DEEP LEARNING-BASED, PASSIVE FAULT TOLERANT CONTROL FACILITATED BY A TAXONOMY OF CYBER-ATTACK EFFECTS

DISSERTATION

Dean C. Wardell, Lieutenant Colonel, USAF

AFIT-ENG-DS-20-D-013

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-DS-20-D-013

DEEP LEARNING-BASED, PASSIVE FAULT TOLERANT CONTROL FACILITATED BY A TAXONOMY OF CYBER-ATTACK EFFECTS

DISSERTATION

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

Dean C. Wardell, BS, MS

Lieutenant Colonel, USAF

23 November 2020

AFIT-ENG-DS-20-D-013

DEEP LEARNING-BASED, PASSIVE FAULT TOLERANT CONTROL FACILITATED BY A TAXONOMY OF CYBER-ATTACK EFFECTS

Dean C. Wardell, BS, MS

Lieutenant Colonel, USAF

Committee Membership:

Dr. Robert F. Mills
Chair

Dr. Gilbert L. Peterson
Member

Dr. Mark E. Oxley
Member

Dr. Adedeji B. Badiru
Dean
Graduate School of Engineering and Management

AFIT-ENG-DS-20-D-013

Abstract

In the interest of improving the resilience of cyber-physical control systems to better operate in the presence of various cyber-attacks and/or faults, this dissertation presents a novel controller design based on deep-learning networks. This research lays out a controller design that does not rely on fault or cyber-attack detection. Being passive, the controller's routine operating process is to take in data from the various components of the physical system, holistically assess the state of the physical system using deep-learning networks and decide the subsequent round of commands from the controller. This use of deep-learning methods in passive fault tolerant control (FTC) is unique in the research literature. The proposed controller is applied to both linear and nonlinear systems. Additionally, the application and testing are accomplished with both actuators and sensors being affected by attacks and /or faults.

The effects of various attacks and faults are identified, and composed into a taxonomy for use in training and testing FTCs. By training the controller on the wide range of cyber-attack and fault effects, the type(s) of attack or fault affecting the system do not need to be known in advance. This research effort challenges the assertion made by some researchers that passive FTC is limited in utility or application because fault information must be known a priori.

Two variants of the proposed controller design are presented and tested; one using a single large deep-learning network and the other using an ensemble of smaller networks. The proposed controller design is compared to an active fault tolerant controller. Both are

applied to simulations of a benchmark non-linear quadruple-tank system. Test results show both versions of the proposed controller design perform monotonically better than the active FTC in keeping or regaining control of the system during a simulated cyber-attack or fault. The proposed controller is also more responsive to changes in the system, not suffering from the delays inherent to active FTCs.

The differences in performance are especially pronounced when the system under control is tested with Stuxnet-like attacks.

Acknowledgments

I would like to express my sincere gratitude to my research committee, Dr. Bob Mills, Dr. Bert Peterson and Dr. Mark Oxley, for their guidance and support throughout the course of this dissertation effort. Their insight, experience and patience are certainly appreciated.

Table of Contents

Page

List of Figures

Page

List of Tables

Page

DEEP LEARNING-BASED, PASSIVE FAULT TOLE RANT CONTROL FACILITATED BY A TAXONOMY OF CYBER-ATTACK EFFECTS

1 Introduction

This chapter provides the context for the research described in this dissertation. Events that exemplify the challenges faced by cyber-physical systems are described and the specific research questions to be explored are presented. The methodology and the scope of the proposed solution are introduced.

1.1. Motivation

In 2014, a German steel mill was attacked by hackers who accessed the operational control system via the company's IT network [1]. The hackers were able to put the mill into an "undefined state" [2] from which a blast furnace could not be shut down properly, resulting in extensive, though unspecified physical damage to furnace systems at the mill. This confirmed case of a cyber-attack causing physical destruction of equipment followed the Stuxnet attack that damaged Iranian centrifuges at a uranium enrichment plant [2]. That attack was discovered in 2010; since then, experts have warned that it would only be a matter of time before other more destructive attacks would occur.

In Maroochy, Australia, a former employee of the shire's water services used radio signals to remotely switch valves in the water treatment plant [3]. He caused extensive environmental damage by routing 800,000 liters of untreated sewage into nearby waterways and parks, contaminating fresh water sources and killing marine life.

More recently, the Ukrainian power grid was taken down in a cyber-attack that required "extensive reconnaissance" [4] and preparation to complete. This is, or should be,

of particular concern to the US Department of Defense (DOD). There are over 2,600 electrical, water, wastewater, and natural gas utility systems on DOD installations [5]. In 1997, the DOD decided that utility privatization was the preferred method for improving utility systems and services; Congress subsequently approved legislative authority for privatizing utility systems at military installations. Since these privatization efforts are currently under way, a closer look at the technology controlling these systems is warranted.

1.2. General Issue

Cyber-physical systems (CPS) are systems of collaborating computational elements controlling physical entities such as: power and water, industrial systems, transportation systems, medical devices, security systems, building automation, emergency management, and many other systems vital to our well-being. Industrial Control Systems (ICS) are a subset of these cyber-physical systems that comprise a diverse group of sensor and control systems used to monitor and control critical infrastructure. These include Supervisory Control and Data Acquisition (SCADA) systems, Distributed Control Systems (DCS) and Programmable Logic Controllers (PLCs).

When any CPS malfunctions, or fails, the operation of the corresponding system(s) in the real world can endanger physical safety or even cause loss of life, as well as property damage or harm to the environment [6]. The priorities for traditional IT systems are usually focused, in order, on confidentiality, integrity, and availability (CIA) of the data. These priorities are generally reversed for ICS. Keeping the system up and running (data availability) is the top priority for ICS. Data integrity and confidentiality usually fall in behind availability.

As applied to ICS, industry recommended solutions include security policies, procedures, tools and products to prevent unauthorized access of the system [7] [8]. These solutions, when used together are commonly referred to as a "Defense in Depth" [9] approach. The elements of Defense in Depth are all useful and should be employed. Unfortunately, focusing too much on "preventing access" results in a cyber "Maginot Line" to protect ICS.

France's WWII defensive line did prevent the German army from simply rolling straight across the border. Instead, the Germans got around it by way of Belgium and easily conquered France. Likewise, Defense in Depth makes a head-on attack of an ICS much more difficult, but ICS are ill equipped to deal with malicious actors if, and when, they do gain access to the system.

There has been (and continues to be) interest in the control theory community in developing controllers that can operate in the presence of a component failure, i.e. fault tolerant controls (FTC). Practitioners in this community have made the logical step of applying the benefits of FTC in developing controllers that can respond to a cyber-attack on any physical component of a system.

Unlike active (reactive) controllers which depend on attack or fault detection to trigger their activity, passive (proactive) controllers are designed to properly control the system whether faults and/or attacks are present or not. Proactive controls are disregarded by many researchers because of the belief that proactive controllers are only effective if detailed information about the fault or attacks the system may face is known *a priori* [10] [11].

1.3. Research Questions

This research will address the following questions relating to the use of fault tolerant controls to improve the resilience of cyber physical systems.

Question 1. What are the physical layer effects in Cyber Physical System that could result from a component fault or cyber-attack?

Question 2. How could a passive Fault tolerant Controller use deep learning to maintain operations, and how could it be realized?

Question 3. Will the approach work for both linear and nonlinear systems?

Question 4. What benefit(s) does passive FTC with deep learning provide compared to an active FTC?

1.4. Methodology

To answer the research questions stated above, this dissertation proposes an ICS control design that is broadly applicable to multiple input – multiple output (MIMO) and multiple input – single output (MISO) systems.

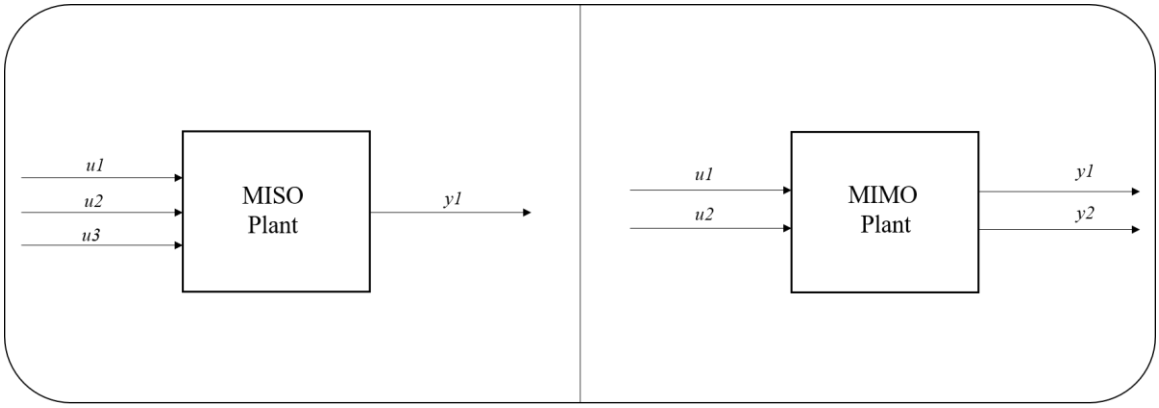

Figure 1.1. Notional MISO and MIMO Systems Diagram.

For systems portrayed in Figure 1.1, the control signals going to the actuators in the system (plant) are labeled (u_i) and the output signals from the system sensors are labelled (y_i). Both systems have more than one actuator to be controlled, thus categorizing them as multiple input systems.

In an effort to represent system faults and cyber-attacks, their possible effects are explored at the fundamental physical component (sensor or actuator) layer. By considering the possible behaviors of a sensor or actuator when being affected by faults or cyber-attacks, we develop and present a taxonomy of effects. This taxonomy is useful in the design and testing of any FTC, but is foundational to the design, training and testing of a proposed controller design.

The proposed controller holistically considers the entire state of the system under control and uses deep-learning networks (either stand-alone or in an ensemble) to determine the next appropriate commands for each actuator in the system under control. It does so without relying on a fault or attack detection mechanism. The proposed design is tested in simulations, on both linear and nonlinear systems, with multiple combinations of fault/attack effects on the sensors and actuators introduced to the system under test. The proposed controller performance is compared to a published active FTC controller design and the test results are analyzed.

Finally, the proposed controller and comparison controller are tested on a system that is experiencing Stuxnet-like attacks. Those test results are also discussed.

1.5 Assumptions/Limitations

The application of this research effort will only be demonstrated on MIMO and MISO systems. While this controller design could be applied to systems with a single input (single actuator), these are of limited interest, because the direct and/or indirect interaction within a system of multiple inputs make the control problem much more challenging and interesting.

Any solution that would require altering the system to be controlled (e.g., adding redundant components or limiting the system's intended operational capabilities) is not considered in this work. The solution controller needs to work with the system as designed.

As described in Chapter 3, there are some fault and attack effects that can render an actuator in the cyber physical system "uncontrollable". Since there is no fault tolerant or adaptive controller that can control an uncontrollable component, such fault and cyber-attack effects will not be addressed in this work unless the system under control is designed with redundant actuators.

1.6. Organization

The remainder of this dissertation is organized as follows: Chapter 2 provides a review of the pertinent literature in the areas of fault tolerant control, reactive control, system modeling and faults and attacks on control systems. Chapter 3 shows the development of the taxonomy of effects and lays out the proposed controller design in detail. Chapter 4 describes a proof-of-concept, linear test subject and specifies how the proposed controller is applied to that test subject system. The method and results of testing the proposed controller design on the linear system are also discussed. Chapter 5 describes

a benchmark nonlinear test subject system and specifies how the proposed controller is applied to that test subject system. The design and training of the deep learning-based controller are presented in detail. The method and results of testing the proposed controller design on the nonlinear system are also discussed. In Chapter 6, a published comparison controller [12] is reviewed and the method and results of testing the comparison controller design on the same nonlinear system are presented and analyzed. This chapter also compares and addresses the results of testing the proposed and comparison controllers on the nonlinear test system. In addition, it also provides a discussion on the results of testing the comparison controller and the proposed controller(s) against Stuxnet-like attacks on the non-linear test system. Concluding remarks and comments are found in Chapter 7.

2 Literature Review

2.1 Chapter Overview

This chapter provides an overview of the relevant existing literature for the proposed research problem. First, is a review of work done in traditional fault tolerant controls, followed by data driven fault tolerant controls. A review of the fault tolerant controls using artificial intelligence is then presented and the section is rounded out with a review of the application of deep-learning methods applied to active fault tolerant controls. The final section reviews how faults and attacks are represented in the various publications.

2.2 Relevant Research

The literature reviewed in this chapter comes from two related areas of controls research: Fault Tolerant Controls (FTC) and controls designed to respond to attacks. The former focuses on designing controllers that can handle one or more faults/malfunctions in the system under control. The latter focuses on designing controllers that can respond to one or more deliberate attacks on the system under control. Both areas offer research worthy of discussion.

The literature can be divided along a few different lines; active vs. passive FTC, model-based vs. data-driven, or those designed to work with linear vs. nonlinear systems. Active vs. passive FTC is a sorting used in much of the literature, and this section will begin there since the controller proposed in this work is passive and the controller against which it will be compared is active.

2.2.1 Fault Tolerant Controls

Before beginning a review of the pertinent literature, it is useful to define the terms that are commonly used in the field of fault tolerant controls. The following definitions come from [13].

• Fault-tolerance: the ability of a controlled system to maintain control objectives, despite the occurrence of a fault. A degradation of control performance may be accepted. Fault-tolerance can be obtained through fault accommodation or through system and/or controller reconfiguration.

• Fault-accommodation: change in controller parameters or structure to avoid the consequences of a fault. The input-output between controller and plant is unchanged. The original control objective is achieved although performance may degrade.

• Reconfiguration: change in input-output between the controller and plant through change of controller structure and parameters. The original control objective is achieved although performance may degrade.

• Supervision: the ability to monitor whether control objectives are met. If not, calculate a revised control objective and a new control structure and parameters that make a faulty closed loop system meet the new modified objective. Supervision should take effect if faults occur and it is not possible to meet the original control objective within the fault tolerant scheme.

Figure 2.1 provides a taxonomy of FTC approaches in each category. The subjects highlighted in bold face are discussed in this section. The others are listed for context, and references to research in those areas are provided in Appendix A.

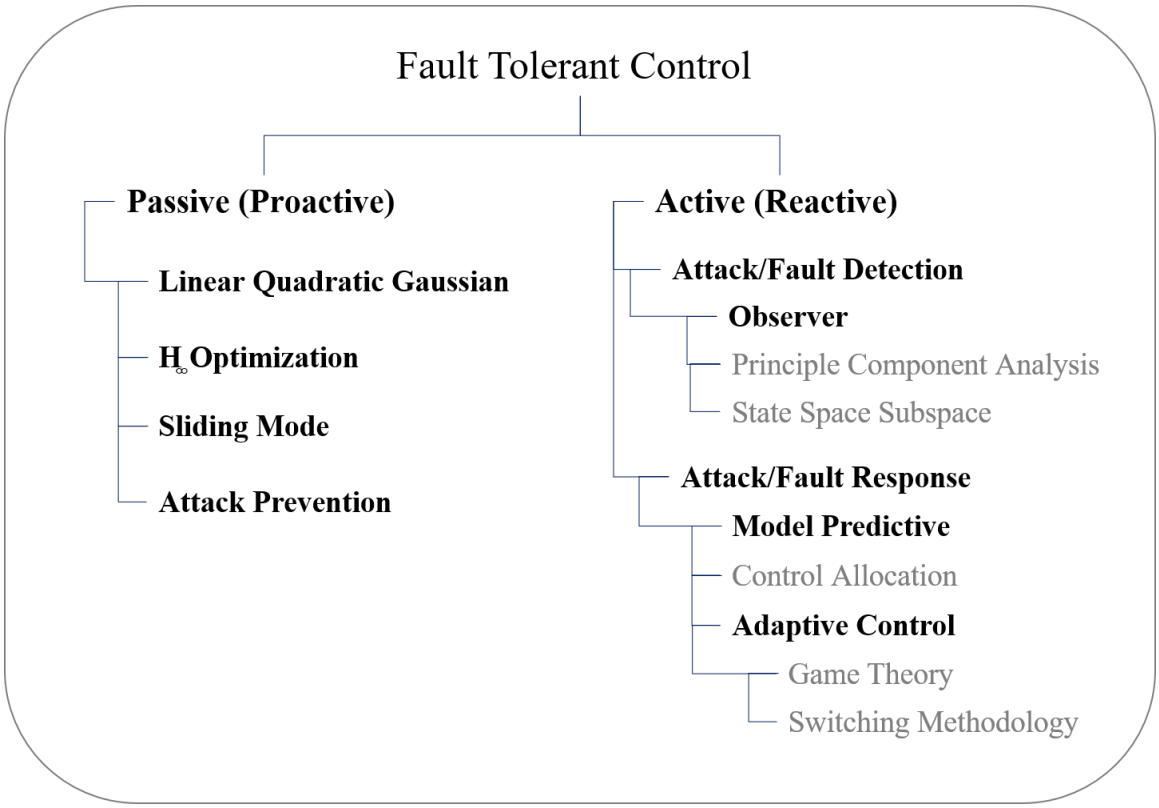

Figure 2.1. Taxonomy of Fault Tolerant Control Methods.

