injection attacks. The estimated false data can then be utilized in the recovery process. One solution to recover sensor measurement data is to employ a virtual sensor, which can replace manipulated data with healthy data. The recovery scheme is illustrated in Figure 3.8.

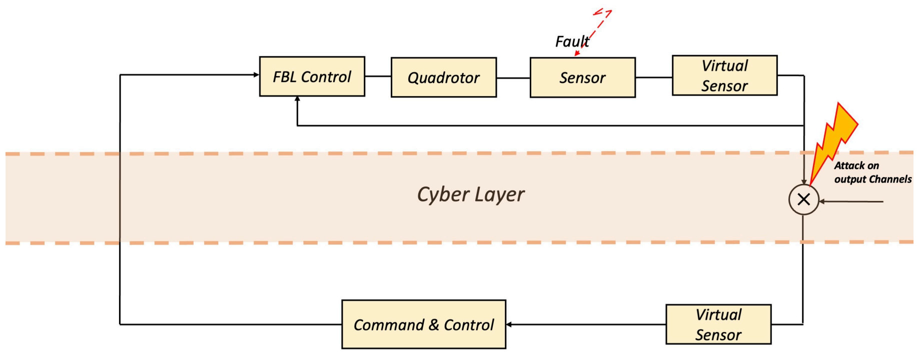

Figure 3.8: Virtual sensor scheme for sensor measurement recovery after the occurrence of multiplicative false data injection attacks.

The nonlinear continuous-time system under multiplicative false data injection attacks can be represented in the general form as:

$$\begin{cases} \dot{x}(t) = f\left(x(t), u(t)\right), \\ y(t) = \alpha\left(x(t), u(t)\right) * Cx(t), \end{cases} \tag{3.27}$$

where $x \in \mathbb{R}^n$ is the state variable, $u \in \mathbb{R}^m$ is the input, $y \in \mathbb{R}^q$ is the output , $f$ is the known nonlinear function and $\alpha$ denotes the unknown multiplicative false data injection attack.

In the virtual sensor block at the command and control center, a neural network-based observer is used to detect, isolate and estimate multiplicative false data injection attacks as shown in Figure 3.4.

The virtual sensor structure for sensor bias faults is defined as follows:

$$\begin{cases} \dot{\hat{x}}_{vs} = A\hat{x}_{vs} + g\left(\hat{x}_{vs}, u\right), \\ \hat{y}_{vs} = \hat{\alpha}^{-1} y, \end{cases} \tag{3.28}$$

where $\hat{\alpha}$ represents the estimate of the multiplicative false data injection attacks which is obtained by using equations (3.5), (3.13), and (3.14).

The virtual sensor block at the command and control center replaces sensor measurement data subject to multiplicative false data injection attacks with healthy data. The recovery of the manipulated data helps the command and control center to generate the proper desired trajectory despite the cyberattacks on the communication link.

## 3.5 Simulation Results

This section presents the simulation results and evaluates the performance of the sensor fault detectors and false data injection attack detectors. Different scenarios are considered to assess the performance of the detectors. Section 3.5.1 represents the operation of the nonlinear quadrotor in a healthy condition with no faults and no false data injection attacks. Section 3.5.2 investigates the system's performance and fault detection mechanism while the sensors are faulty. Section 3.5.3 studies the detection of false data injection attacks in the communication channels by the designed NN-based detectors. Section 3.5.4 analyzes the system under multiplicative false data injection attacks in the communication channel while the sensors are faulty.

The quadrotor model parameters are the same as Ranjbaran et al. in [1], and it is presented in the Table 3.1.

Table 3.1: *OS4 Quadrotor Physical Parameters [1]*

| Name | Parameter | Value | Unit |
|---|---|---|---|
| $mass$ | $m$ | 0.650 | $kg$ |
| $Inertia\ on\ x\ axis$ | $I_{xx}$ | $7.5e^{-3}$ | $kg.m^2$ |
| $Inertia\ on\ y\ axis$ | $I_{yy}$ | $7.5e^{-3}$ | $kg.m^2$ |
| $Inertia\ on\ z\ axis$ | $I_{zz}$ | $1.3e^{-2}$ | $kg.m^2$ |
| $thrust\ coefficient$ | $b$ | $3.13e^{-5}$ | $N.s^2$ |
| $drag\ coefficient$ | $d$ | $7.5e^{-7}$ | $N.s^2$ |
| $arm\ length$ | $l$ | 0.23 | $m$ |
| $propeller\ inertia$ | $J_p$ | $6e^{-5}$ | $kg.m^2$ |
| $gearbox\ efficiency$ | $\eta$ | 90% | |
| $gearbox\ ratio$ | $r$ | 4 : 1 | |
| $motor\ inertia$ | $J_m$ | $6e^{-5}$ | $kg.m^2$ |
| $motor\ internal\ resistance$ | $R_{mot}$ | 0.6 | $\Omega$ |
| $back-EMF\ constant$ | $k_e$ | 5.2 | $V.s.rad^{-1}$ |

The parameters of the neural network-based fault detector are considered as follows:

$A = -0.02 * I_{12\times12}$,

$number\ of\ hidden\ layer = 7$,

$\eta_1 = 20$,

$\eta_2 = 20$,

$\rho_1 = 1e^{-6}$,

$\rho_2 = 1e^{-6}$.

The threshold for the sensor fault detector has been defined through a Monte Carlo test that was performed in the absence of sensor faults and false data injection attacks, while considering noise in the system,

$\delta_{TAi} = 0.05$.

The parameters of the neural network-based multiplicative false data injection attack detector are considered as follows:

$A = -0.02 * I_{12\times 12}$,

$number\ of\ hidden\ layer = 7$,

$\eta_1 = 200$,

$\eta_2 = 200$,

$\rho_1 = 1$,

$\rho_2 = 1$.

The threshold for the multiplicative false data injection attack detector is determined by conducting a Monte Carlo simulation with noise on the normal system in the absence of sensor faults and false data injection attacks,

$\delta_{\alpha i} = 0.03$.

An additive white Gaussian noise is considered in the sensor measurements and the signal to noise ratio (SNR) is considered as:

$SNR = 30db$.

### 3.5.1 Fault-Free and Attack-Free Scenario

The objective of the quadrotor operation is to start from the origin (0,0,0), with an initial roll, yaw, and pitch angle set to zero, and follow the desired trajectory consisting of specific points and then return to the origin. The real-time path planning topic is a broad and important subject that can be studied from different aspects, such as generating cost-effective paths. In this work, a simple real-time path generator is selected to highlight the significance of data recovery from sensor faults and multiplicative false data injection attacks. The path-planning process is based on time and the current position of the quadrotor.

The specific points are defined, and the quadrotor will go to the points and hover for a determined time as shown in Figure 3.9.

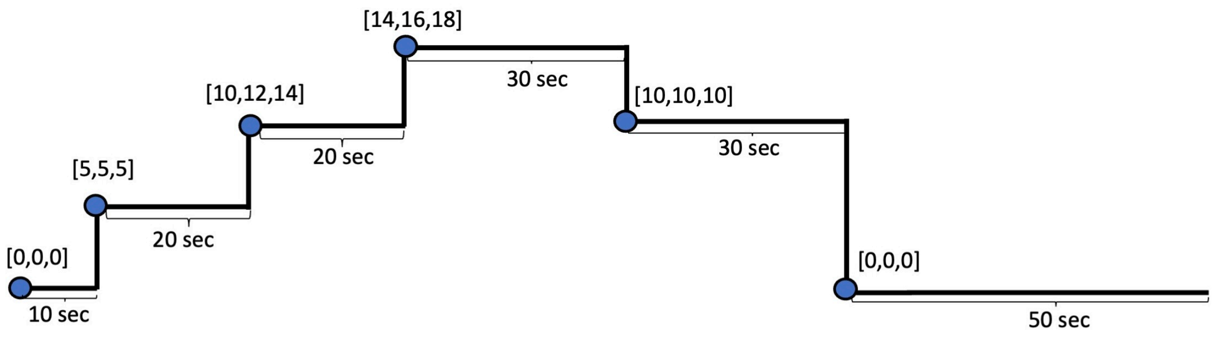

Figure 3.9: Quadrotor path planning model.

The function that depends on the position of the quadrotor and time is considered as follows:

$$r_d(t) = y(t) + s(f(t) - y(t)), \tag{3.29}$$

where $r_d$ represents the desired trajectory generated at the command and control center. $y$ is the sensor measurement data sent from the quadrotor to the command and control center, and $s$ is step size, and it is the time interval between two successive updates of the current position in the path generation process. $f$ is the final point and the desired target position that the algorithm is trying to reach.

Figure 3.10 depicts the quadrotor's position and Euler angles in time following the commanded trajectory from the command and control which is shown in Figure 3.9 .

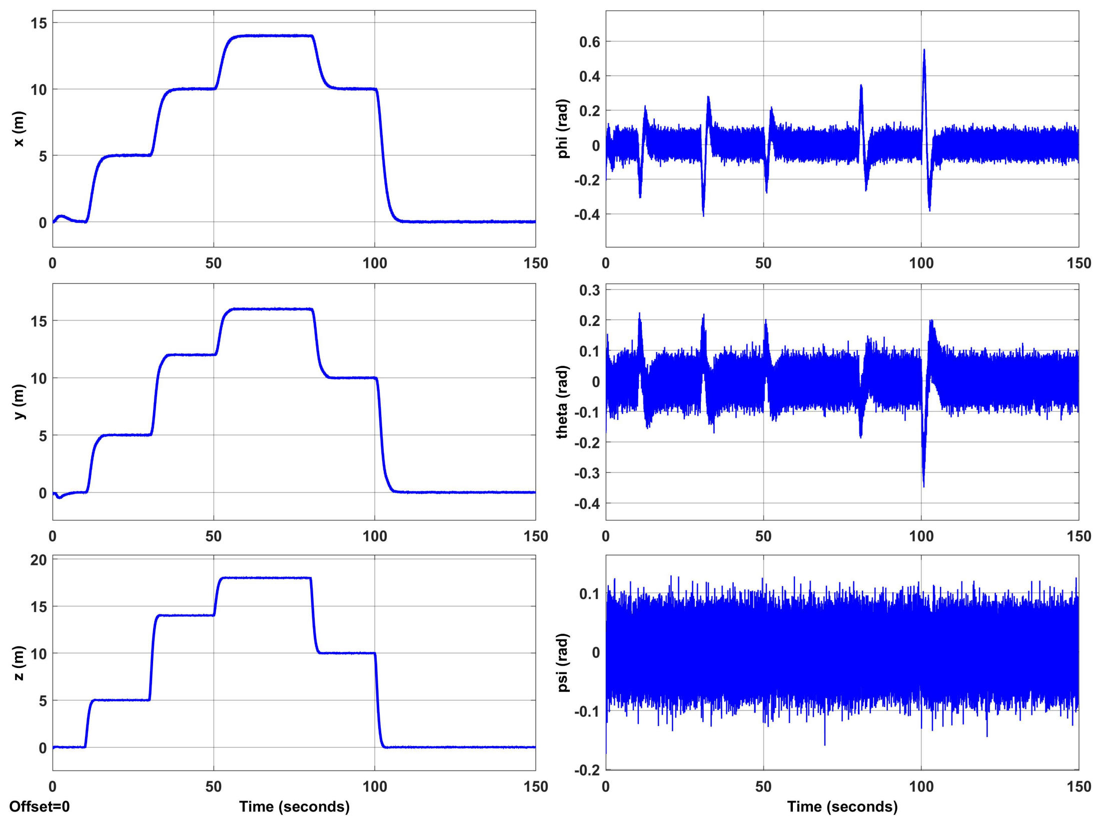

Figure 3.10: Linear position and Euler angles of the quadrotor.

In this work, the $x, y$, and $z$ states are used to generate the trajectory of the quadrotor. The generated trajectory is then transmitted from the command and control center to the quadrotor. The actual trajectory represents the path the quadrotor physically follows, while the planned trajectory refers to the intended path. The trajectory generation process is based on the planned trajectory. Figures 3.11, 3.12, and 3.13 show the actual, planned, and generated trajectory of the quadrotor. It is shown that quadrotor follows the planned trajectory effectively and the generated path matching the planned path.

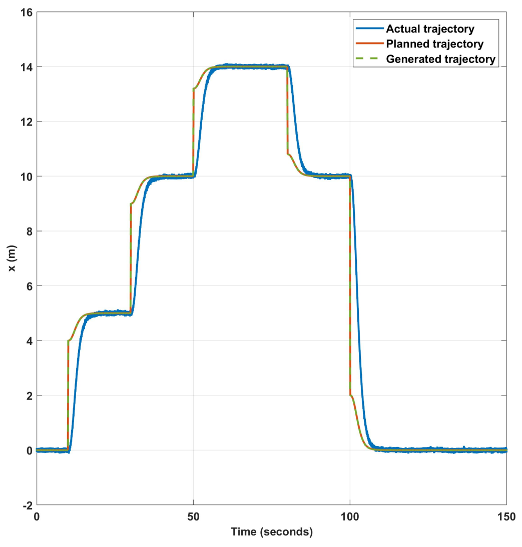

Figure 3.11: The actual, planned, and generated trajectory of the quadrotor in $x$ direction.

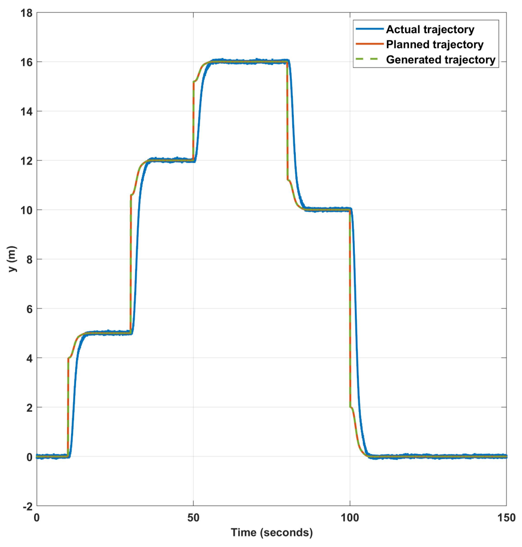

Figure 3.12: The actual, planned, and generated trajectory of the quadrotor in $y$ direction.

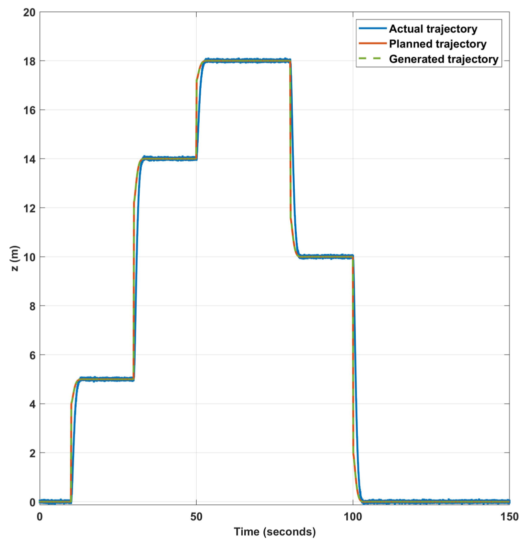

Figure 3.13: The actual, planned, and generated trajectory of the quadrotor in $z$ direction.

Tracking error between the actual and generated trajectory of the quadrotor, shown in Figure 3.14. The sudden variations in the tracking error are a result of the abrupt changes in the trajectory of the quadrotor.

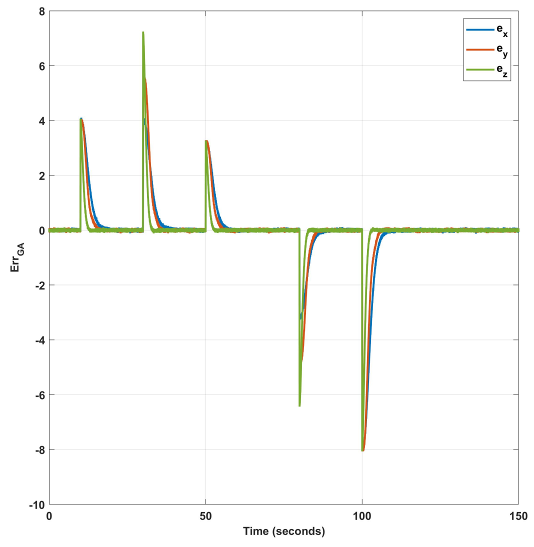

Figure 3.14: The tracking error between the actual, and generated trajectory of the quadrotor.

Tracking error between the planned and generated trajectory of the quadrotor, shown in

Figure 3.15. It is shown that the generated trajectory is perfectly matching with the intended path.

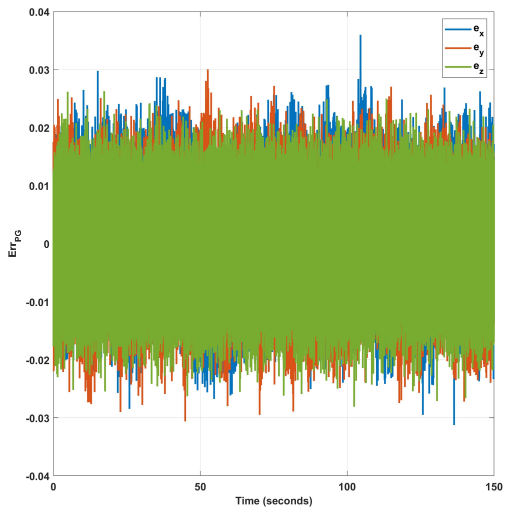

Figure 3.15: The tracking error between the planned, and generated trajectory of the quadrotor.

Tracking error between the planned and actual trajectory of the quadrotor, shown in Figure 3.16. The abrupt changes in the trajectory of the quadrotor result in sudden variations in the tracking error between planned and actual trajectory.

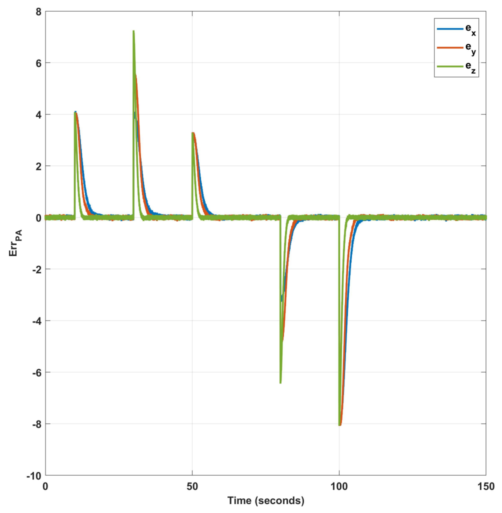

Figure 3.16: The tracking error between the planned, and actual trajectory of the quadrotor.

Figure 3.17 illustrates the behavior of the sensor fault detector under normal operating conditions when there are no sensor faults or false data injection attacks. As shown in the figure, the detector briefly activates for less than one second when the path of the system changes, but it does not exceed the predefined threshold (as described on page 63), indicating that no faults are present in the system.The full-state measurement system uses twelve sensors, and the sensor fault detector provides twelve outputs that estimate faults in each sensor. These outputs are shown in the Figure 3.17.

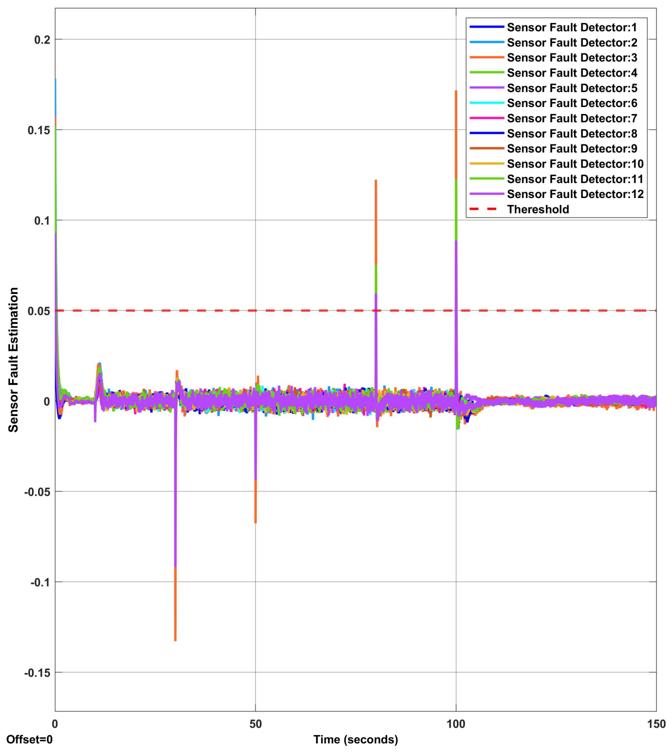

Figure 3.17: The response of the sensor fault detection diagnosis on the plant side in normal condition. Refers to page 63 for the selection of the threshold.

Figure 3.18 depicts the performance of the multiplicative false data injection attack detector under normal operating conditions, where no faults or attacks are present. As can be seen, the detector remains inactive throughout the entire operation, as it never reaches the predefined threshold (as described on page 64) at any point in time. This indicates that there are no multiplicative false data injection attacks present in the system during the duration of the experiment.

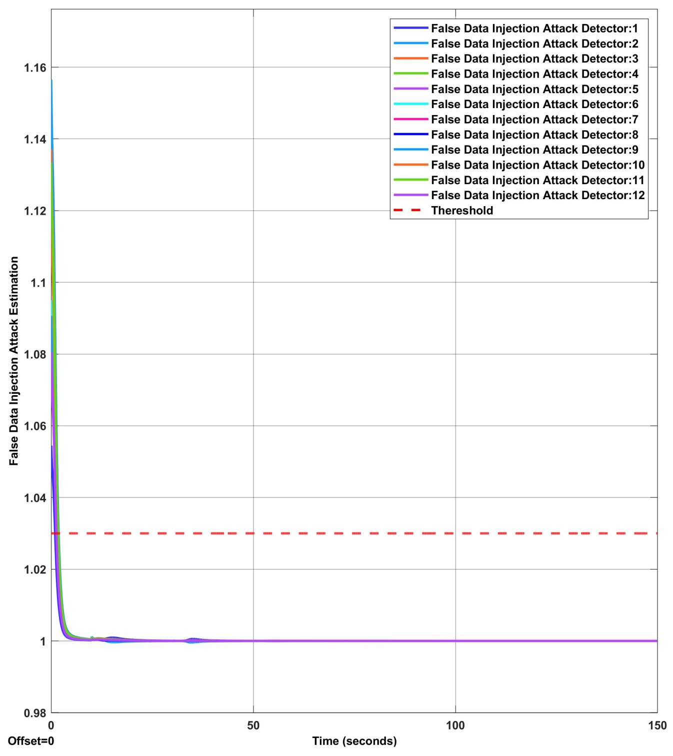

Figure 3.18: The response of the false data injection attack detection diagnosis on the C&C side in normal condition. Refers to page 64 for the selection of the threshold.

### 3.5.2 Bias Fault In Sensors and Attack-Free Scenario

In this section, the faults in the sensors of the quadrotors are considered, and it is assumed that the system is not under false data injection attacks. In this case, some sensors of the quadrotors are faulty, and the performance of the sensor fault detector on the plant side is studied. The sensor faults are considered in the time interval of [20,120], and it is considered that ten of the sensors are faulty. The fault signal is defined as:

$T_s = [0.4 \quad 0.6 \quad 0.5 \quad 0.45 \quad 0.7 \quad 0.25 \quad 0 \quad 0.1 \quad 0.2 \quad 0.28 \quad 0 \quad 0.15]$.

**A) Simulation results without considering virtual sensor**

This part presents the simulation results for the quadrotor under bias sensor faults. Figure

3.19 shows the impact of sensors faults on the quadrotor's linear position and Euler angles. It is shown that during the time interval of [20,120], the system experiences bias sensor faults, which have an impact on the system states.

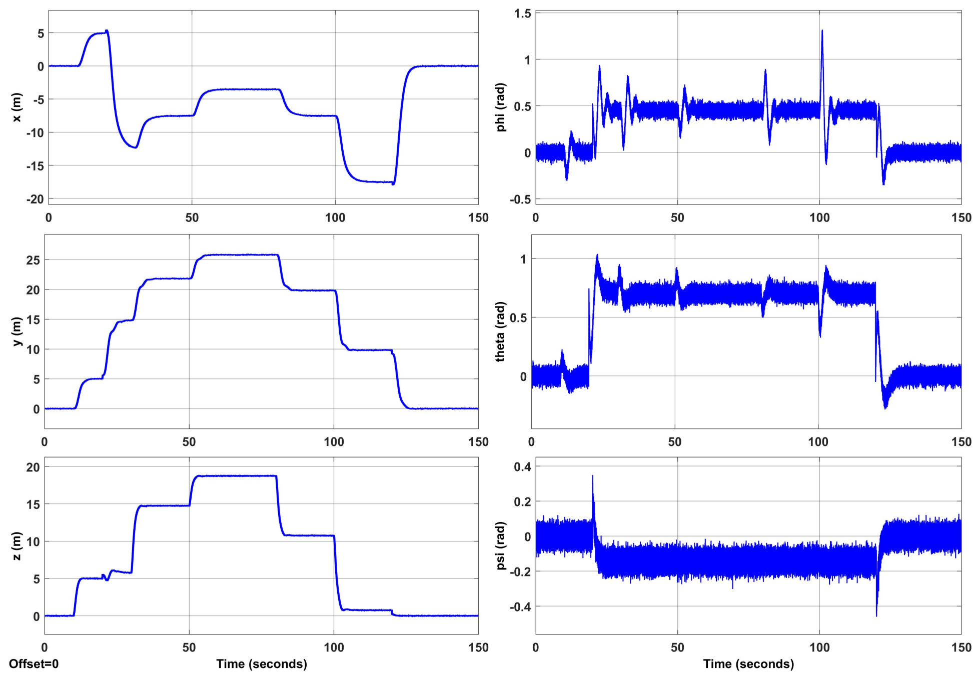

Figure 3.19: Linear position and Euler angles of the quadrotor under sensor fault.

Figures 3.20, 3.21, and 3.22 show the actual, planned, and generated trajectory of the quadrotor. It is shown that the bias sensor faults impact the quadrotor's generated trajectory and actual trajectory, and the quadrotor does not follow the intended trajectory anymore.

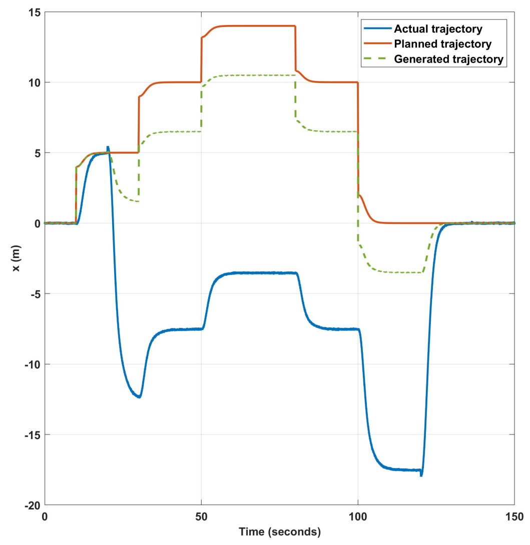

Figure 3.20: The actual, planned, and generated trajectory of the quadrotor in $x$ direction in the presence of bias sensor faults.

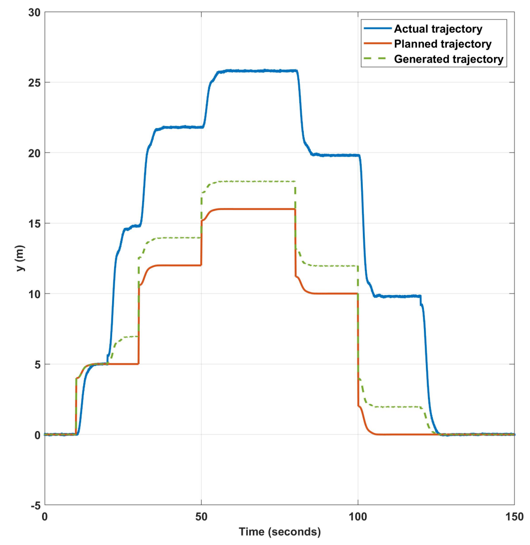

Figure 3.21: The actual, planned, and generated trajectory of the quadrotor in $y$ direction in the presence of bias sensor faults.

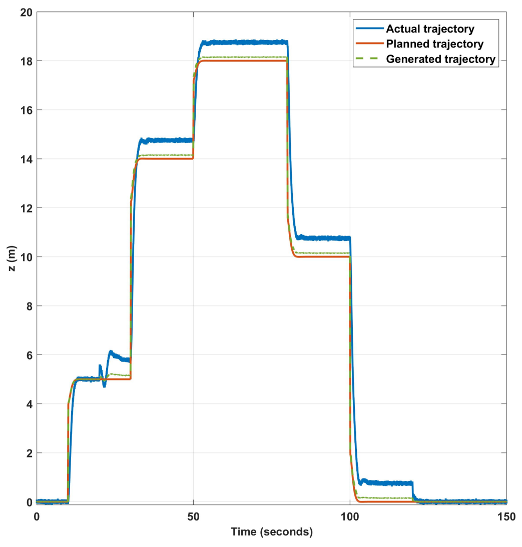

Figure 3.22: The actual, planned, and generated trajectory of the quadrotor in $z$ direction in the presence of bias sensor faults.

Tracking error between the actual and generated trajectory of the quadrotor, shown in Figure 3.23. The tracking error between the generated and actual trajectories in all three directions is evident due to bias sensor faults in the system.

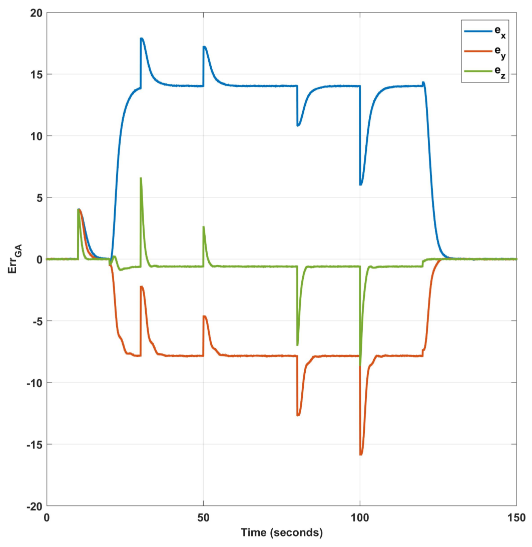

Figure 3.23: The tracking error between the actual, and generated trajectory of the quadrotor in the presence of bias sensor faults.

Tracking error between the planned and generated trajectory of the quadrotor, shown in Figure 3.24. It is shown that during the time interval of [20,120], the planned and generated trajectories do not match, and the presence of sensor faults has impacted the path generation process.

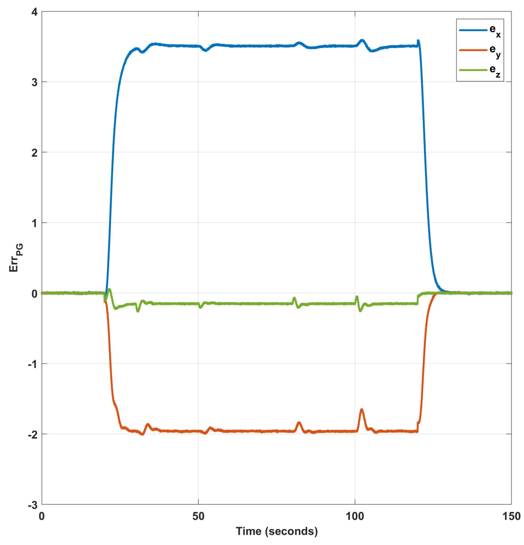

Figure 3.24: The tracking error between the planned, and generated trajectory of the quadrotor in the presence of bias sensor faults.

Tracking error between the planned and actual trajectory of the quadrotor, shown in Figure 3.25. The existence of bias sensor faults results in tracking errors between the intended and real-time trajectories of the quadrotor.