Current efforts on FTC design can be classified as either passive (proactive), or active. The passive approach focuses on making the system insensitive to faults by designing a more robust controller. Passive Fault Tolerant Controllers (PFTC) are attractive for their comparative design simplicity. The advantage of the PFTC approach can be explained as follows. When a fault occurs, it takes some time for the Fault Detection and Diagnosis (FDD) module of an active FTC to detect the fault and then isolate and identify the fault. There may also be some delay due to the controller reconfiguration. During this period, the system is working with the nominal controller. Performance of the system in this period is mainly dependent on the severity of the fault and the robustness of the nominal controller. It is clear that the controlled system may become unstable during

this period [14]. For safety-critical systems, such as aircraft flight controls or nuclear power plants, the time window may be too short to perform accurate fault isolation and estimation. In such cases, a PFTC system is preferable because it does not need an FDD scheme [15].

2.2.1.1 Passive Fault Tolerant Control

Passive controllers can employ robust control methods such as Linear Quadratic Gaussian (LQG) control like that used in [16] and [17]. The problem concerns linear systems driven by additive white Gaussian noise. The challenge is to determine an output feedback law that is optimal in the sense of minimizing the expected value of a quadratic cost criterion. The LQG controller is simply a combination of a Kalman filter (linear quadratic state estimator (LQE)) as part of a linear quadratic regulator (LQR) as shown in Figure 2.2.

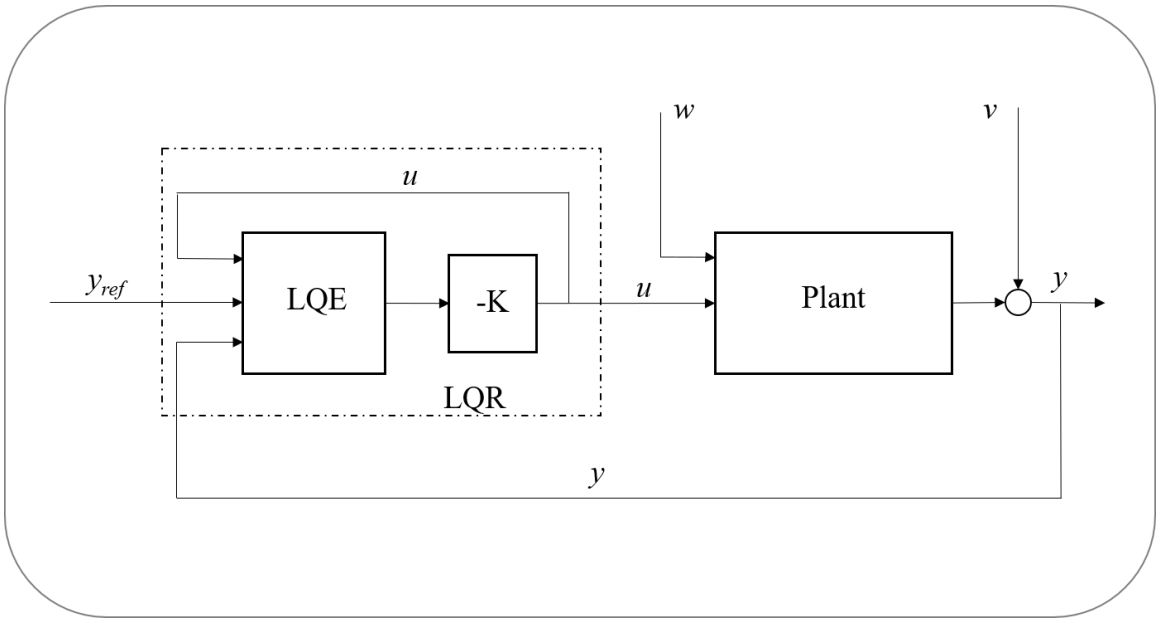

Figure 2.2. Linear Quadratic Gaussian Controller and System.

The output measurements are assumed to be corrupted by Gaussian noise (v) and the initial state (w) is assumed to be a random Gaussian vector. The LQG controller

provides reliable linear quadratic state feedback control such that it can tolerate actuator outages. The work in [18] combines LQC and H_∞ optimization to control flight tracking in the Advanced Tactical Fighter when actuator faults or flight surface impairments occur.

H_∞ optimization methods are used to synthesize controllers to achieve stabilization with guaranteed performance [19] [20] [21] [22]. To use H_∞ methods, the control problem is expressed as a mathematical optimization problem, and the challenge is to find the controller that solves this optimization. H_∞ techniques have the advantage over classical control techniques in that they are readily applicable to problems involving multivariate systems with cross-coupling between channels. The disadvantages of H_∞ techniques include the level of mathematical understanding needed to apply them successfully and the need for a reasonably accurate model of the system to be controlled. The general form of an H_∞ controller with plant is shown in Figure 2.3.

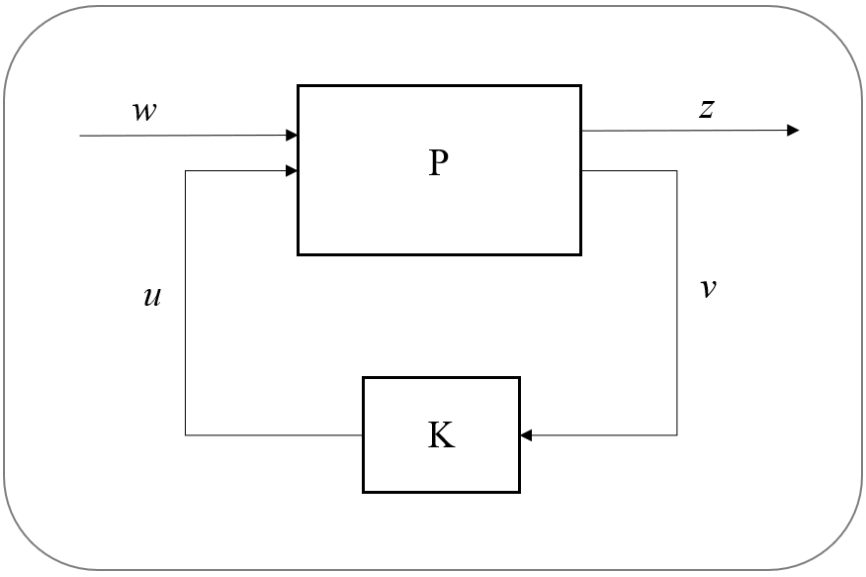

Figure 2.3. General Form of H-infinity Control.

The plant (P) takes as input a reference signal and disturbances (w) as well as the manipulated signal (u). The outputs from the plant are an error signal (z) and the measured

variables (v) used to control the system. The controller (K) uses v to manipulate u. The signals w, u, z and v are usually vectors while P and K are matrices. The mathematical formula for the system is

$$\begin{bmatrix} z \\ v \end{bmatrix} = P(s) \begin{bmatrix} w \\ u \end{bmatrix} = \begin{bmatrix} P_{11}(s) & P_{12}(s) \\ P_{21}(s) & P_{22}(s) \end{bmatrix} \begin{bmatrix} w \\ u \end{bmatrix}$$

where

$$u=\mathrm{K}(s)v$$

The dependency of z on w is expressed as:

$$z = F_\ell(\mathbf{P}, \mathbf{K})w$$

F_ℓ is called the *lower* linear fractional transformation and is defined as

$$F_\ell(\mathbf{P}, \mathbf{K}) = P_{11} + P_{12}\mathbf{K}(I - P_{22}\mathbf{K})^{-1}P_{21}$$

Thus, the objective of H_∞ control is to find $\mathbf{K}$ such that $F_\ell(\mathbf{P}, \mathbf{K})$ is minimized according to the H_∞ norm.

It is important to keep in mind that the resulting controller is only optimal with respect to the prescribed cost function and does not necessarily represent the best controller in terms of the usual performance measures used to evaluate controllers such as settling time, energy expended, etc. As an example, the method used in [18], guarantees the H_∞ performance of the normal systems, as well as the faulty system to be less than a predetermined bound.

In the cases of [23] and [24] faults are modeled as uncertainties and a robust control is designed such that it can tolerate uncertainties. Robust control of uncertain piecewise linear systems using state feedback and continuous piecewise Lyapunov functions helps achieve the desired H_∞ performance.

Another approach is to use simple redundancy as described in [25]. In this example, the presence of multiple redundant (identical) controllers to ensure H_∞ performance is achieved even in the presence of sensor and/or actuator faults.

Sliding Mode control (SMC) is a nonlinear control method that alters the dynamics of a nonlinear system by application of a discontinuous control signal that forces the system to "slide" along a cross-section of the system's normal behavior as depicted in Figure 2.4.

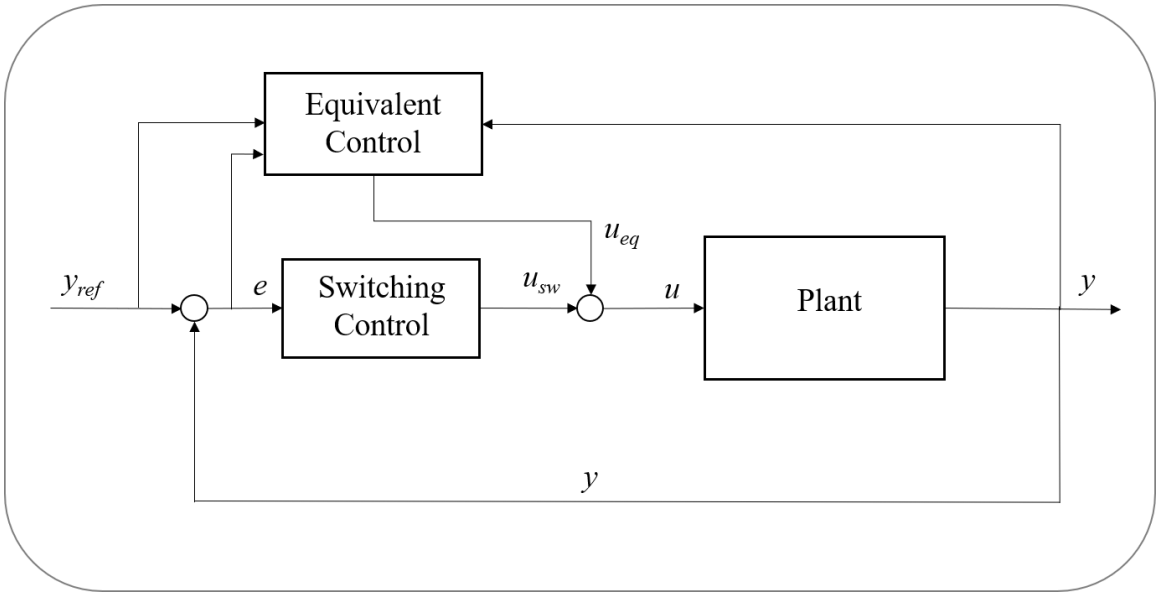

Figure 2.4. Sliding Mode Control and System.

The example in Figure 2.4 depicts a generic sliding mode control circuit. The discontinuous signal that forces the system to slide is u_{sw} and the signal being acted upon is $u_{eq.}$

The feedback control law is not a continuous function of time. Instead, it can switch from one continuous structure to another based on the current position in the state space. Thus, sliding mode control is a variable structure control method. The multiple control structures are designed so that trajectories always move toward an adjacent region with a

different control structure, and so the ultimate trajectory will not exist entirely within one control structure. Instead, it will *slide* along the boundaries of the control structures [26], [27] [28] as shown in Figure 2.5.

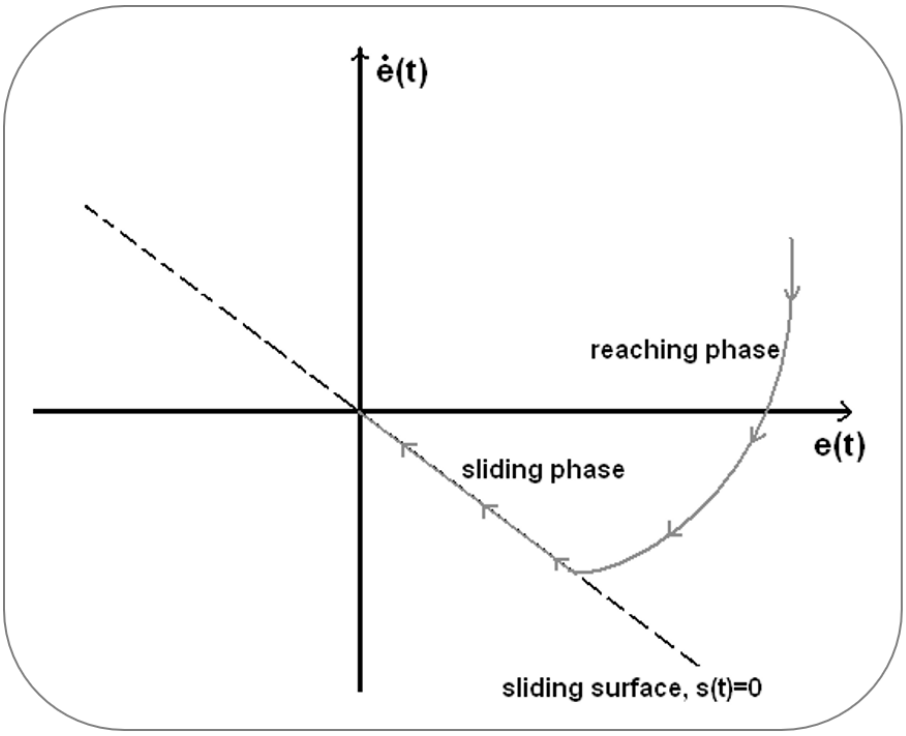

Figure 2.5. Phase Portrait of Sliding Mode Control [29].

One advantage of this method is that the dynamic behavior of the system may be directly tailored by the choice of switching function; essentially the switching function is a measure of desired performance [30]. The sliding mode approach is also used in [31] with application to aircraft control.

Attack preventive approaches (also called proactive) seek to make the controller more robust by identifying the system vulnerabilities and changing the physical structure, changing parameters of the system, or by designing new control methodologies. They are not dependent on an attack detection system. The primary criticism of preventive

approaches is that designers are required to have prior knowledge of the types of attacks [11]. Work in preventive action controllers has focused on modifying the system structure (e.g., adding redundancy), or by designing more robust controllers.

In [32] the authors use observability and controllability Gramian matrices as a means of measuring and minimizing network vulnerability. The vulnerabilities of a power system state estimator are explored in [33] where modifications of routing and data authentication are used to mitigate data integrity attacks. Authors in [34] find the optimal estimator that minimizes estimation error based on multiple sensor measures.

The work in [35] investigates the problems of robust passive fault-tolerant control for systems with time-delays and uncertain parameters or unmodeled dynamics. The authors design a state feedback controller, which ensures robust stability of the closed loop system for all "admissible uncertainties". The challenge is to design a controller that meets desired performance metrics and maintains stability despite the uncertainties in the system. The authors develop a linear matrix inequality (LMI) approach to solve these problems. This work considers only actuator faults and/or unknown parameters.

2.2.1.2 Active Fault Tolerant Control

In contrast to passive control, active FTC approaches focus on responding to an identified, or diagnosed fault. In order to respond to a fault, active controllers require a mechanism for detecting and isolating faults. Typically, an active FTC will have a Fault Detection and Identification (FDI) scheme and a controller reconfiguration mechanism paired with a reconfigurable controller [36]. The structure of an FTC system is depicted in Figure 2.6.

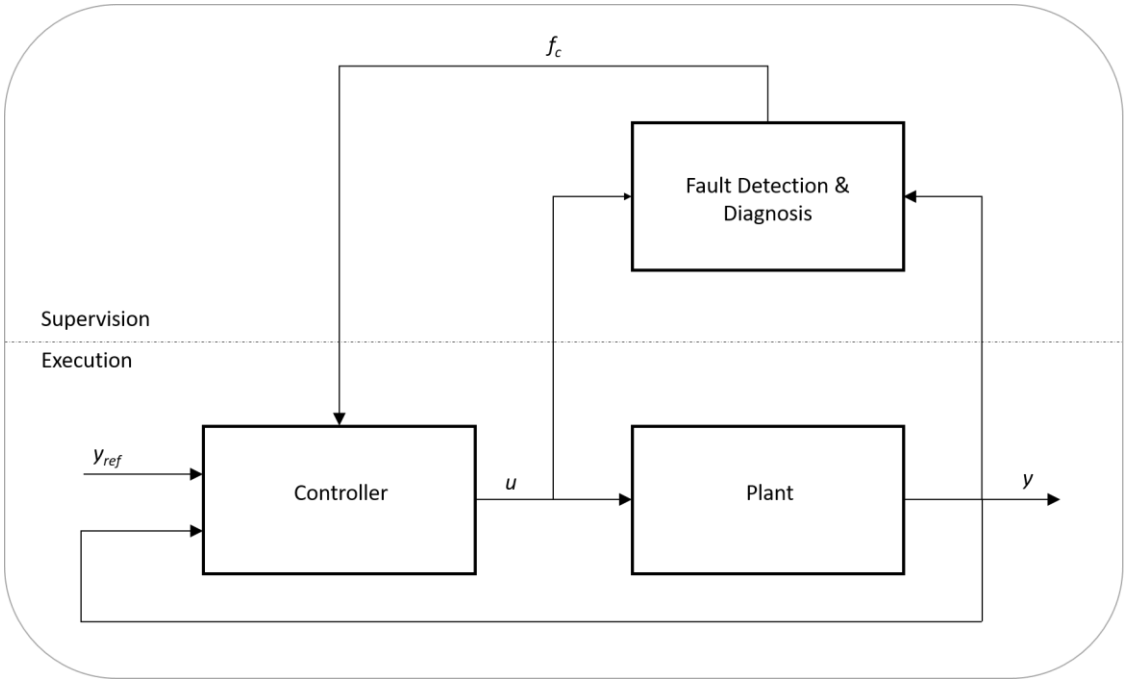

Figure 2.6. Traditional Control Loop with Fault Detection & Diagnosis Block.

In Figure 2.6 y_{ref} is the desired output of the system under control (i.e., condition the controller is trying to reach). The output of the controller is designated by *u* while the output of the system under control (the plant) is designated as *y*. Under normal operations, the controller compares y_{ref} to *y* and adjusts the next value of *u* to compensate for any error in the last value of *y*.

The fault diagnosis block receives the input *(u)* and output *(y)* from the system, and checks its consistency with the behavior of the system. If the I/O sequence is consistent with the normal behavior of the system, then the system is considered to be working in the normal condition by the FDD block and will continue working with the nominal controller. If the I/O sequence is not consistent with the nominal behavior of the system, then the FDD block detects occurrence of a fault. Specifically, the measured difference between normal

and abnormal behavior is tracked and accumulates as a residual. Once the residual crosses a predetermined threshold, the fault is “detected”.

Next, the FDD block tries to find out which fault has occurred by checking the consistency of the I/O sequence with the faulty behaviors of the system. The result is a fault candidate *fc*; the controller is then informed by the FDD block that fault *fc* has occurred. A new control scheme, should be designed online, or be selected among pre-designed controllers such that the faulty system can achieve the control objective.

If such a controller exists, then the system is considered to be fault-tolerant with respect to the fault *fc* and the control objective. But if the controller objective cannot be achieved, the system is not fault tolerant. In this case, a possible solution is to change the control objective, e.g. by allowing some degradation in system performance.

Set-membership based detection and isolation approaches have been proposed for the detection of specific faults. These methods generally operate by discarding models of a system or subsystem that are not compatible with observed data rather than selecting the most likely model.

There is a multiplicity of methods to perform active Fault Tolerant Control. Although each method has its own advantages (speed, accuracy, ease of implementation, etc.) and weaknesses (need for closed-form mathematical models, inability to detect multiple sensor faults, inability to distinguish between sensor and system faults, and the need to integrate different approaches together to accomplish different tasks such as modeling, fault detection, fault isolation, etc.). As shown in Figure 2.1, the approaches used can be divided into a few broad categories.

2.2.1.2.1 Observer Based Controllers

An observer is a system that provides an estimate of the internal state of a given real system, from measurements of the input and output of the real system. A typical configuration of a system with an internal observer is shown in Figure 2.7.

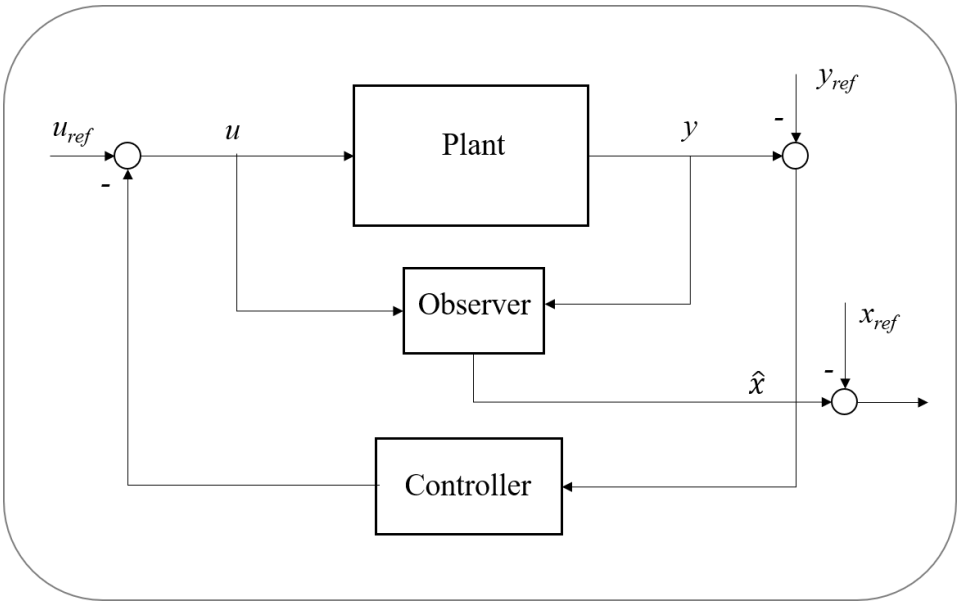

Figure 2.7. Control System with Internal Observer.

Observers are a popular tool employed in the active FTC area for fault detection. The work in [37] introduces an observer-based approach to estimate the actuator fault in a system and design an FTC strategy that compensates for the effects of the fault. The authors use a quadratic boundedness approach to design the observer so that the state and fault estimation errors converge to the origin.

In [38] the authors estimate simultaneous sensor and actuator faults using an adaptive observer based on the quadratic Lyapunov approach. The solution is described as an optimization problem formulated in terms of linear matrix inequalities. The proposed

controller is applied to a VTOL (vertical take-off and landing) aircraft system. The work in [39] also uses an adaptive observer but it is applied to aircraft engines in this case.

2.2.1.2.2 Model Predictive Control

Model Predictive Control (MPC) relies explicit model of the system to predict a future output chain ($\hat{y}$). The basic structure of MPC is shown in Figure 2.8.

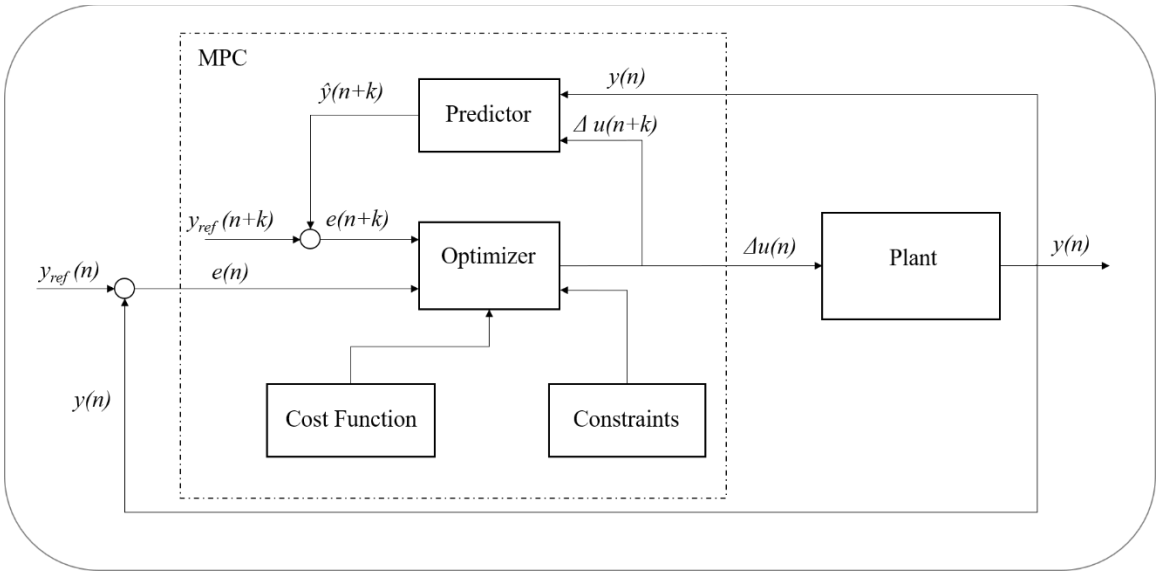

Figure 2.8. Basic Structure of Model Predictive Control.

Based on predicted system output ($\hat{y}$) and current system output (y), the error (e) is calculated. The error is fed into the optimizer which calculates the future optimal control sequence (Δu) using the cost function and constraints. The sampling instant is represented by k.

As an example, the work in [40] develops multiple models off-line then uses multiple MPC on-line to switch between models as faults are detected. The approach is demonstrated on a simulated DC servo from the MATLAB® controls toolbox.

The work in [41] takes on a more challenging effort in using a discrete controller with an analog continuous system (these combinations are known as hybrid systems).

Intended to control an aircraft conducting air-to-air refueling, this work uses Hybrid Bond Graphs to capture the interactions in the physical system and temporal causal graphs to derive state-space equations for fault isolation and to estimate fault magnitudes.

2.2.1.2.3 Adaptive Control

Adaptive control changes the control law with parameters that may be uncertain or at least variable over time. A generic adaptive control diagram is shown in Figure 2.9.

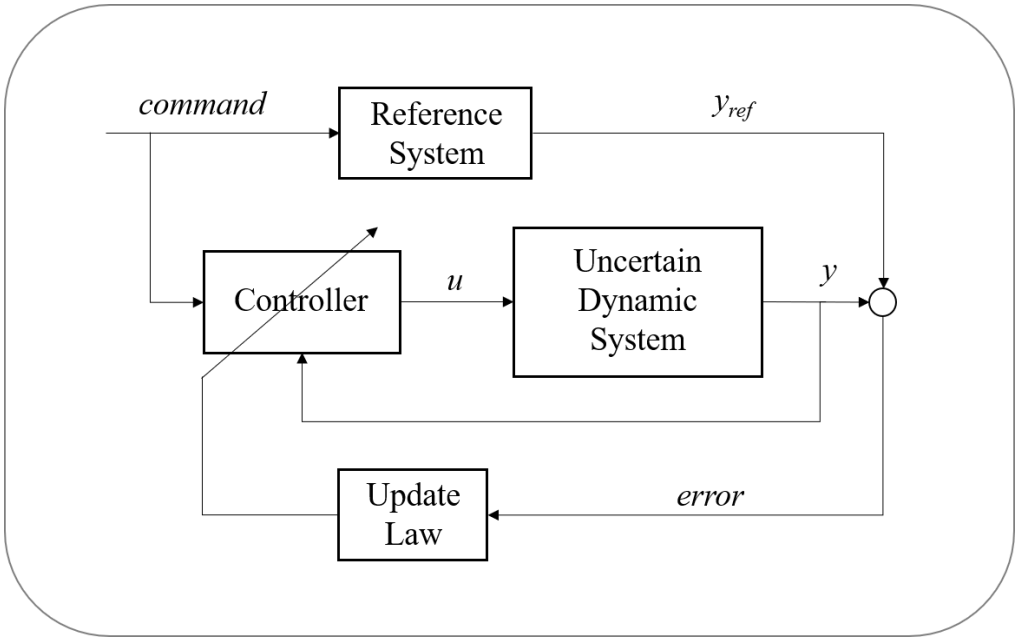

Figure 2.9 Generic Adaptive Control Diagram.

Unlike robust control, adaptive control requires no a priori knowledge of the bounds of the parameters or variables in question. Adaptive controls can be direct or indirect. Indirect adaptive control estimates the parameters of the controller using a model of the process. Direct adaptive control uses the I/O data to calculate the controller parameters.

Reactive responses only take effect once the attack has been detected. They then reconfigure the control action to mitigate the impact of the attack on the system.

Many reactive controllers require switching between models or modes to maintain control. Upon detection of a fault, the switching methodologies in [42] and [43] generate a model or vector for the controller to compensate for the detected abnormality.

Model-based responses use models of the system under normal conditions (i.e., not under attack) and compares them to the values received from the system under control to determine the best response to keep the system in a desirable state.

There are two general types of model-based adaptive control:

Model reference adaptive controllers (MRACs) which incorporate a *reference model* defining desired closed loop performance [44] as shown in Figure 2.10.

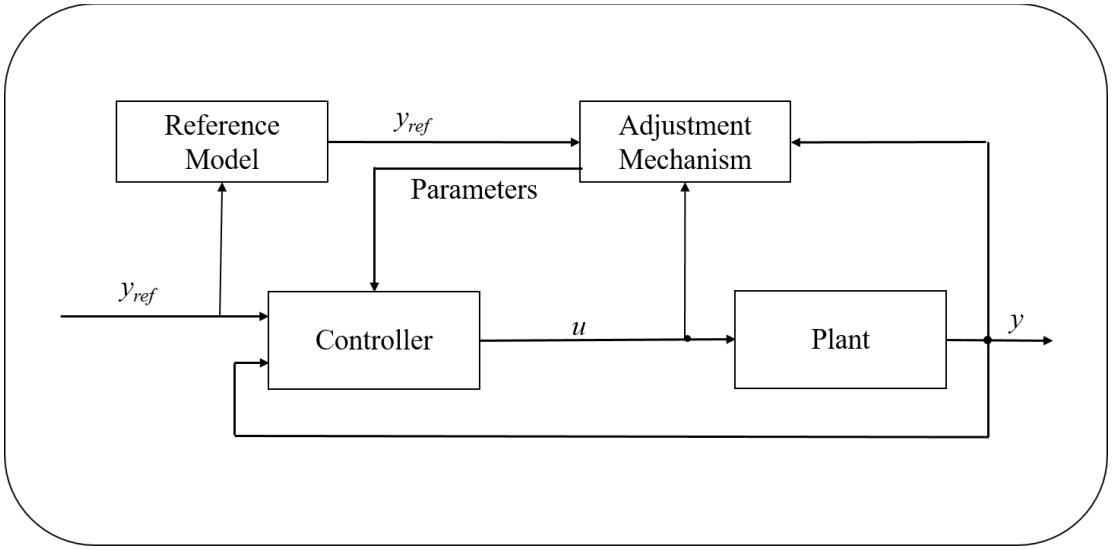

Figure 2.10. Model Reference Adaptive Control Diagram.

Examples of this approach are found in [45] and [46] in which the authors present a method for handling “locked in place” actuator faults by using a model of the system with known actuator faults as a reference to design a control structure to dealing with unknown actuator faults.