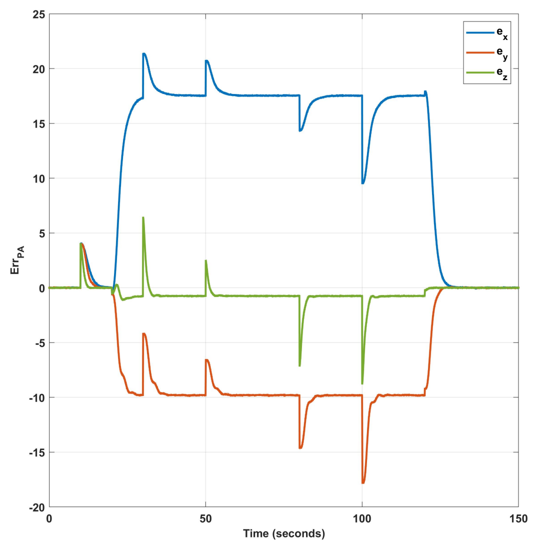

Figure 3.25: The tracking error between the planned, and actual trajectory of the quadrotor in the presence of bias sensor faults.

Figure 3.26 shows the output of the neural network-based sensor fault detector when bias sensor faults are present. As can be seen from the figure, the detector is able to estimate bias sensor faults, which are considered as residuals. When these residuals exceed a certain threshold which is defined in 63, the detector is able to detect the presence of sensor faults. Additionally, the detector is capable of identifying which specific sensors are affected by bias sensor faults.

3.27 illustrates the output of the neural network-based false data injection attack detector when bias sensor faults are present. It can be seen that this detector is triggered during the bias sensor faults. This highlights the significance of virtual sensors on the plant side. The occurrence of bias sensor faults leads to changes in the sensor outputs, which are then transmitted to the command and control center. Thus, the detection mechanism in the command and control center is unable to differentiate between sensor faults and false data injection attacks, as both result in altered measurement data. Virtual sensors are essential for data recovery and isolation of sensor faults and false data injection attacks.

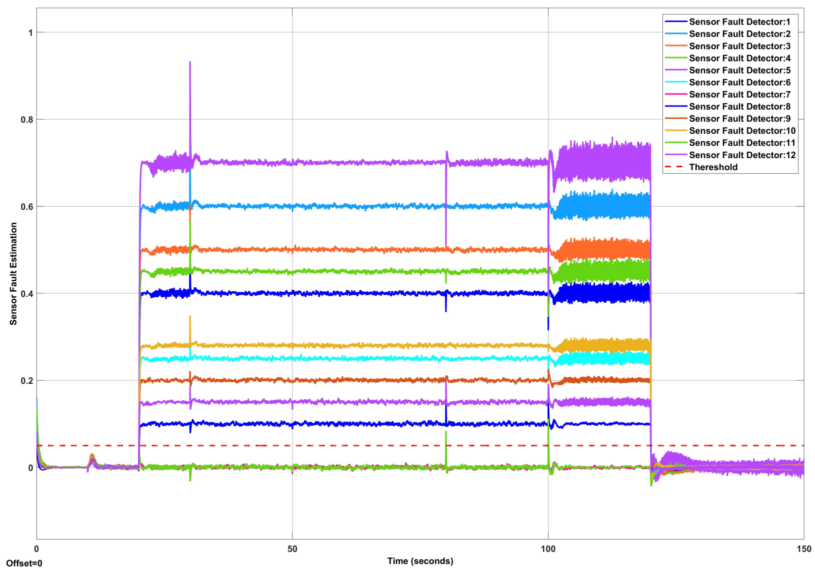

Figure 3.26: The sensor fault detection diagnosis response demonstrates its ability to detect, isolate, and estimate bias sensor faults in the presence of bias sensor faults.

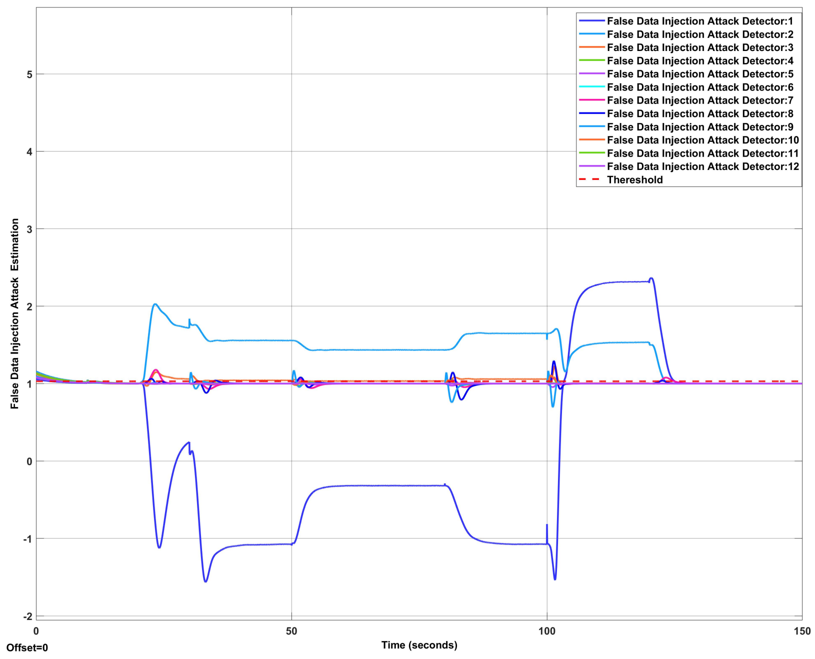

Figure 3.27: The response of the false data injection attack detection diagnosis on the C&C side while the system is under bias sensor faults.

**B) Simulation results with considering virtual sensor**

In this part the simulation results are presented with considering virtual sensors. Figure 3.28 shows how the impact of sensors faults on the quadrotor's linear position and Euler angles is eliminated by using virtual sensors. Therefore, even with the existence of sensor faults, the quadrotor is going to follow the desired trajectory. At the initial stages of the bias sensor faults, small fluctuations or spikes can be seen in the graphs, which represent the time taken by the virtual sensor to effectively mitigate the impact of the sensor faults.

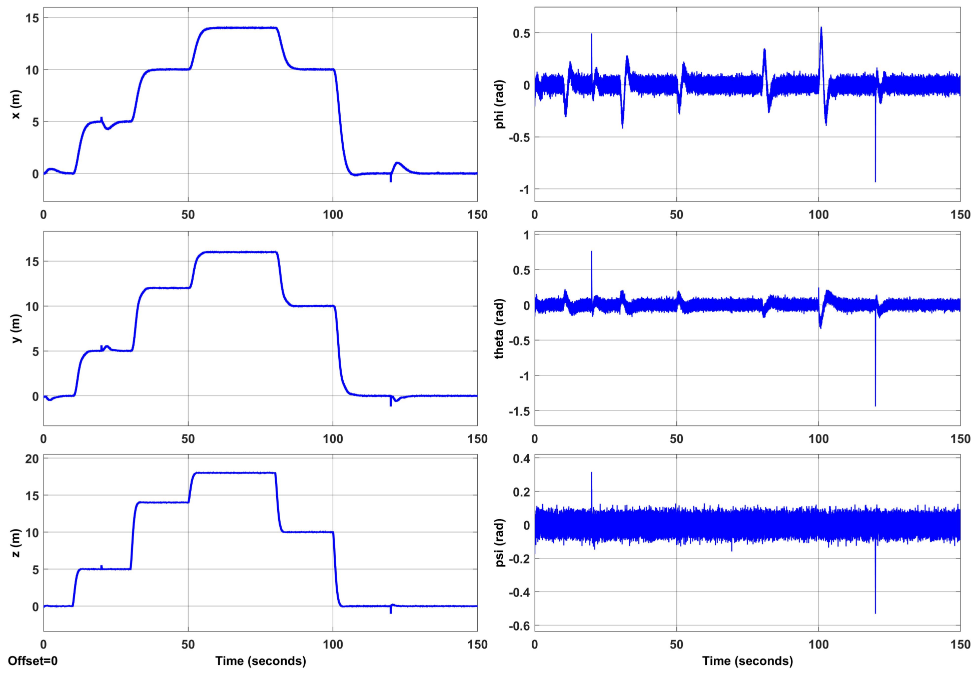

Figure 3.28: Linear position and Euler angles of the quadrotor under sensor fault using a virtual sensor.

Figures 3.29, 3.30, and 3.31 show the actual, planned, and generated trajectory of the quadrotor It has been shown that the effect of bias sensor faults on the generated and actual trajectories of the quadrotor can be mitigated by incorporating a virtual sensor at the plant side. The small fluctuations in the graphs indicate the time taken by the virtual sensor to cancel the impact of the bias sensor faults.

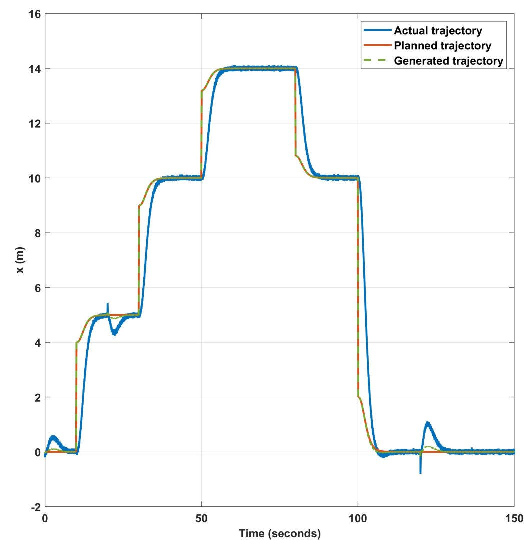

Figure 3.29: The actual, planned, and generated trajectory of the quadrotor in $x$ direction in the presence of bias sensor faults and using a virtual sensor.

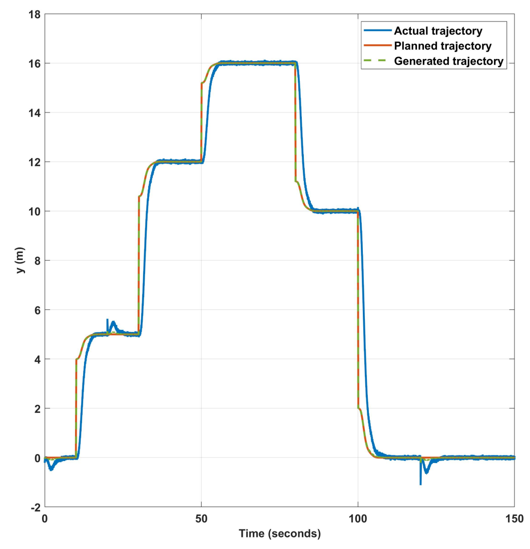

Figure 3.30: The actual, planned, and generated trajectory of the quadrotor in $y$ direction in the presence of bias sensor faults and using a virtual sensor.

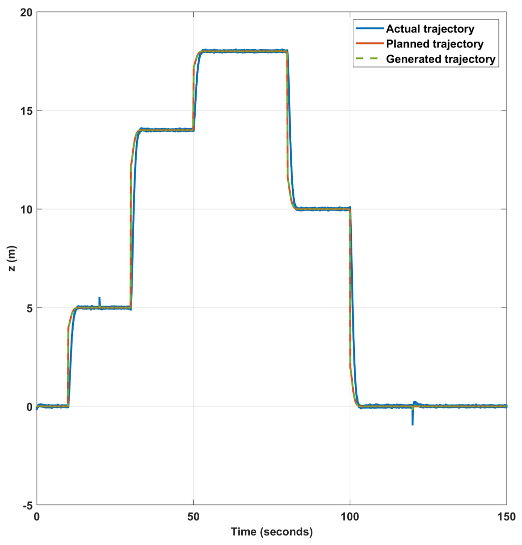

Figure 3.31: The actual, planned, and generated trajectory of the quadrotor in $z$ direction in the presence of bias sensor faults and using a virtual sensor.

Tracking error between the actual and generated trajectory of the quadrotor, shown in Figure 3.32. The abrupt changes in the trajectory of the quadrotor lead to sudden changes in the tracking error. Moreover, some bumps are shown in the graph caused by the virtual sensor.

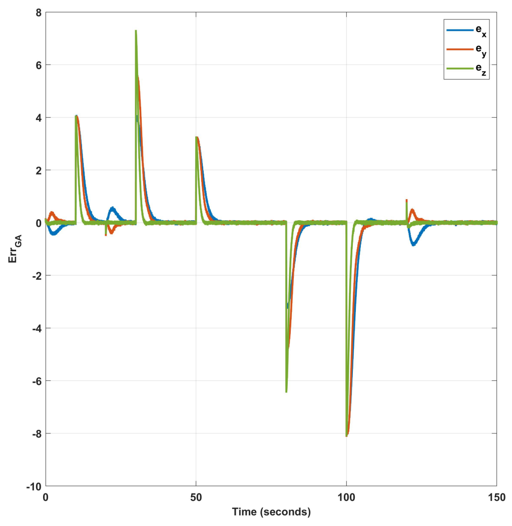

Figure 3.32: The tracking error between the actual, and generated trajectory of the quadrotor in the presence of bias sensor faults and using a virtual sensor.

Tracking error between the planned and generated trajectory of the quadrotor, shown in Figure 3.33. It is shown that the planned and generated trajectories of the quadrotor match, except for some deviations and bumps due to the presence of a virtual sensor on the plant side. It is important to note that the mismatch between the generated and planned trajectories is not as apparent as it was in Figure 3.23.

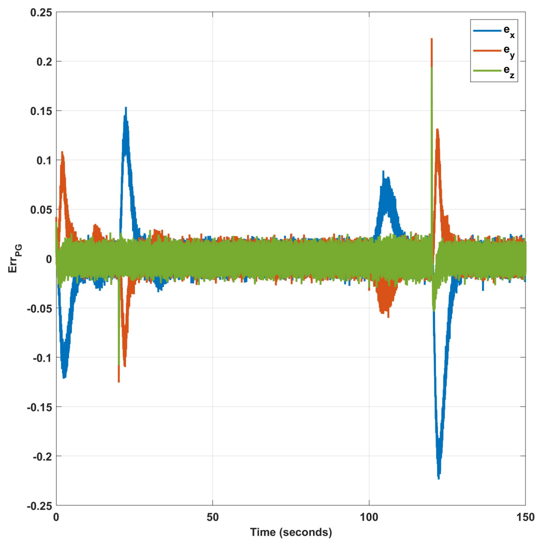

Figure 3.33: The tracking error between the planned, and generated trajectory of the quadrotor in the presence of bias sensor faults and using a virtual sensor.

Tracking error between the planned and actual trajectory of the quadrotor, shown in Figure 3.34. Using a virtual sensor at the plant side ensures that the actual trajectory of the quadrotor follows the planned trajectory. However, there may be some small deviations at the start of the trajectory and at the beginning and end of the fault period, which are caused by the utilization of the virtual sensor at the plant side.

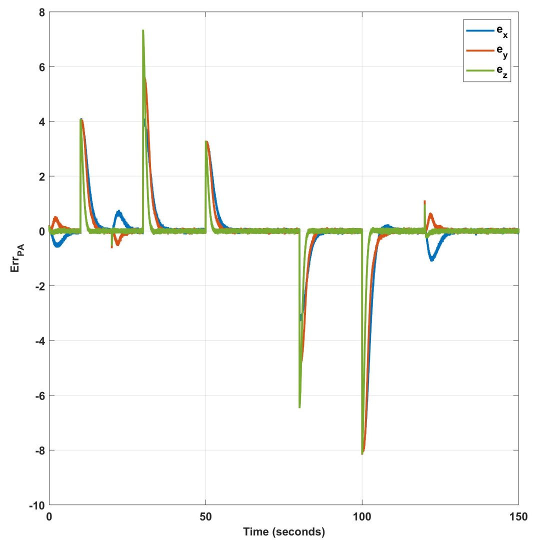

Figure 3.34: The tracking error between the planned, and actual trajectory of the quadrotor in the presence of bias sensor faults and using a virtual sensor.

Figures 3.35 displays the output of the neural network-based sensor fault detector in the presence of bias sensor faults. As shown in the figure, the detector can estimate bias sensor faults and treat them as residuals. When the magnitude of these residuals exceeds a threshold value (defined in page 63), the detector can detect the presence of sensor faults. Furthermore, the detector can identify which specific sensors are affected by bias sensor faults. It is shown that the presence of a virtual sensor does not affect the performance of the bias sensor fault detector at the plant side.

Figure 3.36 depicts the output of the neural network-based detector designed for detecting multiplicative false data injection attacks at the command and control center. The figure shows the detector's performance in the presence of bias sensor faults and with the use of a virtual sensor at the plant side. As can be seen from the figure, the detector does not trigger false alarms in the presence of bias sensor faults when a virtual sensor is employed. The detector briefly activates for less than five seconds before falling below the threshold, as determined through Monte Carlo tests and discussed on page 64.

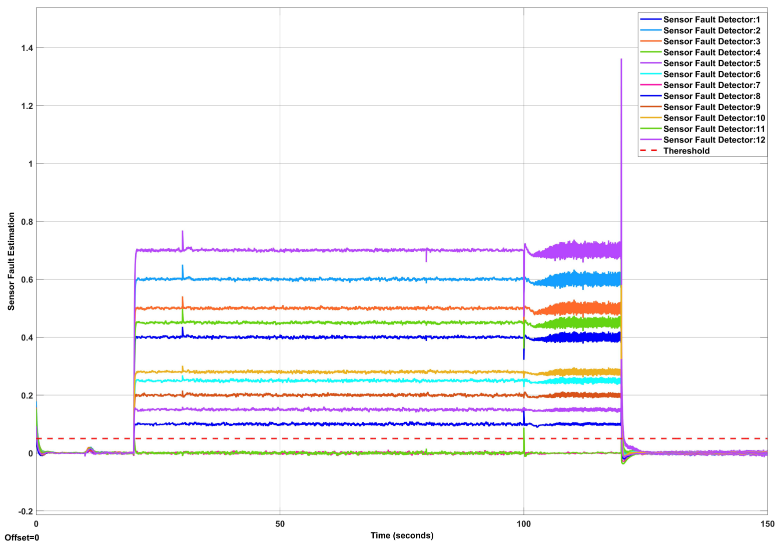

Figure 3.35: The sensor fault detection diagnosis response demonstrates its ability to detect, isolate, and estimate bias sensor faults in the presence of bias sensor faults using virtual sensors.

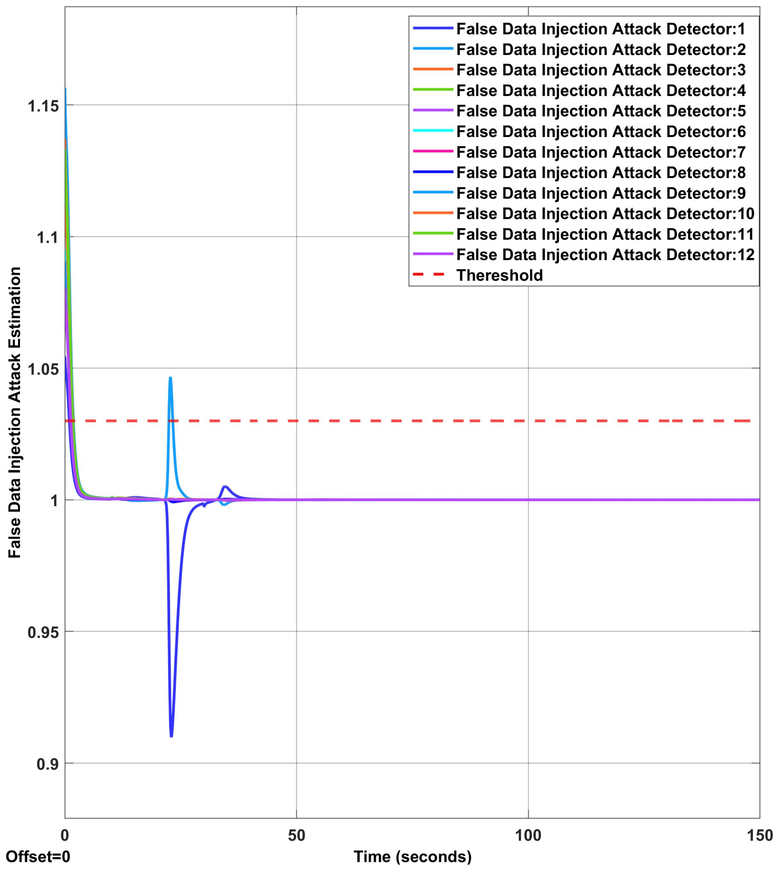

Figure 3.36: The response of the false data injection attack detection diagnosis on the C&C side while the system is under bias sensor faults using virtual sensors.

### 3.5.3 Fault Free and False Data Injection Attacks on the Communication Channel Scenario

In this section, the false data is injected through the output communication channels. It is assumed that the sensors of the quadrotor are not faulty and the system is just under false data injection attacks. The false data is injected through a communication channel in the time interval of [35,80] and multiplied by the sensor data sent from the quadrotor to the command and control center. The attack signal is defined as :

$y_a = [2 \quad 1.5 \quad 1.5 + sin(t) \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1].$

**A) Simulation results without considering virtual sensor**

The Figure 3.37 shows the effect of multiplicative false data injection attacks on the linear position and Euler angles of the quadrotor. As seen in the figure, the multiplicative false data injection attacks result in significant deviations from the intended trajectory, causing the quadrotor's actual path to deviate from its planned trajectory.

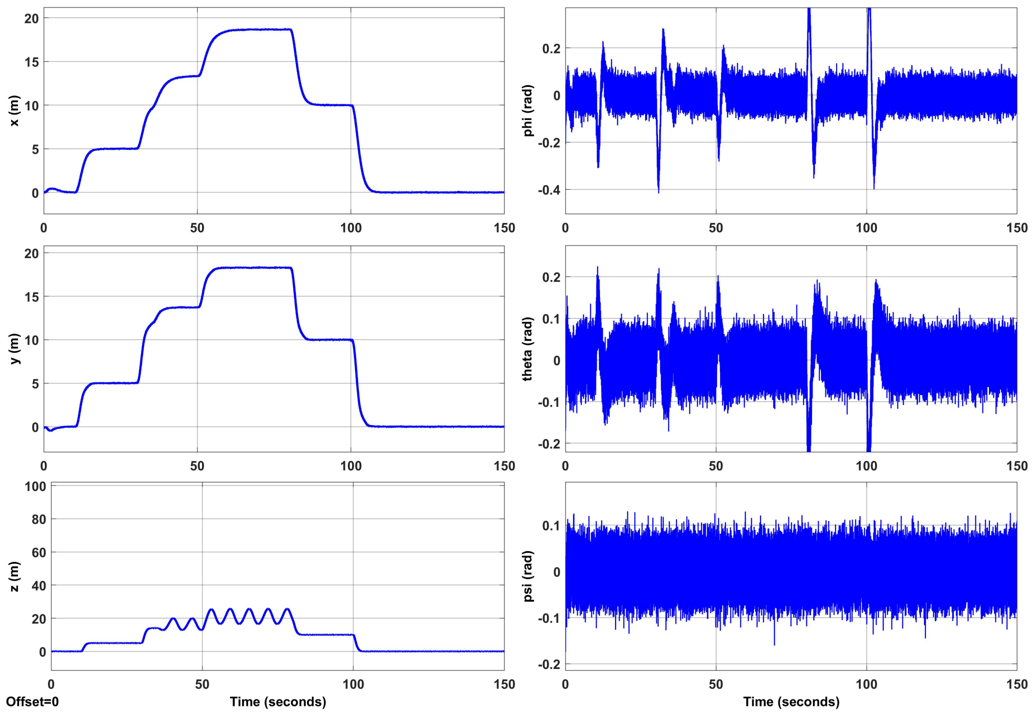

Figure 3.37: Linear position and Euler angles of the quadrotor under false data injection attacks.

Figures 3.38, 3.39, and 3.40 show the actual, planned, and generated trajectory of the quadrotor. It has been shown that when the sensor measurement data are multiplied by the false data injection attacks, the generated trajectory and the actual path in all three directions will not follow the intended trajectory.

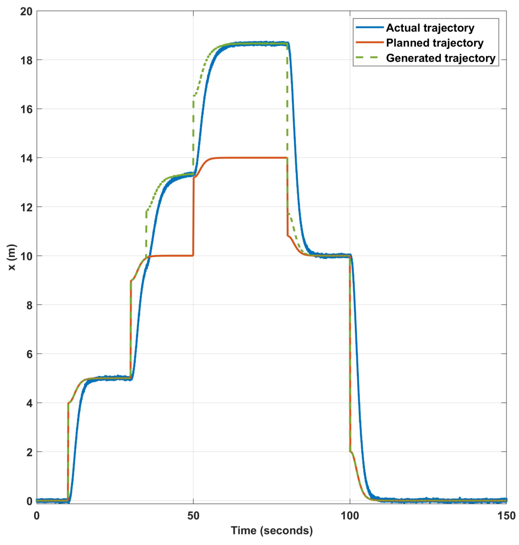

Figure 3.38: The actual, planned, and generated trajectory of the quadrotor in $x$ direction in the presence of multiplicative false data injection attacks.

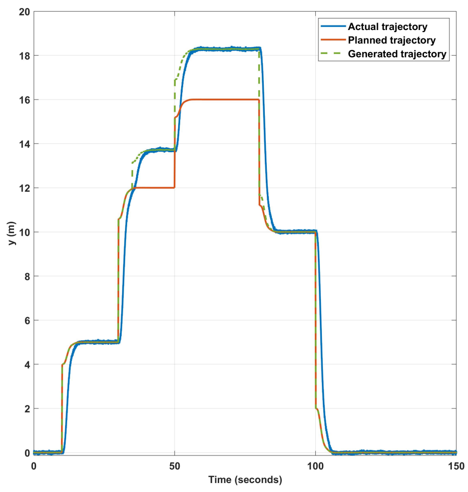

Figure 3.39: The actual, planned, and generated trajectory of the quadrotor in $y$ direction in the presence of multiplicative false data injection attacks.

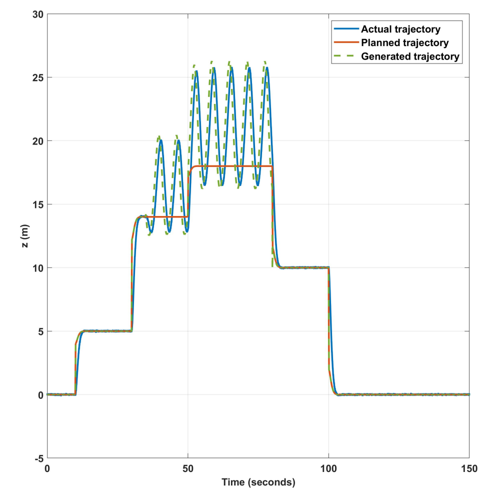

Figure 3.40: The actual, planned, and generated trajectory of the quadrotor in $z$ direction in the presence of multiplicative false data injection attacks.

Tracking error between the actual and generated trajectory of the quadrotor, shown in Figure 3.41. Due to the multiplicative false data injection attacks in the communication link of the quadrotor, a noticeable tracking error between the generated and actual trajectories is evident in all three directions.

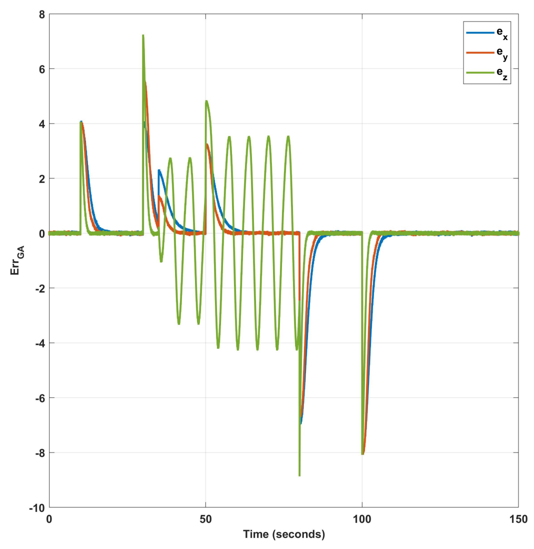

Figure 3.41: The tracking error between the actual, and generated trajectory of the quadrotor in the presence of multiplicative false data injection attacks.

Tracking error between the planned and generated trajectory of the quadrotor, shown in Figure 3.42. It is shown that during the time interval of [35,80], the planned and generated trajectories do not match, and the multiplicative false data injection attacks in the communication link has impacted the path generation process.

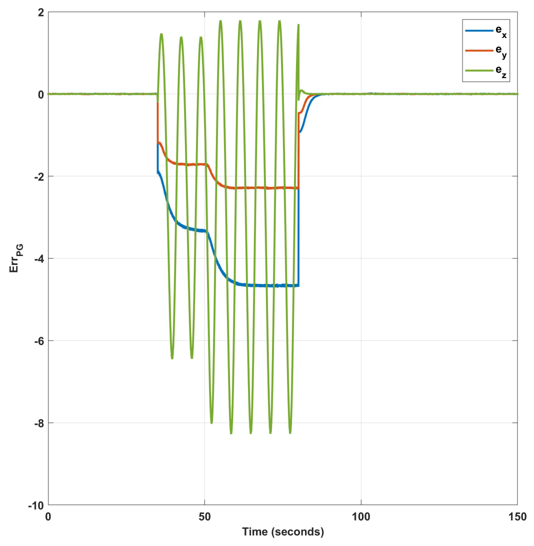

Figure 3.42: The tracking error between the planned, and generated trajectory of the quadrotor in the presence of multiplicative false data injection attacks.

Tracking error between the planned and actual trajectory of the quadrotor, shown in Figure 3.43. The existence of multiplicative false data injection attacks in the communication link leads to rising tracking errors between the intended and real-time trajectories of the quadrotor.

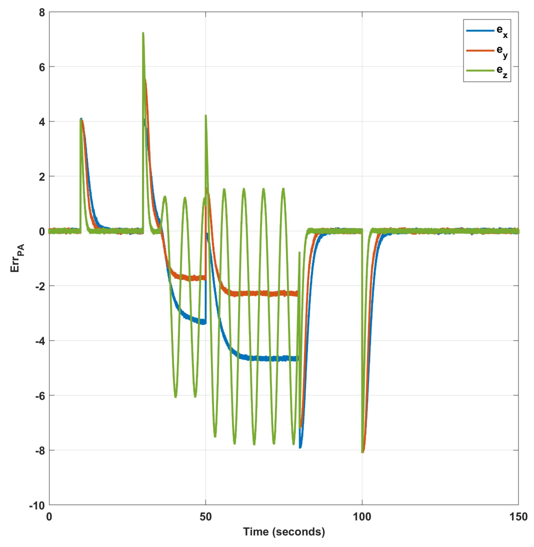

Figure 3.43: The tracking error between the planned, and actual trajectory of the quadrotor in the presence of multiplicative false data injection attacks.

Figures 3.44 shows the output of the neural network-based sensor fault detector in the presence of multiplicative false data injection attacks. As it is shown in figure, the sensor fault detector does not trigger when multiplicative false data injection attacks are present. However, the detector experiences bumps at the starting points where the path suddenly changes.

Figure 3.45 depicts the effectiveness of the neural network-based false data injection attack detector, which can detect, isolate, and estimate the multiplicative false data injection attacks at the appropriate time and with high accuracy. The figure shows that the detector can estimate the multiplicative false data injection attacks and consider them as residuals. When the magnitude of the residuals exceeds a threshold value (defined in page 64), the detector can detect the presence of multiplicative false data injection attacks. Furthermore, the detector can identify which specific sensor measurement data are affected by multiplicative false data injection attacks.