The other method is Model identification adaptive controllers (MIACs). This approach builds a mathematical model of the system while the system is running as shown in Figure 2.11.

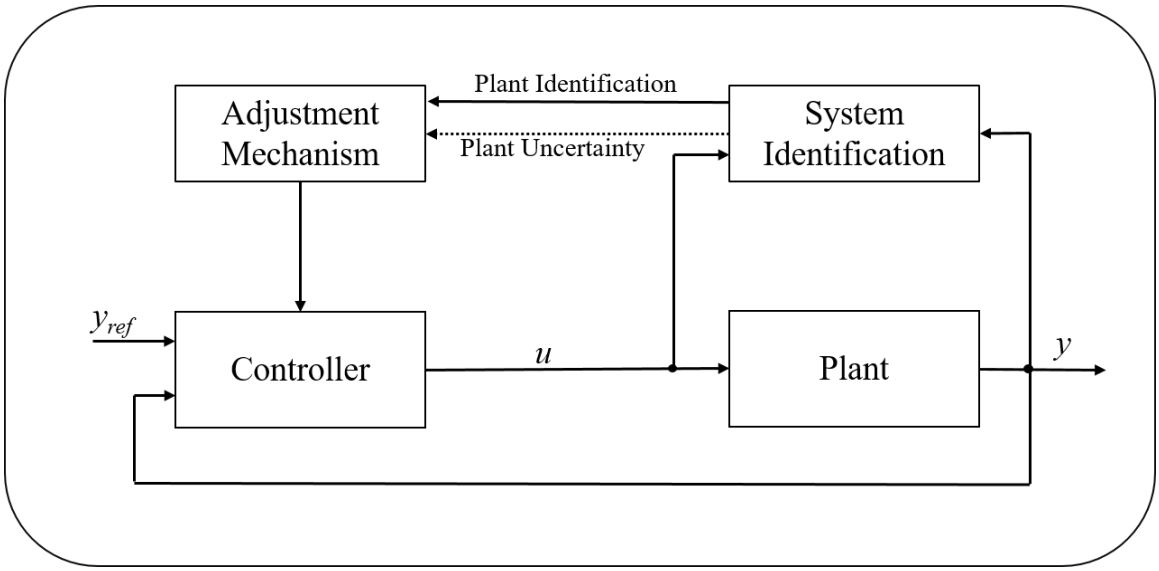

Figure 2.11. Model Identification Adaptive Control Diagram.

An example of this approach is found in [47]. Here the authors use online model and control derivative identification to control the X-33 reusable launch vehicle in ascent mode.

For both approaches, different types of models are used such as finite state machines [48] or linear models that can be used to represent nonlinear systems [49]. Others use redundant back-up controllers [50] or models of the process controller itself [51] [52] [53]. For further related work in the field of active FTC, the reader is referred to references [A1-A82] in Appendix A.

In summary, the examples of published work in the area fault tolerant controls can be grouped as either active or passive as shown in Table 2.1.

Table 2.1. Passive and Active Fault Tolerant Control References.

Passive FTC	Active FTC
[16] [17] [18] [19] [20] [21] [22] [23] [24] [25] [26] [27] [28] [31] [32] [33] [34] [35]	[37] [38] [39] [40] [41] [42] [43] [44] [45] [46] [47] [48] [49] [50] [51] [52] [53] A: [A1-A82]

2.2.2 Data-Driven Control vs. Model-Based Control

Up to this point, the control literature reviewed has been model-based. There is another separation in control approaches that cuts across both active and passive FTC; that is Model-Based Control (MBC) vs. Data Driven Control (DDC) [54]. Since at least one application of the proposed controller in this dissertation uses a data-driven design, it is appropriate to review the fundamentals of DDC.

Data-driven controls are the control theories and methods in which the controller is designed directly using on-line or off-line I/O data of the controlled system or knowledge from the data processing without using explicit or implicit information of the mathematical model of the controlled process, and whose stability, convergence, and robustness can be guaranteed by rigorous mathematical analysis under certain reasonable assumptions [55].

The reason to make the division between MBC and DDC is that MBC theory can only solve control problems when reliable mathematical models are available and the uncertainties are constrained within a known moderate bound. DDC methods are the better choice for control problems in which the mathematical model is difficult to establish or

unavailable or if the mathematical model is complicated with too high of an order or too much nonlinearity [54].

Data direct controls generally fall into one of two structure types. The first is those with a fixed controller structure. This kind of DDC method involves controller design that depends only on plant I/O measurements with a pre-specified fixed controller structure. Controller parameters are obtained from some optimization procedures, such as batched or recursive algorithms. Here the controller design problem is transformed into the controller parameter identification with the help of the assumption that the controller structure is known prior and linear in controller parameters. No information regarding the plant model or dynamics is involved. The main issue in this kind of methods is how to determine controller structure for a given controlled plant.

Obtaining a good controller structure with unknown parameters, especially for general nonlinear systems, is quite difficultAnother issue with this kind of DDC method is the lack of the stability and analysis methodologies.

The second (and more interesting) type of DDC is made up of those systems with an unknown controller structure. These can further be divided into 2 subsets; Apparent DDC methods and Model-free DDC methods.

Apparent DDC methods. In this DDC method the controller design is dependent only on measured plant I/O data and the plant model structure and the dynamics are implicitly involved in controller design. The controller design and methods of theoretical analysis are similar to those of MBC designs. However, they are also more robust when they are used in practice. The direct adaptive control and predictive control methods (described previously) are typical of this [56].

Model-free DDC methods. This kind of DDC method implies that the controller is designed directly simply using the measured plant I/O data, without explicitly or implicitly using model information. This is the ideal type of DDC method. The outstanding feature of this kind of DDC methods is that it has a systematic controller design framework and systematic means of analyzing stability. The main difference between this kind of DDC method and others is that the effectiveness or rationality of the controller structure or controller designing is theoretically guaranteed using rigorous mathematics. This strategy can deal with the system control problems using a uniform way both for linear and nonlinear systems.

2.2.2.1 Passive DDC

A passive, model-free FTC method is presented in [57]. In this work several candidate controllers are designed without any knowledge of the plant model or fault model(s). Using an approach called unfalsified Control (UC) the controllers that do not achieve the required performance set by the designer are discarded (falsified) and switched out of the loop. That controller is then replaced by an unfalsified controller. This process continues until a controller is placed in the loop that can maintain proper performance. No fault detection or fault identification steps are involved. The decision to switch controllers is driven by poor performance of the system, not in response to a fault or attack. This controller in [57] was demonstrated on a MIMO system.

2.2.2.2 Active DDC

The work in [58] used active model predictive control (described previously) combined with a data-driven approach to determining the post-fault actions for system

recovery. The authors applied the control approach to a 4-motor drone experiencing partial, single and multiple faults.

In [59] the authors present a data driven compensation controller to make a traditional PID controller more fault tolerant. This is an active FTC system that is demonstrated on a DC motor controller experiencing a single fault.

In [60] an integrated data-driven adaptive FTC system design scheme is proposed for MIMO systems with unknown plant model dynamics. The nominal feedback controller is first designed with a data-driven observer-based residual generator, which produces a residual signal when a fault occurs. Then, the data-driven adaptive FTC compensator is developed using only the online system data. This Model Free Adaptive Control (MFAC) system is compared to an iterative feedback controller (IFC) on a stirred, heated tank system.

In an unusual application of active fault detection [61] proposes an unsupervised model-free approach to detect insulin pump malfunctioning in an artificial pancreas, relying on data-driven techniques for anomaly detection. It is model-free, thus avoiding the complex subtask of identifying a model of patient physiology and it is also unsupervised, thus not requiring labeled data for training, which is difficult to obtain in actual practice. The authors extract a feature set capable of accounting for the dynamics in type-1 diabetic physiology and designed to highlight anomalous behaviors associated with pump faults. They then applied established anomaly detection methods to the data set.

The work in [62] reviews a residual generator-based (active) data-driven fault-tolerant control framework. The two data driven methods for deriving the optimal compensator control design are based on gradient decent or machine learning. By analyzing

the differences between the cost function of the two methods, it is confirmed that the solution obtained by the gradient descent method is locally optimal while the reinforcement learning method's solution is globally optimal. Both methods are demonstrated on a DC motor benchmark case study. These systems are nonlinear. The fault used in this work is a single step disturbance. No discussion about multiple faults.

For more examples of published work in the area of active Data-Driven FTC, the reader is referred to references [A84-A90] in Appendix A.

In review, the published work in the area of Data-Driven Control can be grouped in to active and passive DDC as shown in Table 2.2.

Table 2.2. Active and Passive Data Driven Control References.

Passive	**Active**	
[16] [17] [18] [19] [20] [21] [22] [23] [24] [25] [26] [27] [28] [31] [32] [33] [34] [35]	[37] [38] [39] [40] [41] [42] [43] [44] [45] [46] [47] [48] [49] [50] [51] [52] [53] A: [A1-A82]	**FTC**
[57]	[58] [59] [60] [61] [62] A: [A84-A90]	**DDC**

This last example brings up another area of research applicable to fault tolerant controls. That is the application of artificial intelligence methods in FTC.

2.2.3 Artificial Intelligence in FTC

The use of artificial intelligence techniques in fault tolerant control has been researched since 1988. Part of the reason for the interest is AI methods can be readily

applied to nonlinear systems which can be a challenge with crisp mathematical methods. Like conventional FTC methods, AI related FTC can be grouped as either passive or active.

2.2.3.1 Passive FTC using Artificial Intelligence

The first published work on using AI with passive FTC or adaptive controls was the application of Fuzzy Control theory to the flight controls of an F-16 experiencing actuator faults [63] also found in [64]. In this work, the authors use fuzzy model learning to teach the controller normal aircraft responses to various pilot inputs. With the "normal" model in memory, the controller used fuzzy control principles to adapt the controller outputs when the aircraft behavior varied from the normal model. The approach is effective to a point. The authors found that trying to model all possible fault conditions resulted in degraded performance of the controller under normal conditions as well as being very computationally demanding.

Like all fuzzy controllers, this approach is well suited to non-linear systems. While this work could theoretically handle more than one simultaneous actuator faults, the possibility of co-occurring sensor faults is not addressed in this work.

In 2001, the authors of [65] published work on a fuzzy logic-based damage-mitigation controller. The system uses a dual level control scheme. In addition to the normal (lower level) control functions the controller also employs a structural model of the system and a damage prediction model. Using data from the system during operation, the damage prediction model informs the fuzzy damage controller (higher level control) when conditions indicate a higher likelihood of damage to a critical component. The fuzzy damage controller then adjusts the control signals to the system to lessen the likelihood of

damage to a component This early work added to the body of research, but is not actually FTC.

In [66], which was published in 2011, the authors investigate the position and velocity tracking control problem with high-speed trains with multiple vehicles connected through couplers. A dynamic model reflecting nonlinear and elastic impacts between adjacent vehicles as well as traction/braking nonlinearities and actuation faults is derived. They developed neuro-adaptive fault-tolerant control algorithms to account for various factors such as input nonlinearities, actuator failures, and uncertain impacts of in-train forces in the system simultaneously. The resulting control scheme is essentially independent of any system model and is primarily data-driven because with the appropriate input–output data, the proposed control algorithms are capable of automatically generating the intermediate control parameters, neuro-weights, and the compensation signals, literally producing the traction/braking force based upon input and response data only. The process does not require precise information on system models or system parameters.

Finally, in [67] the authors propose a passive fault-tolerant tracking controller for use with a class of uncertain, non-linear systems with intermittent faults and time delays. The proposed controller uses a fuzzy control system combined with a PID controller on a two-tank system experiencing intermittent actuator faults.

2.2.3.2 Active FTC Using Artificial Intelligence

Research in using artificial intelligence in active FTC was first published in 1988 when the authors of [68] proposed adding a learning portion to a rule-based FTC system so that new rules for controlling the system can be generated to respond to an unanticipated fault in the system. The controller used redundant software and hardware to ensure accurate

data (at varying levels of fidelity) was available when required. The rules and normal operating data were stored in a lattice structure. Rules for all anticipated faults were included in the original controller design. When an unanticipated fault was detected, the controller would search by depth and by breadth across the lattice for the existing data and rules that best match the conditions of the unanticipated fault and generate control actions.

The authors of [69] use a multi-layer perceptron network as the process model and adapt it on-line using extended Kalman filters to learn changes in process dynamics. In this way, the adaptive model learns the post-fault dynamics caused by actuator or component faults. Then, the inversion of the neural model is used as a controller to maintain the system stability and control performance after fault occurrence. The convergence of the model inversion control is verified using the Lyapunov method.

In [70] the authors provide an overview of applying a neural network to the control of a nonlinear system for the purpose of detecting, identifying and accommodating a fault in the system. The fault(s) are unspecified but categorized as either “incipient” or “abrupt” and are, therefore, only described their time profile matrix. A learning algorithm is used to approximate the function of the fault thus acting as detector and identifier all in one. No mention is made of detecting more than one fault at a time.

The authors of [71] integrated model-based adaptive control and reconfiguration based on fault detection and diagnosis. The adaptive controller and the fault detection scheme are based on a fuzzy model process. In this case it is applied to a heat exchanger which is a non-linear system. The topic of addressing multiple simultaneous faults is not addressed.

The authors of [72] show that a dynamic fuzzy logic controller (DFLC) could be shown to be equivalent to a sliding mode controller, but without the chattering they experience during disturbances. Sliding mode controllers have a “dead zone” in which the control equations converge to zero for second order (and lower) systems. This work applies the DFLC to higher order nonlinear systems.

Similarly, in [73] the authors apply fuzzy logic techniques to affect the gain on a sliding mode controller to design a discontinuous controller for nonlinear multivariable systems to eliminate the chattering they experience during disturbances.

In [74] a genetic algorithm is used to design a fault tolerant controller for the active magnetic bearings used by the Rolls Royce turbomachine’s rotor. This is a nonlinear system. The controller has a fault compensator which responds by shifting the workload to neighboring coils when the controller detects that one has failed (which is the anticipated fault). The authors do discuss the application of this approach to multiple coil failures but do not implement in under those conditions.

A genetic algorithm is also used in [75] to train a multi-layer neural network. The controller relies on redundancy to ensure there is a fault-free data path and the neural network selects which data path to use when a fault is detected by the active detection circuit. The genetic algorithm retrains the network weights. The controller is implemented on a Field Programmable Gate Array (FPGA).

The authors of [76] use a genetic algorithm to redesign the sliding mode controller after a fault is detected. This approach preserves much of the dynamics of the system unaffected by the failure.

For more published work in the area of active FTC using artificial intelligence techniques, the reader is referred to references [A91-A117] in Appendix A.

In summary, the examples of published work in Artificial Intelligence in fault tolerant and adaptive controls can be grouped as either active or passive as shown in Table 2.3.

Table 2.3. Passive and Active References on Artificial Intelligence in FTC.

Passive	**Active**	
[16] [17] [18] [19] [20] [21] [22] [23] [24] [25] [26] [27] [28] [31] [32] [33] [34] [35]	[37] [38] [39] [40] [41] [42] [43] [44] [45] [46] [47] [48] [49] [50] [51] [52] [53] A: [A1-A82]	FTC
[57]	[58] [59] [60] [61] A: [A84-A90]	DDC
	[62]	
[63] [64] [65] [66] [67]	[68] [69] [70] [71] [72] [73] [74] [75] [76] A: [A91-A117]	AI

2.2.4 Deep Learning in Fault Tolerant Controls

Deep learning is a comparatively recent development in Artificial Intelligence. While deep-learning networks are a type of Artificial Neural Network (ANN) they differ from the traditional neural networks referenced previously. Deep-learning networks are

capable of effectively working with much larger data sets than traditional neural networks. Additionally, deep-learning networks do not require any type of a priori feature extraction.

2.2.4.1 Active Fault Tolerant Control Using Deep Learning

There are a few examples of deep-learning methods being used in active FTC systems. The earliest is from 2016 in which the authors used a stacked sparse auto encoder to learn the deep architectures of fault data to minimize the loss of information in the event of a fault. Their proposed method was used to improve the divisibility between faults and normal process, and classify faults in the chemical benchmark, Tennessee Eastman Process (TEP) data [77].

In 2018 the authors of [78] used a hybrid recurrent/convolutional neural network model to estimate adaptation parameters for F-16 aircraft dynamics experiencing actuator/engine faults. The model is trained offline from a database of different failure scenarios. When a fault occurs, the model identifies adaptation parameters and feeds this information to an adaptive controller, which changes its configuration to better handle the fault. This controller works especially well in the case of significant actuator faults/failures.

Another active FTC application of deep learning is found in [79] published in 2019. Here the authors proposed a real-time fault diagnostic model for air-handling units (AHUs); the model used deep learning to improve the operational efficiency of AHUs. EnergyPlus® simulation software was used to establish different types of fault operation behavior data to serve as reference for the deep-learning network, thus improving the reliability of the diagnostic model. The proposed deep neural network features five hidden layers, each comprised of 200 neurons.

The authors of [80] propose using deep-learning tools, specifically auto-encoders, to actively identify faults in in the assemblies of automobile instrument clusters.

In [81] the authors propose a fault diagnosis method based on convolutional neural network (CNN) for aero-engine sensors. Convolutional layers and pooling layers in a CNN are used to extract correlation features among sensed signals, based on which fully connected layers are used to actively diagnose sensor faults. The inception module method is used to extract features on different data sizes among sensors, making the method capable of detecting multi-sensor faults.

More recently, a Long Short-Term Memory (LSTM) network was used to actively diagnose sensor bias in an air-cooled chiller system [82].

In [83], a mapless movement policy for robots is presented. Deep reinforcement learning is used to create a policy which is capable of handling a robot even when some of its sensors are broken. The policy is based on three neural models capable not only of moving the robot, but also learning the best movement behavior to adapt it to its perception needs.

The other FTC-related deep learning work is based around bearing fault detection and fault identification [84]. The publications in this survey use, autoencoder networks, restricted Boltzman machines and convolutional neural networks to detect and/or identify failing bearings.

2.2.4.2 Passive Fault Tolerant Control Using Deep Learning

We are not aware of any passive FTC systems that employ deep-learning methods. Thus, passive FTC using deep-learning methods appears to be a research area of potential interest.

The examples of published work in deep learning in active fault tolerant and adaptive controls are added to the active FTC group as shown in Table 2.4.

Table 2.4. Passive and Active FTC Methods with Deep Learning Included.

Passive	**Active**	
[16] [17] [18] [19] [20] [21] [22] [23] [24] [25] [26] [27] [28] [31] [32] [33] [34] [35]	[37] [38] [39] [40] [41] [42] [43] [44] [45] [46] [47] [48] [49] [50] [51] [52] [53] A: [A1-A82]	**FTC**
[57]	[58] [59] [60] [61] A: [A84-A90]	**DDC**
	[62]	
[63] [64] [65] [66] [67]	[68] [69] [70] [71] [72] [73] [74] [75] [76] A: [A91-A117]	**AI**
	[78] [79] [80] [81] [82] [83] [84]	**DL**

The control approach proposed in this dissertation is a passive controller that uses deep-learning methods.

All of the works discussed thus far have some form of limitation. Active FTC controllers must rely on detecting and identifying a fault. To prevent the poor performance that would result from a high rate of false positives or false negatives in the detection mechanism, some form of cumulative residual is calculated thus forcing a delay between

the onset of the attack or fault and the detection of it. Once detected, the active FTC system must select the appropriate corrective action. Combined, these steps result in a delay that could allow the system under control to experience a critical failure before control is regained.

Among the passive FTC works cited, the limitations are primarily system-based (linear or piece-wise linear only) [16] [17] [19] [23] [24] or component-based (sensor faults only [26] [34] or actuator faults only [22] [18] [27] [31] [35] [63] [64] [67]). The other limitations are the requirement to add redundant components [25] [32] or limiting the types of attacks or faults that can occur [20] [21] [28]. Table 2.5 summarizes the limitations as described.

Table 2.5. References Sorted by Limitations.

Limitation	**Referenced Work**
Response Delay from Fault/Attack Onset	All Active FTC
Linear (Piecewise Linear) Systems Only	[16] [17] [19] [23] [24]
Sensor Faults Only	[26] [34]
Actuator Faults Only	[22] [18] [27] [31] [35] [63] [64] [67]
Redundancy Required	[25] [32]
Specific Attack/Fault Types Only	[20] [21] [28]

The ideal controller would adapt to the effects of faults and attacks as a matter of routine operations, not relying on a detection mechanism. This controller would be able to deal with attacks and faults on both sensors and actuators. It would be able to work on linear and nonlinear systems alike and would not require adding additional redundant components to function properly. In this work, we propose such a controller.

2.2.5 Faults and Cyber-Attacks on FTC as Represented in Literature

In the body of FTC work, the faults that can affect the system under control are represented in different ways. In [70] the speed at which the faults occur was addressed by categorizing them as either *abrupt* or *incipient*. Abrupt faults occur quickly and are generally easier for active FTCs to detect and identify, while the effects of incipient faults come on gradually over time and can be more difficult for active FTCs to detect or identify.

In identifying the faults that could affect a wind turbine [85], the authors separate out the faults as sensor faults and actuator faults. The sensor faults are labeled as *Fixed Value* or *Gain Factor*. A Fixed Value fault occurs when the sensor repeatedly reports the same value when that value should be changing. A Gain Factor fault is one in which the sensor reports a higher value than is correct, but the reported value does change over time. For actuators, the faults are labelled as either *Offset* or *Changed Dynamics*. An Offset fault is said to have occurred when an actuator consistently misses the intended end state by the error size. A Changed Dynamic fault occurs when the actuator responds differently to a given command.

A popular approach in much of the research is to assume the fault is manifested as a fading component. For example, the work in [86] represents the faulty actuator as responding to the control command at some level below that which is intended.

Mathematically, faults are often treated as additive or multiplicative terms [87] appended to the in the model of the system.

Taken together, these articles provide useful examples of different ways that component faults can be represented, but none provide a comprehensive picture of the effects of faults on industrial control systems.

Three studies were conducted in 2017. The authors of [88] present a model-based software development framework integrated with a hardware-in-the-loop (HIL) testbed for rapidly deploying CPS attack experiments. The framework provides the ability to emulate low level attacks and obtain platform specific performance measurements that could be difficult to obtain in a traditional simulation environment. This work focused on the attack vectors and types of attacks (DDoS, packet sniffing, side-channel attacks, etc.) rather than looking at the effects of the attacks on control components.

In [89] the authors capture and systematize existing research on CPS security under a unified framework. The framework consists of three orthogonal coordinates: (1) from the security perspective, they follow the well-known taxonomy of threats, vulnerabilities, attacks and controls; (2) from the CPS components perspective, they focus on cyber, physical, and cyber-physical components; and (3) from the CPS systems perspective, they explore general CPS features as well as representative systems (e.g., smart grids, medical CPS and smart cars). Here again the research does not drill down to the physical component level.

In [90] the authors present an analysis method (STPA-SafeSec) to consider both system safety and security. This work references the physical layer components as part of the process analysis, but does not address specific effects at that layer.

In work leading up to this dissertation [91] industrial control systems were modeled along with inputs from an attacker and the system response was tested using formal verification. That research highlighted the necessity of knowing all the effects a fault or cyber-attack can have on a CPS.

There is no published work that addresses the full range of effects that cyber-attacks and faults can have at the physical layer of a cyber-physical system. We address this in Chapter 3.

2.3 Summary

This chapter reviewed the relevant research in the area of fault tolerant controls (both active and passive), data driven fault tolerant controls, the use of artificial intelligence in FTC and, specifically, the application of deep-learning methods. The limitations of previously published works were identified and the area of passive FTC using deep learning was identified as an open area of research. There was also a review of the research addressing the types of faults and attacks and how they are represented in the TC research.

3 Background and Methodology

3.1 Chapter Overview

As stated in Chapter 1, this dissertation presents a novel, passive controller for industrial control systems designed to operate properly in the presence of multiple cyber-attacks and/or component faults. This chapter initially presents background information on the network structure and protocols used by industrial control systems. This is followed by the development and introduction of a taxonomy of the effects faults and cyber-attacks can have on the physical components of the system. The chapter also introduces the design of a deep learning-based, passive controller design for ICS networks. The application of the taxonomy of effects to training and testing this controller is discussed. Finally, a version of the controller that uses an ensemble of smaller networks instead of a single large network is described.

3.2 ICS Networks

Industrial Control Systems can be very complex with multiple layers of activity, as shown in Figure 3.1 (originally from the DoD Committee on National Security Systems Instruction (CNSSI) No. 1253, "Security Categorization and Control Selection for National Security Systems"). In this layered representation, the highest layer represents the interface between the ICS and the outside world while the lowest layer represents the physical components that interface the controller to the system under control. Layers 2 through 5 use well known Internet Protocols (IPs).

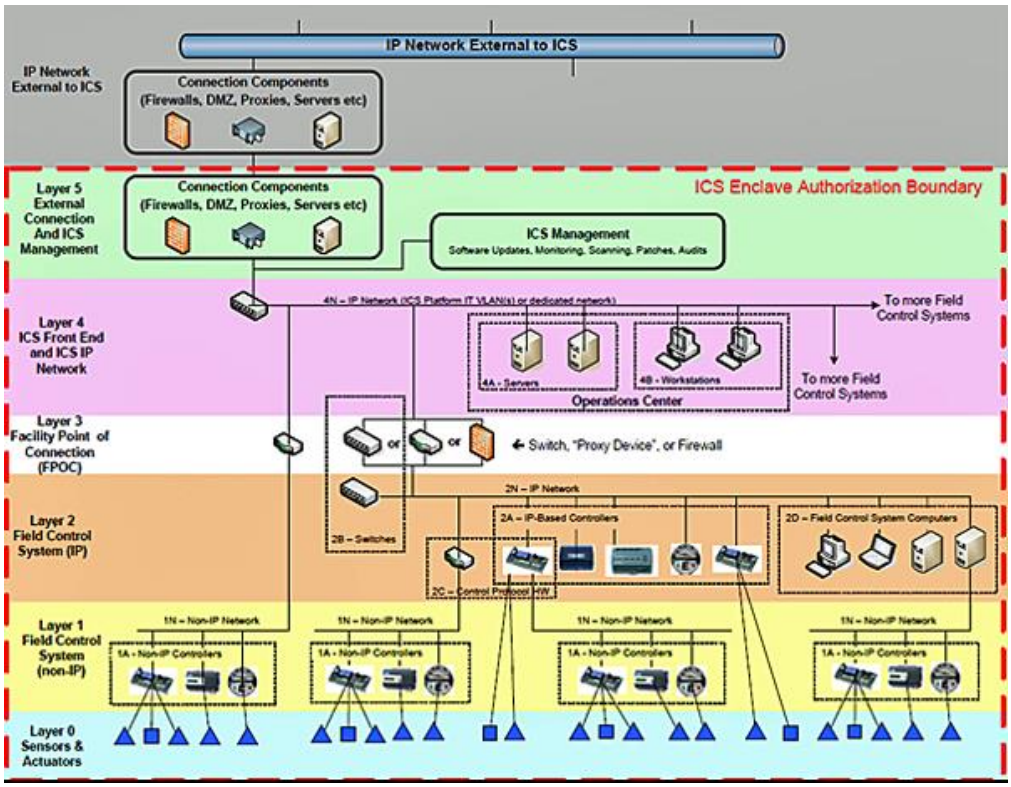

Figure 3.1. CNSSI 1235 ICS Overlay Enclave Authorization Boundary & Layers [92].

Less well known are the non-IP connections appearing in layers 0 through 2, as shown in Figure 3.2.

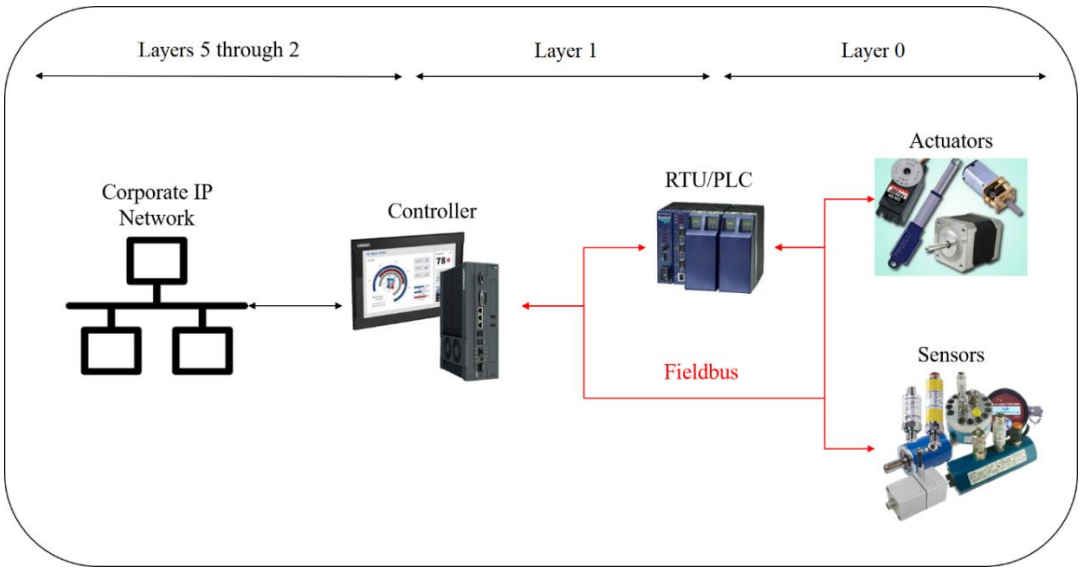

Figure 3.2. Notional ICS Network.

Depending on the size and complexity of the ICS network, it may or may not include Programmable Logic Controllers (PLCs) or Remote terminal Units (RTUs). A small control system may have the controller tied directly to the individual components (actuators and sensors) that interact with the physical system (or plant) under control.

The connections from layer 2 down to layer 0 can be radio, cellular or wired (telephone, coaxial or Ethernet). These connections are referred to as a "fieldbus". Fieldbus was the original protocol used for these communications. According to the International Electrotechnical Commission (IEC), there are 16 different Communication Profile Families (CPFs) or protocols used in process controls [93]. Fieldbus is the generic term for any of them. The most common are Foundation Fieldbus, Profibus, Modbus, and the Control and Information Protocol (CIP). For the purposes of this research effort no distinction is made between various fieldbus protocols. All of them carry data to and from the controller. The particular format of the data and the protocol(s) used are irrelevant to this research effort.

3.3 Effects of Cyber-Attacks and Faults

The survey [89] cited in Chapter 2 provides a taxonomy of cyber threats (Criminal, Espionage, Insider, etc.) and identifies the various types of cyber-attacks that can affect cyber-physical systems. However, the effects discussed are limited to the system level and the proposed "controls" are all focused on limiting network access to the system. Nowhere in the literature is there discussion of the component-level effects that cyber-attacks can have on an industrial control system. The remainder of this section uses the ICS enclave layers described in Figure 3.1 and the list of ICS attacks listed in [89] as the starting point

to develop a taxonomy of cyber-attack effects on ICS. The effects of faults are also rolled in so the resulting taxonomy can be used in the training and/or testing of a control system meant to address the faults and cyber-attacks that can impact an ICS.

The cyber-attack examples include the attack on the Maroochy Water Services in Queensland Australia. In that case, the attacker accessed the system via a wireless communications link (layer 2) and used his system knowledge to send false commands to the system pumps (actuators) at layer 0.

In the Stuxnet example, the malware was written to exploit the lack of encryption on the Siemens motor controllers and the fieldbus that connected them to infect the system. This resulted in harmful control commands being injected into the system and false sensor signals being sent to the controllers so the human monitors were not aware of the misbehaving systems. With Stuxnete, the attack entered the system at layer 5 and propagated through the ICS until it found its way to the motor controllers (layer 1).

The German steel mill attack [1] was an example of hackers accessing the corporate IT network (layer 5) and pivoting via layer 4 to the operational network (layer 3). In that case the hackers used the controllers in layers 2 and 1 to eventually manipulate multiple actuators (layer 0) and put the system into a state from which the blast furnace could not easily be shut down.

The Modbus worm [94] exploits the lack of authentication in the Modbus protocol. The worm has been demonstrated to cause DoS attacks by causing servers to repeatedly reset and injecting false command signals to the system actuators.