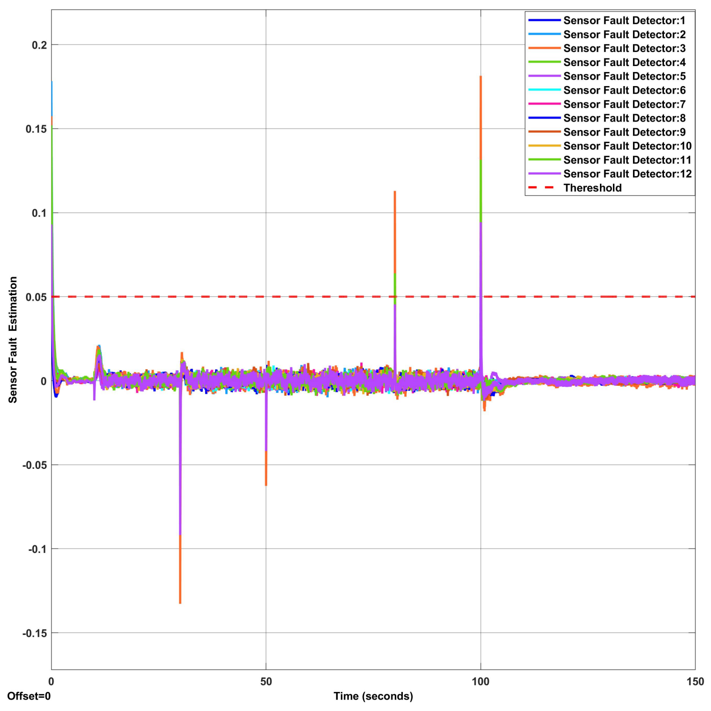

Figure 3.44: The response of the sensor fault detection diagnosis on the plant side while the system is under multiplicative false data injection attacks.

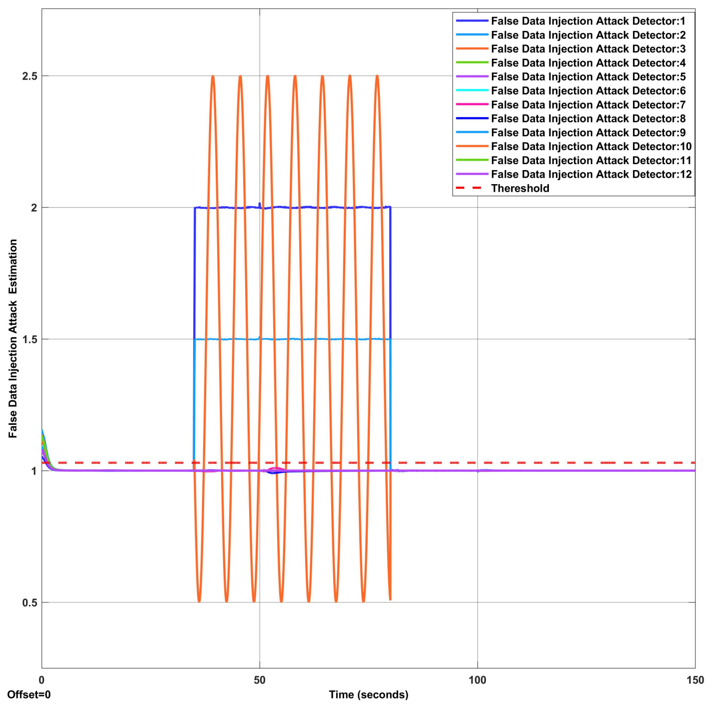

Figure 3.45: The response of the False Data Injection attack detection diagnosis on the C&C side demonstrates its ability to detect, isolate, and estimate the severity of an attack in the presence of multiplicative false data injection attacks.

**B) Simulation results with considering virtual sensor**

In this part, the simulation results are presented by considering a virtual sensor at the command and control center to recover sensor measurement data from multiplicative false data injection attacks. Figure 3.46 demonstrates the elimination of the impact of multiplicative false data injection attacks on the linear position and Euler angles of the quadrotor through the utilization of virtual sensor at the command and control center. As a result, the quadrotor will follow the intended trajectory even in the presence of multiplicative false data injection attacks.

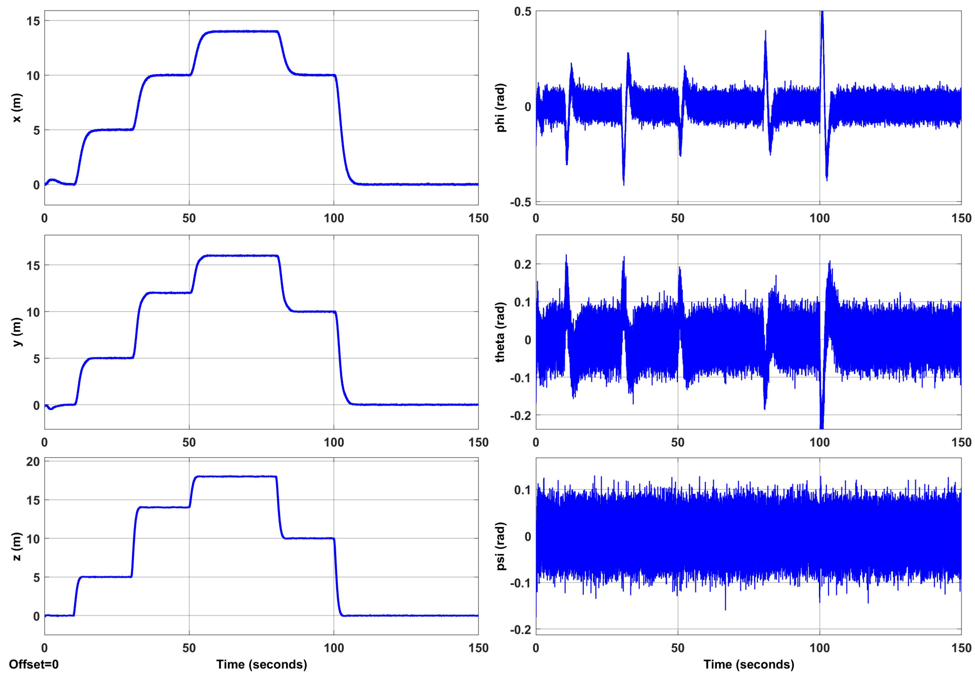

Figure 3.46: Linear position and Euler angles of the quadrotor under false data injection attacks using a virtual sensor.

Figures 3.47, 3.48, and 3.49 show the actual, planned, and generated trajectory of the quadrotor. It has been shown that the effect of multiplicative false data injection attacks on the generated and actual trajectories of the quadrotor can be mitigated by incorporating a virtual sensor at the command and control side.

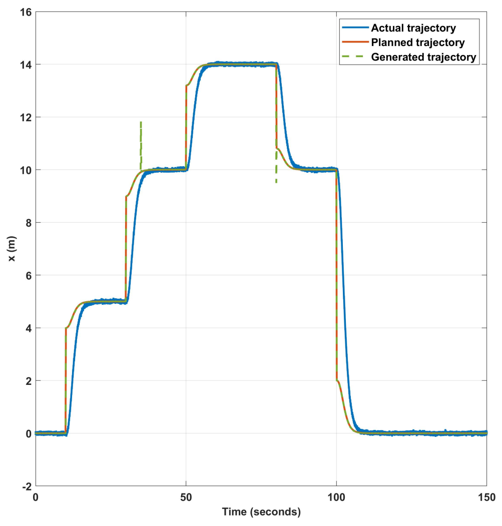

Figure 3.47: The actual, planned, and generated trajectory of the quadrotor in $x$ direction in the presence of multiplicative false data injection attacks and using a virtual sensor.

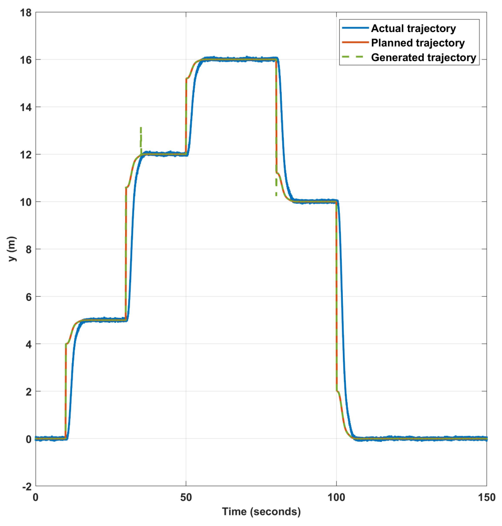

Figure 3.48: The actual, planned, and generated trajectory of the quadrotor in $y$ direction in the presence of multiplicative false data injection attacks and using a virtual sensor.

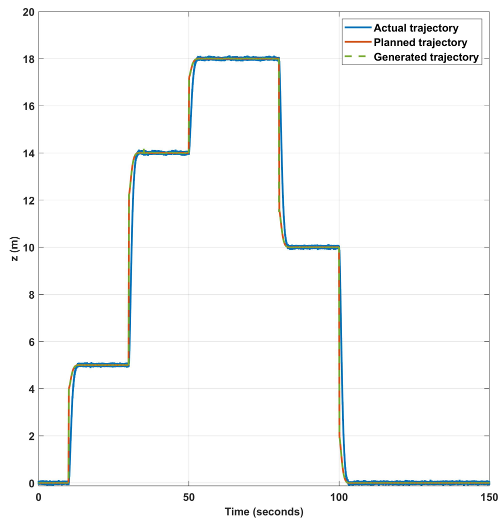

Figure 3.49: The actual, planned, and generated trajectory of the quadrotor in $z$ direction in the presence of multiplicative false data injection attacks and using a virtual sensor.

Tracking error between the actual and generated trajectory of the quadrotor, shown in Figure 3.50. The abrupt changes in the trajectory of the quadrotor lead to sudden changes in the tracking error.

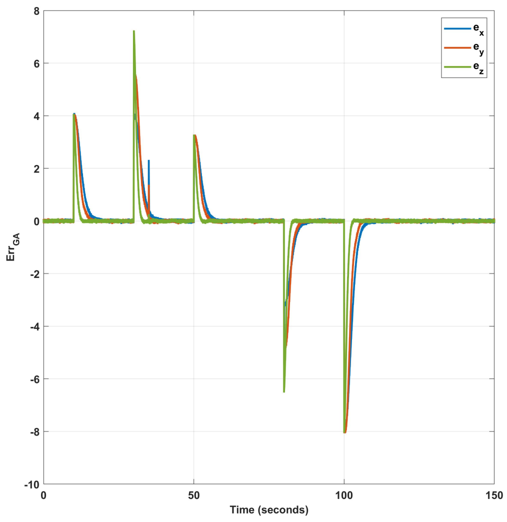

Figure 3.50: The tracking error between the actual, and generated trajectory of the quadrotor in the presence of multiplicative false data injection attacks and using a virtual sensor.

Tracking error between the planned and generated trajectory of the quadrotor, shown in Figure 3.51. It is demonstrated that the planned and generated trajectories of the quadrotor match, except for minor deviations at the start and end time intervals where false data injection attacks have been multiplied by the sensor measurement data. It is worth noting that the mismatch between the generated and planned trajectories is less noticeable compared to what is shown in Figure 3.41.

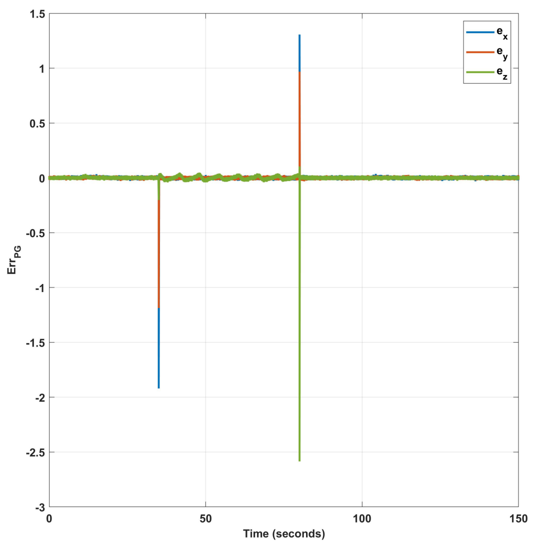

Figure 3.51: The tracking error between the planned, and generated trajectory of the quadrotor in the presence of multiplicative false data injection attacks and using a virtual sensor.

Tracking error between the planned and actual trajectory of the quadrotor, shown in Figure 3.52. It can be seen that utilizing a virtual sensor at the command and control side ensures that the actual trajectory of the quadrotor follows the intended trajectory.

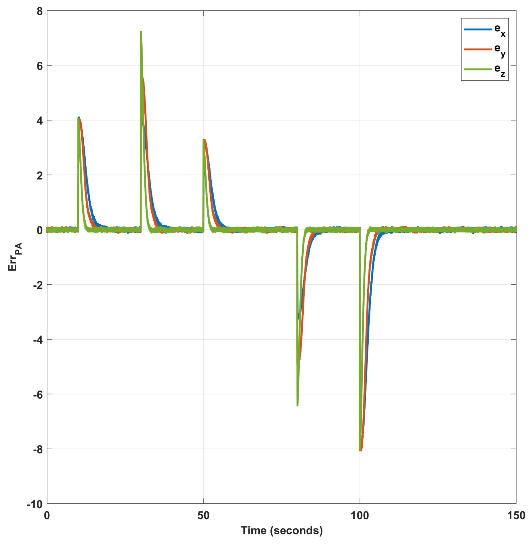

Figure 3.52: The tracking error between the planned, and actual trajectory of the quadrotor in the presence of multiplicative false data injection attacks and using a virtual sensor.

Figures 3.53 and 3.54 show the outputs of the detectors during multiplicative false data injection attacks on the system. The performance of these two neural network-based detectors is similar to the situation in which there is no virtual sensor on the command and control side. The sensor fault detector does not trigger during multiplicative false data injection attacks. Still, it shows a bump at the start time of the path changing. The neural network-based false data injection attack detector can effectively detect, isolate, and estimate the attack with high accuracy.

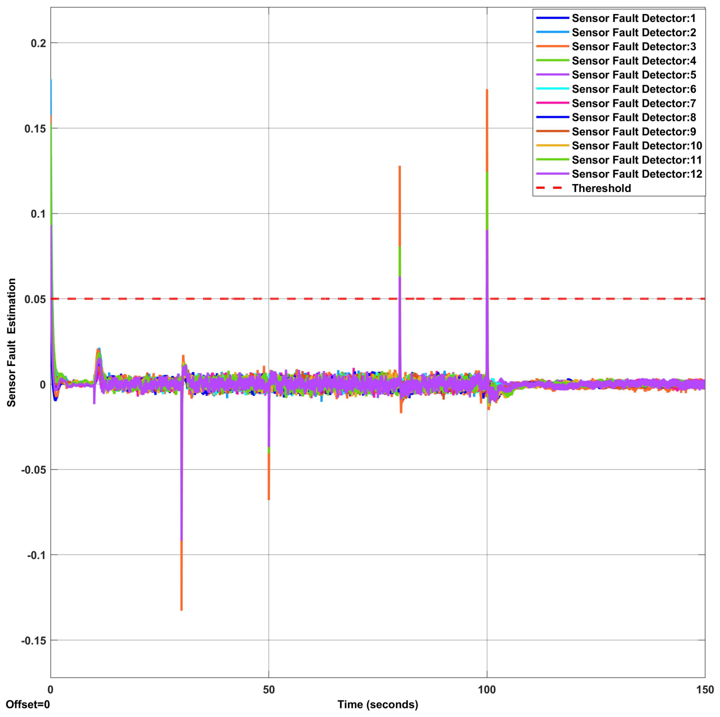

Figure 3.53: The response of the sensor fault detection diagnosis on the plant side while the system is under multiplicative false data injection attacks using virtual sensors.

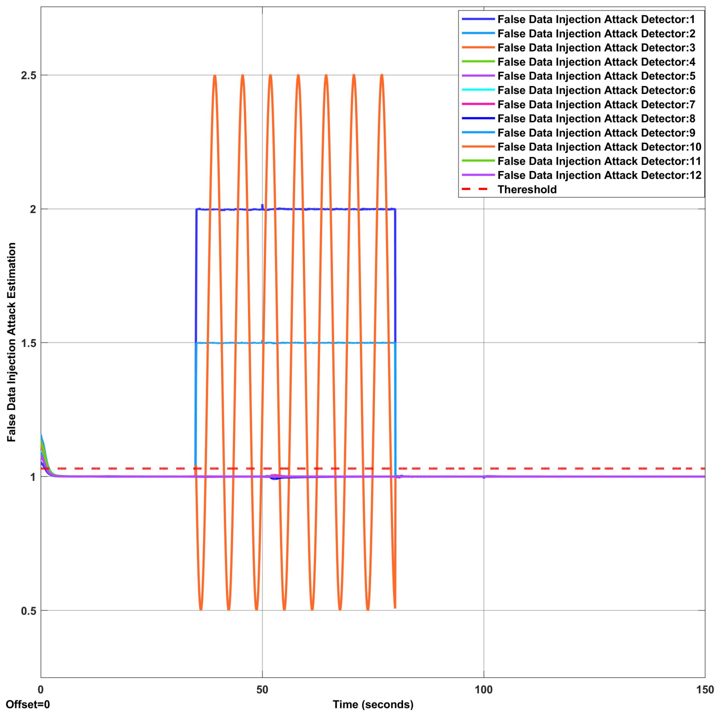

Figure 3.54: The response of the False Data Injection attack detection diagnosis on the C&C side demonstrates its ability to detect, isolate, and estimate the severity of an attack in the presence of multiplicative false data injection attacks and using virtual sensors.

### 3.5.4 Bias Faults in Sensors and Multiplicative False Data Injection Attacks on the Communication Channels Scenario

This section considers the scenario with multiplicative false data injection at the output communication channel and sensor faults. The false data is injected through a communication channel in the time interval of [35,80] and multiplied by the sensor data sent from the quadrotor to the command and control center. The attack signal is defined as :

$y_a = [2 \quad 1.5 \quad 1.5 + sin(t) \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1]$.

The bias sensor faults are considered in the time interval of [20,120], and it is considered that ten of the sensors are faulty. The fault signal is defined as:

$T_s = [0.4 \quad 0.6 \quad 0.5 \quad 0.45 \quad 0.7 \quad 0.25 \quad 0 \quad 0.1 \quad 0.2 \quad 0.28 \quad 0 \quad 0.15]$.

**A) Simulation results without considering virtual sensors**

The Figure 3.55 shows the effect of multiplicative false data injection attacks and sensor faults on the linear position and Euler angles of the quadrotor. As seen in the figure, the presence of multiplicative false data injection attacks and bias sensor faults result in significant changes in the states of the quadrotor.

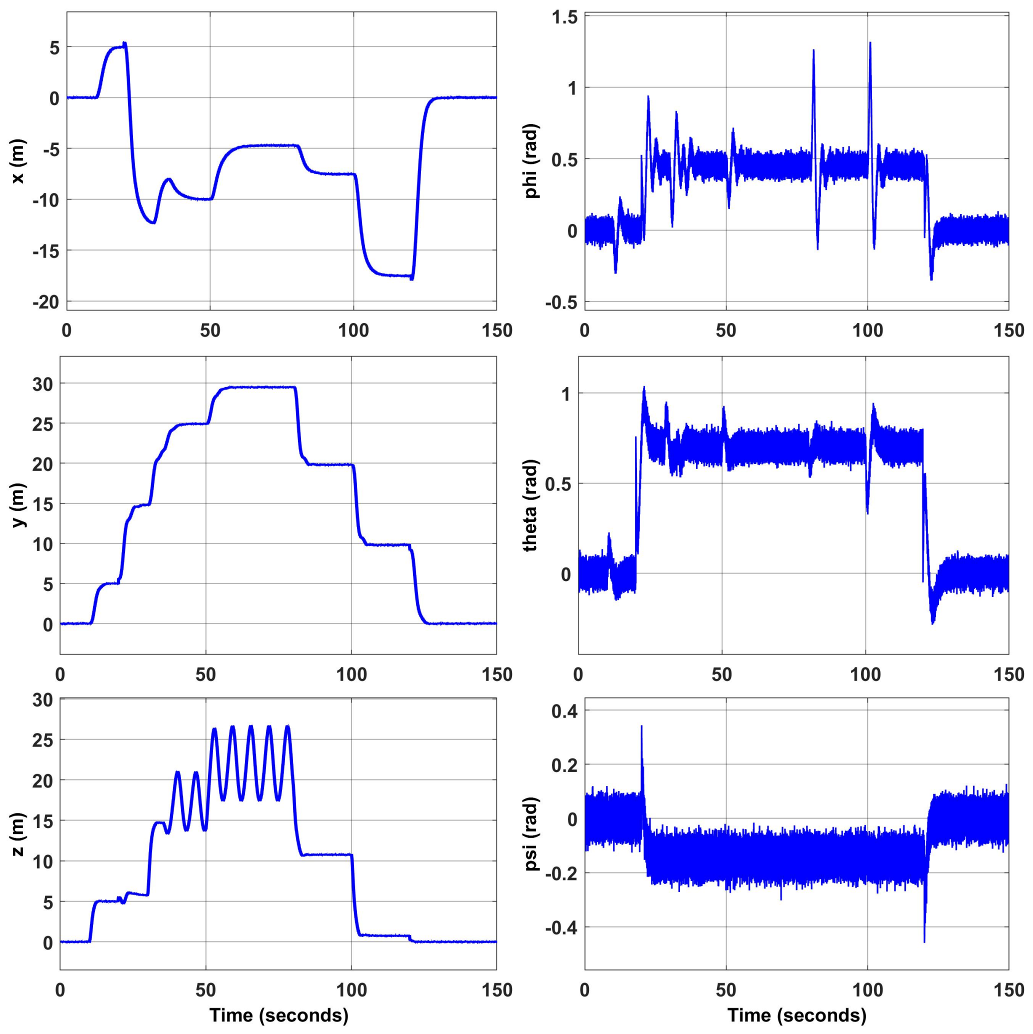

Figure 3.55: Linear position and Euler angles of the quadrotor under false data injection attacks and sensor faults.

Figures 3.56, 3.57, and 3.58 show the actual, planned, and generated trajectory of the quadrotor. It has been demonstrated that when the bias sensor faults exist, and sensor measurement data are affected by multiplicative false data injection attacks in the communication channel, the generated trajectory and actual path in all three directions deviate significantly from the planned trajectory and do not follow the intended path.

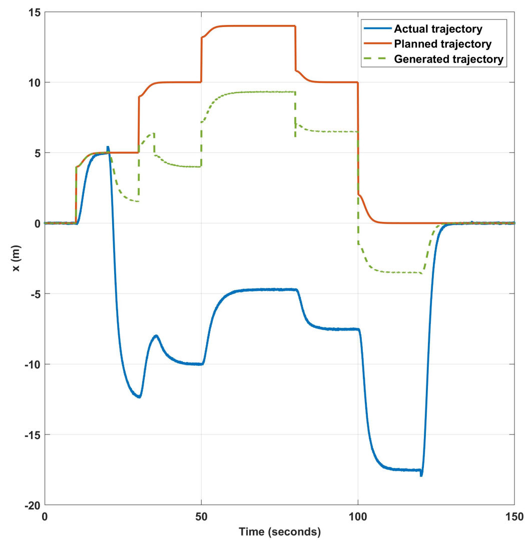

Figure 3.56: The actual, planned, and generated trajectory of the quadrotor in $x$ direction in the presence of multiplicative false data injection attacks and sensor faults.

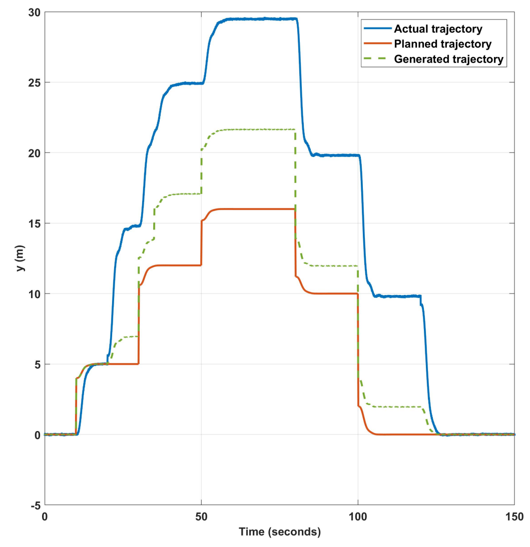

Figure 3.57: The actual, planned, and generated trajectory of the quadrotor in $y$ direction in the presence of multiplicative false data injection attacks and sensor faults.

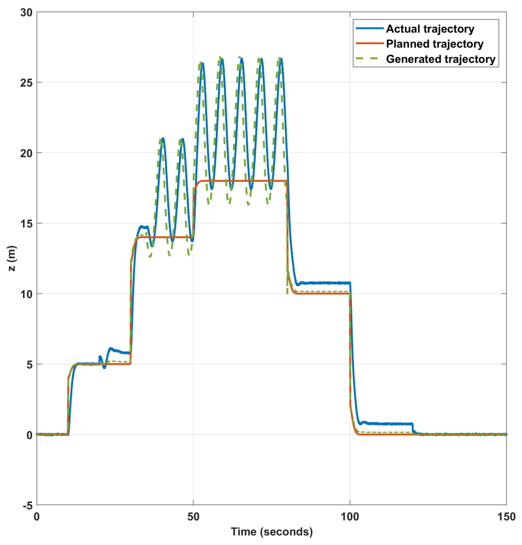

Figure 3.58: The actual, planned, and generated trajectory of the quadrotor in $z$ direction in the presence of multiplicative false data injection attacks and sensor faults.

Tracking error between the actual and generated trajectory of the quadrotor, shown in Figure 3.59. Due to the multiplicative false data injection attacks in the communication link of the quadrotor and sensor faults, a significant tracking error between the generated and actual trajectories is apparent in all three directions.

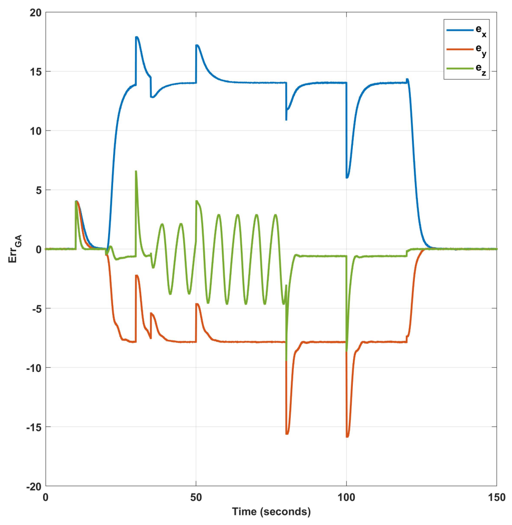

Figure 3.59: The tracking error between the actual, and generated trajectory of the quadrotor in the presence of multiplicative false data injection attacks and sensor faults.

Tracking error between the planned and generated trajectory of the quadrotor, shown in Figure 3.60. It is shown that the planned and generated trajectories do not match, and the multiplicative false data injection attacks in the communication link and sensor faults have impacted the path generation process.

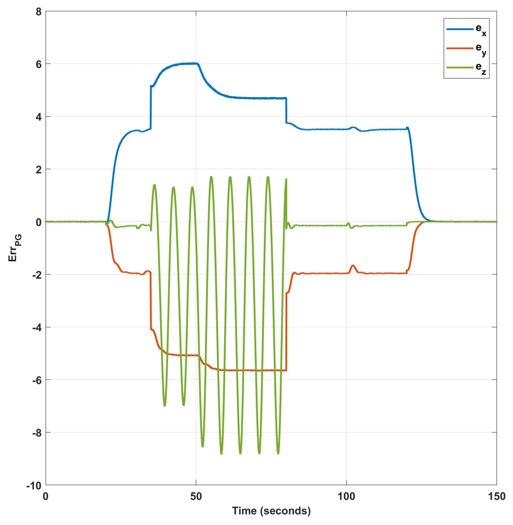

Figure 3.60: The tracking error between the planned, and generated trajectory of the quadrotor in the presence of multiplicative false data injection attacks and sensor faults.

Tracking error between the planned and actual trajectory of the quadrotor, shown in Figure 3.61. The existence of multiplicative false data injection attacks in the communication link and sensor faults resulted in significant tracking errors between the intended and real-time trajectories of the quadrotor.

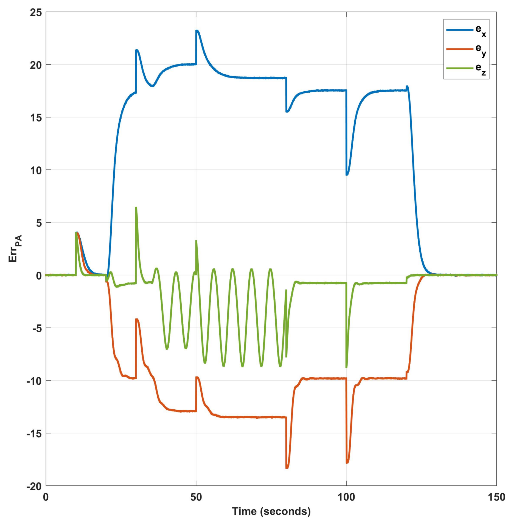

Figure 3.61: The tracking error between the planned, and actual trajectory of the quadrotor in the presence of multiplicative false data injection attacks and sensor faults.

Figure 3.62 illustrates the output of the neural network-based sensor fault detector when both bias sensor faults and multiplicative false data injection attacks are present. As can be seen from the figure, the detector is capable of estimating bias sensor faults and considering them as residuals. When the magnitude of these residuals exceeds the threshold value, which is defined in page 63, the detector can detect the presence of sensor faults and identify which specific sensors are affected by bias sensor faults. Importantly, the detector does not trigger false alarms for the multiplicative false data injection attacks, as demonstrated in the figure. It should be noted that because of sudden path changes, the detector has bumps at the starting points the path is going to change.

Figure 3.63 shows that the false data injection attack detector is triggered for both sensor faults and false data injection attacks. Unfortunately, the bias sensor faults impact the neural network-based detector in the command and control center, making it disable to isolate both sensor faults and false data injection attacks, as well as leading to improper estimation of the false data injection attacks.

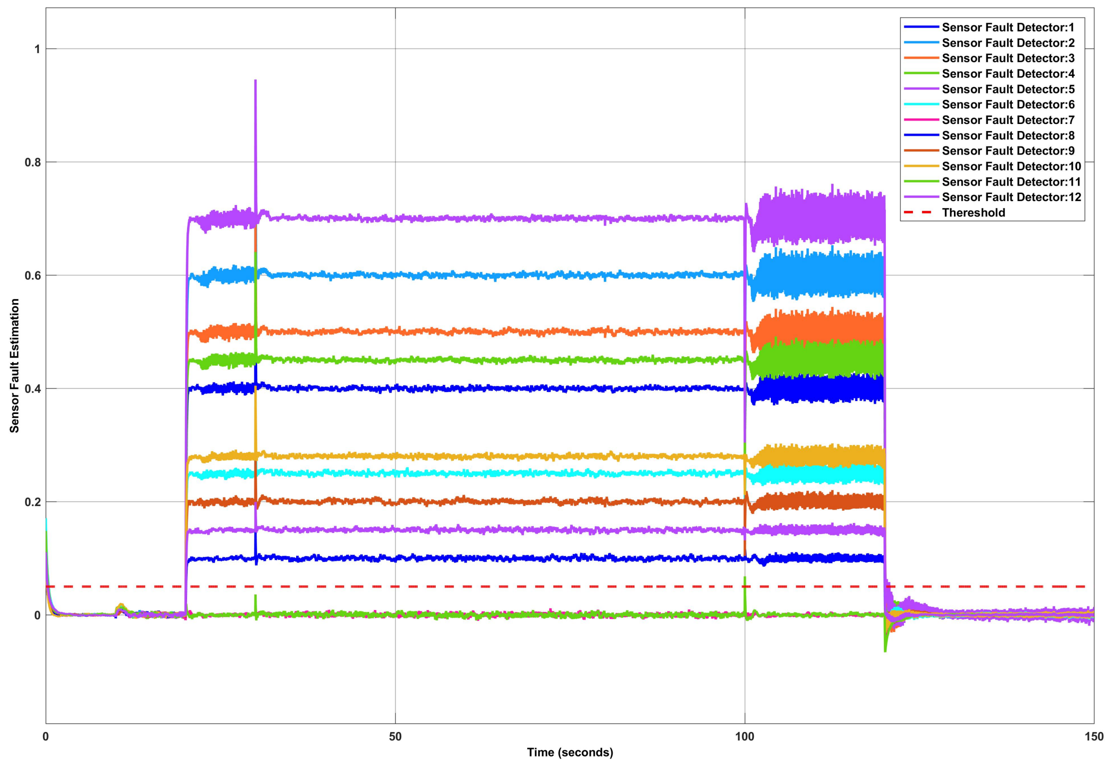

Figure 3.62: The sensor fault detection diagnosis response demonstrates its ability to detect, isolate, and estimate bias sensor faults in the presence of bias sensor faults and False Data Injection attacks.