A research effort using the Shodan program [95] found over 14,000 ICS directly connected to the internet. Web-based attacks have been used to leave devices in an open

connection state thus allowing an illegitimate user to access the field devices remotely and execute malicious commands or cause a denial of service to the legitimate users.

Looking across these examples, there is no single means of access or type of attack that was common to all. The only common factor in each was that the field devices (sensors and/or actuators) of the system were affected. In the case of the Denial of Service attacks, the field devices never received the command signals and therefore did not respond. In those attacks where malicious control signals were introduced to the system. The actuators responded to those modified signals. The Stuxnet attack was a good example of falsified sensor data being introduced into the system thus preventing any notification that the system was behaving abnormally.

3.4 Taxonomy of Effects on Physical Components

At layer 0, sensors and actuators are the components which interface between the cyber side and the physical side in cyber physical systems as shown in Figure 3.3.

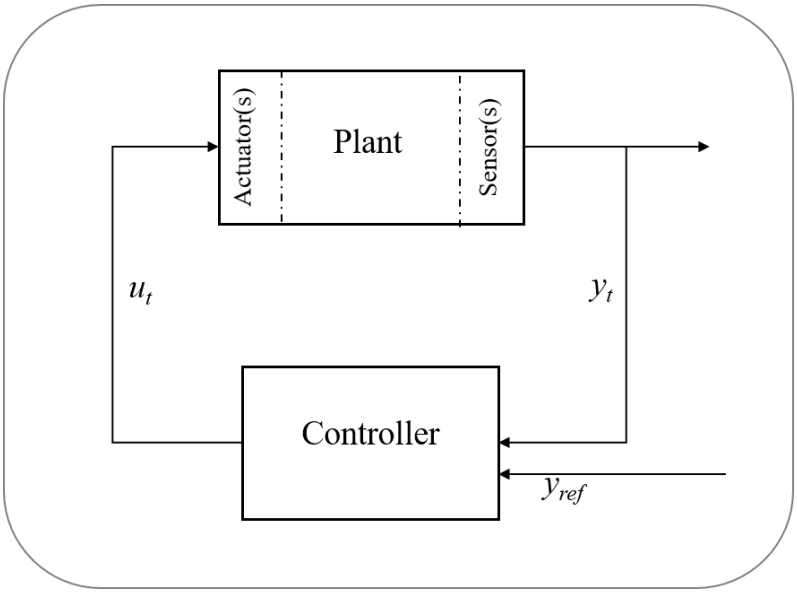

Figure 3.3. Basic Control System

From the perspective of the controller, the information of interest is the last control signal sent to the actuator(s) of the plant u_t and the output reported by the sensor(s) of the plant y_t. A "dumb" controller assumes the actuators are behaving as commanded and that the sensor readings it is receiving are accurate. Therefore, minimizing any difference (or error) between the reported output y_t and the desired output y_{ref} is addressed by adjusting the commands to the plant actuators u_t. Other than the sensor readings y_t, the plant is a black box operation to the controller.

In reality the output error could be caused by an attack or a fault of the actuator (shown as $\hat{u}_t$ in Figure 3.4) or the error may be incorrectly perceived by the controller because of an attack or a fault of the sensor (shown as $\hat{y}_t$ in Figure 3.4). Even more challenging still, is a combination of both.

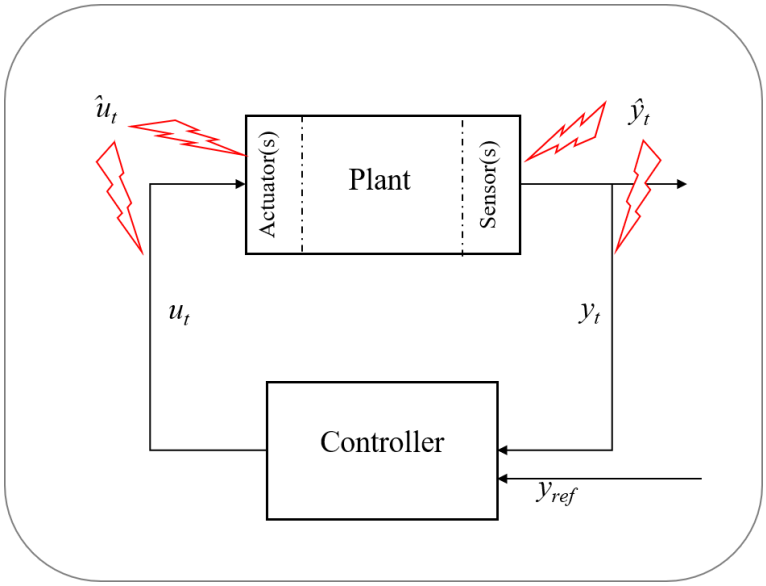

Figure 3.4. Basic Control System with Attacks/Faults

A control system, like that shown in Figure 3.4. can be modeled using simple differential equations

$$\dot{x} = Ax + Bu$$

$$y = Cx + Du$$

Where x is the system state, u is the input to the system actuator(s) and y is the output from the system sensor(s). A common way of representing a fault in the system is to add the fault term to the equations. So, an actuator fault would be represented in the differential equations as

$$\dot{x} = Ax + B\hat{u}$$

$$y = Cx + D\hat{u}$$

Where $\hat{u}$ is the resulting faulty actuator signal resulting from some function of u and an erroneous or malicious input v.

$$\hat{u} = f(u, v)$$

Similarly, a sensor fault would be a function of the output equation and a faulty or malicious input s.

$$\hat{y} = f(Cx + Du, s)$$

This $\hat{u}$ and $\hat{y}$ nomenclature will be used through this section and we will return to these control system equations shortly.

By focusing on the effects of faults and/or attacks, rather than how the attack or fault occurred, we can enumerate and describe the effects of cyber-attacks and faults in terms of how the system sensors and actuators behave. We present the development of a taxonomy of these abnormal behaviors which will be referenced in later chapters of this work.

3.4.1.1 Sensors

The term *sensor* refers to a device that measures something from the physical system under control and reports that measurement to the controller. All modern ICS sensors operate on common principles; a physical change in the system produces a corresponding electrical (voltage, current, capacitance, etc.) change in the sensor. Figure 3.5 shows an example of a water level measurement sensor.

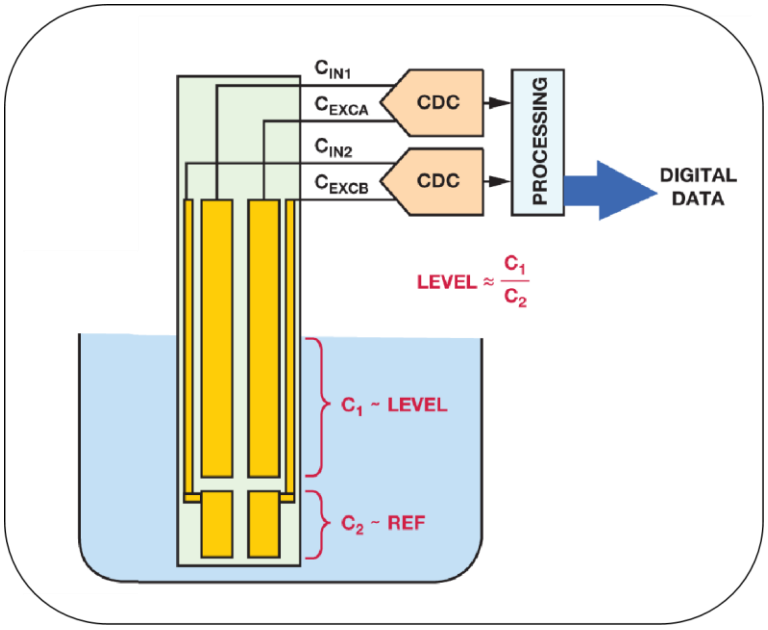

Figure 3.5. Example Water Level Sensor [96]

In the case of this example sensor, changing water level causes a change in the capacitance of the sensor. That capacitance measurement is converted (via Capacitance-to-Digital Converters) to a digital signal which is then sent to the controller via the fieldbus.

From one time increment (t) to the next ($t + 1$), the value of the measurement sent (reported) from a properly functioning sensor can only change in one of three ways as shown in Table 3.1, where *y* is the value of the reported signal from the sensor.

Table 3.1. Possible Changes of Sensor Measurement Values

Value Change	
Increase	$y_t < y_{t+1}$
Decrease	$y_t > y_{t+1}$
No change	$y_t = y_{t+1}$

Any fault in the sensor can cause the value of the reported measurement (y) to be different than the actual value. Similarly, a cyber-attack on the system can alter or replace the value being reported via the fieldbus from the sensor. In either case, if the reported value is incorrect, there are only four comparative ways in which the measured value can differ from the actual value over a single time step as shown in Table 3.2, where $\hat{y}$ is the corrupt reported value, and y is the correct value of the measurement (i.e., the value that should have been reported).

Table 3.2. Possible Changes in Reported Sensor Measurement Values

Reported Value Differs from Actual Value	
High	$\hat{y}_t = y_t$ AND $\hat{y}_{t+1} > y_{t+1}$
Low	$\hat{y}_t = y_t$ AND $\hat{y}_{t+1} < y_{t+1}$
No change	$\hat{y}_t = y_t$ AND $\hat{y}_t = \hat{y}_{t+1}$ AND $y_t \neq y_{t+1}$
No value	$\hat{y}_t = y_t$ AND $\hat{y}_{t+1} = \{\ \}$

These four comparative variations between the reported and actual values can be considered the building blocks for creating the effects of a fault or cyber-attack on the sensor. By expanding the time frame to cover several consecutive time steps, we begin to see more possible effects. If the same type of effect persists over time, the resulting effect remains the same. If different basic effects are combinations of these effects can be added

to those already identified. If there is a pattern to the combination the effect is cyclical, otherwise it is stochastic. Thus, the effects on the reported value, as compared to the actual value, can be represented by the six different effects described in Table 3.3. These six representative effects can be set to start abruptly or gradually (incipient faults) and, of course, the specific values of the variables should be set to match the system to be tested.

Table 3.3. Fundamental Effects on Sensors

	Sensor Effect Descriptions		
1	Increased Value	$\hat{y}_{t+z} = \alpha + y_{t+z}\ OR\ \breve{\alpha}(y_{t+z})$	Where $\alpha > 0$ and $\breve{\alpha} > 1$
2	Decreased Value	$\hat{y}_{t+z} = \beta + y_{t+z}\ OR\ \breve{\beta}(y_{t+z})$	Where $\beta < 0$ and $0 < \breve{\beta} < 1$
3	Stochastic Value	$\hat{y}_{t+z} = y_{t+z} + \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(z-\mu)^2}{2\sigma}}$	Where μ is mean and σ is the standard deviation of additive Gaussian noise
4	Cyclical Value	$\hat{y}_{t+z} = y_{t+z} + \gamma\ sin(\delta\pi z)$	Where γ determines the magnitude of an additive cyclical effect and δ determines the frequency
5	Fixed Value	$\hat{y}_{t+z} = \varepsilon$	Where ε is the fixed value
6	No Value	$\hat{y}_{t+z} = \{\}$	

Returning to the control theory equation for the sensor output

$$\hat{y} = f(Cx + Du, s)$$

The variable s is drawn from the third column of Table 3.3. If, for example, the control designer wants to represent an additive effect (# 1 through #4) to their system model, the output equation would take the form $\hat{y} = Cx + Du + s$. For multiplicative effects (#1 or #2) the output equation would take the form $\hat{y} = s(Cx + Du)$. Effects #5 and #6 are simple replacement functions (i.e., $\hat{y} = s$).

3.4.1.2 Actuators

An *actuator* is a device that effects a change in the physical system under control. The actuator receives an electronic command signal from the controller via the fieldbus and converts that signal into physical action.

The process to develop the effects of cyber-attacks and faults on actuators is very similar to that done for sensors. From one time increment (t) to the next ($t+1$), the value of the command signal sent to a properly functioning actuator can only change in one of three ways as shown in Table 3.4, where u is the value of the command signal.

Table 3.4. Possible Changes in Actuator Command Values

Value Change	
Increase	$u_t < u_{t+1}$
Decrease	$u_t > u_{t+1}$
No change	$u_t = u_{t+1}$

A cyber-attack on the system can alter or replace the command signal value to the actuator. Similarly, a mechanical or electrical fault in the actuator can cause the actuator to respond to the command signal as if the value of the signal were something different than the actual value u. In either case, if the actuator does not respond as desired, there are only four ways in which the action value can differ from the commanded value over a single time step as depicted in Table 3.5, where $\hat{u}$ is the altered or misinterpreted command signal. These four comparative variations between the received and actual values are the building blocks for creating the effects of a fault or cyber-attack on the actuator. By expanding the time frame to cover several consecutive time steps, we begin to see more possible effects.

Table 3.5. Possible Variations from Actuator Value

Reported Value Differs from Actual Value	
High	$\hat{u}_t = u_t$ AND $u_{t+1} < \hat{u}_{t+1}$
Low	$\hat{u}_t = u_t$ AND $u_{t+1} > \hat{u}_{t+1}$
No change	$\hat{u}_t = u_t$ AND $u_t \neq u_{t+1}$ AND $\hat{u}_t = \hat{u}_{t+1}$
No Value	$\hat{u}_t = u_t$ AND $\hat{u}_{t+1} = \{\}$

If the same type of effect persists over time, the resulting effect remains the same. Combinations of these effects can be added to those already identified. If there is a pattern to the combination, the effect can be considered cyclical, otherwise it is stochastic. Thus, the effects on the received command signal value, as compared to the actual value, can be represented by the six different effects described in Table 3.6. These six representative effects can be set to start abruptly or gradually (incipient faults) and, of course, the specific values of the variables should be set to match the system to be tested.

Table 3.6. Fundamental Effects on Actuators

	Actuator Effect Descriptions		
1	Increased Value	$\hat{u}_{t+z} = \alpha + u_{t+z} \;\; OR \;\; \breve{\alpha}(u_{t+z})$	Where $\alpha > 0$ and $\breve{\alpha} > 1$
2	Decreased Value	$\hat{u}_{t+z} = \beta + u_{t+z} \;\; OR \;\; \breve{\beta}(u_{t+z})$	Where $\beta < 0$ and $0 < \breve{\beta} < 1$
3	Stochastic Value	$\hat{u}_{t+z} = u_{t+z} + \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(z-\mu)^2}{2\sigma}}$	Where μ is mean and σ is the standard deviation of Gaussian noise
4	Cyclical Value	$\hat{u}_{t+z} = u_{t+z} + \gamma \, sin(\delta \pi z)$	Where γ sets the magnitude of a cyclical effect and δ determines the frequency
5	Fixed Value	$\hat{u}_{t+z} = \varepsilon$	Where ε is the fixed value
6	No Value	$\hat{u}_{t+z} = \{\}$	

Returning to the control theory equations for the system, the variable v is drawn from the third column of Table 3.6. For all of the actuator effects, the system equations would take the form

$$\dot{x} = Ax + B\hat{u}$$

$$y = Cx + D\hat{u}$$

If, for example, the control designer wants to represent an additive effect (#1 through #4) to their system model, then $\hat{u} = v + u$. For multiplicative effects (#1 or #2) the altered input signal is derived from $\hat{u} = v(u)$. Effects #5 and #6 are simple replacement functions (i.e., $\hat{u} = v$).

The 12 representative effects (from Table 3.3 and Table 3.6) constitute a taxonomy of effects that cyber-attacks and faults can have on the sensors and actuators of an ICS. Using this taxonomy, the control designer can represent the effects as abrupt or incipient, as intermittent or continuous, as a single effect or in combination with other effects.

This taxonomy is a basis upon which any fault tolerant or adaptive control can be tested. Because the effects can be applied to any Industrial Control System or any Cyber-physical system, they can be used to test both active and passive FTC. The flexibility of setting variable values associated with each effect also means they can be readily applied to the training of control systems that employ any kind of learning in their design. This taxonomy is foundational to the controllers designed and presented in later chapters of this dissertation.

3.5 Proposed Controller Considerations

To be generally applicable to ICS, the controller must work with both linear and non-linear systems. The controller must also be capable of handling faults or cyber-attacks that affect both the sensors and the actuators of the system. The controller must also be capable of handling any type of cyber-attack or fault effect.