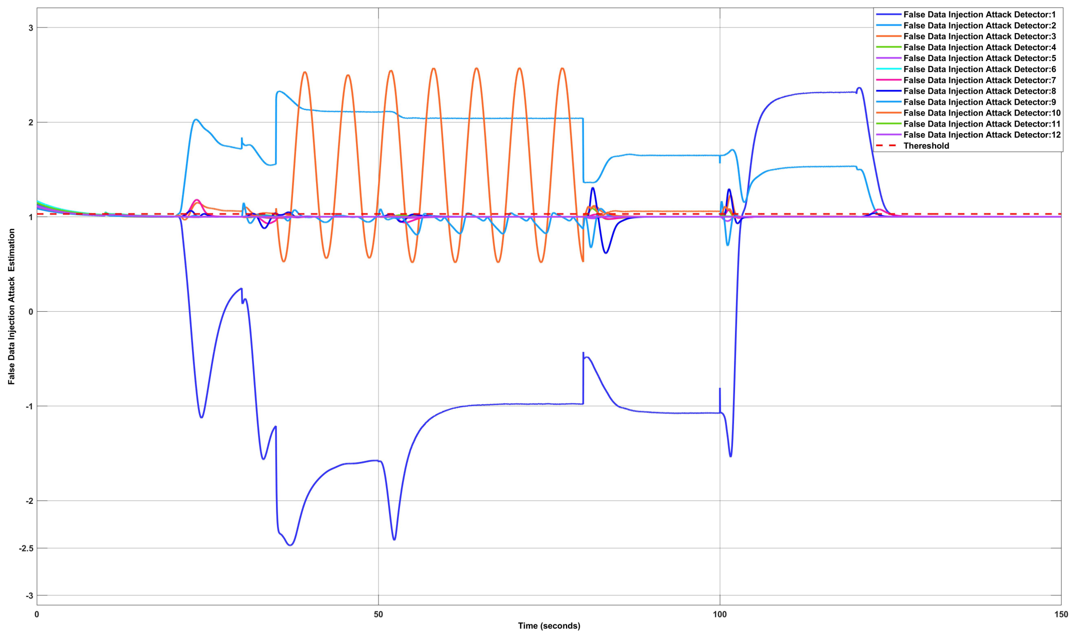

Figure 3.63: The response of the false data injection attack detection diagnosis on the C&C side while the system is under multiplicative false data injection attacks and sensor faults.

**B) Simulation results with considering virtual sensors**

This section presents the simulation results by considering virtual sensors at the command and control center and plant side. The virtual sensor at the plant side is used to recover the sensor measurement data from bias faults, while at the command and control side, a virtual sensor is employed to recover sensor measurement data that are subjected to multiplicative false data injection attacks.

Figure 3.64 depicts the elimination of the impact of multiplicative false data injection attacks and sensor faults on the linear position and Euler angles of the quadrotor through the utilization of virtual sensors at the command and control center and plant side. As a result, the quadrotor will follow the intended trajectory even in the presence of multiplicative false data injection attacks and sensor faults. There are some bumps in the quadrotor states graphs which are resulted from using virtual sensors and the time that it takes to recover the data.

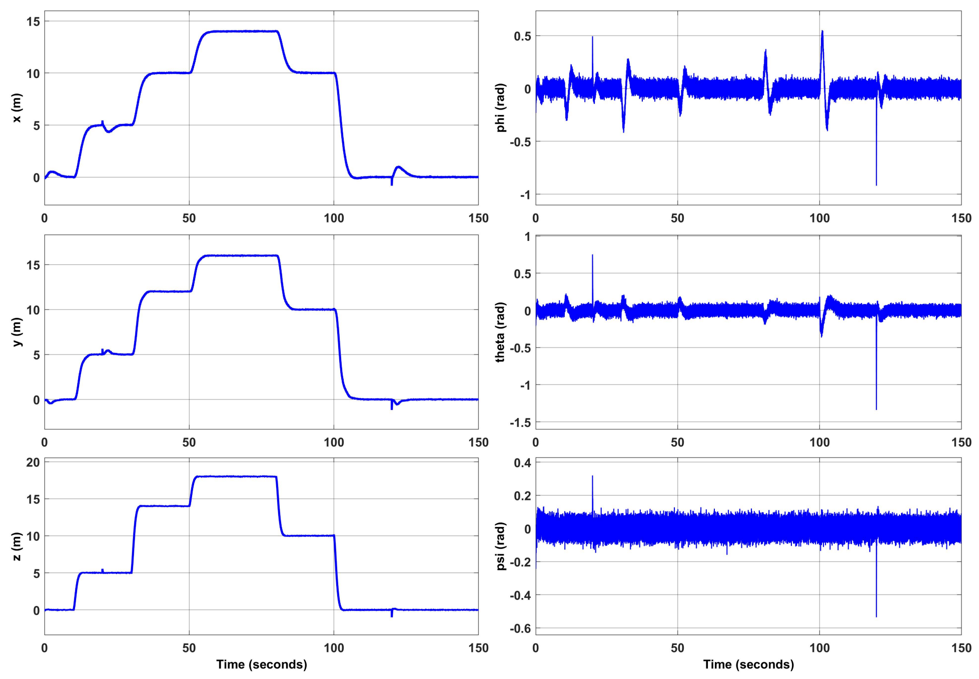

Figure 3.64: Linear position and Euler angles of the quadrotor under false data injection attacks and sensor faults using virtual sensors.

Figures 3.65, 3.66, and 3.67 show the actual, planned, and generated trajectory of the quadrotor. It has been shown that the effect of multiplicative false data injection attacks on the generated and actual trajectories of the quadrotor can be mitigated by incorporating virtual sensors at the plant side and command and control side. The small rises visible in the figures indicate the time taken by the virtual sensors to recover data.

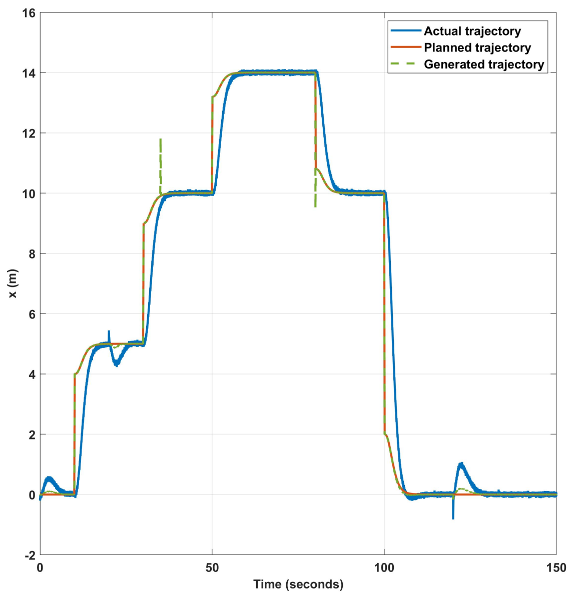

Figure 3.65: The actual, planned, and generated trajectory of the quadrotor in $x$ direction in the presence of sensor faults and multiplicative false data injection attacks and using virtual sensors.

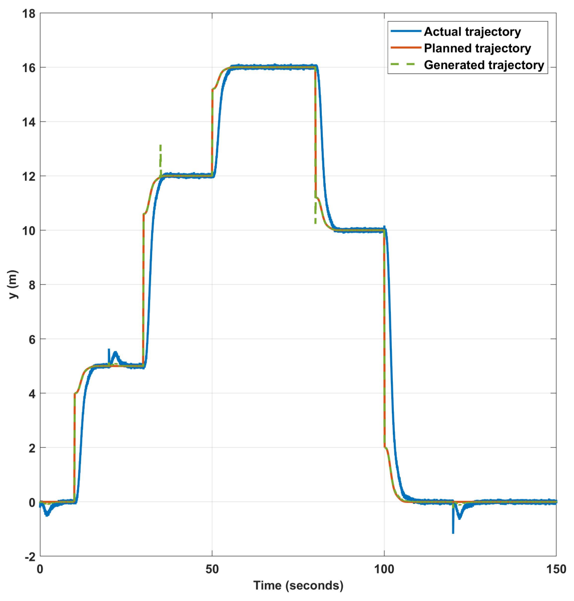

Figure 3.66: The actual, planned, and generated trajectory of the quadrotor in $y$ direction in the presence of sensor faults and multiplicative false data injection attacks and using virtual sensors.

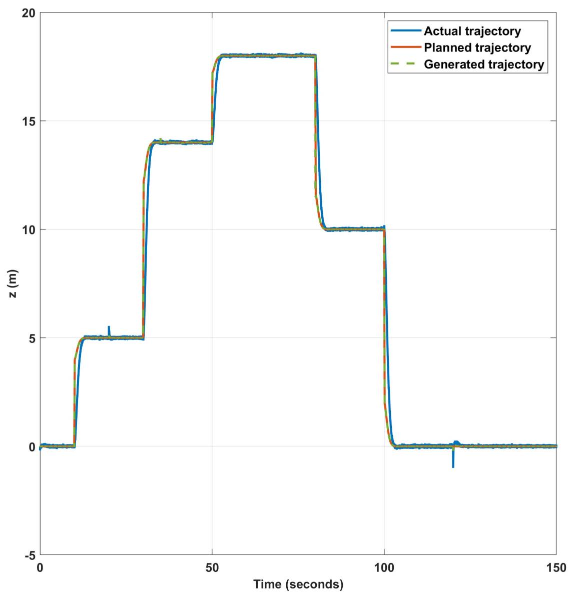

Figure 3.67: The actual, planned, and generated trajectory of the quadrotor in $z$ direction in the presence of sensor faults and multiplicative false data injection attacks and using virtual sensors.

Tracking error between the actual and generated trajectory of the quadrotor, shown in Figure 3.68. The abrupt changes in the trajectory of the quadrotor lead to sudden changes in the tracking error. Moreover, some rises are shown in the figure and are caused by the virtual sensors.

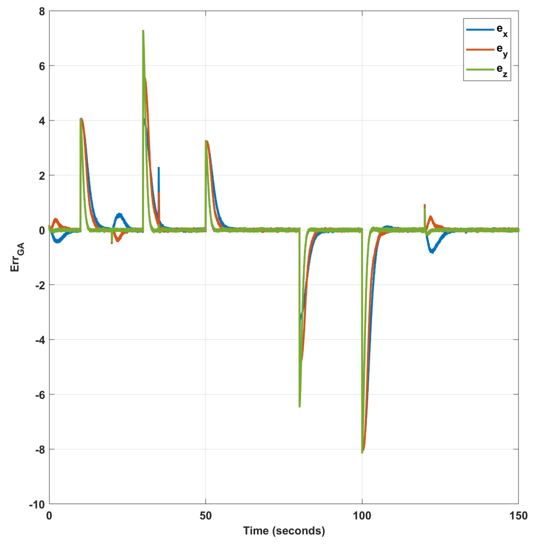

Figure 3.68: The tracking error between the actual, and generated trajectory of the quadrotor in the presence of sensor faults and multiplicative false data injection attacks and using virtual sensors.

Tracking error between the planned and generated trajectory of the quadrotor, shown in Figure 3.69. It is shown that the planned and generated trajectories of the quadrotor match, except for minor deviations at the start and end time intervals where false data injection attacks have been multiplied by the sensor measurement data. Moreover, some jumps are shown in the figure, which is resulted from the virtual sensor that is used at the plant side for sensor fault recovery. It should be mentioned that the mismatch between the generated and planned trajectories is less noticeable compared to what is shown in Figure 3.59.

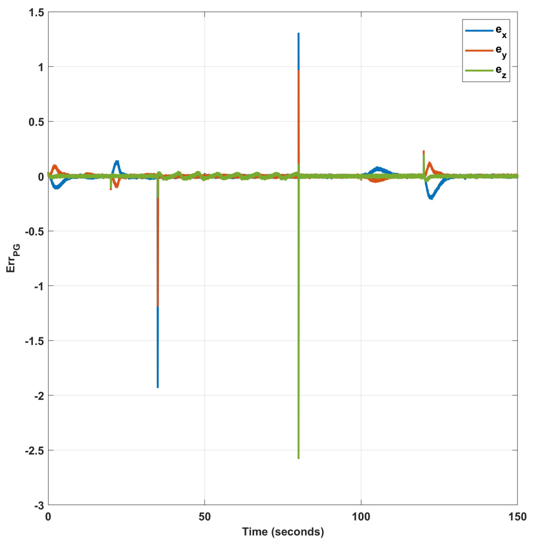

Figure 3.69: The tracking error between the planned, and generated trajectory of the quadrotor in the presence of sensor faults and multiplicative false data injection attacks and using virtual sensors.

Tracking error between the planned and actual trajectory of the quadrotor, shown in Figure 3.70. It can be seen that employing virtual sensors at the command and control and plant sides confirms that the real-time trajectory of the quadrotor follows the intended trajectory. However, because of sudden path changes, the detector has raises at the starting points the path is going to change. Additionally, some bumps are existed due to the virtual sensor, which is used for sensor fault recovery.

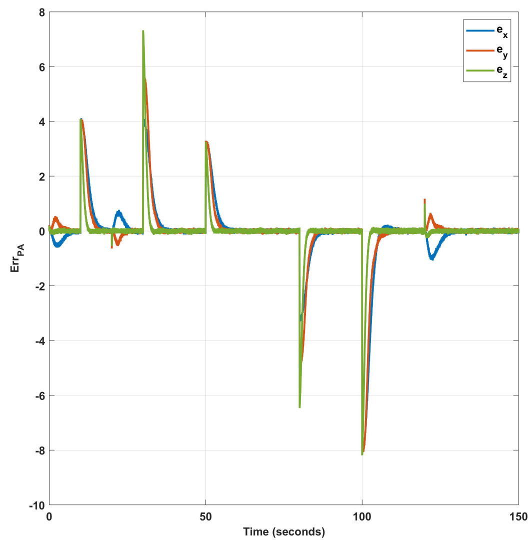

Figure 3.70: The tracking error between the planned, and actual trajectory of the quadrotor in the presence of sensor faults and multiplicative false data injection attacks and using virtual sensors.

Figures 3.71 shows the output of the neural network-based sensor fault detector when bias sensor faults and multiplicative false data injection attacks are present, and virtual sensors are utilized for data recovery at both the plant and command and control sides. As demonstrated in the figure, the detector can estimate bias sensor faults and treat them as residuals. When the magnitude of these residuals exceeds the threshold value, as defined in page 63, the detector can detect the presence of sensor faults and identify which specific sensors are affected by bias sensor faults. The detector does not trigger false alarms for the multiplicative false data injection attacks, as shown in the figure. However, due to sudden path changes, the detector has bumps at the starting points where the path changes.

Figure 3.72 displays the output of the false data injection attack detector when the system is under both multiplicative false data injection attacks and sensor faults, and virtual sensors are employed for data recovery at both the plant and command and control sides. The figure shows that the detector can estimate the multiplicative false data injection attacks and treat them as residuals. When the magnitude of these residuals exceeds the threshold value, as defined in page 64, the detector can detect the presence of multiplicative false data injection attacks and identify which specific sensor measurements are affected. Notably, the detector is not triggered by sensor faults when a virtual sensor is utilized at the plant side, and it is only sensitive to the multiplicative false data injection attacks.

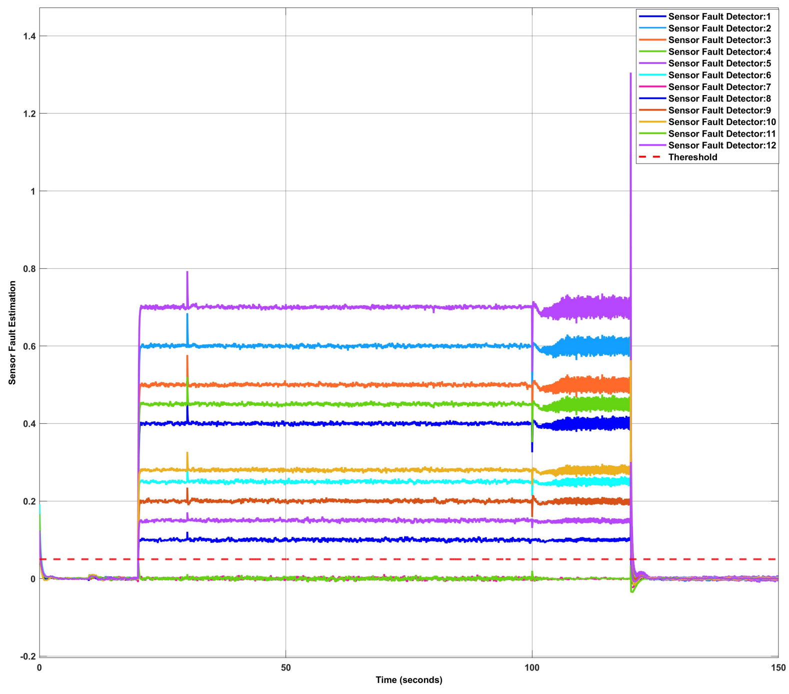

Figure 3.71: The sensor fault detection diagnosis response demonstrates its ability to detect, isolate, and estimate bias sensor faults in the presence of bias sensor faults and multiplicative false data injection and using virtual sensors.

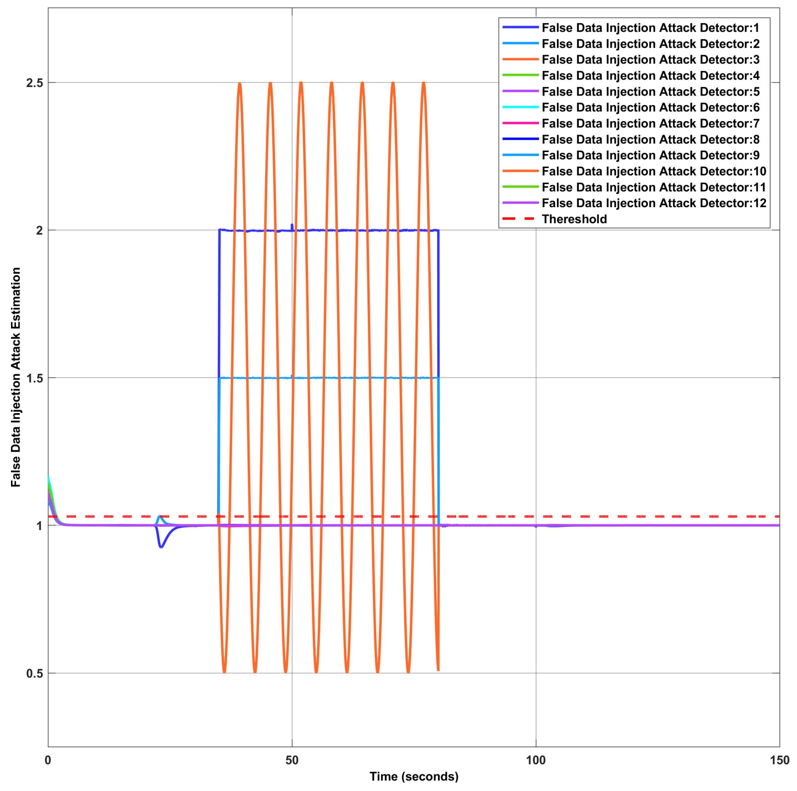

Figure 3.72: The response of the False Data Injection attack detection diagnosis on the C&C side demonstrates its ability to detect, isolate, and estimate the severity of an attack in the presence of sensor faults and multiplicative false data injection attacks and using virtual sensors.

## 3.6 Performance Analysis

In this section, the performance of the designed NN-based diagnosis is investigated. In this case, the confusion matrix will be used to analyze the performance by calculating the detection, isolation, and estimation rate of the diagnosis. In this regard, four possibilities for the detector results are considered consisting of True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). According to [85], two different metrics, precision and accuracy, can be defined. This work will calculate the precision metric and accuracy metric for evaluating NN-based fault and false data injection attack detectors. The precision metric is defined as [85] :

$$PPV = \frac{TP}{TP+FP}. \tag{3.30}$$

The accuracy metric is defined as [85] :

$$ACC = \frac{TP+TN}{TP+TN+FP+FN}. \tag{3.31}$$

**System under bias sensor faults**

In this section, it is assumed that system is just under bias sensor faults and there is no virtual sensor in the system. The values in these tables are derived from 100 simulation runs.The confusion matrices for the two detectors are represented in Tables 3.2 and 3.3 when the system is under bias sensor faults. Table 3.2 presents the results of the performance evaluation of the neural network-based diagnosis system in terms of its ability to detect, isolate, and estimate sensor faults. The Table 3.3 presents the performance of the false data injection attack detector in detecting the bias sensor faults. Since there is no recovery scheme in place to address these faults, the performance of the detector is degraded.

Table 3.2: *Confusion matrix for the sensor fault diagnosis when the system is under bias sensor faults.*

| | TN | FP | TP | FN |
|---|---|---|---|---|
| Bias Sensor Fault | 97.79 | 2.21 | 98.81 | 1.19 |
| PPV | 97.81% | | | |
| ACC | 98.3% | | | |

Table 3.3: *Confusion matrix for the false data injection attack diagnosis when the system is under bias sensor faults.*

| | TN | FP | TP | FN |
|---|---|---|---|---|
| Bias Sensor Fault | 87.34 | 12.66 | 92.08 | 7.92 |
| PPV | 87.91% | | | |
| ACC | 89.71% | | | |

**System under bias sensor faults using a virtual sensor at plant side**

In this section a virtual sensor is employed at the plant side to recover the sensor measurement data from bias sensor faults. The performance of the sensor fault detector is the same as the mentioned performance in Table 3.2. The performance of the false data injection attack detector has been enhanced in terms of its ability to detect, isolate, and estimate false data injection attack and it is now not affected by bias sensor faults. The new confusion matrix is presented in Table 3.4.

Table 3.4: *Confusion matrix for the false data injection attack diagnosis when the system is under bias sensor faults and using a virtual sensor.*

| | TN | FP | TP | FN |
|---|---|---|---|---|
| Bias Sensor Fault | 98.73 | 1.27 | 98.19 | 1.81 |
| PPV | 98.19% | | | |
| ACC | 98.46% | | | |

**System under bias sensor faults and multiplicative false data injection attacks using virtual sensors**

In this section it is assumed that system is under multiplicative false data injection attacks and bias sensor faults. The performance of two neural network-based diagnosis approaches is evaluated in terms of their ability to detect, isolate, and estimate sensor faults and false data injection attacks and are shown in Tables 3.5 and 3.6 respectively.

Table 3.5: *Confusion matrix for the sensor fault diagnosis when the system under bias sensor faults and multiplicative false data injection attacks using virtual sensors.*

| | TN | FP | TP | FN |
|---|---|---|---|---|
| Faults & False Data Injection attacks | 98.15 | 1.85 | 93.83 | 6.16 |
| PPV | 98.06% | | | |
| ACC | 95.99% | | | |

Table 3.6: *Confusion matrix for the false data injection attack diagnosis when the system under bias sensor faults and multiplicative false data injection attacks using virtual sensors.*

| | TN | FP | TP | FN |
|---|---|---|---|---|
| Faults & False Data Injection attacks | 97.21 | 2.79 | 94.62 | 5.38 |
| PPV | 97.13% | | | |
| ACC | 95.91% | | | |

## 3.7 Conclusions

This chapter provides the detection, isolation, and estimation of bias sensor faults and multiplicative false data injection attacks in the quadrotor. An online neural network-based diagnosis technique is proposed, which can be used for the detection, isolation, and estimation of false data injection attacks. The estimated sensor faults can be used in virtual sensors, making possible simultaneous detection of sensor faults and false data injection attacks. Moreover, a virtual sensor module is utilized in order to recover sensor measurement data which are subjected to multiplicative false data injection attacks. The performance of the diagnoses are also calculated in this chapter.

# Chapter 4

# Detection and Isolation of Sensor Faults and Replay Attacks

This chapter proposes a new online neural network-based approach for detecting, isolating, and estimating replay attacks on the input of CPS. The great contribution of this chapter is the detection, isolation, and estimation of replay attacks by considering two neural network-based detectors at the plant and command and control sides. In the case that a replay attack occurs, just the detector on the plant side is triggered. Therefore, by comparing the output of these detectors and assuming that the binary flag indicating the activation of the plant side detector, which is sent to the command and control center, is not under attack and is sent through a secure channel, the replay attack can be detected. Moreover, the other contribution is that bias sensor faults and replay attacks can be isolated by employing the virtual sensor at the plant side.

Section 4.1 presents a detection approach for false data injection attacks on the input of CPS and provides a scheme for isolating false data injection attacks from bias sensor faults. Section 4.2 provides the simulation results by considering different scenarios of sensor faults and replay attacks. Section 4.3 investigates the performance of the designed detector and provides confusion matrices results for the detectors.

## 4.1 Detection of Replay Attacks and Isolation from Sensor Faults on the CPS

In this section, the detection of replay attacks on the nonlinear quadrotor has been studied. In the replay attack case, the adversary tries to replay the sensors' measurements in the system to damage the control system without being detected. For instance, it is considered that the adversary can inject false data into the control signal while replaying the previous sensors' measurement data to fool the controller and also not being detected in the command and control center.

This section proposes a schematic that can detect a replay attack while the adversary injects false data into the control signals. Figure 4.1 shows how the adversary tries to damage the system with replay attacks. In this case, the adversary replays the output of virtual sensors, which is sent to the controller and the command and control center, meanwhile injecting false data into the control signal to damage the system.

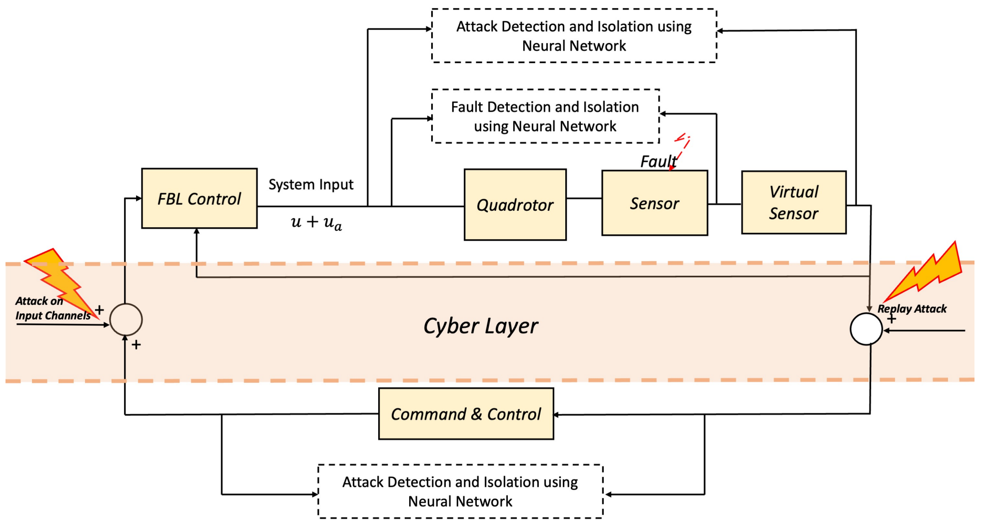
Attack Detection and Isolation using Neural Network
Fault Detection and Isolation using Neural Network
Fault
FBL Control
System Input
$u + u_a$
Quadrotor
Sensor
Virtual Sensor
Attack on Input Channels
+
+
Cyber Layer
Replay Attack
+
Command & Control
Attack Detection and Isolation using Neural Network

Figure 4.1: Structure of the adaptive neural network-based detection strategy for replay attacks with considering sensor faults

Figure 4.1 illustrates that two cyberattack detectors are considered. When the system is under replay attack, the adversary hides the attack's impact from the command and control center, but it can be detected at the plant side. Therefore, if the plant-side attack detection is triggered and the one in the command and control is not activated, it can be concluded that the system is under replay attack.

One of the significant advantages of this schematic is that it can isolate sensor faults from replay attacks. By using the virtual sensor, the signature of bias sensor faults is mitigated, and the recovered data is sent for feedback linearization control and the command and control center. This approach helps to mitigate the impact of bias sensor faults before the data is sent to the controller and the command and control center, thus ensuring that the replay attacks do not affect the process of fault detection, isolation, and estimation. The sensor faults detection, isolation, and estimation methodology are completely explained in the previous chapter, Section 3.3. Section 3.4.2 explains the process of recovering sensor measurement data from bias sensor faults using virtual sensors.

### 4.1.1 Replay Attack Detection strategy

In the first phase of the replay attack the nonlinear system can be written as [67; 68] :

$$\begin{cases} \dot{x}(t) = f\left(x(t), u(t)\right), \\ y(t) = Cx(t), \end{cases} \quad t_0 \leq t \leq t_0 + w, \tag{4.1}$$

where $x \in \mathbb{R}^n$ is the state variable, $u \in \mathbb{R}^m$ is the input, $y \in \mathbb{R}^q$ is the output , $f$ is the known nonlinear function, and $w$ is the time frame that the attacker record the data.

During the second stage, the adversary manipulates the control input to cause damage to the system while replaying the system output to the controller and command and control center. The nonlinear CPS can be represented as [67; 68]:

$$\begin{cases} \dot{x}(t) = f\left(x(t), \tilde{u}(t)\right), \\ \tilde{u}(t) = u(t) + u_a, & t_1 + (N_f - 1)\,w \leq t \leq t_1 + N_f\,w, \quad N_f \in \mathbb{N}, N_f \geq 1, \\ \tilde{y}(t) = y(t + t_0 - t_1 - (N_f - 1)\,w). \end{cases} \tag{4.2}$$

In this work, a neural network-based detector is considered to detect replay attacks. The neural network-based observer estimates the injected attack, and it is placed at both the plant and command and control sides.

During a replay attack, the adversary replays recorded data for some time, and the output simulates the system's normal condition while injecting false data into the input signal. As shown in Figure 4.2, the control signal $u$ generated at the command and control center is unaffected, and the received manipulated sensor measurement $\tilde{y}$ is the healthy recorded data resulting in no detector activation.

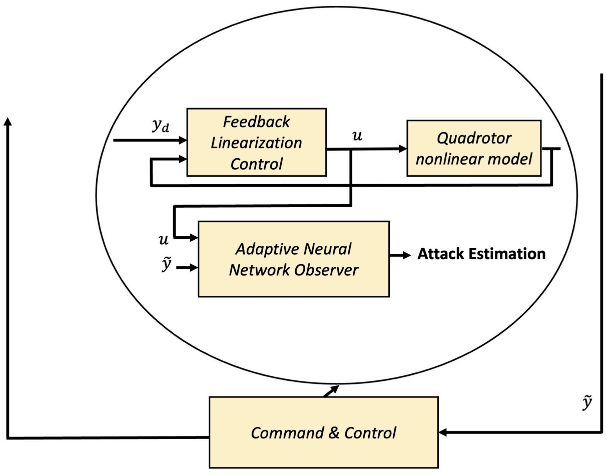
$y_d$
Feedback Linearization Control
$u$
Quadrotor nonlinear model
$u$
$\tilde{y}$
Adaptive Neural Network Observer
Attack Estimation
$\tilde{y}$
Command & Control

Figure 4.2: Structure of the neural network-based detection strategy at the command and control center.

On the other hand, as shown in Figure 4.3 the detector at the plant side receives the manipulated control signal $\tilde{u}$ and manipulated sensor measurement $\tilde{y}$, which is the healthy sensor measurement data, and this leads to triggering the neural network-based detector. Therefore, the attack signal is detected, isolated, and estimated at the plant side neural network-based detector.

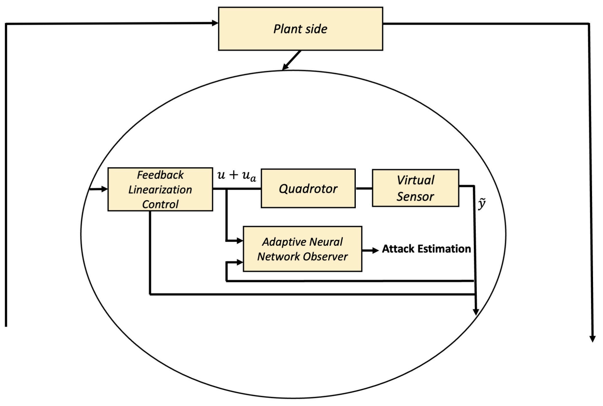

Figure 4.3: Structure of the neural network-based detection strategy at the plant.

If the system is subjected to false data injection attacks on the control input signal, the command and control side detector is triggered. However, if the system is under replay attack, since the attacker replays healthy data and sends it to the command and control center, only the detector on the plant side is triggered. It is assumed that the binary flag indicating the activation of the plant side detector is sent securely to the command and control center. By comparing the output of the neural network detectors, it is possible to recognize that a replay attack has occurred in the system.

### Detection of the attack signal injected to the control input

By considering a Hurwitz matrix $A$ the nonlinear model in the second phase can be rewritten as:

$$\begin{cases} \dot{x} = Ax + g(x, \tilde{u}), \\ \tilde{u}(t) = u(t) + u_a, & t_1 + (N_f - 1)w \le t \le t_1 + N_f w, \quad N_f \in \mathbb{N}, N_f \ge 1, \\ \tilde{y}(t) = y(t + t_0 - t_1 - (N_f - 1)w), \end{cases} \tag{4.3}$$

where $g(x, \tilde{u}) = f(x, \tilde{u}) - Ax$.

The control signal which is manipulated by the adversary can be represented as:

$$\tilde{u}(t) = u + T_A(x, u), \tag{4.4}$$

where $T_A$ represents the additive false data that is injected into the control signal. Four assumptions which are introduced in [63], should be taken into account, such as open-loop observability of the nonlinear system, stability of the nominal closed-loop system and etc.

The structure of the neural network-based observer for the detection, isolation, and estimation of injected false data into the control input is shown in Figure 4.4. As it is shown, a neural network (NN2) is used to map $T_A(x, u)$.

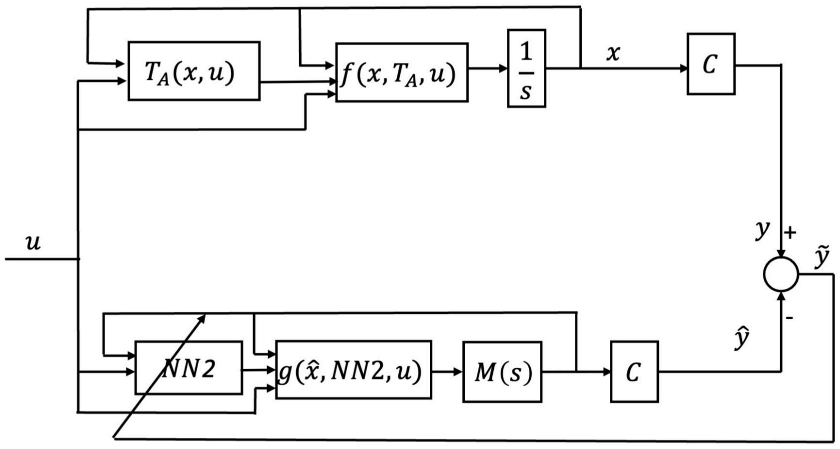

Figure 4.4: Structure of the adaptive neural network-based scheme for the detection of false data injection attacks on the control input signal.

The nonlinear state space representation of the observer considering false data injection attack on the CPS input can be rewritten as below:

$$\begin{cases} \dot{\hat{x}} = A\hat{x} + g\left(\hat{x}, \tilde{u}\right), \\ \tilde{u} = \hat{T}_A\left(\hat{x}, \hat{W}\right) + u, \\ \hat{y} = C\hat{x}, \end{cases} \tag{4.5}$$

where $A$ is a Hurwitz matrix and $g\left(\hat{x}, \tilde{u}\right) = f\left(\hat{x}, \tilde{u}\right) - A\hat{x}$. The transfer function $M(s)$ is defined as below [63]:

$$M(s) = (sI - A)^{-1}. \tag{4.6}$$

The NN2 estimates false data injection attack which is shown by $\hat{T}_A\left(\hat{x}, \hat{W}\right)$ with adaptable variables which are weight matrices.

$$\hat{T}_A\left(\hat{\bar{x}}, \hat{W}\right) = \hat{W}\sigma\left(\hat{V}\hat{\bar{x}}\right), \tag{4.7}$$

where $\hat{\bar{x}} = [\hat{x} \quad u]^T$, and $\hat{W}$ and $\hat{V}$ are neural network weights matrices. The activation function is defined as below [63]:

$$\sigma_i(V^i\hat{\bar{x}}) = \frac{2}{1 + exp^{-2V^i\hat{\bar{x}}}} - 1, \qquad i = 1, 2, ..., N, \tag{4.8}$$

where $N$ is the number of neurons in the hidden layer.

The weights of this NN are updated based on the theorem stated in the work of Talebi et al.[63].

$$\begin{aligned}\dot{\hat{W}} &= -\eta_1\left(\frac{\partial J}{\partial \hat{W}}\right) - \rho_1 ||\left(y-\hat{y}\right)||\hat{W}, \\ \dot{\hat{V}} &= -\eta_2\left(\frac{\partial J}{\partial \hat{V}}\right) - \rho_2 ||\left(y-\hat{y}\right)||\hat{V},\end{aligned} \tag{4.9}$$

where $\eta_1$ and $\eta_2 > 0$ are learning rates and $\rho_1$ and $\rho_2$ are small positive constants and the objective function of neural network is defined as [63]:

$$J = \frac{1}{2}((y-\hat{y})^T(y-\hat{y})). \tag{4.10}$$

In order to define learning rules, the first two variables are defined as [63]:

$$\begin{aligned}&net_{\hat{v}} = \hat{V}\hat{\bar{x}}, \\ &net_{\hat{w}} = \hat{W}S_1, \\ &S_1 = \sigma(\hat{V}\hat{\bar{x}}).\end{aligned} \tag{4.11}$$

Therefore, $\frac{\partial J}{\partial \hat{W}}$ and $\frac{\partial J}{\partial \hat{V}}$ can be determined as [63; 78]:

$$\begin{aligned}\frac{\partial J}{\partial \hat{W}} &= \frac{\partial J}{\partial net_{\hat{w}}} \cdot \frac{\partial net_{\hat{w}}}{\partial \hat{W}}, \\ \frac{\partial J}{\partial \hat{V}} &= \frac{\partial J}{\partial net_{\hat{v}}} \cdot \frac{\partial net_{\hat{v}}}{\partial \hat{V}}.\end{aligned} \tag{4.12}$$

Then,

$$\begin{aligned}&\frac{\partial J}{\partial net_{\hat{w}}} = \frac{\partial J}{\partial (y-\hat{y})} \cdot \frac{\partial (y-\hat{y})}{\partial \hat{x}} \cdot \frac{\partial \hat{x}}{\partial net_{\hat{w}}} \\ &= -(y-\hat{y})^T C \frac{\partial \hat{x}}{\partial net_{\hat{w}}}, \\ &\frac{\partial J}{\partial net_{\hat{v}}} = \frac{\partial J}{\partial (y-\hat{y})} \cdot \frac{\partial (y-\hat{y})}{\partial \hat{x}} \cdot \frac{\partial \hat{x}}{\partial net_{\hat{v}}} \\ &= -(y-\hat{y})^T C \frac{\partial \hat{x}}{\partial net_{\hat{v}}}.\end{aligned} \tag{4.13}$$

And,

$$\begin{aligned}\frac{\partial net_{\hat{w}}}{\partial \hat{W}} &= S_1^T,\\ \frac{\partial net_{\hat{v}}}{\partial \hat{V}} &= \hat{\bar{x}}.\end{aligned} \tag{4.14}$$

By utilizing (4.5) and the notion of $net_{\hat{v}}$ and $net_{\hat{w}}$ the following formulas can be obtained:

$$\begin{aligned}\frac{\partial \hat{\dot{x}}}{\partial net_{\hat{w}}} &= A\frac{\partial \hat{x}}{\partial net_{\hat{w}}} + \frac{\partial g}{\partial \hat{x}}\frac{\partial \hat{x}}{\partial net_{\hat{w}}} + \frac{\partial g}{\partial \hat{\bar{u}}}\frac{\partial \hat{\bar{u}}}{\partial net_{\hat{w}}},\\ \frac{\partial \hat{\dot{x}}}{\partial net_{\hat{v}}} &= A\frac{\partial \hat{x}}{\partial net_{\hat{v}}} + \frac{\partial g}{\partial \hat{x}}\frac{\partial \hat{x}}{\partial net_{\hat{v}}} + \frac{\partial g}{\partial \hat{\bar{u}}}\frac{\partial \hat{\bar{u}}}{\partial net_{\hat{v}}}.\end{aligned} \tag{4.15}$$

The gradients can be defined as below:

$$\begin{aligned}\dot{d}_{xw} &= (A + A_c)d_{xw} + B,\\ \dot{d}_{xv} &= (A + A_c)d_{xv} + B\hat{W}S_2,\end{aligned} \tag{4.16}$$

where $d_{xw} = \dfrac{\partial \hat{x}}{\partial net_{\hat{w}}}$ and $d_{xv} = \dfrac{\partial \hat{x}}{\partial net_{\hat{v}}}$ and the parameters $B$ and $S_2$ are defined as:

$$\begin{aligned}S_2 &= I - diag\{\sigma_i^2\left(V^l\hat{\bar{x}}\right)\}, \quad i = 1, 2, ..., m,\\ B &= \frac{\partial g}{\partial \hat{u}_a}|_{\hat{x}=0,\hat{u}=u},\\ A_c &= \frac{\partial g}{\partial \hat{x}}|_{\hat{x}=0,\hat{u}=u}.\end{aligned} \tag{4.17}$$

Therefore, the learning rules in (4.9) become:

$$\begin{aligned}\hat{\dot{W}} &= -\eta_1\left(\left(y-\hat{y}\right)^T Cd_{xw}\right)^T S_1^T - \rho_1||y-\hat{y}||\hat{W},\\ \hat{\dot{V}} &= -\eta_2\left(\left(y-\hat{y}\right)^T Cd_{xv}\right)^T \hat{\bar{x}}^T - \rho_2||y-\hat{y}||\hat{V}.\end{aligned} \tag{4.18}$$

By using this method, the false data injection attack that is injected into the control signal is estimated. The estimated attack which is the output of the NN is derived by (4.7) and can be represented as:

$$\hat{T}_A = [\hat{T}_{A1} \quad \hat{T}_{A2} \quad \hat{T}_{A3} \quad \hat{T}_{A4}], \tag{4.19}$$

where each element of $\hat{T}_A$ represents the estimated false data injection attacks for each of the four input signals.

When the system operates in normal conditions, the neural network's output $\hat{T}_A$ is close to zero. However, when an attack is injected, the neural network estimates the attack, and the output can be considered as the residual. To reduce false alarms caused by measurement noise in the system, a Monte Carlo test is conducted under different conditions to determine a suitable threshold. If the residual, which is the estimation of the attack signal, exceeds the determined threshold, the attack is detected,

$$|\hat{T}_{Ai}| > \delta_{TAi}, \qquad i = 1, \ldots, 4, \tag{4.20}$$

where $\delta_{TAi}$ represents the threshold for the $i^{th}$ control signal.

This detector estimates the false data injected into each of the control signals, producing a residual for each signal. This estimated value is considered the residual for the cyberattack diagnosis procedure. By analyzing these residuals, the neural network can determine which control inputs are under attack. Therefore, the neural network-based detection approach can achieve attack isolation if the estimation of the attack is nonzero. If the neural network estimates the cyberattack signal properly, it can simultaneously detect and isolate the false injection data attack in the control signal.

## 4.2 Simulation Results

This section presents the simulation results and evaluates the performance of the sensor faults detectors and false data injection attack detectors. Different scenarios are considered to assess the performance of the detectors.

The parameter of the neural network-based detectors are considered as below:

$A = -5 * I$,

$number \quad of \quad hidden \quad layer = 7$,

$\eta_1 = 115$,

$\eta_2 = 115$,

$\rho_1 = 0.08$,

$\rho_2 = 0.08$.

The threshold for the replay attack detector is determined by conducting a Monte Carlo simulation with noise on the normal system in the absence of sensor faults and false data injection attacks,

$\delta_{TA} = 0.08$.

A band-limited white noise is considered in the first six sensor measurements and the noise power defined as:

$Noise\ power = 0.0001$.

The threshold for the sensor fault detector has been defined through a Monte Carlo test that was performed in the absence of sensor faults and replay attacks, while considering noise in the system,

$\delta_{Ts} = 0.03$.

### 4.2.1 Detection of Replay Attacks and Isolation from Sensor Faults on the Quadrotor- First Scenario

#### 4.2.1.1 Fault-Free and Attack-Free Scenario

In the first scenario, it is considered that the mission of the quadrotor is to start from the origin (0,0,0) by considering the roll, yaw, and pitch angles initially set to zero and reach the point (10,10,10) and hover at this point.

Figure 4.5 depicts the quadrotor's position and Euler angles in time following the commanded trajectory from the command and control.

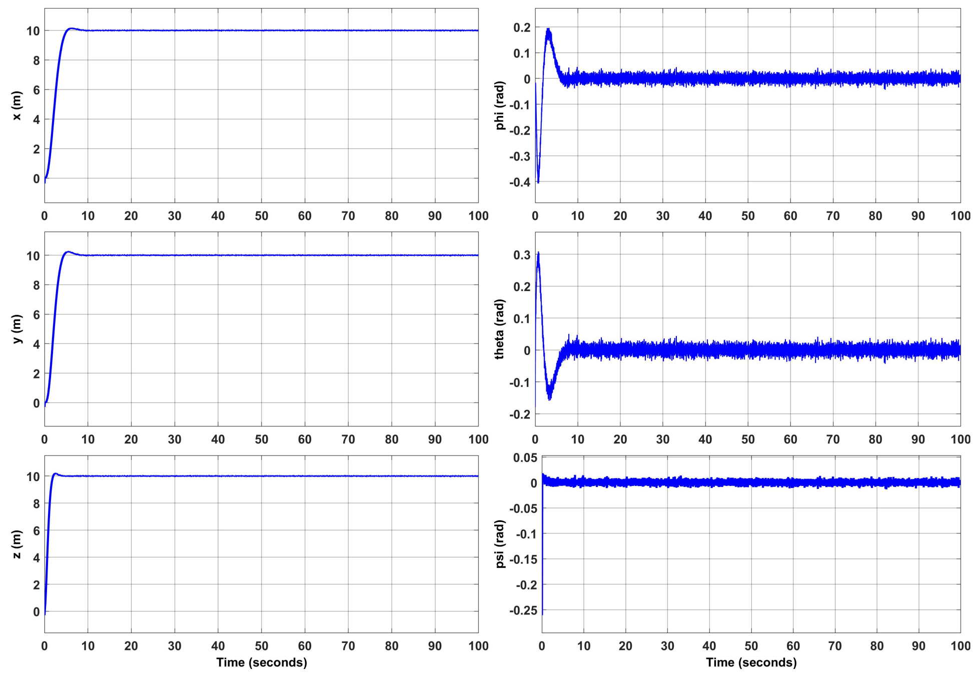

Figure 4.5: Linear position and Euler angles of the quadrotor while there are no attack and sensor faults.

Figures 4.6 and 4.7 show the performance of the detectors while there is no sensor fault and replay attacks in the system. It is shown that these detectors are not triggered when there are no faults and replay attacks in the system.

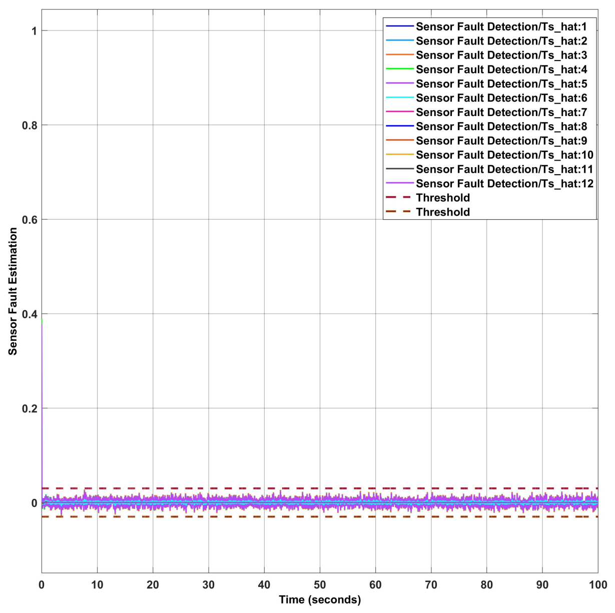

Figure 4.6: The response of the sensor fault detection diagnosis on the plant side in normal condition. Refers to page 124 for the selection of the threshold.

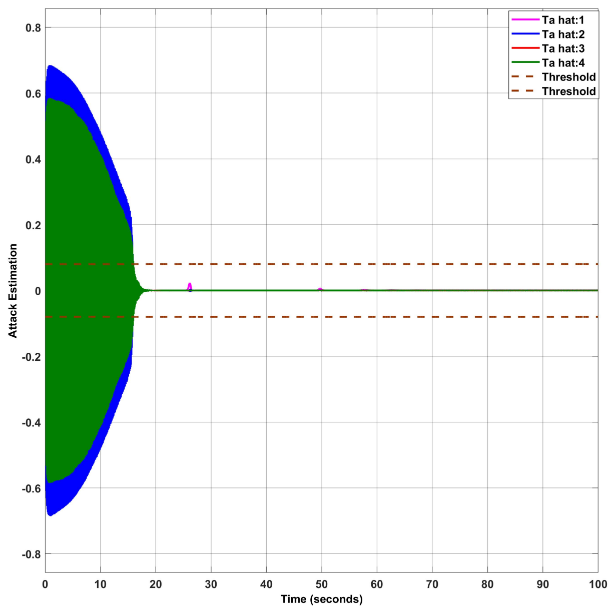

Figure 4.7: The response of the replay attack detection diagnosis on the C&C side and plant side in normal condition. Refers to page 124 for the selection of the threshold.

#### 4.2.1.2 Fault-Free and Replay Attack Scenario

In this section, the detection of replay attacks on the nonlinear quadrotor has been investigated. In the replay attack case, the adversary first tries to replay the sensors' measurements in the system to damage the control system without being detected. In this case, the measurement data from the time interval [20,60] is recorded

$$\begin{cases} \dot{x}(t) = f\left(x(t), u(t)\right), \\ y(t) = Cx(t), \end{cases} \quad 20 \le t \le 60, \tag{4.21}$$

In the second stage, the attacker replayed the previously recorded data between the time window of [60,100]. In this stage, attacker wants to cause damage to the system without being detected on the command and control center, by manipulating the control input. The sensor measurements were replaced by the previously recorded data, and the abnormality of the system's operation was hidden from the command and control center. In order to damage the system, the attacker injected false data into the control input, as represented:

$$\begin{cases} \dot{x}(t) = f\left(x(t), \tilde{u}(t)\right), \\ \tilde{u}(t) = u(t) + u_a, \quad 60 \le t \le 100, \\ \tilde{y}(t) = y(t-40), \end{cases} \tag{4.22}$$

where the attack signal is defined as :

$u_a = [0.4sin(t) \quad 0.2 \quad 0.1 \quad 0.5]$.

Figure 4.8 shows the real linear position and Euler angles of the quadrotor. It is shown that the replay attack can harmfully affect the quadrotor and destabilize the system. However, the impact of replay attack is hidden from controller and command and control center and it is shown in Figure 4.9.

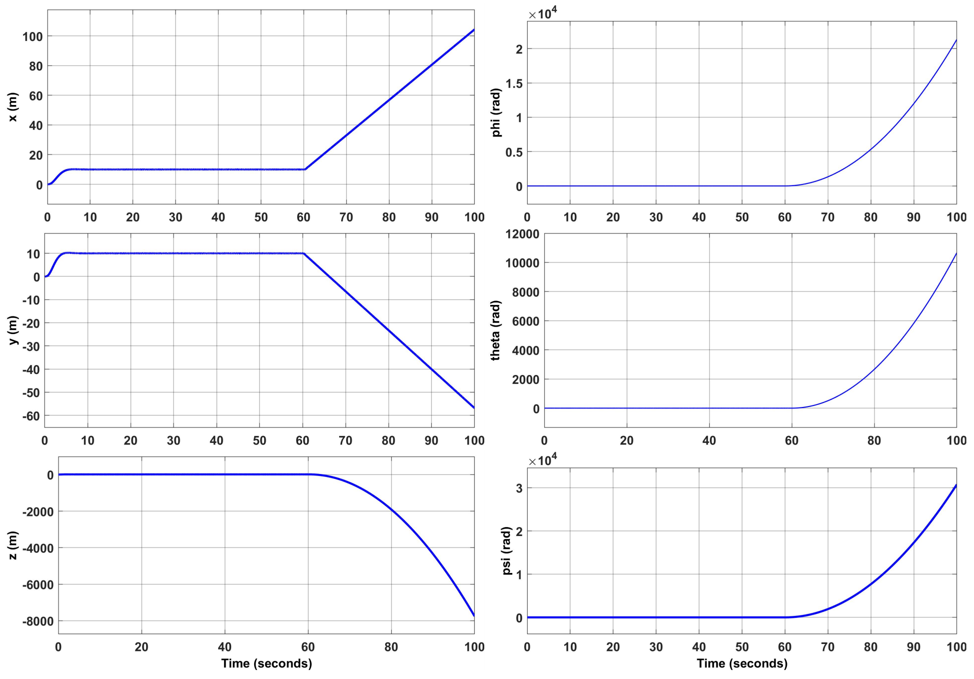

Figure 4.8: Quadrotor linear position and Euler angles under the replay attack.

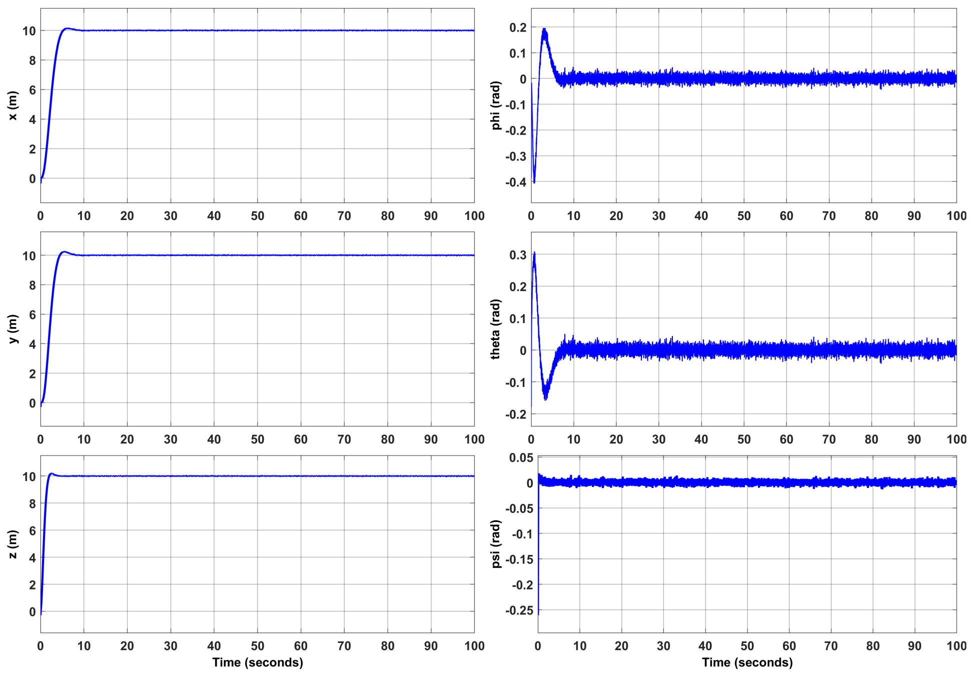

Figure 4.9: The sensor measurements data for linear position and Euler angles of the quadrotor under the replay attack.

Figure 4.10 represents the output of the sensor fault detector. It is shown that this detector works accurately for sensor fault detection, and replay attacks do not trigger it.

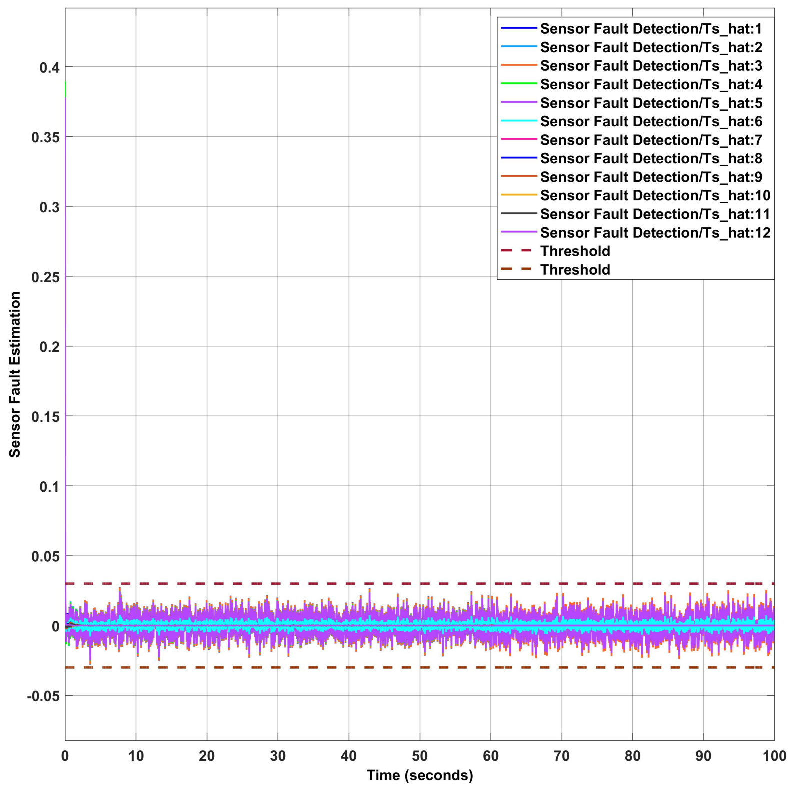

Figure 4.10: The response of the sensor fault detection diagnosis on the plant side while the system is under the replay attack.

Two neural network-based detectors are employed on both the plant and the command and control sides to detect replay attacks. Figure 4.11 depicts the output of the neural network-based detector on the plant side. The figure shows that the detector can successfully estimate false data injection attacks on the control input signals, which are then considered residuals. These residuals exceed a predefined threshold, enabling the detector to detect cyberattacks. Also, the detector can isolate false data injection attacks in different control input signals. It should be mentioned that the estimated attack signal has an inverted sign compared to the actual attack signal because of the structure of both the neural network-based detector and the replay attack. However, the neural network-based detector on the command and control side does not trigger, as shown in Figure 4.12, showing that the attack signal is not detectable at the command and control systems. This means that a replay attack has occurred in the system.

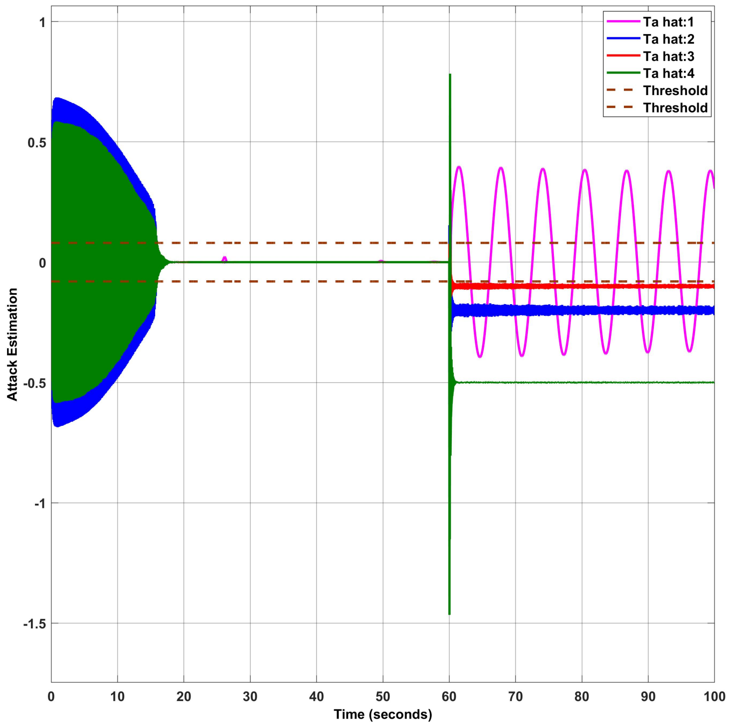

Figure 4.11: The response of a replay attack detection diagnosis on the plant side demonstrates its ability to detect, isolate, and estimate the severity of an attack.

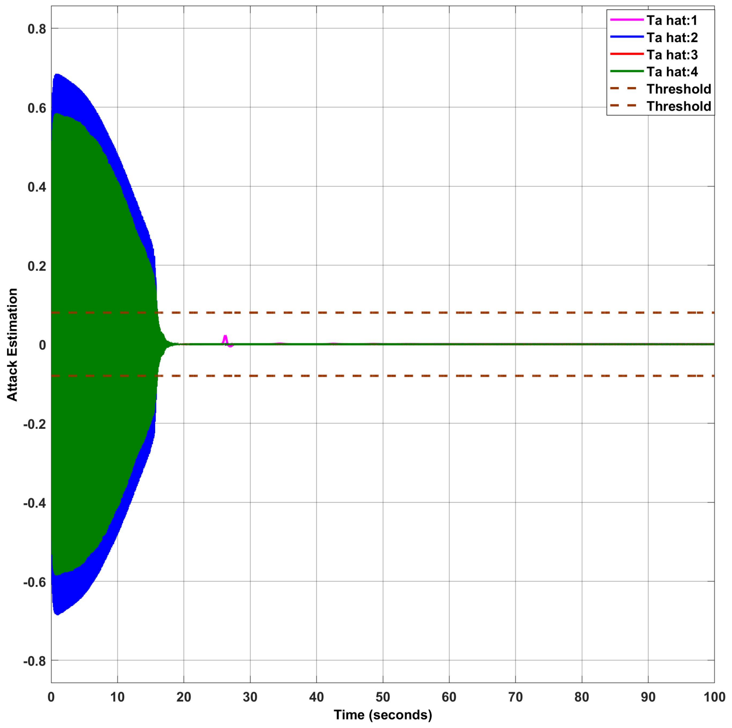

Figure 4.12: The response of a replay attack detection diagnosis on the C&C side.

#### 4.2.1.3 Sensor Fault and Replay Attack Scenario without using a Virtual Sensor

In this scenario, in addition to the replay attack, the system has bias sensor faults in the interval [50,90]. In this case, the virtual sensor for recovery of bias sensor faults is not considered at the plant side. The bias sensor faults are considered as:

$T_s = [0.9 \quad 0 \quad 0.7 \quad 0 \quad 0.3 \quad 0 \quad 0 \quad 0 \quad 0.5 \quad 0 \quad 0.6 \quad 0]$.

Figure 4.13 shows the sensor measurement data for the linear position and Euler angles of the quadrotor, which is sent to the command and control center and controller. The figure shows that sensor faults occur at time 50 but do not persist until the end of the period because the data is replayed from the time window of [20, 60]. Since, in the replay attack scenario, the sensor measurements are replaced by the previously recorded data, the impact of the replay attack is not evident.