As mentioned in Chapter 2, the goal of active FTC is ensuring the stability and performance, possibly degraded, of a system experiencing one or more faults, by reconfiguring the controller on-line, using a fault detection and diagnosis (FDD) component that detects, isolates and estimates the current fault. Contrary to this active approach, the passive solution uses a unique robust controller that, will deal with all the expected faults. The passive approach has the advantage of avoiding the time delay required in active FTC for on-line fault diagnosis and control reconfiguration. In addition, passive controller designs are generally more simple in their design.

A primary criticism of passive (proactive) controllers is that a knowledge of the types of attacks or faults the controller may face is required a priori. As seen in Chapter 2, the existing literature regarding passive FTC always places some sort of limitation on the type of fault, or type of component or type of system to which the controller can be applied. Since almost any conceivable type of component effect can be represented using the taxonomy described previously, these limitations can be set aside and a truly universal passive FTC can be considered.

Because it would be used with complex systems, such a controller must necessarily be capable of considering multiple simultaneous inputs. Which specific inputs are the most important will vary from system to system, so the controller should be able to select the

inputs used to determine the correct control command. Because the effects of cyber-attacks and faults are characterized in part by how they change over a segment of continuous time, the controller must also be able to leverage the change in inputs over a segment in time. These considerations drove the decision to design the proposed controller around a Long Short-Term Memory (LSTM) network [97].

3.5.1 Long Short-Term Memory Networks

LSTM networks are a type of deep-learning, recurrent neural network (RNN) designed to classify activities over time. LSTMs are a favored tool in applications such as natural language recognition or activity classification from video inputs. Their ability to use time series inputs makes LSTMs a good candidate for use with industrial control systems. ICS are commonly continuous-time (analogue) plants controlled by discrete-time (digital) controllers.

Intended for use with time series inputs, RNNs not only process an input vector but they also include previous hidden layer values as well. This allows the network to use previous iterations as part of its calculations for selecting the best output. Because RNNs use back-propagation to update the network weights during training, they are prone to phenomena known as “vanishing gradient” and “exploding gradient” [98].

The relationship between layers in a RNN is multiplicative. As a result, in deep networks (those with many hidden layers), the values being passed back through the network to update the weights can quickly become so small that they have little or no effect on the front layers (vanishing gradient). It also easy for the opposite to occur where some values grow exponentially causing some weights to be grossly over-emphasized (exploding gradient).

LSTM networks address these problems by including a "forget gate" which allows the individual nodes (called cells) of the network to either carry useful information through to the next layer to update the weights or dismiss information that will overly affect the weights. Figure 3.6 shows a notional LSTM cell, also called a block. As shown, an individual cell takes as input the output of a previous cell h_{t-1}, the previous cell state c_{t-1}, and the new system input x_t, also called a feature. The output from a given cell is denoted as h_t and the state of that cell is c_t. The internal gates are the forget gate f, the input gate i, the input modulation gate g and the output gate o.

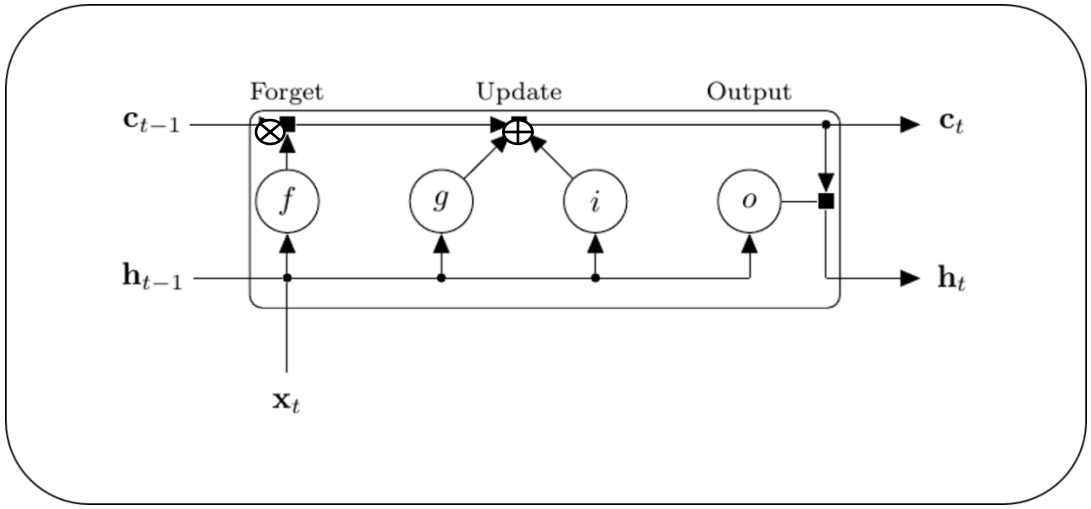

Figure 3.6. Typical LSTM Block [99].

The equations for the individual cell are given below, in which the subscript t denotes time.

$$f_t = \sigma\big(w_f[h_{t-1}, x_t] + b_f\big)$$
$$i_t = \sigma(w_i[h_{t-1}, x_t] + b_i)$$
$$o_t = \sigma(w_o[h_{t-1}, x_t] + b_o)$$

where

f_t represents the forget gate

i_t represents the input gate

o_t represents the output gate

σ represents the sigmoid function

w_x represents the weights for the respective gate (x) neurons

h_{t-1} represents the output of the previous LSTM block (at time $t-1$)

x_t represents the input at the current time stamp

b_x represents the biases for the respective gates x

Since the LSTM network works with time series input data, multiple cells (blocks) are connected to create the network as shown in Figure 3.7.

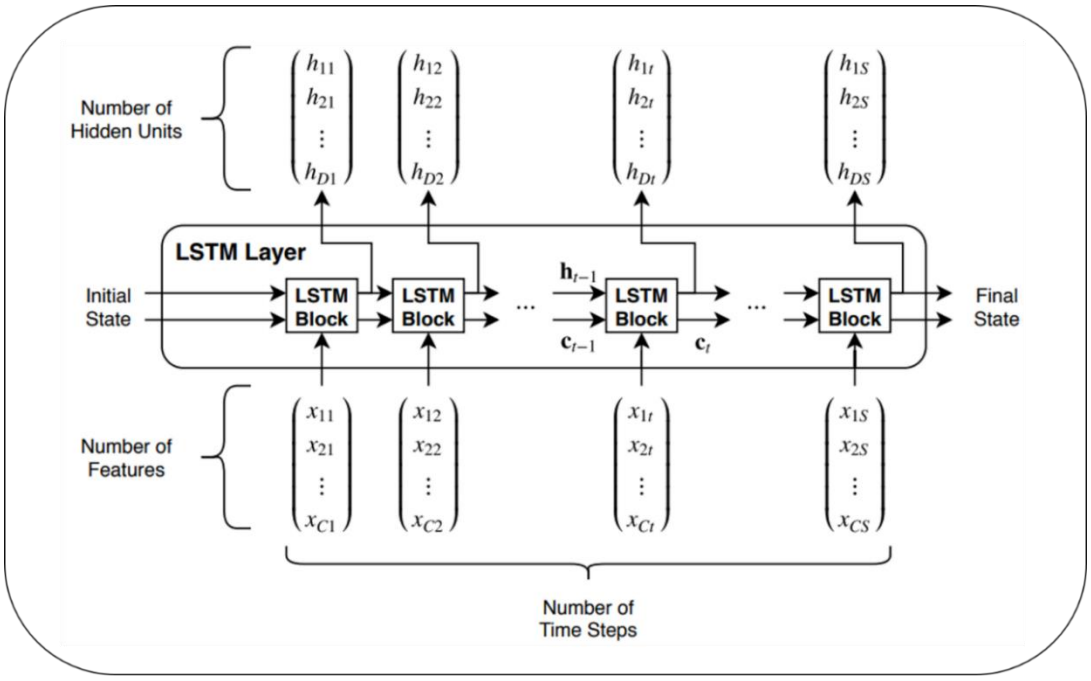

Figure 3.7. LSTM Network Logic Structure [100]**.**

The input to the LSTM network is a 2-dimensional array. One dimension is the number of features (designated as *x* in Figure 3.7) and the other is the number of time steps (designated as c in Figure 3.7). The number of features is driven by the input data to the network and is set when the network is designed. The number of prior time steps can vary

from feature to feature, but large variations in the number of time steps from one feature to another can adversely affect the performance of the network [101].

The number of hidden layers and nodes in the network is also determined when the network is designed. Increasing the number of hidden units can improve the performance of the network, but comes at a cost of longer training times and increased time to calculate a solution. Increasing the number of hidden layers in the network can *significantly* increase the time required for the network to calculate a solution.

3.5.2 Proposed Controller Design

An overview of the proposed controller is shown in Figure 3.8.

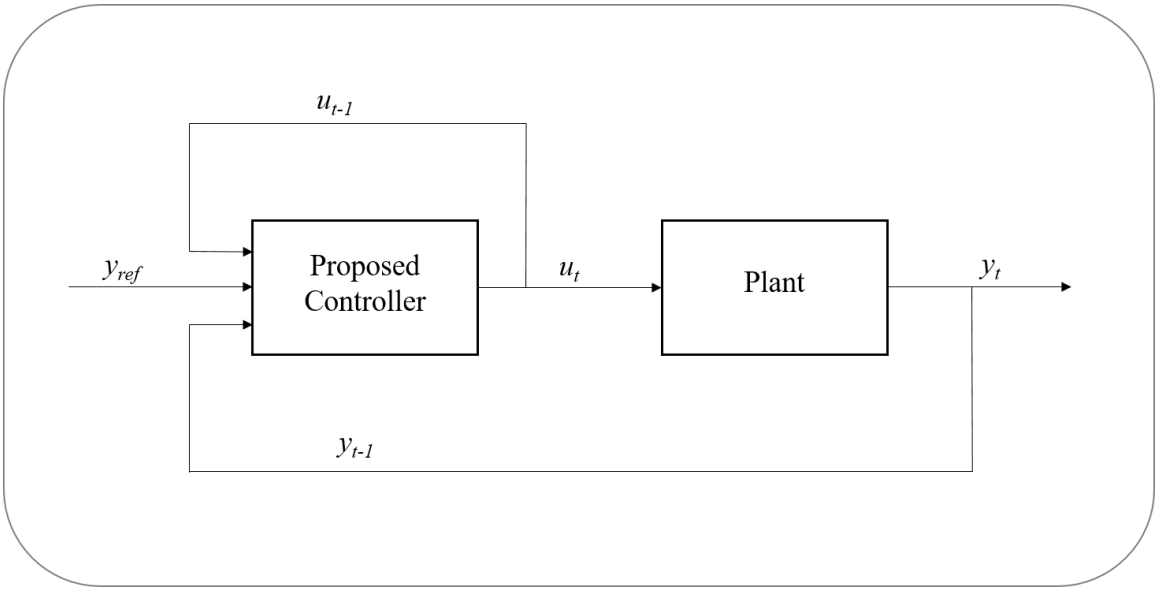

Figure 3.8. Overview of Proposed Controller with Plant.

It takes as input the desired output value of the plant (y_{ref}), the previous output value of the plant (y_{t-1}) and the previous input to the plant (u_{t-1}) that resulted in y_{t-1}. The output from the controller is the new input to the plant, u_t.

From the perspective of the controller, there may be no way to tell if a change in behavior and associated sensor readings is the result of a faulty sensor, a system fault, or if it is an accurate measure of a system being affected by an actuator that is in fact under attack. The normal functioning of the controller must therefore keep the plant performance within established bounds of "acceptable behavior" regardless of how or why the system is affected. The proposed controller uses deep learning to learn a policy, or mapping, of various inputs to the appropriate output. The mapping is a function of the controller inputs which results in a single desired controller output (u_t) to the actuator to which it is assigned.

$$u_t = f(y_{t-1}, u_{t-1})$$

The specific stages of the proposed controller are shown in Figure 3.9 and described below.

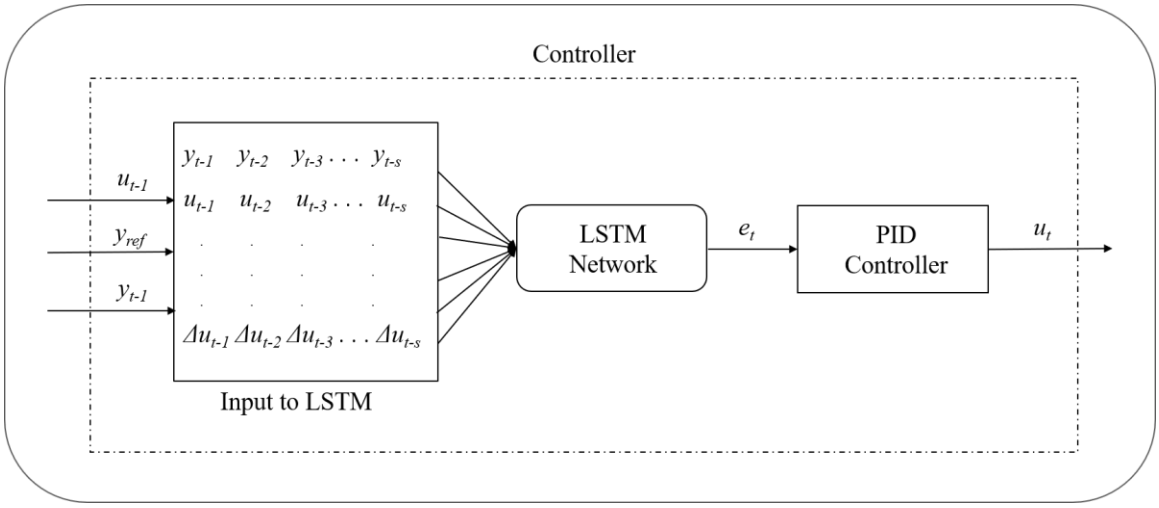

Figure 3.9. Stages of Controller Using LSTM Network.

3.5.3 Input to the LSTM

The input to the controller includes the values reported by the system sensors and the control value(s) previously sent to the actuators. These values, along with others derived from these values are internally calculated. These inputs (called features) are

tracked, normalized and presented to the LSTM network as shown in Figure 3.8. The number of features, their composition and their length (time steps) are all determined by the designer and are based on the specific system to be controlled.

3.5.4 Data Driven and Model-Based Control

Depending on the system to be controlled, the LSTM-based controller can be purely data-driven (as with the test subject in Chapter 4) or may be a synthesis of model-based and data-driven control (as with the test subject in Chapter 5). If the latter is the case, then along with the input features, the control program also generates values from multiple internal models of the system under control. The classification decision made by the LSTM is which model's output to select based on the input features provided. Because the LSTM is selecting the model, not the model's output value, there is only one correct solution, even if more than one of the models happens to be calculating the same value.

3.5.5 Converting LSTM Output to a Usable Control Signal

The raw output of the LSTM needs to be converted into a signal that is useable by the system under control. The exact conversion mechanism varies based on the system. In the case of a simple linear system (such as the test subject in Chapter 4) a set of *if-then-else* logic statements may be sufficient. In the case of the test subject in Chapter 5, the output of the internal models from which the LSTM is selecting are possible values for a sensor y_i. Once the LSTM network decides which model to select, the output value of the selected model is fed into a classical controller such as a Proportional-Integral-Derivative (PID) controller. That PID controller then determines the control signal value u_t to the actuator to which it is assigned.

The block diagram in Figure 3.10 shows the principles of how these terms are generated and applied. The figure shows a classical PID controller that continuously receives an error value as the difference between a desired reference $r(t)$ and a measured process variable $y(t)$, and applies a correction based on proportional, integral, and derivative terms. The controller attempts to minimize the error $e(t)$ over time by adjustment of a control variable $u(t)$, such as the speed of a pump motor, to a new value determined by a weighted sum of the control terms.