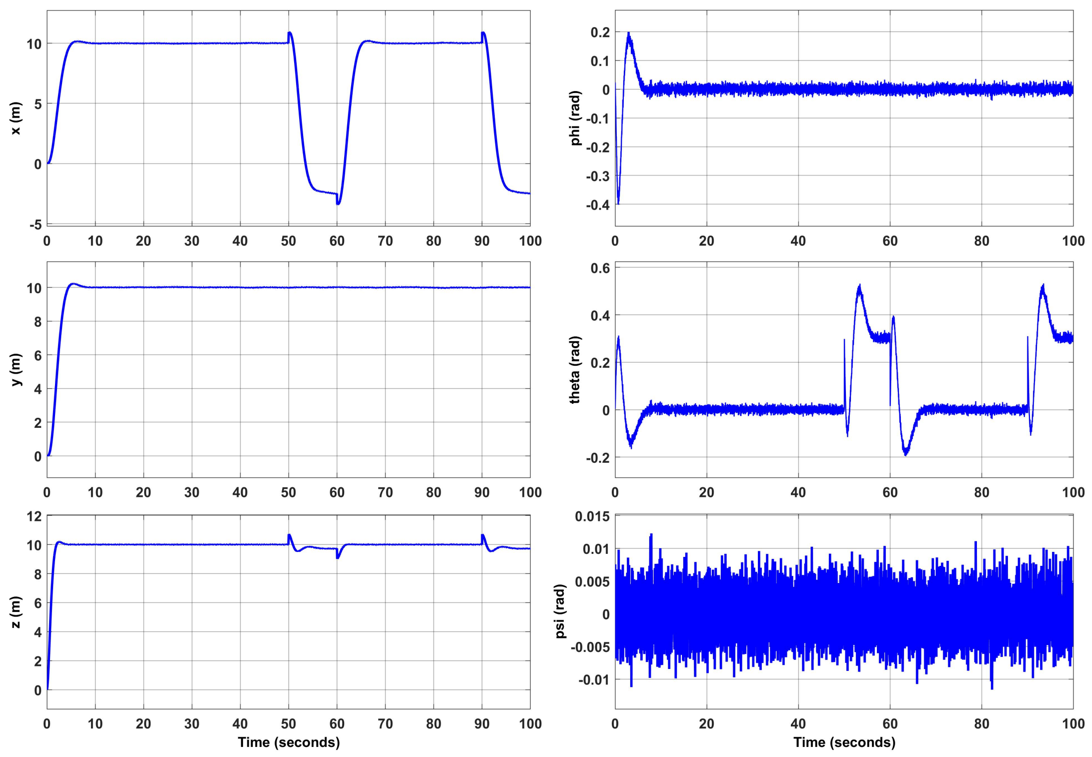

Figure 4.13: The sensor measurements data for linear position and Euler angles of the quadrotor under replay attack and sensor faults.

Figure 4.14 depicts the actual linear position and Euler angles of the quadrotor. The figure illustrates that a replay attack can cause significant harm to the quadrotor and destabilize the system. However, the impact of the replay attack is hidden from the command and control center.

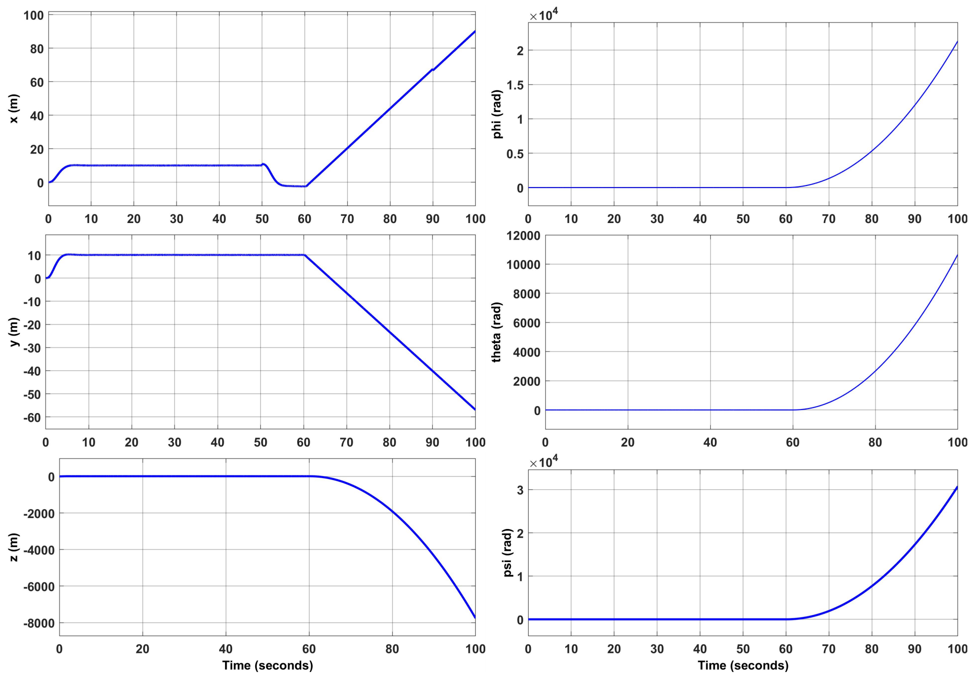

Figure 4.14: Quadrotor linear position and Euler angles under replay attack and sensor faults.

In Figure 4.15, the sensor fault detector's output is displayed. The figure demonstrates that the detector can successfully estimate bias sensor faults, which are then considered as residuals. These residuals exceed a certain threshold, enabling the detector to detect sensor faults. Furthermore, the detector is capable of isolating bias sensor faults in different sensors. It can be noted that this detector is not triggered for replay attacks, and it is just sensitive to bias sensor faults.

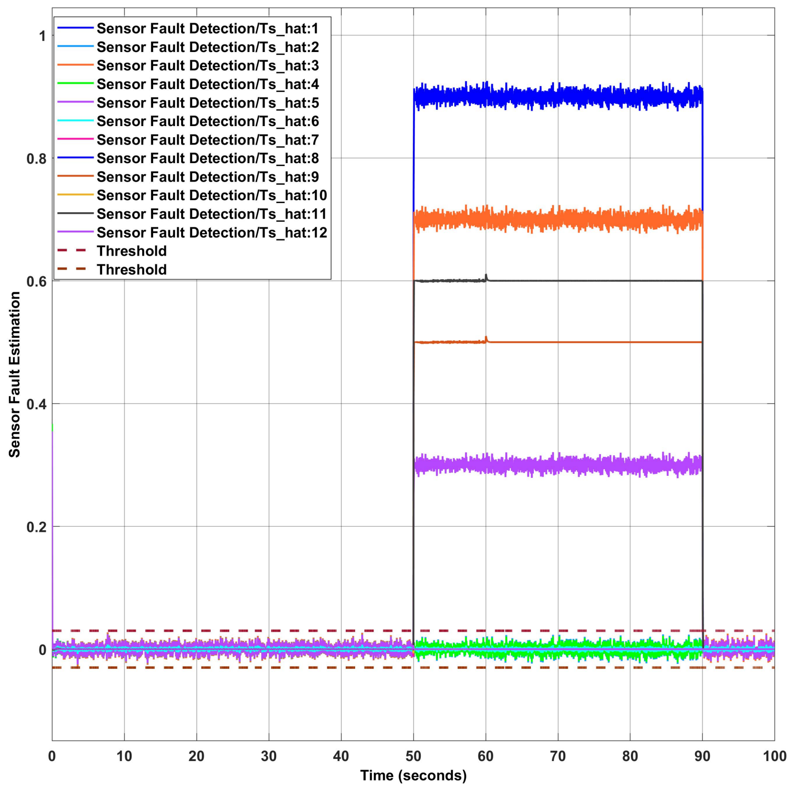

Figure 4.15: The sensor fault detection diagnosis response demonstrates its ability to detect, isolate, and estimate bias sensor faults in the presence of replay attacks and bias sensor faults.

Figure 4.16 depicts the output of the neural network-based detector. The figure shows that the detector on the plant side is triggered by the bias sensor faults since there is no recovery scheme for bias sensor faults. Figure 4.17 displays a zoomed-in view of the time frame in which the replay attack occurred. It can be seen that the detector can detect the attack on the last three input channels, although the estimation of the false data injected into the first input control signal is not correct. This indicates that the sensor faults triggered the replay attack detector and also affected the detector's performance when replay attacks were present.

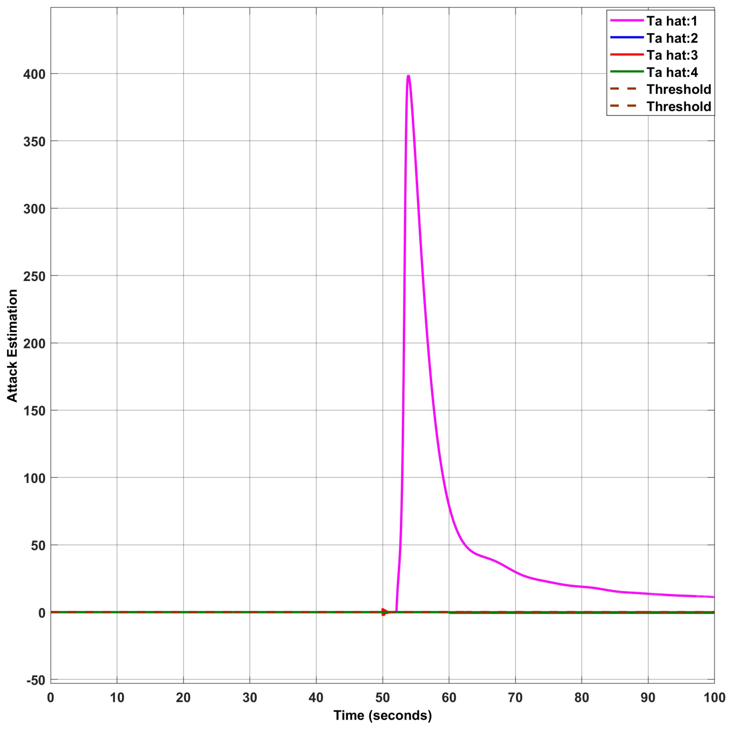

Figure 4.16: The response of the replay attack detection diagnosis on the plant side while the system under replay attack and sensor faults.

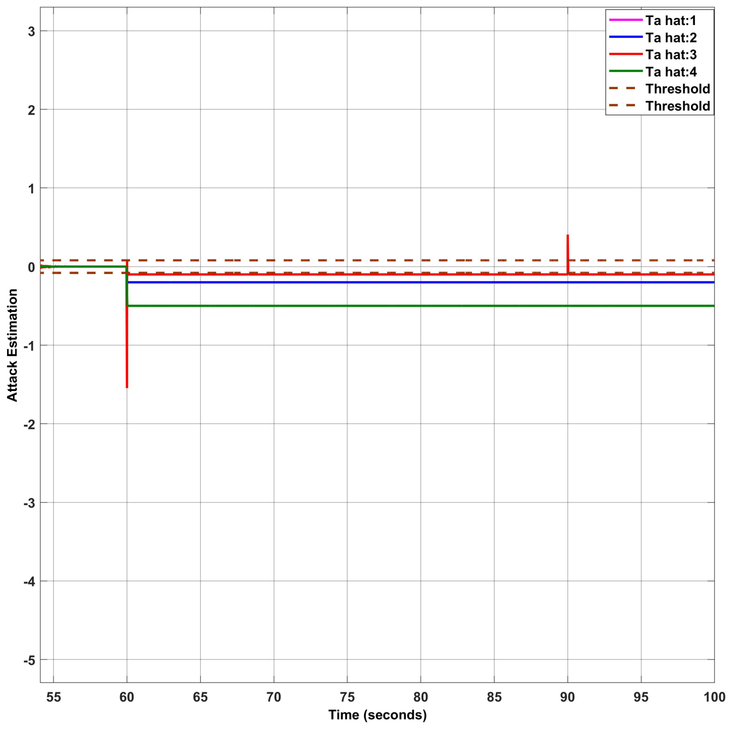

Figure 4.17: The response of the replay attack detection diagnosis on the plant side during a system under replay attack and sensor faults zoomed-in view.

Figure 4.18 represents the neural network-based detector on the command and control center. This figure shows the sensitivity of the detector to the bias sensor faults, and it indicates that the faults trigger the detector.

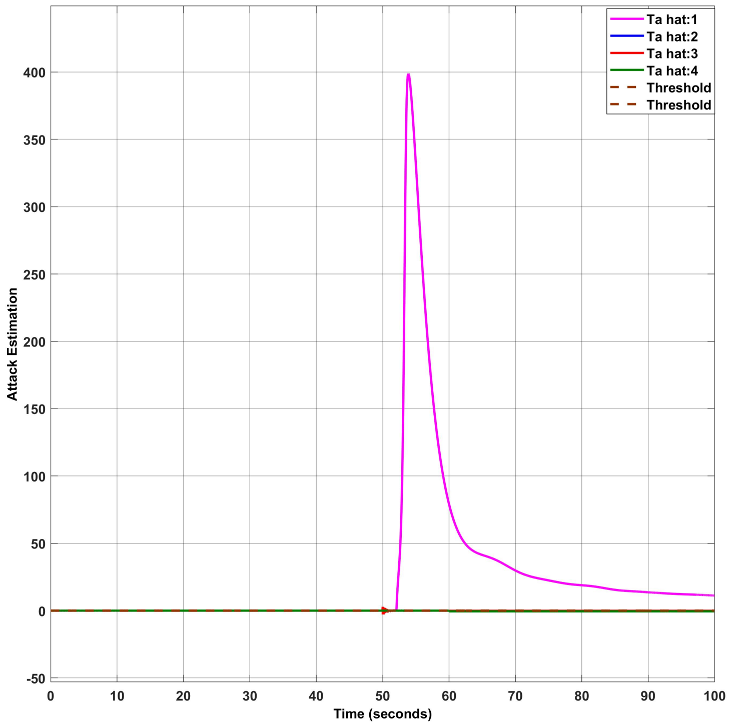

Figure 4.18: The response of the replay attack detection diagnosis on the C&C side during a system under replay attack and sensor faults.

#### 4.2.1.4 Sensor Fault and Replay Attack Scenario using a Virtual Sensor

In this scenario, the virtual sensor for data recovery of bias sensor faults is placed at the plant side. Figure 4.19 shows the sensor measurements data for linear position and Euler angles of the quadrotor. The impact of bias sensor faults is recovered by using a virtual sensor at the plant side. Since in the replay attack, the sensor measurements are replaced by the previously healthy recorded data, the impact of the replay attack is not shown.

Figure 4.20 depicts the actual linear position and Euler angles of the quadrotor. It is evident from the figure that a replay attack can cause significant damage to the quadrotor and destabilize its dynamics. However, the effects of the replay attack may not be apparent to the controller and the command and control center.

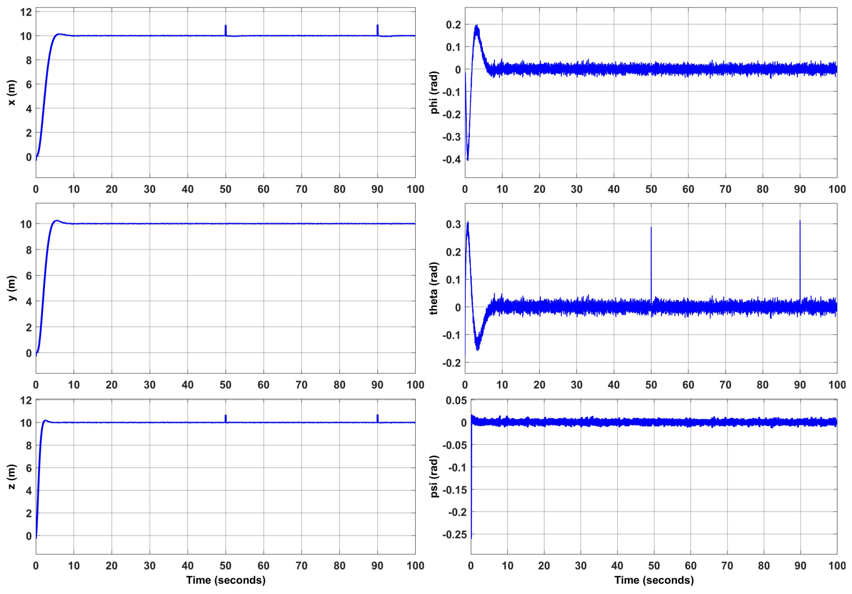

Figure 4.19: The sensor measurements data for linear position and Euler angles of the quadrotor under replay attack and sensor faults using a virtual sensor.

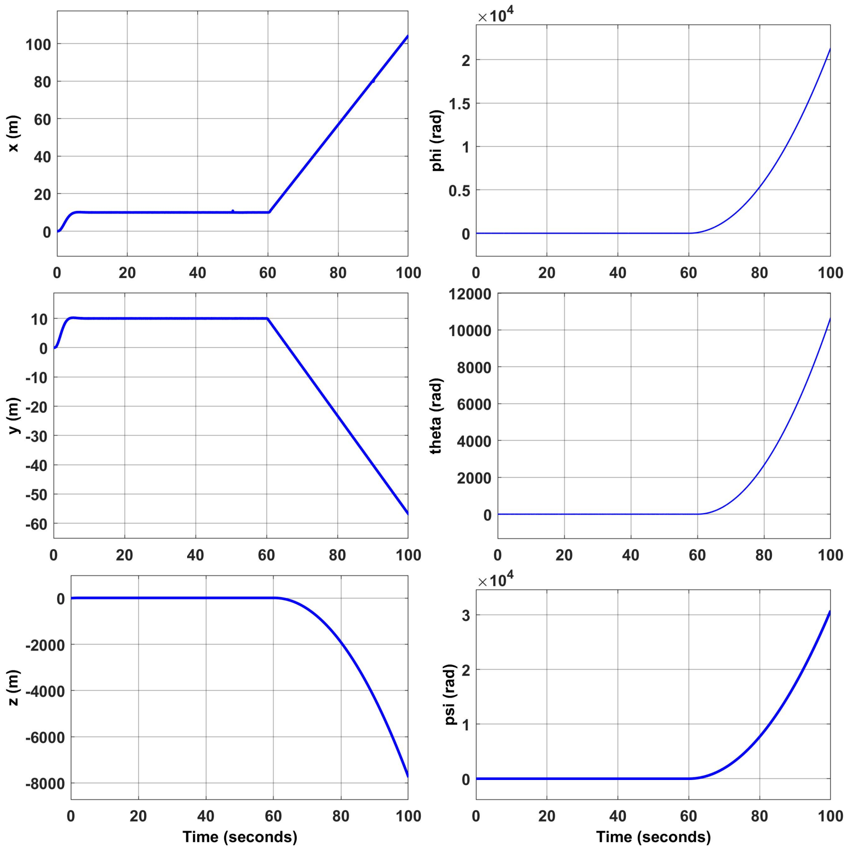

Figure 4.20: Quadrotor linear position and Euler angles under replay attack and sensor faults using a virtual sensor.

The output of the sensor fault detector is shown in Figure 4.21. The figure illustrates that the detector can accurately estimate bias sensor faults, which are then treated as residuals. These residuals exceed a predefined threshold, and the bias sensor faults can be detected. Additionally, the detector can isolate bias sensor faults in various sensors. It is worth noting that the detector does not detect replay attacks and is only sensitive to bias sensor faults.

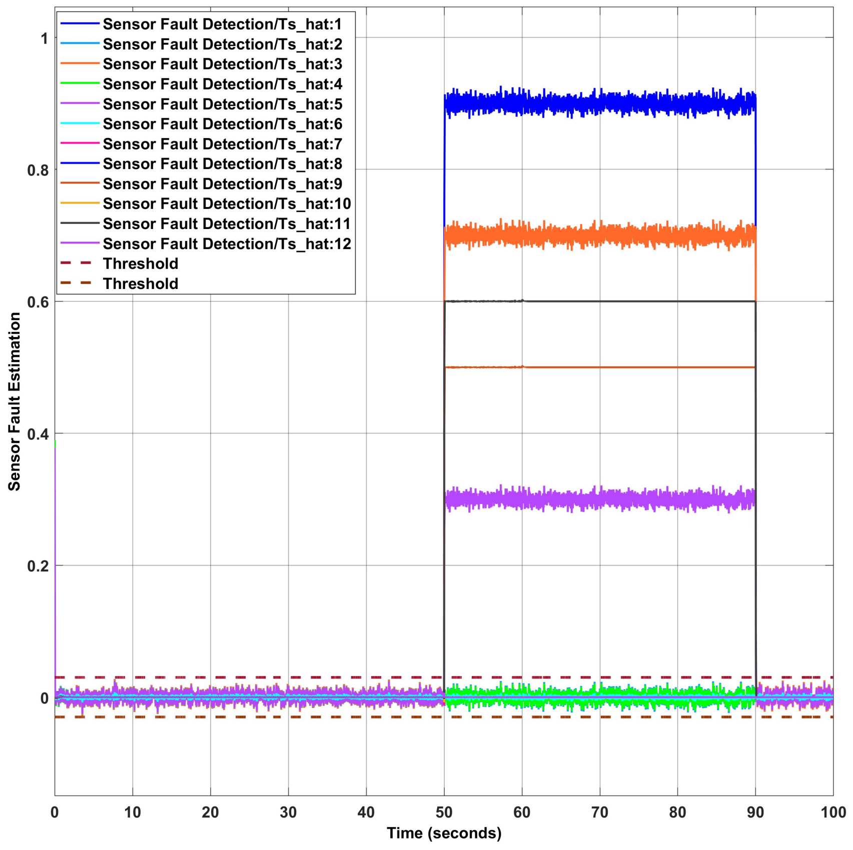

Figure 4.21: The sensor fault detection diagnosis response demonstrates its ability to detect, isolate, and estimate bias sensor faults in the presence of replay attacks and bias sensor faults using a virtual sensor.

Figure 4.22 shows the output of the replay attack detector on the plant side. This detector shows that by utilizing a virtual sensor, the detector is only triggered by replay attacks, while it is not affected by bias sensor faults. The figure highlights the neural network-based detector's ability to detect, isolate, and estimate the severity of an attack. The figure demonstrates that the detector estimates false data injection attacks and considers them as residuals. These residuals are compared to a threshold to determine whether a cyberattack has occurred. Additionally, the detector can identify the control input signals affected by the cyberattack for attack isolation. The figure displays two transient bumps that last for less than two seconds. These bumps are caused by bias sensor faults occurring at times 50 and 90, which are shown in the figure.

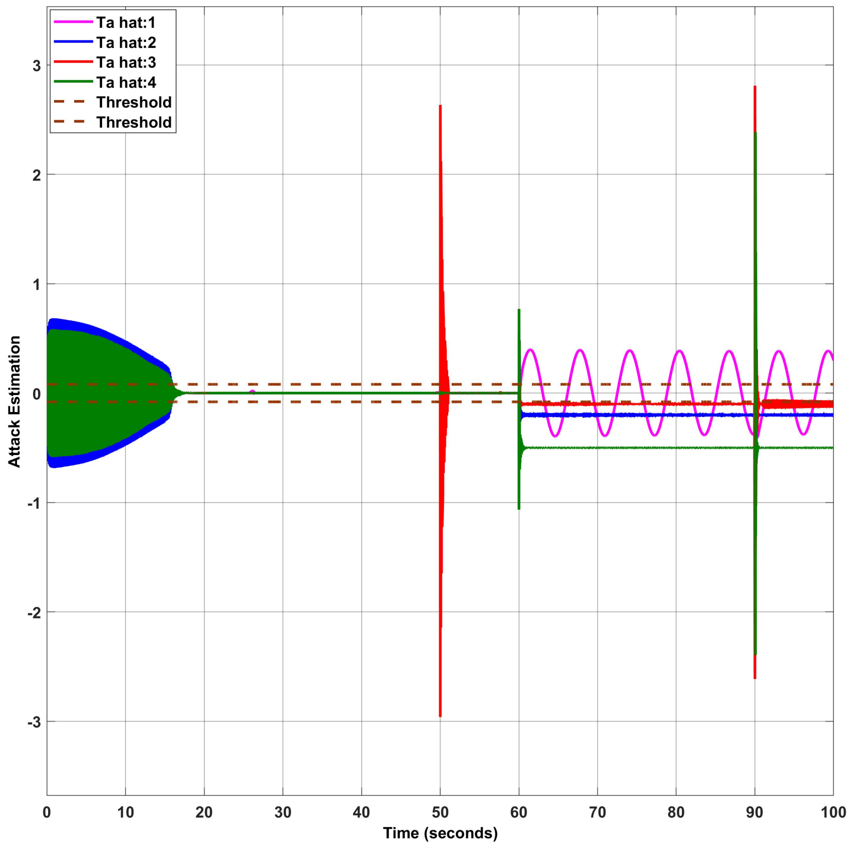

Figure 4.22: The response of a replay attack detection diagnosis on the plant side demonstrates its ability to detect, isolate, and estimate the severity of an attack in the presence of a replay attack and bias sensor faults using a virtual sensor.

Figure 4.23 depicts the output of the replay attack detector on the command and control side. It is observed that this detector is unable to detect the false data injection attack because the sensor measurement data are manipulated, and healthy data is replayed. By comparing the results of the detectors on the plant and command and control sides, it can be concluded that a replay attack scenario has occurred. The figure depicts two transient bumps lasting for less than two seconds, occurring at times 50 and 90. It is important to note that these bumps are due to the occurrence of bias sensor faults.

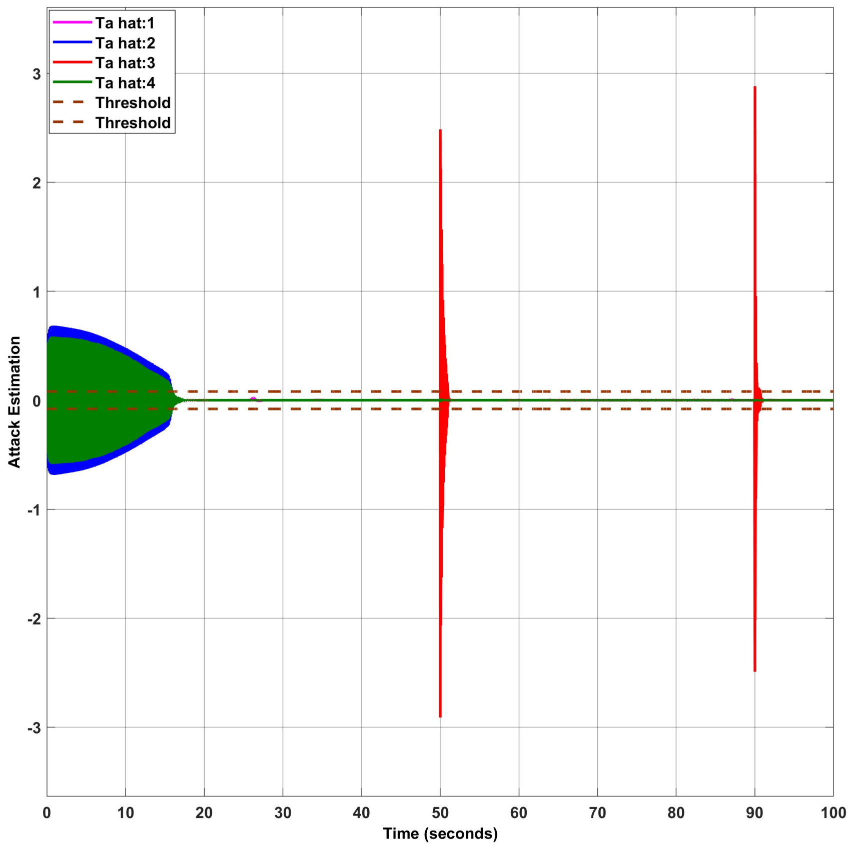

Figure 4.23: Replay attack detector on the C&C side while the system is under replay attack and sensor faults using a virtual sensor.

### 4.2.2 Detection of Replay Attacks and Isolation from Sensor Faults on the Quadrotor- Second Scenario

#### 4.2.2.1 Fault-Free and Attack-Free Scenario

In the second scenario, The performance of the neural network-based detector while the desired trajectory is in the periodic steady-state condition is investigated. The desired trajectory in $x$ and $z$ direction is to start from the origin and reach the point (10,10). However, the desired path in $y$ direction is defined as follows:

$y = 0.75 * sin(0.5t)$.

In Figure 4.24, the quadrotor's position and Euler angles are displayed as follows a commanded trajectory from the command and control. The figure shows that the quadrotor maintains a hovering position at the point (10,10) in the $x$ and $z$ directions while indicating a sinusoidal behavior in the $y$ direction. No attack or sensor faults are present during this time period.

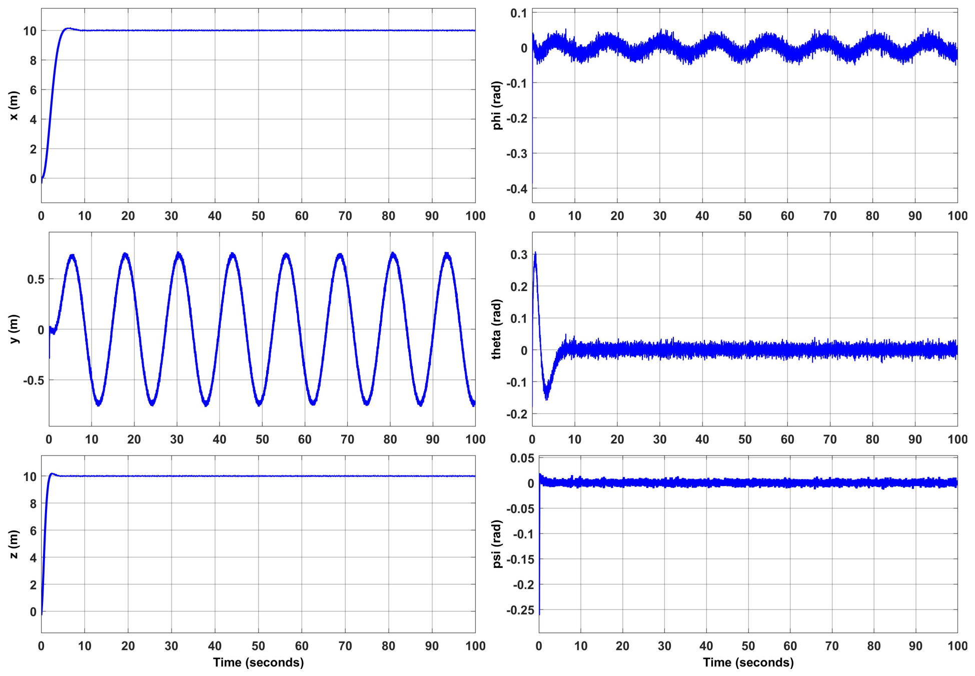

Figure 4.24: Linear position and Euler angles of the quadrotor while there are no attack and sensor faults.

Figures 4.25 and 4.26 display the performance of the detectors in the absence of sensor faults and replay attacks. The figures demonstrate that these detectors remain inactive when the system operates without sensor faults or attacks.

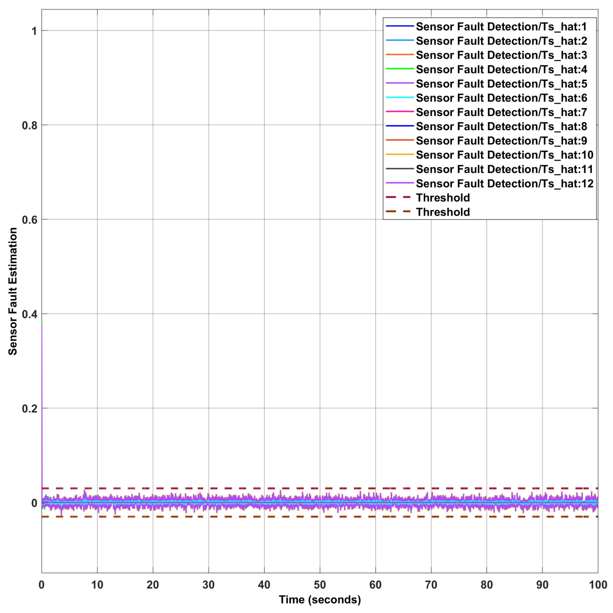

Figure 4.25: The response of the sensor fault detection diagnosis on the plant side in normal condition. Refers to page 124 for the selection of the threshold.

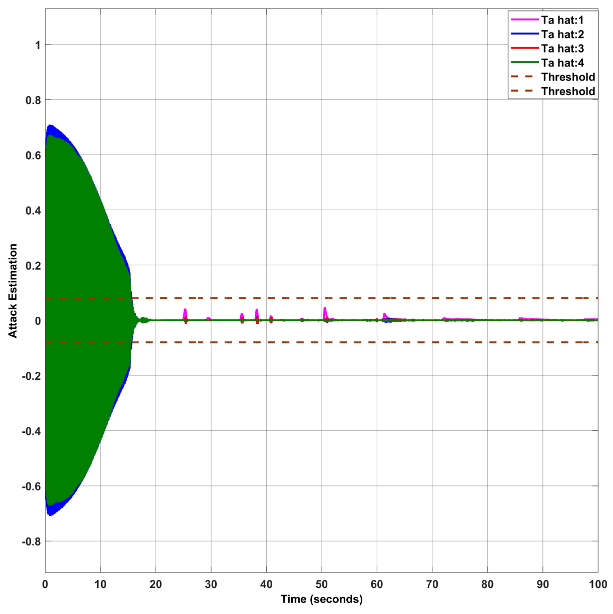

Figure 4.26: The response of the replay attack detection diagnosis on the C&C side and plant side in normal condition. Refers to page 124 for the selection of the threshold.

#### 4.2.2.2 Fault-Free and Replay Attack Scenario

This section explores the detection of replay attacks on a nonlinear quadrotor. In a replay attack, the attacker replays sensor measurements in the system to disrupt the control system without being detected. To achieve this, the attacker records measurement data from a specific time interval, such as the [20,60] interval, and replays it at a later time.

$$\begin{cases} \dot{x}(t) = f\left(x(t), u(t)\right), \\ y(t) = Cx(t), \end{cases} \quad 20 \le t \le 60, \tag{4.23}$$

In the second stage, the attacker replayed the recorded data from the time window of [60,100] to cause damage to the system without being detected. During this stage, the sensor measurements were replaced with previously recorded data, and the system's abnormal behavior was concealed from the command and control center. The attacker manipulated the control input to achieve this goal. The false data injected into the control input is represented as:

$$\begin{cases} \dot{x}(t) = f\left(x(t), \tilde{u}(t)\right), \\ \tilde{u}(t) = u(t) + u_a, \\ \tilde{y}(t) = y(t-40), \end{cases} \quad 60 \le t \le 100, \tag{4.24}$$

where the attack signal is defined as :

$u_a = [0 \quad 0.5 * cos(t) \quad 0 \quad 0.2]$.

The real linear position and Euler angles of the quadrotor under a replay attack scenario are shown in Figure 4.27. The figure highlights that the replay attack can significantly impact and destabilize the quadrotor's operation. However, the effect of the replay attack is concealed from the controller and the command and control center. The impact of the replay attack on the quadrotor's position and Euler angles is shown in Figure 4.28. It is shown that from the command and control point of view, nothing abnormal happened.

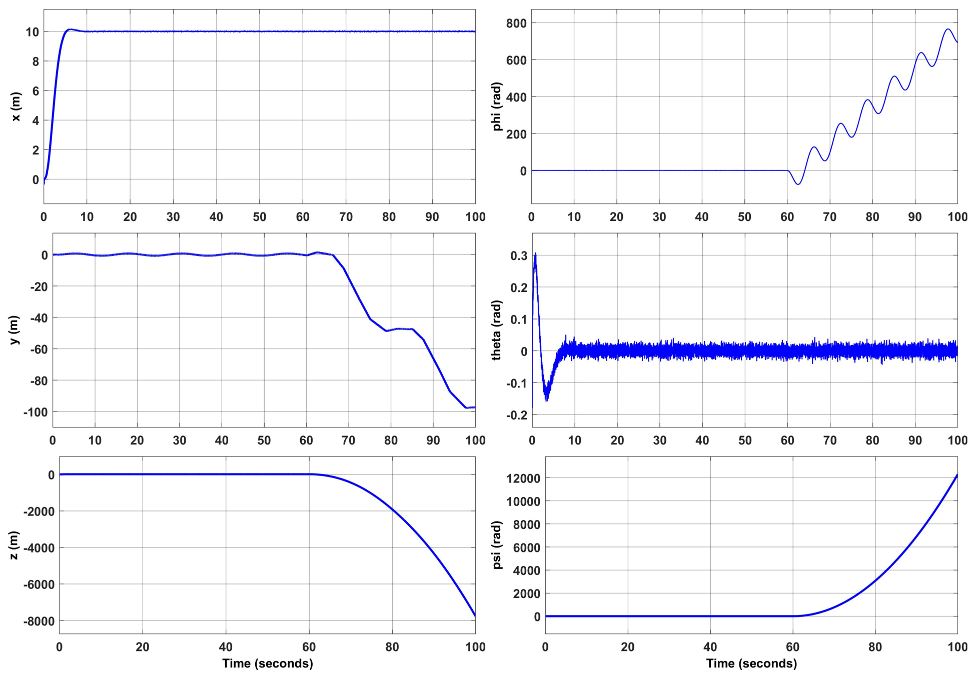

Figure 4.27: Quadrotor linear position and Euler angles under the replay attack.

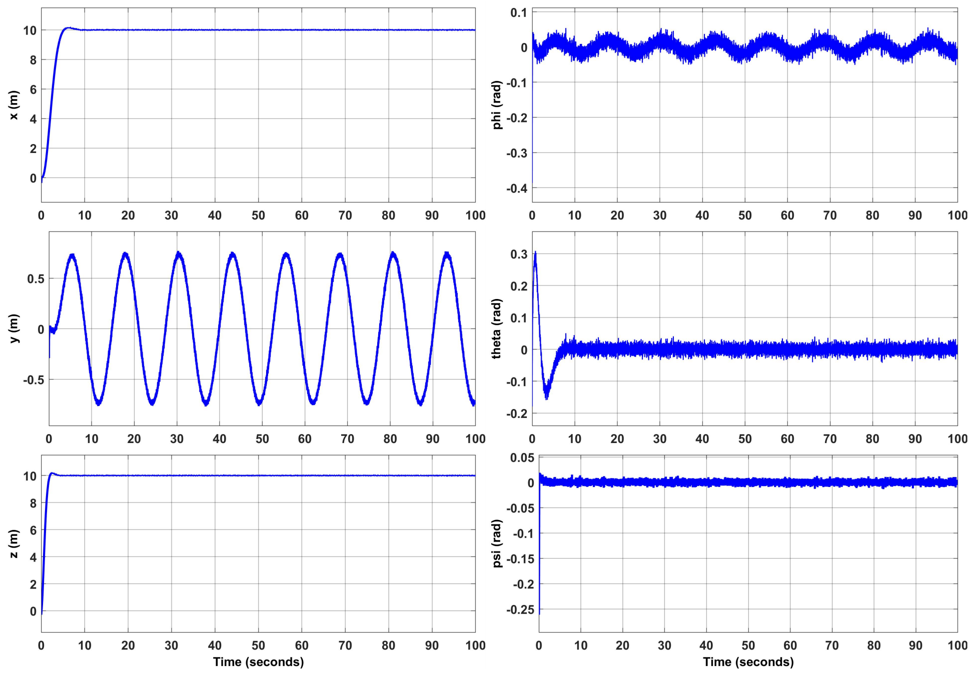

Figure 4.28: The sensor measurements data for linear position and Euler angles of the quadrotor under the replay attack.

The output of the sensor fault detector is depicted in Figure 4.29. The figure indicates that the sensor fault detector is not triggered by replay attacks.

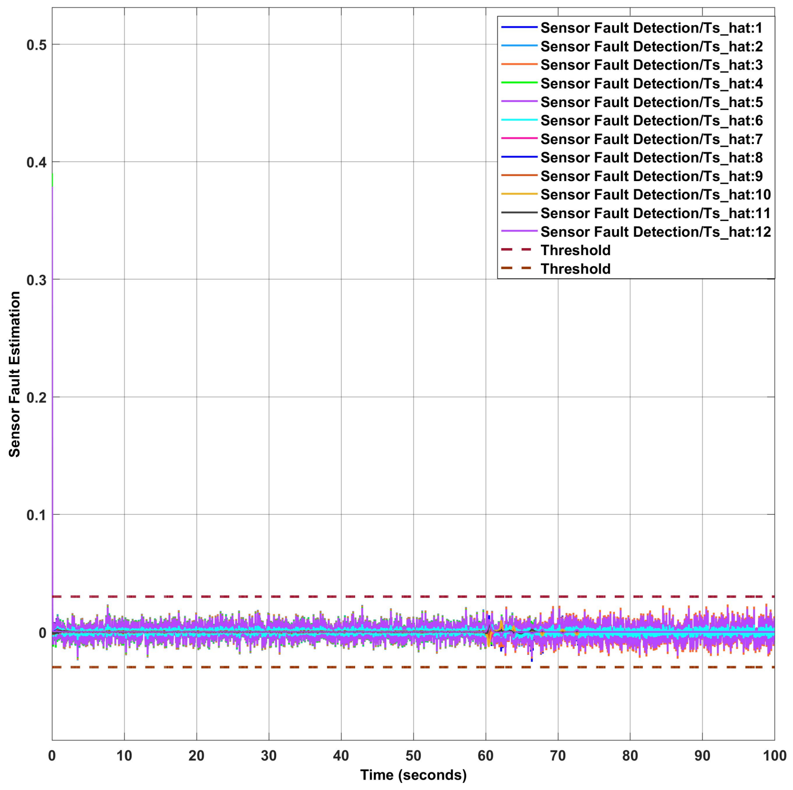

Figure 4.29: The response of the sensor fault detection diagnosis on the plant side while the system is under the replay attack.

Two neural network-based detectors have been employed for detecting replay attacks on both the plant side and the command and control side. Figure 4.30 illustrates the output of the neural network-based detector on the plant side. The figure shows that the detector can successfully estimate false data injection attacks on the control input signals, which are then considered residuals. These residuals overreach a predefined threshold, and allow the detector to detect cyberattacks. Also, the detector can isolate false data injection attacks in two control input signals. It should be noted that, due to the structure of the neural network-based detector and the replay attack, the estimated attack signal has the opposite sign of the actual attack signal. However, the neural network-based detector on the command and control side does not trigger, as shown in Figure 4.31, indicating that the attack signal is not detected at the command and control center. Since the attack is detected on the plant side and is not detected on the command and control side, it means that a replay attack has occurred on the system.

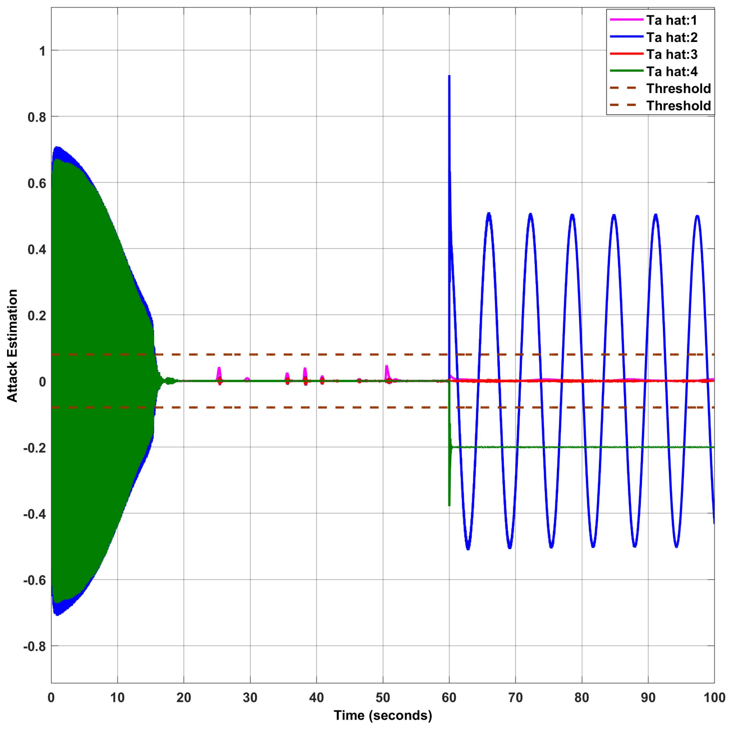

Figure 4.30: The response of a replay attack detection diagnosis on the plant side demonstrates its ability to detect, isolate, and estimate the severity of an attack.

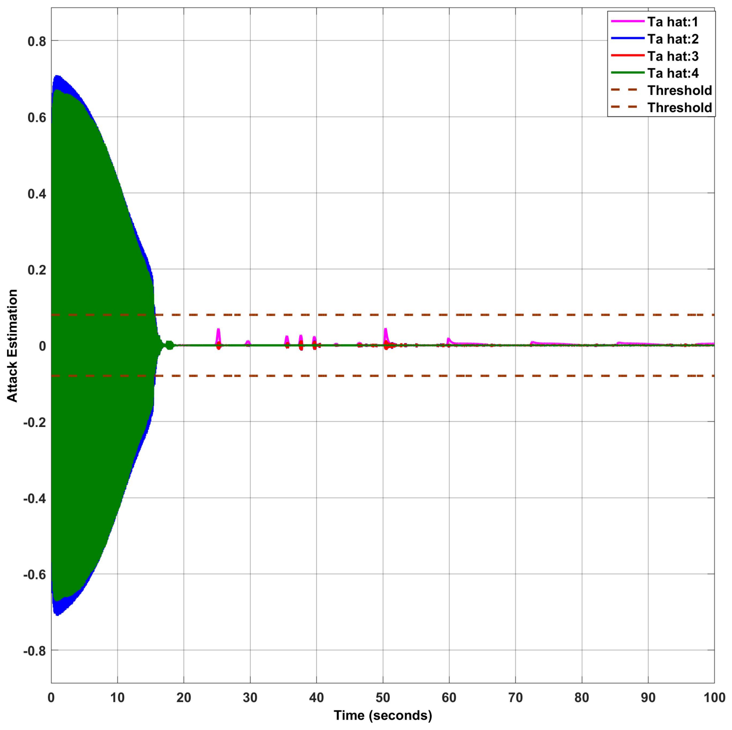

Figure 4.31: The response of a replay attack detection diagnosis on the C&C side.

#### 4.2.2.3 Sensor Fault and Replay Attack Scenario without using a Virtual Sensor

In this scenario, in addition to the replay attack, the system has bias sensor faults in the interval [50,90]. In this case, the virtual sensor for recovery of bias sensor faults is not considered at the plant side. The bias sensor faults are considered as:

$T_s = [0 \quad 0.2 \quad 0.25 \quad 0.3 \quad 0 \quad 0.1 \quad 0 \quad 0.32 \quad 0 \quad 0.15 \quad 0 \quad 0.45]$.

Figure 4.32 shows the sensor measurement data for the linear position and Euler angles of the quadrotor, which is sent to the command and control center and controller. The figure shows that sensor faults occur at time 50 but do not persist until the end of the period at 90 because the data is replayed from the time interval [20, 60]. However, at 90, the fault presence is shown again in the linear position and Euler angles of the quadrotor. Since, in the replay attack scenario, the sensor measurements are replaced by the previously recorded data, the severe impact of the replay attack is not evident.

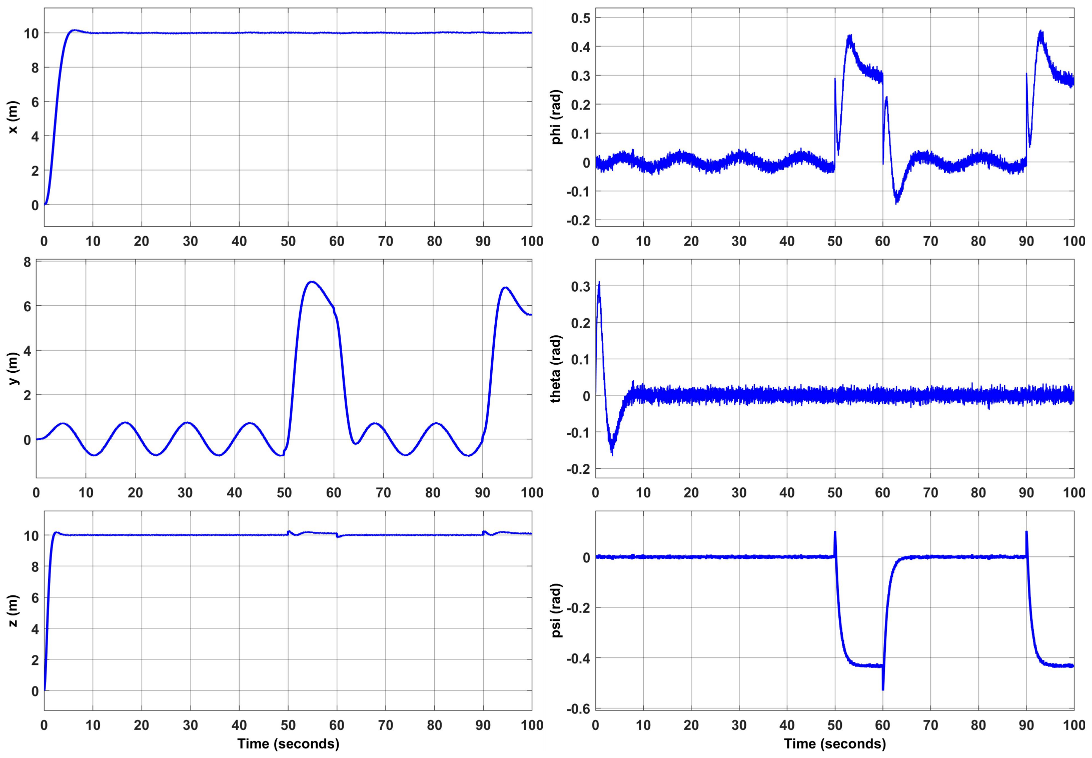

Figure 4.32: The sensor measurements data for linear position and Euler angles of the quadrotor under replay attack and sensor faults.

The linear position and Euler angles of the quadrotor are shown in Figure 4.33, revealing the damaging effects of a replay attack on the system. Despite the significant harm caused to the quadrotor and its destabilization, the command and control center remains unaware of the attack's impact..

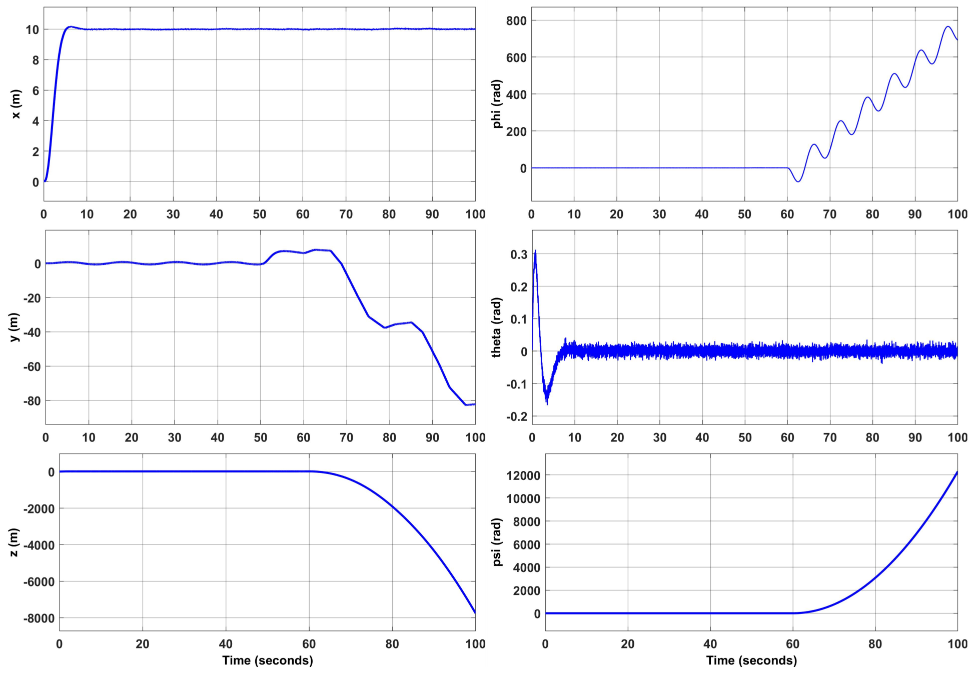

Figure 4.33: Quadrotor linear position and Euler angles under replay attack and sensor faults.

Figure 4.34 presents the output of the sensor fault detector, presenting its effectiveness in detecting bias sensor faults. The detector estimates the faults, considers them residuals, and triggers an alert when they exceed a specific threshold. Moreover, the detector can isolate bias sensor faults in multiple sensors. Additionally, the detector does not detect replay attacks, as it correctly detects bias sensor faults.

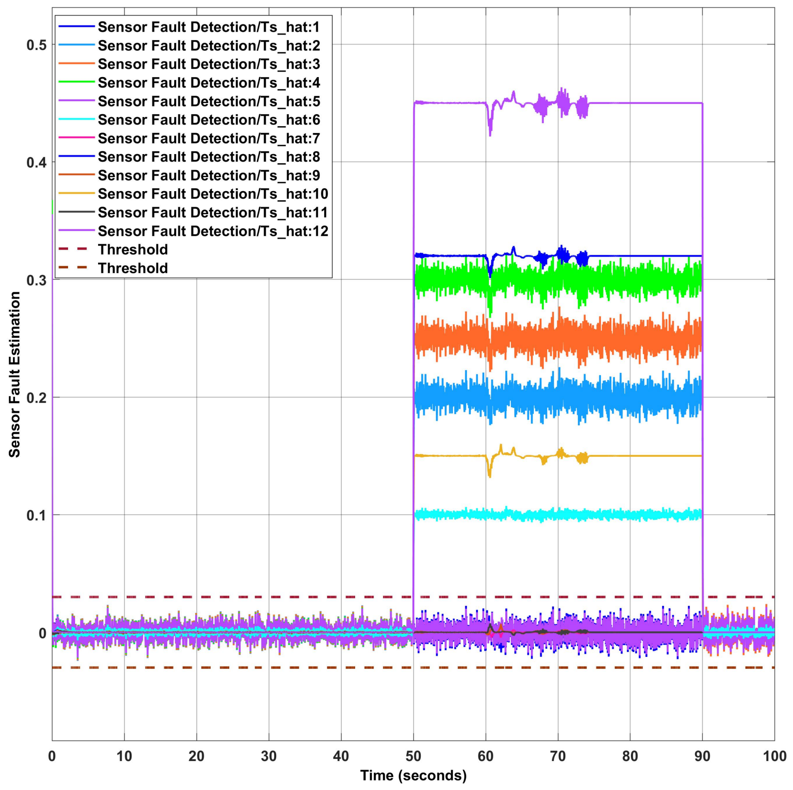

Figure 4.34: The sensor fault detection diagnosis response demonstrates its ability to detect, isolate, and estimate bias sensor faults in the presence of replay attacks and bias sensor faults.

Figure 4.35 displays the output of the replay attack detector on the plant side. As there is no virtual sensor to recover data from bias sensor faults on the plant side, the detector is triggered by such faults. However, at time 60, a replay attack occurs, and the neural network-based detector on the plant side detects it. Without a virtual sensor, the replay attack detector cannot function correctly regarding isolating and estimating the false data injected into input control signals. Moreover, sensor faults and replay attacks cannot be isolated in this case.

Figure 4.36 presents the output of the neural network-based detector on the command and control side. Bias sensor faults trigger the detector, but it cannot detect replay attacks.

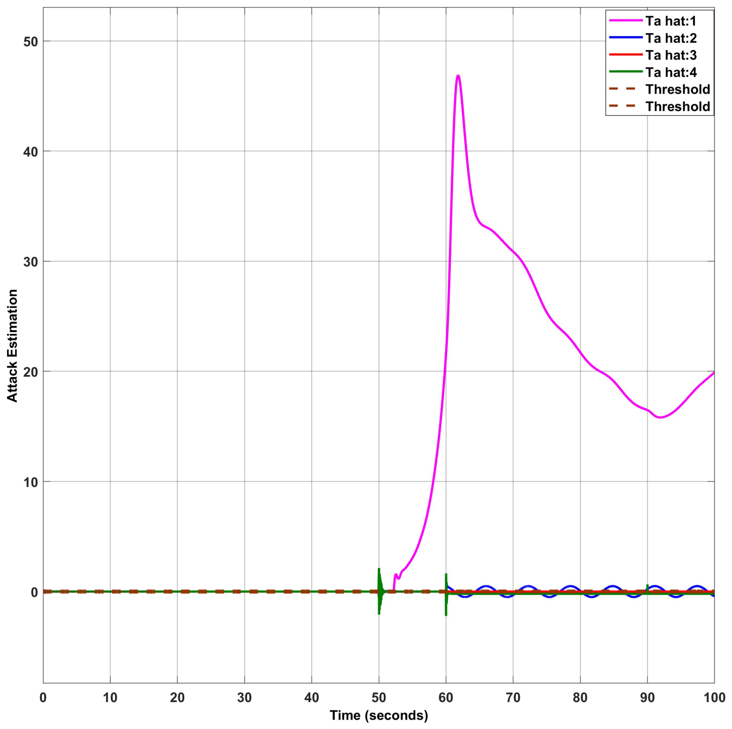

Figure 4.35: The response of the replay attack detection diagnosis on the plant side during a system under replay attack and sensor faults.

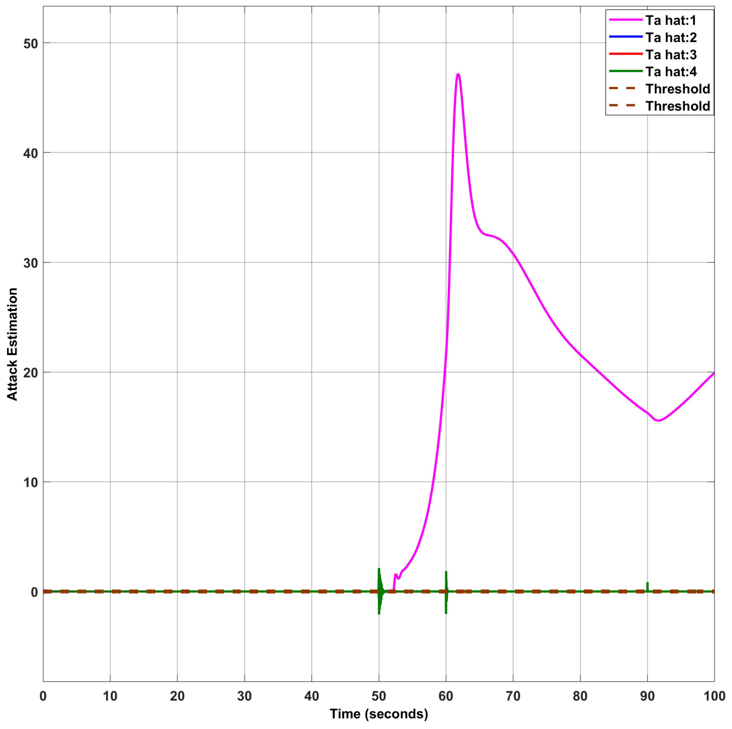

Figure 4.36: The response of the replay attack detection diagnosis on the C&C side during a system under replay attack and sensor faults.

#### 4.2.2.4 Sensor Fault and Replay Attack Scenario using a Virtual Sensor

In this scenario, a virtual sensor for data recovery of bias sensor faults is employed on the plant side. Figure 4.37 displays the sensor measurement data for the quadrotor's linear position and Euler angles, with the impact of bias sensor faults, successfully recovered. However, since replay attacks replace the sensor measurements with previously recorded data, their impact is not visible in this figure.

Figure 4.38 shows the actual linear position and Euler angles of the quadrotor, indicating the significant damage and destabilization that a replay attack can cause. However, these effects may not be evident to the controller and the command and control center.

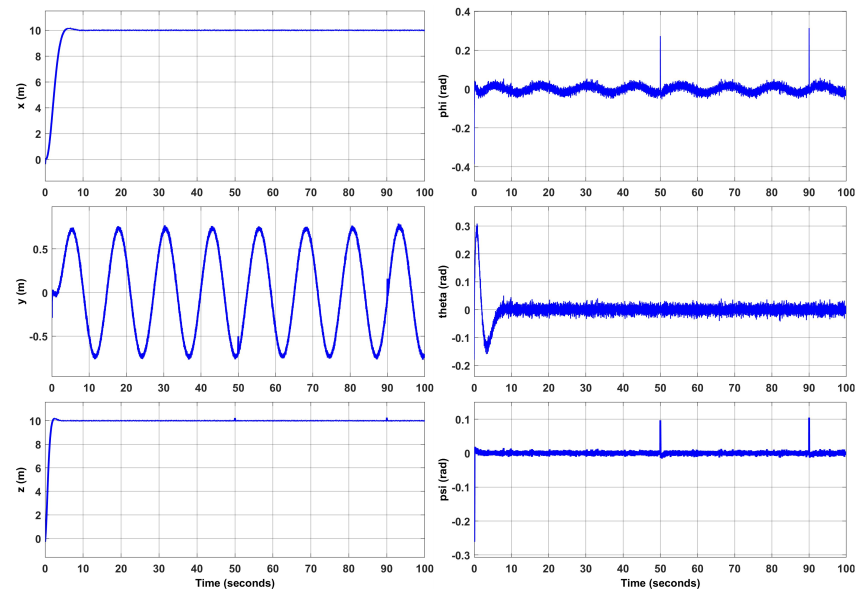

Figure 4.37: The sensor measurements data for linear position and Euler angles of the quadrotor under replay attack and sensor faults using a virtual sensor.

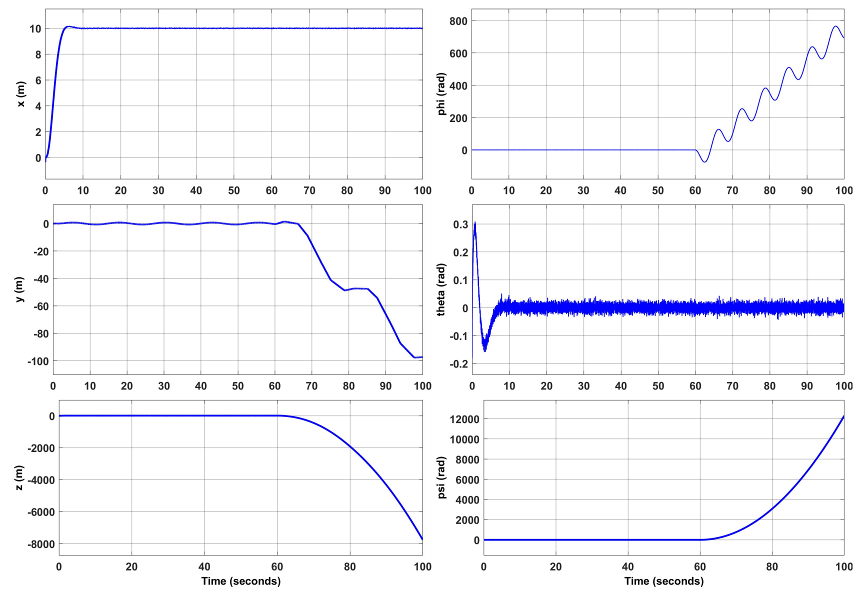

Figure 4.38: Quadrotor linear position and Euler angles under replay attack and sensor faults using a virtual sensor.

Figure 4.39 displays the output of the sensor fault detector. The figure demonstrates the detector's ability to estimate bias sensor faults and then consider them as residuals. These residuals exceed a predefined threshold, allowing the detector to detect the bias sensor faults. Furthermore, the detector can isolate bias sensor faults in different sensors. It is important to note that the detector is not capable of detecting replay attacks and is only designed to detect bias sensor faults.

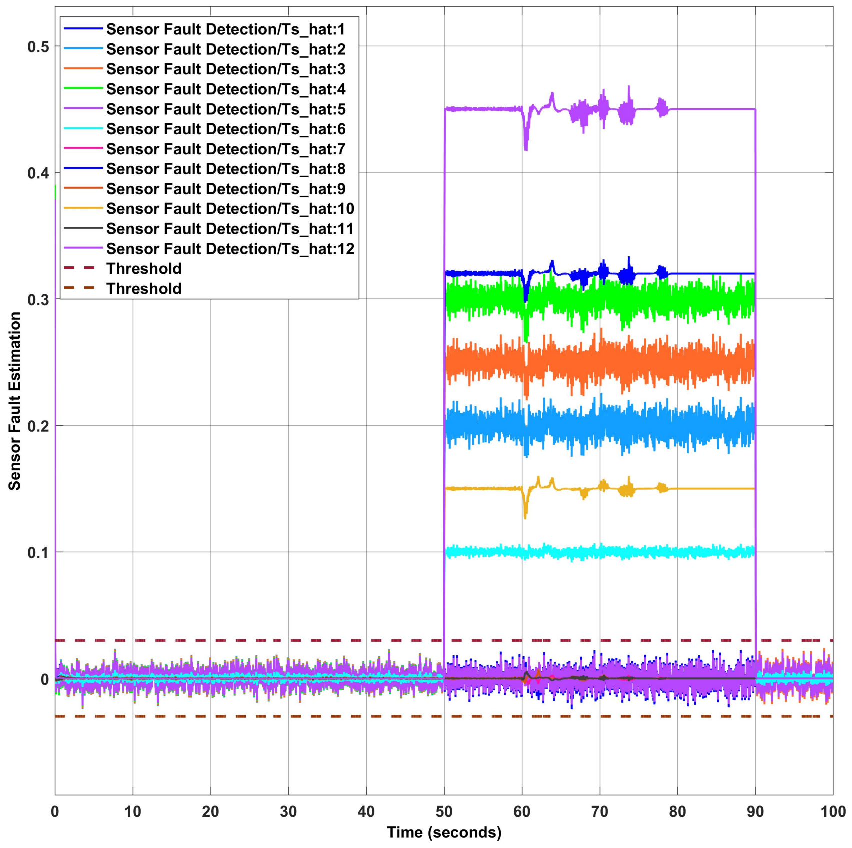

Figure 4.39: The sensor fault detection diagnosis response demonstrates its ability to detect, isolate, and estimate bias sensor faults in the presence of replay attacks and bias sensor faults using a virtual sensor.

Figure 4.40 presents the output of the neural network-based detector on the plant side. By employing a virtual sensor on the plant side, this detector is just triggered by replay attacks, whereas bias sensor faults do not impact it. The figure shows the detector's ability to detect, isolate, and estimate the severity of a cyberattack. The figure illustrates that the detector estimates false data injection attacks by treating them as residuals, which are then compared to a predefined threshold to determine the occurrence of a cyberattack. Additionally, the detector can identify the control input signals affected by the attack and can isolate the attack. The figure features two transient bumps that last for less than two seconds, which are caused by bias sensor faults occurring at times 50 and 90, as displayed in the figure. It should be taken into account that the estimated attack signal has the opposite sign due to the structure of the neural network-based detector and the replay attack.

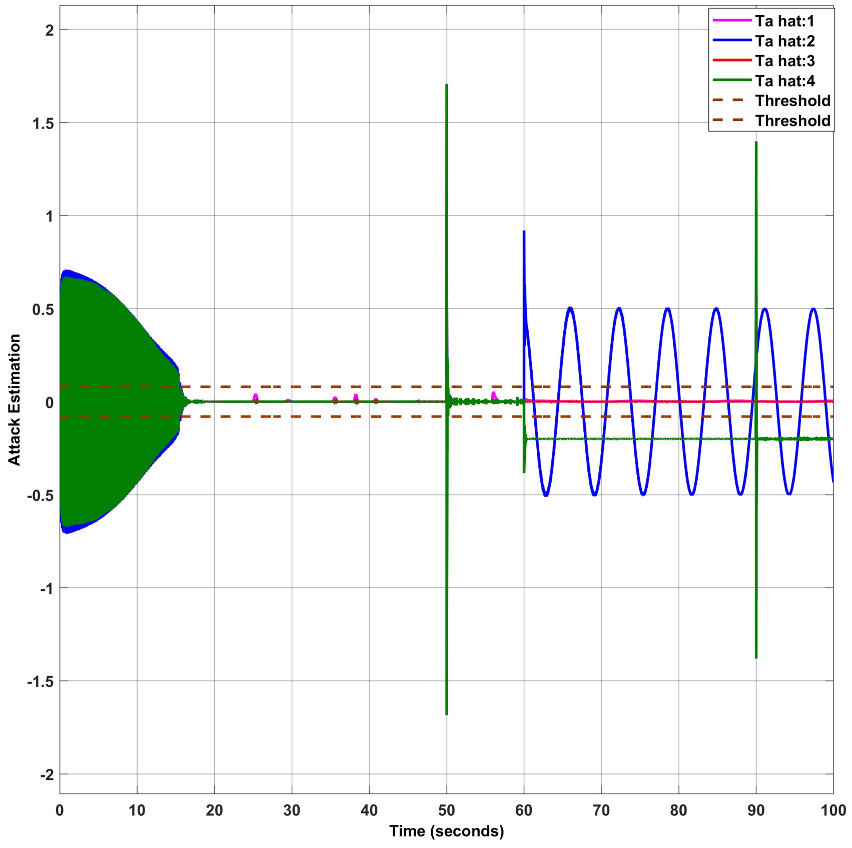

Figure 4.40: The response of a replay attack detection diagnosis on the plant side demonstrates its ability to detect, isolate, and estimate the severity of an attack in the presence of a replay attack and bias sensor faults using a virtual sensor.

In Figure 4.41, the output of the replay attack detector on the command and control side is displayed. The figure shows that the detector is unable to detect the false data injection attack into control input signals as the sensor measurement data are manipulated, and healthy data is replayed. By comparing the results of the detectors on the plant and command and control sides, it can be concluded that a replay attack scenario has occurred. The figure displays two transient bumps occurring at times 50 and 90, lasting for less than two seconds. It is important to note that bias sensor faults cause these bumps.

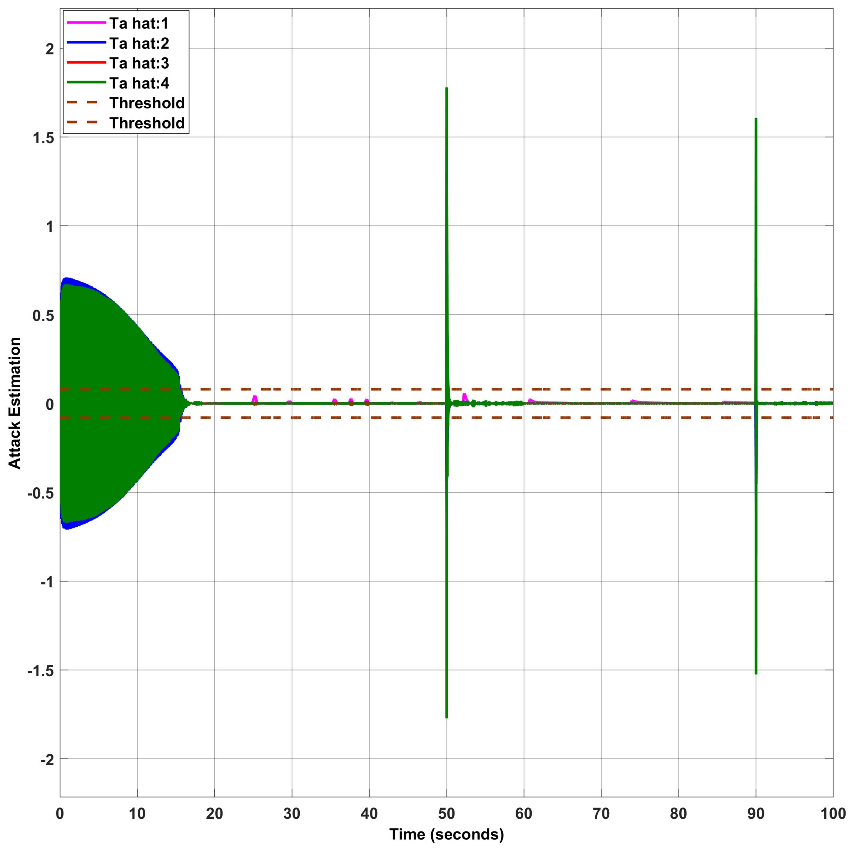

Figure 4.41: Replay attack detector on the C&C side while the system is under replay attack and sensor faults using a virtual sensor.

## 4.3 Performance Analysis

In this section, the performance of the designed NN-based diagnosis is investigated. The approach for calculating confusion matrix is explained in the previous chapter in the section 3.6.

**System under bias sensor faults and replay attacks**

It is assumed that system is under bias sensor faults and replay attacks, and there is no virtual sensor on the plant side. The values in these tables are derived from 100 simulation runs. Table 4.1 presents the results of the performance evaluation of the neural network-based diagnosis system in terms of its ability to detect, isolate, and estimate sensor faults. Table 4.2

presents the performance of the replay attack detector on the C&C side in detecting the bias sensor faults and replay attacks.

Table 4.1: *Confusion matrix for the sensor fault diagnosis when the system under bias sensor faults and replay attacks.*

| | TN | FP | TP | FN |
|---|---|---|---|---|
| Faults&replay attack | 98.01 | 1.99 | 97.63 | 2.37 |
| PPV | 98% | | | |
| ACC | 97.82% | | | |

Table 4.2: *Confusion matrix for the replay attack diagnosis when the system under bias sensor faults and replay attacks.*

| | TN | FP | TP | FN |
|---|---|---|---|---|
| Faults&replay attack | 73.42 | 26.58 | 81.23 | 18.77 |
| PPV | 75.34% | | | |
| ACC | 77.32% | | | |

**System under bias sensor faults and replay attacks using a virtual sensor on the plant side**

It is assumed that system is under bias sensor faults and replay attacks and there is a virtual sensor on the plant side. The values in these tables are derived from 100 simulation runs. The performance of two neural network-based diagnosis approaches is evaluated in terms of their ability to detect, isolate, and estimate sensor faults and replay attacks, and it is presented in Tables 4.3 and 4.4.

Table 4.3: *Confusion matrix for the sensor fault diagnosis when the system under bias sensor faults and replay attacks using a virtual sensor.*

| | TN | FP | TP | FN |
|---|---|---|---|---|
| Faults&replay attack | 97.24 | 2.76 | 98.86 | 1.14 |
| PPV | 97.28% | | | |
| ACC | 98.05% | | | |

Table 4.4: *Confusion matrix for the replay attack diagnosis when the system under bias sensor faults and replay attacks using a virtual sensor.*

| | TN | FP | TP | FN |
|---|---|---|---|---|
| Faults&replay attack | 93.78 | 6.22 | 92.28 | 7.72 |
| PPV | 93.68% | | | |
| ACC | 93.03% | | | |

## 4.4 Conclusions

This chapter provides a neural network-based detection technique for the detection, isolation, and estimation of replay attacks, as well as isolating them from sensor faults. Unlike the method proposed in Chapter 3, this approach is capable of detecting replay attacks. The virtual sensor used on the plant side for bias sensor faults recovery makes it possible to isolate sensor faults and replay attacks. The performance of the diagnoses are also calculated in this chapter.

# Chapter 5

# Conclusions and Future Work

In this chapter, the outcomes of the thesis are concluded, and some potential directions for future work are provided.

## 5.1 Conclusions

This thesis investigates the problem of isolating faults and cyberattacks in the nonlinear CPS. Isolation of faults and cyberattacks is considered one of the challenging problems in control systems. The main focus of this thesis is to provide schematics to isolate sensor faults, false data injection attacks, and replay attacks.

Firstly, the quadrotor mathematical model is derived using the Newton-Euler formulations in the second chapter. The feedback linearization control approach, one of the nonlinear control techniques, is introduced. The possible faults and cyberattacks in the CPS are briefly introduced.

The third chapter investigates the problem of detecting multiplicative false data injection attacks and isolation from bias sensor faults. The neural network-based detection approach is used for detecting, isolating, and estimating faults and false data injection attacks. Two neural network-based detectors are used in this chapter. In order to detect sensor faults, a detector is placed at the plant side. The estimated sensor faults can be used in virtual sensors, which enables simultaneous detection of sensor faults and false data injection attacks. The other detector is placed at the command and control center to detect false data injection attacks in the output communication channel. The false data injection attacks on the sensor measurement data sent from the quadrotor to GCS and the desired trajectory sent from GCS to the quadrotor can be detected in this approach. Since the desired trajectory is generated on the command and control it is affected by the multiplicative false data injection attacks. Therefore, a virtual sensor is considered at the command and control center for data recovery. The proposed approach is not able to detect replay attacks. Therefore, a new neural network-based detector is proposed in chapter four.

In chapter four, the problem of detecting replay attacks is considered, as well as the isolation of replay attacks from sensor faults. A new online neural network-based detection approach is introduced. In order to detect replay attacks, one detector is placed on the plant side, and the other one is placed on the command and control side. If there is a replay attack on the system, the one on the plant side is just triggered. Observing these two detectors makes it possible to distinguish between false data injection attacks and replay attacks.

Simulation results are provided to show the effectiveness of the proposed detectors for the detection of sensor faults, false data injection attacks, and replay attacks in different scenarios.

## 5.2 Future Work

In this work, it is assumed that a full-state measurement is available. Considering this assumption is not possible in most cases. Hence, an extension to this work can be using a state estimator for general output feedback scenarios.

This work just considered the faults in the sensor of the quadrotors. However, the actuator faults can also affect the system's performance. Therefore, one of the great potential extensions to this work is considering actuator faults in the quadrotor and designing a detector for fault estimation. If an online neural network-based detection approach is considered, it can be used in virtual actuators.

The neural network-based detection mechanisms which are proposed in this thesis are able to estimate the attack signal. Thus, presenting a mitigation scheme for the mitigation of replay attacks can be an exciting topic for further research.

If a secure channel is considered for data transformation by using the proposed detector, it is possible to detect covert attacks. Accordingly, another direction of future studies can be working on the necessary condition for detecting covert attacks in the nonlinear CPS and isolating covert attacks and faults.

# REFERENCES

[1] M. Ranjbaranhesarmaskan, *Fault recovery of an under-actuated quadrotor aerial vehicle.* PhD thesis, Concordia University, 2010.

[2] H. Fawzi, P. Tabuada, and S. Diggavi, “Secure estimation and control for cyber-physical systems under adversarial attacks,” *IEEE Transactions on Automatic control*, vol. 59, no. 6, pp. 1454–1467, 2014.

[3] M. H. Amini, K. G. Boroojeni, S. Iyengar, F. Blaabjerg, P. M. Pardalos, and A. M. Madni, “A panorama of future interdependent networks: From intelligent infrastructures to smart cities,” in *Sustainable interdependent networks*, pp. 1–10, Springer, 2018.

[4] J. Liu, Z.-G. Wu, D. Yue, and J. H. Park, “Stabilization of networked control systems with hybrid-driven mechanism and probabilistic cyber attacks,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 2, pp. 943–953, 2019.

[5] D. Ding, Q.-L. Han, Y. Xiang, X. Ge, and X.-M. Zhang, “A survey on security control and attack detection for industrial cyber-physical systems,” *Neurocomputing*, vol. 275, pp. 1674–1683, 2018.

[6] A. Teixeira, D. Pérez, H. Sandberg, and K. H. Johansson, “Attack models and scenarios for networked control systems,” in *Proceedings of the 1st international conference on High Confidence Networked Systems*, pp. 55–64, 2012.

[7] M. Ciampa, *Security+ guide to network security fundamentals.* Cengage Learning, 2012.

[8] H. S. Sánchez, D. Rotondo, T. Escobet, V. Puig, and J. Quevedo, “Bibliographical review on cyber attacks from a control oriented perspective,” *Annual Reviews in Control*, vol. 48, pp. 103–128, 2019.

[9] A. Teixeira, I. Shames, H. Sandberg, and K. H. Johansson, “A secure control framework for resource-limited adversaries,” *Automatica*, vol. 51, pp. 135–148, 2015.

[10] Y. Z. Lun, A. D’Innocenzo, F. Smarra, I. Malavolta, and M. D. Di Benedetto, “State of the art of cyber-physical systems security: An automatic control perspective,” *Journal of Systems and Software*, vol. 149, pp. 174–216, 2019.

[11] X.-M. Zhang, Q.-L. Han, X. Ge, D. Ding, L. Ding, D. Yue, and C. Peng, “Networked control systems: A survey of trends and techniques,” *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 1, pp. 1–17, 2019.

[12] C. Kwon, W. Liu, and I. Hwang, “Security analysis for cyber-physical systems against stealthy deception attacks,” in *2013 American control conference*, pp. 3344–3349, IEEE, 2013.

[13] A. Gupta, A. Sikdar, and A. Chattopadhyay, “Quickest bayesian and non-bayesian detection of false data injection attack in remote state estimation,” *arXiv e-prints*, pp. arXiv–2010, 2020.

[14] Z. Qu, J. Zhang, Y. Wang, P. M. Georgievitch, and K. Guo, “False data injection attack detection and improved wls power system state estimation based on node trust,” *Journal of Electrical Engineering & Technology*, pp. 1–15, 2021.

[15] K. Manandhar, X. Cao, F. Hu, and Y. Liu, “Detection of faults and attacks including false data injection attack in smart grid using kalman filter,” *IEEE transactions on control of network systems*, vol. 1, no. 4, pp. 370–379, 2014.

[16] S. Amin, X. Litrico, S. S. Sastry, and A. M. Bayen, “Cyber security of water scada systems—part ii: Attack detection using enhanced hydrodynamic models,” *IEEE Transactions on Control Systems Technology*, vol. 21, no. 5, pp. 1679–1693, 2013.

[17] S. A. Foroutan and F. R. Salmasi, “Detection of false data injection attacks against state estimation in smart grids based on a mixture gaussian distribution learning method,” *IET Cyber-Physical Systems: Theory & Applications*, vol. 2, no. 4, pp. 161–171, 2017.

[18] A. Sargolzaei, K. Yazdani, A. Abbaspour, C. D. Crane III, and W. E. Dixon, “Detection and mitigation of false data injection attacks in networked control systems,” *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4281–4292, 2019.

[19] A. Abbaspour, M. Sanchez, A. Sargolzaei, K. K. Yen, and N. Sornkhampan, “Adaptive neural network based fault detection design for unmanned quadrotor under faults and cyber attacks.,” in *ICSEng*, pp. 77–84, 2017.

[20] M. Kravchik and A. Shabtai, “Detecting cyber attacks in industrial control systems using convolutional neural networks,” in *Proceedings of the 2018 workshop on cyber-physical systems security and privacy*, pp. 72–83, 2018.

[21] Y. Chen, S. Huang, F. Liu, Z. Wang, and X. Sun, “Evaluation of reinforcement learning-based false data injection attack to automatic voltage control,” *IEEE Transactions on Smart Grid*, vol. 10, no. 2, pp. 2158–2169, 2018.

[22] F. Farivar, M. S. Haghighi, A. Jolfaei, and M. Alazab, “Artificial intelligence for detection, estimation, and compensation of malicious attacks in nonlinear cyber-physical systems and industrial iot,” *IEEE transactions on industrial informatics*, vol. 16, no. 4, pp. 2716–2725, 2019.

[23] A. Abbaspour, K. K. Yen, S. Noei, and A. Sargolzaei, “Detection of fault data injection attack on uav using adaptive neural network,” *Procedia computer science*, vol. 95, pp. 193–200, 2016.

[24] Y. Mo and B. Sinopoli, “Secure control against replay attacks,” in *2009 47th annual Allerton conference on communication, control, and computing (Allerton)*, pp. 911–918, IEEE, 2009.

[25] Y. Mo, R. Chabukswar, and B. Sinopoli, “Detecting integrity attacks on scada systems,” *IEEE Transactions on Control Systems Technology*, vol. 22, no. 4, pp. 1396–1407, 2013.

[26] Y. Mo, S. Weerakkody, and B. Sinopoli, "Physical authentication of control systems: Designing watermarked control inputs to detect counterfeit sensor outputs," *IEEE Control Systems Magazine*, vol. 35, no. 1, pp. 93–109, 2015.

[27] T. Irita and T. Namerikawa, "Detection of replay attack on smart grid with code signal and bargaining game," in *2017 American Control Conference (ACC)*, pp. 2112–2117, IEEE, 2017.

[28] D. Ye, T.-Y. Zhang, and G. Guo, "Stochastic coding detection scheme in cyber-physical systems against replay attack," *Information Sciences*, vol. 481, pp. 432–444, 2019.

[29] F. Miao, M. Pajic, and G. J. Pappas, "Stochastic game approach for replay attack detection," in *52nd IEEE conference on decision and control*, pp. 1854–1859, IEEE, 2013.

[30] M. Ma, P. Zhou, D. Du, C. Peng, M. Fei, and H. M. AlBuflasa, "Detecting replay attacks in power systems: A data-driven approach," in *Advanced Computational Methods in Energy, Power, Electric Vehicles, and Their Integration*, pp. 450–457, Springer, 2017.

[31] M. Taheri, K. Khorasani, I. Shames, and N. Meskin, "Cyber attack and machine induced fault detection and isolation methodologies for cyber-physical systems," *arXiv preprint arXiv:2009.06196*, 2020.

[32] M. Ramadan and F. Abdollahi, "An active approach for isolating replay attack from sensor faults," *European Journal of Control*, p. 100725, 2022.

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

[34] S. K. Kanev, "Robust fault-tolerant control.," 2004.

[35] E. Sobhani-Tehrani and K. Khorasani, *Fault diagnosis of nonlinear systems using a hybrid approach*, vol. 383. Springer Science & Business Media, 2009.

[36] L. Li, K. Qu, and K.-Y. Lin, "A survey on attack resilient of uav motion planning," in *2020 IEEE 16th International Conference on Control & Automation (ICCA)*, pp. 558–563, IEEE, 2020.

[37] G. Rong-Xiao, T. Ji-wei, W. Bu-hong, and S. Fu-te, "Cyber-physical attack threats analysis for uavs from cps perspective," in *2020 International Conference on Computer Engineering and Application (ICCEA)*, pp. 259–263, IEEE, 2020.

[38] T. Shima and S. Rasmussen, *UAV cooperative decision and control: challenges and practical approaches.* SIAM, 2009.

[39] G. K. Fourlas and G. C. Karras, "A survey on fault diagnosis and fault-tolerant control methods for unmanned aerial vehicles," *Machines*, vol. 9, no. 9, p. 197, 2021.

[40] X. Qi, D. Theilliol, J. Qi, Y. Zhang, J. Han, D. Song, L. Wang, and Y. Xia, "Fault diagnosis and fault tolerant control methods for manned and unmanned helicopters: a literature review," in *2013 Conference on Control and Fault-Tolerant Systems (SysTol)*, pp. 132–139, IEEE, 2013.

[41] H. Shraim, A. Awada, and R. Youness, "A survey on quadrotors: Configurations, modeling and identification, control, collision avoidance, fault diagnosis and tolerant control," *IEEE Aerospace and Electronic Systems Magazine*, vol. 33, no. 7, pp. 14–33, 2018.

[42] A. Drak, H. Noura, M. Hejase, and Y. A. Younes, "Sensor fault diagnostic and fault-tolerant control for the altitude control of a quadrotor uav," in *2015 IEEE 8th GCC Conference & Exhibition*, pp. 1–5, IEEE, 2015.

[43] H. Shi, S. Hu, and J. Zhang, "Research on fault diagnosis of three degrees of freedom gyroscope redundant system," in *2019 CAA Symposium on Fault Detection, Supervision and Safety for Technical Processes (SAFEPROCESS)*, pp. 232–239, IEEE, 2019.

[44] F. R. López-Estrada, J.-C. Ponsart, D. Theilliol, Y. Zhang, and C.-M. Astorga-Zaragoza, "Lpv model-based tracking control and robust sensor fault diagnosis for a quadrotor uav," *Journal of Intelligent & Robotic Systems*, vol. 84, no. 1, pp. 163–177, 2016.

[45] R. C. Avram, X. Zhang, J. Campbell, and J. Muse, "Imu sensor fault diagnosis and estimation for quadrotor uavs," *IFAC-PapersOnLine*, vol. 48, no. 21, pp. 380–385, 2015.

[46] S. Hansen and M. Blanke, "Diagnosis of airspeed measurement faults for unmanned aerial vehicles," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 50, no. 1, pp. 224–239, 2014.

[47] K. Guo, L. Liu, S. Shi, D. Liu, and X. Peng, "Uav sensor fault detection using a classifier without negative samples: A local density regulated optimization algorithm," *Sensors*, vol. 19, no. 4, p. 771, 2019.

[48] R. Sun, Q. Cheng, G. Wang, and W. Y. Ochieng, "A novel online data-driven algorithm for detecting uav navigation sensor faults," *Sensors*, vol. 17, no. 10, p. 2243, 2017.

[49] Y. Zhong, Y. Zhang, W. Zhang, J. Zuo, and H. Zhan, "Robust actuator fault detection and diagnosis for a quadrotor uav with external disturbances," *IEEE Access*, vol. 6, pp. 48169–48180, 2018.

[50] R. C. Avram, X. Zhang, and J. Muse, "Quadrotor actuator fault diagnosis and accommodation using nonlinear adaptive estimators," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 6, pp. 2219–2226, 2017.

[51] J. Fu, C. Sun, Z. Yu, and L. Liu, "A hybrid cnn-lstm model based actuator fault diagnosis for six-rotor uavs," in *2019 chinese control and decision conference (ccdc)*, pp. 410–414, IEEE, 2019.

[52] R. Shakeri, M. A. Al-Garadi, A. Badawy, A. Mohamed, T. Khattab, A. K. Al-Ali, K. A. Harras, and M. Guizani, "Design challenges of multi-uav systems in cyber-physical applications: A comprehensive survey and future directions," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3340–3385, 2019.

[53] Y. Gu, X. Yu, K. Guo, J. Qiao, and L. Guo, "Detection, estimation, and compensation of false data injection attack for uavs," *Information Sciences*, vol. 546, pp. 723–741, 2021.

[54] G. Vasconcelos, G. Carrijo, R. Miani, J. Souza, and V. Guizilini, "The impact of dos attacks on the ar. drone 2.0," in *2016 XIII Latin American Robotics Symposium and IV Brazilian Robotics Symposium (LARS/SBR)*, pp. 127–132, IEEE, 2016.

[55] C. Gudla, M. S. Rana, and A. H. Sung, “Defense techniques against cyber attacks on unmanned aerial vehicles,” in *Proceedings of the international conference on embedded systems, cyber-physical systems, and applications (ESCS)*, pp. 110–116, The Steering Committee of The World Congress in Computer Science, Computer . . . , 2018.

[56] S. Dahiya and M. Garg, “Unmanned aerial vehicles: Vulnerability to cyber attacks,” in *International Conference on Unmanned Aerial System in Geomatics*, pp. 201–211, Springer, 2019.

[57] J. Chen, Z. Feng, J.-Y. Wen, B. Liu, and L. Sha, “A container-based dos attack-resilient control framework for real-time uav systems,” in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1222–1227, IEEE, 2019.

[58] S. Ouiazzane, M. Addou, and F. Barramou, “A multiagent and machine learning based denial of service intrusion detection system for drone networks,” in *Geospatial Intelligence*, pp. 51–65, Springer, 2022.

[59] S. Garg, G. S. Aujla, N. Kumar, and S. Batra, “Tree-based attack–defense model for risk assessment in multi-uav networks,” *IEEE Consumer Electronics Magazine*, vol. 8, no. 6, pp. 35–41, 2019.

[60] H. Lin, P. Sun, C. Cai, S. Lu, and H. Liu, “Secure lqg control for a quadrotor under false data injection attacks,” *IET Control Theory & Applications*, vol. 16, no. 9, pp. 925–934, 2022.

[61] M. Bahrami and H. Jafarnejadsani, “Detection of stealthy adversaries for networked unmanned aerial vehicles,” *arXiv preprint arXiv:2202.09661*, 2022.

[62] H. A. Talebi, K. Khorasani, and S. Tafazoli, “A recurrent neural-network-based sensor and actuator fault detection and isolation for nonlinear systems with application to the satellite’s attitude control subsystem,” *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 45–60, 2008.

[63] H. A. Talebi and K. Khorasani, “A neural network-based multiplicative actuator fault detection and isolation of nonlinear systems,” *IEEE Transactions on Control Systems Technology*, vol. 21, no. 3, pp. 842–851, 2012.

[64] S. M. Dibaji, M. Pirani, D. B. Flamholz, A. M. Annaswamy, K. H. Johansson, and A. Chakrabortty, “A systems and control perspective of cps security,” *Annual reviews in control*, vol. 47, pp. 394–411, 2019.

[65] H. Yuan and Y. Xia, “Resilient strategy design for cyber-physical system under dos attack over a multi-channel framework,” *Information Sciences*, vol. 454, pp. 312–327, 2018.

[66] D. Mikhaylenko and P. Zhang, “Stealthy local covert attacks on cyber-physical systems,” *IEEE Transactions on Automatic Control*, 2021.

[67] H. S. Sanchez, D. Rotondo, T. Escobet, V. Puig, J. Saludes, and J. Quevedo, “Detection of replay attacks in cyber-physical systems using a frequency-based signature,” *Journal of the Franklin Institute*, vol. 356, no. 5, pp. 2798–2824, 2019.

[68] N. Babadi and A. Doustmohammadi, “A moving target defence approach for detecting deception attacks on cyber-physical systems,” *Computers and Electrical Engineering*, vol. 100, p. 107931, 2022.

[69] M. Mahmoud, J. Jiang, and Y. Zhang, *Active fault tolerant control systems: stochastic analysis and synthesis*, vol. 287. Springer Science & Business Media, 2003.

[70] J. A. Farber and D. G. Cole, “Nonlinear zero-dynamics attacks targeting nuclear power plants,” in *Dynamic Systems and Control Conference*, vol. 84270, p. V001T04A004, American Society of Mechanical Engineers, 2020.

[71] A. Q. Khan, *Observer-based fault detection in nonlinear systems*. PhD thesis, Duisburg, Essen, Univ., Diss., 2010, 2010.

[72] T. Höfling and T. Pfeufer, “Detection of additive and multiplicative faults-parity space vs. parameter estimation,” *IFAC Proceedings Volumes*, vol. 27, no. 5, pp. 515–520, 1994.

[73] Z. Gao, C. Cecati, and S. X. Ding, “A survey of fault diagnosis and fault-tolerant techniques—part i: Fault diagnosis with model-based and signal-based approaches,” *IEEE transactions on industrial electronics*, vol. 62, no. 6, pp. 3757–3767, 2015.

[74] K. M. Thu and A. Gavrilov, “Designing and modeling of quadcopter control system using l1 adaptive control,” *Procedia Computer Science*, vol. 103, pp. 528–535, 2017.

[75] H. Suresh, A. Sulficar, and V. Desai, “Hovering control of a quadcopter using linear and nonlinear techniques,” *International Journal of Mechatronics and Automation*, vol. 6, no. 2-3, pp. 120–129, 2018.

[76] F. Sabatino, “Quadrotor control: modeling, nonlinearcontrol design, and simulation,” 2015.

[77] “https://cfdflowengineering.com/working-principle-and-components-of-drone/.”

[78] F. Abdollahi, H. A. Talebi, and R. V. Patel, “A stable neural network-based observer with application to flexible-joint manipulators,” *IEEE Transactions on Neural Networks*, vol. 17, no. 1, pp. 118–129, 2006.

[79] Y. Wang, D. Rotondo, V. Puig, and G. Cembrano, “Fault-tolerant control based on virtual actuator and sensor for discrete-time descriptor systems,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 12, pp. 5316–5325, 2020.

[80] Y. Wang, D. Rotondo, V. Puig, and G. Cembrano, “Fault-tolerant control based on virtual actuator and sensor for discrete-time descriptor systems,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 12, pp. 5316–5325, 2020.

[81] I. Hameed, E. I. El-Madbouly, and M. I. Abdo, “Sensor and actuator fault-hiding reconfigurable control design for a four-tank system benchmark,” *International Journal of Innovative Computing, Information and Control*, vol. 11, no. 2, pp. 679–690, 2015.

[82] D. Rotondo and V. Puig, “Virtual sensors and actuators,” *Diagnosis and Fault-tolerant Control Volume 2: From Fault Diagnosis to Fault-tolerant Control*, p. 193, 2021.

[83] S. Ghimire, J. Sarraipa, C. Agostinho, and R. Jardim-Goncalves, “Fault tolerant sensing model for cyber-physical systems,” in *Proceedings of the Symposium on Model-Driven Approaches for Simulation Engineering*, pp. 1–9, 2017.

[84] T. Steffen, “Reconfiguration using a virtual sensor,” in *Control reconfiguration of dynamical systems*, pp. 69–79, Springer, 2005.

[85] E. Naderi and K. Khorasani, “Data-driven fault detection, isolation and estimation of aircraft gas turbine engine actuator and sensors,” *Mechanical Systems and Signal Processing*, vol. 100, pp. 415–438, 2018.