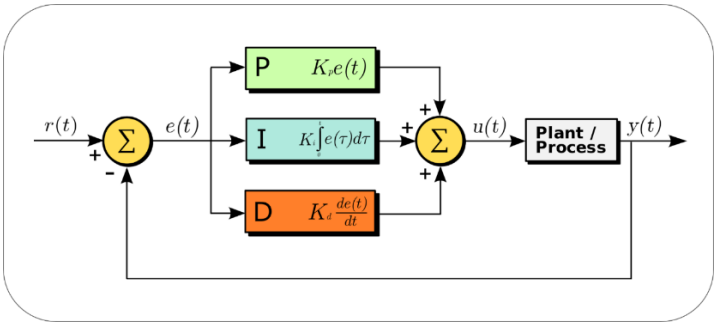

Figure 3.10. Traditional PID Controller Block Diagram [102].

The overall control function is

$$u(t) = K_p e(t) + K_i \int_0^t e(t')dt' + K_d \frac{de(t)}{dt}$$

where K_p, K_i and K_d are all non-negative and represent the coefficients for the proportional, integral, and derivative terms respectively (sometimes denoted *P*, *I*, and *D*).

In the standard form of the equation, K_i and K_d are replaced by K_p/T_i and $K_p T_d$, respectively. This is useful because T_i and T_d have some understandable physical meaning,

since they represent the integration time and the derivative time. Factoring out K_p results in the standard form equation:

$$u(t) = K_p \left(e(t) + \frac{1}{T_i} \int_0^t e(t')dt' + T_d \frac{de(t)}{dt} \right)$$

In the proposed controller design, the LSTM network provides the value of the current, or corrected, sensor reading $y(t)$ to the PID as shown in Figure 3.9.

In this application the PID controller is employed to allow fine tuning of the overall controller. The primary goal of fine tuning is to ensure the controller precisely controls the system when no fault or attack is present. It also simplifies the controller calculations in cases where the reference level y_{ref} may change during the course of normal operations. Instead of the LSTM having to learn the proper output to the actuator for all possible y_{ref} levels, the error e_t is simply calculated and used as a matter of course.

It should be noted that the use of a PID controller was influenced by the specific test case systems being controlled in this dissertation. A different type of system using this LSTM-based controller may be able to use a Proportional, Proportional-Integral, Proportional-Derivative, or other type of controller.

3.6 Training the LSTM Network

Regardless of the number, composition or length of the features, training the LSTM network(s) requires samples of the features as generated when the system is operating. Specifically, the samples need to be representative of the system under normal operating conditions as well as while being affected by faults and/or cyber-attacks that cause the various behaviors in the sensors and actuators described previously.

The data used to train the proposed controller was generated by building a computer simulation of the test subject systems in MATLAB®. The test subject system in this data generation step is controlled by a traditional controller that meets the control requirements when no faults or attacks are affecting the system. This controller has access to true system values, not just the sensor readings or control signals provided to the system previously. With the benefit of access to ground truth, the simulated control program can select the best output for any situation it faces. A combination of correct and incorrect control outputs are saved and used for training the LSTM network.

The simulation is run, iteratively introducing representative values of each of the effects of faults and cyber-attacks on each component. Table 3.7 shows how the number and combinations of data generating simulations runs are determined.

Table 3.7. Numbers and Combinations of Data Generation Runs.

	Components			
	A	**S**	**Runs per Effect**	**Total Runs**
Number of Effects	X	-	A	A(X)
	-	Y	B	B(Y)
	X	Y	C	C(XY)
	Total			∑

The example in Table 3.7 is for a controller with two components that will be affected for testing and training. For some component effects, additional runs are made to better balance out the number of data samples between single and multiple components being tested (i.e., $A(X) + B(Y) \approx C(XY)$).

After each component (sensors and actuators) has been run with each of the different effects, the process is repeated with two components being simultaneously affected. This repeated increasing the number of components affected until the desired number of components have been affected. The information saved from each data generation run are the input variables that will be provided to the LSTM during testing.

Before the data from the simulation runs can be used to train the LSTM networks there are several preprocessing steps it must go through.

Since the training data is generated sequentially, each training data sample is an *m* by *n* matrix where *m* is the number of features and *n* is the length of each input sample. The input length *n* is determined when the controller is designed. The length should be selected as the shortest that provides good results. Inputs longer than this may result in degraded network performance.

The data must also be normalized to a range of (-1,1). This is done by subtracting the mean value for each element and dividing by the maximum value for each element. With the data normalized, 20% of the data is randomly selected and set aside for interim testing and validation.

A key step in data preprocessing for LSTMs is data balancing. If, for example, a significant portion of the data has a correct classification of “1” and only a comparative few with correct classification of 2 or 3, the LSTM will learn it can get good results by simply classifying all inputs as “1”. Therefore, the data is balanced by identifying which classification group is most represented in the data set and duplicating the inputs from the less represented classes until all classes have the same number of samples in the training data. This step is repeated on the sequestered testing inputs as well.

3.6.1 Ensemble or Single LSTM

There are circumstances in which an ensemble of smaller LSTM networks may be preferable to a single larger network. Examples of such cases could include, limited memory on processing hardware which precludes the possibility of training or executing a very large network, inputs to the network that are too difficult for a single network to separate, or a requirement for the classifier to operate under tight time constraints, (i.e., the controller must be capable of a higher sampling rate).

Each of the LSTM networks within the ensemble is trained using a separate but equal portion of the training data. Individually, each of these networks is a reasonably accurate classifier. However, when aggregated together in an ensemble, their combined performance is improved.

The decision of how many classifiers to use in an ensemble depends on the number of class labels (different possible outputs) used for that specific ensemble.

Research indicates, for example, using more than two classifiers to make a binary classification decision offered no improvement in performance and in many cases decreased the accuracy of the ensemble. This “law of diminishing returns in ensemble construction” was first identified and applied by Bonab and Can. Their work in [103] demonstrates that using the same number of classifiers as class labels provides the highest accuracy.

3.6.2 Many Classifiers - Single Solution

There are several possible methods for resolving the results of the networks within an ensemble to a single output. The following approaches were investigated for this research: majority vote scheme, multiple linear regression (MLR), k Nearest Neighbor

(kNN) classifier, and Boosting and Bootstrap Aggregation Algorithms (Bagging). Each is described below and the results of testing with each are described in Chapter 5.

Majority Vote Method. As the name implies, this method takes the output from each classifier (u_z) in the stacked ensemble and counts it as a vote for that particular output to be the output of the entire ensemble (u_t). Whichever output receives the most votes is selected as the overall output of the ensemble. In the case of a tie, a tie-breaking policy is required.

Multiple Linear Regressions (AKA multiple regression). Multiple Linear Regression (MLR) is an extension of ordinary least squares regression. It seeks to predict the value of a dependent variable using the values of more than one explanatory variables by calculating a weighting for each explanatory variable. In this application, the explanatory variables are the outputs from the individual LSTM networks in the ensemble (u_i). The weightings are represented as b_i and the dependent variable is the single overall output of the entire ensemble (u_t). The equation for MLR is

$$u_t = b_0 + b_1 u_1 + b_2 u_2 + b_3 u_3 + \cdots .. + b_z u_z$$

k Nearest Neighbor. The k Nearest Neighbor (kNN) is an instance-based classifier. Classifying an input using the kNN classifier can be as simple as locating the nearest known neighbor in the instance space and labeling the unknown instance with the same class label as that of the located (known) neighbor(s).

One kNN classifier is used for each LSTM ensemble. The classifier takes the output of each LSTM network in the ensemble ($u_1, u_2, \ldots u_z$) and provides the single output u_t. During training, the kNN classifier builds a multi-dimensional space and places each training point q_i in that space. When the kNN classifier is used, the outputs of the LSTMs

in the ensemble ($u_1, u_2, \ldots\ u_z$) are used as a measure of their respective dimension in that space.

In this application "nearest" is defined using the Euclidean distance D between each training point q_i and the new point to be classified which is described by the dimensions ($u_1, u_2, \ldots\ u_z$); refer to the equation shown below:

$$D_{(u,q)} = \sqrt{\sum_{i=1}^{n} (u_i - q_i)^2}$$

The training point p that is closest to the new point described by those dimensions is used to classify the new point u_t. To maximize the performance of the classifier, the number of neighbors chosen k is set to 1.

Boosting and Bootstrap Aggregation Algorithms. Boosting and Bootstrap Aggregation (commonly called Bagging) both strengthen the performance of an ensemble of weaker learners by minimizing bias and variance in the learning algorithms. Bagging randomly selects training samples for each learner and develops them in parallel. Boosting develops the learners in sequence, weighting training data that was not learned well in in one learner so that it can be used again in subsequent learners.

By using the ensemble algorithms built into MATLAB®, the outputs of the LSTMs are not treated as an ensemble, but rather as separate, parallel inputs to an ensemble of decision trees which the *fitcensemble* function in MATLAB® generates. By choosing to let MATLAB® automatically optimize the hyperparameters of the *fitcensemble* function, the algorithm tries both Boosting and Bagging algorithms and selects the one that provides the best results.

Each of these methods for determining the single output u_t of the LSTM stack is tested and the results of testing with each are described in Chapter 5.

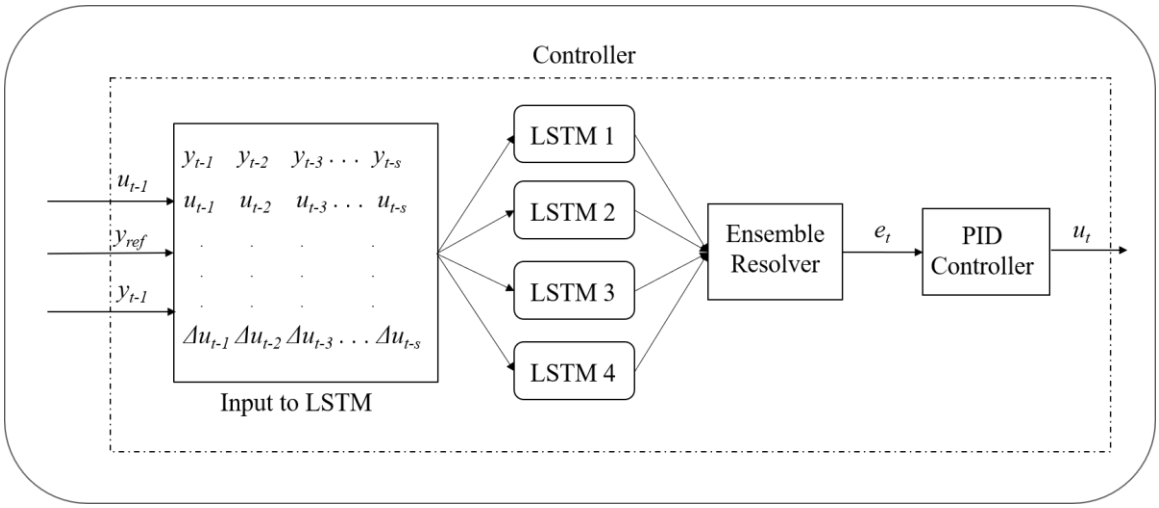

Figure 3.11. Information Flow through an Ensemble to Control a Single Actuator.

Figure 3.11 depicts the logic flow through the controller to generate output signals for a single system actuator. This is duplicated for each system actuator, with each having its own LSTM stack, stack solver and output.

3.7 Test Methodology

The method for testing the controller is very similar to the method for generating the training data. For all testing, the system under test (SUT) and the associated control algorithm are realized as MATLAB® scripts. The tests are performed on an Intel® Core i7 CPU platform with 32 GB of RAM running Windows 10. Each test run is conducted by running the script for a predetermined number of iterations. Prior to each run the attack/fault effects(s) (described previously) is/are set to be activated at an iteration after the system has reached a normal steady state level. Each of the sensor effects are applied on individual runs, then each of the actuator effects were applied on individual runs.

Finally, the sensor and actuator effects are applied simultaneously on the last several test runs, with each run using a different combination of the different effects. In all of the test runs the value of the variable(s) were randomly selected, from the uniform range described previously, prior to the beginning of the test run. The test metrics of interest are:

- Did a catastrophic failure occur? This is defined differently for each system under test.
- Did the controller maintain proper control? – Proper control is defined as keeping the system output values within ±1% of the set reference value.
- If proper control was lost, how long did it take to regain proper control? – This is measured in iterations of the controller algorithm.
- From the onset of the attack/fault effect(s), how long did it take for the controller to respond appropriately? – Definitions for this will vary from system to system and controller to controller.
- Percentage of total iterations in which the controller responded appropriately - Definitions for this will vary from system to system and controller to controller.

3.8 Summary

This chapter provided the development of a taxonomy of the effects that faults and cyber-attacks can have on the physical components of a system. The proposed controller design was explained in detail and an alternate ensemble version was also discussed. Considerations for training and testing the proposed controller based on the taxonomy were discussed and the methodology for testing the proposed controller(s) was described.

4 Proof of Concept - Test Subject: Steam Boiler System

4.1 Chapter Overview

This chapter discusses the use of the linear steam boiler test subject system to demonstrate the utility of the proposed control approach. The specifics of how the proposed controller is applied to the system are described in detail. The taxonomy described in Chapter 3 is used to develop training data a test methodology for the proposed controller design. Finally, the results of the testing on the steam boiler with the proposed controller are discussed.

4.2 The Steam Boiler System

The steam boiler is a linear system with requirements that specify several safety measures and multiple operating modes (separate from the controller). The steam boiler uses binary actuators for which a single classifier type LSTM controller will suffice.

The Steam-boiler Control Specification Problem [104] [105] was initially published by Jean-Raymond Abrial and was provided to participants of the Dagstuhl meeting, “Methods for Semantics and Specification”, in June 1995.

The specification describes a control program serving to control the water level in a steam boiler like those used in nuclear power plants. Based on a specification by the “Institute for Risk Research” and the “Protection and Nuclear Safety Institute” (Institut de Protection et de Sureté Nucléaire) the requirements for the controller emphasize safety and are very formal. Figure 4.1 depicts the steam boiler system.

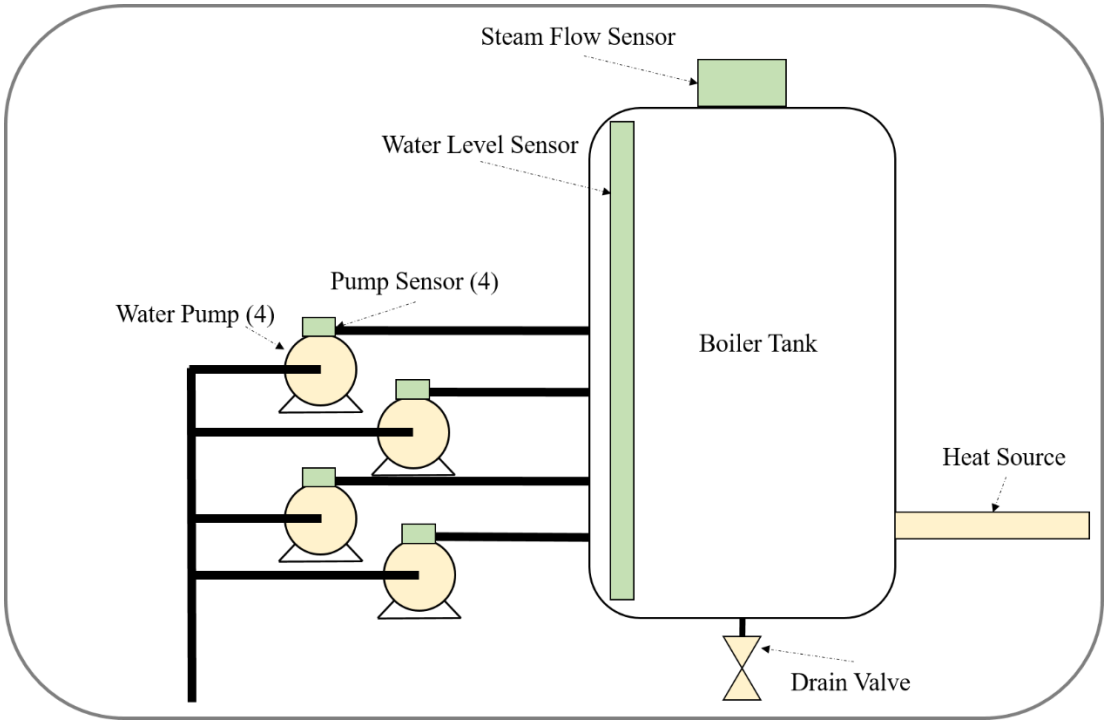

Figure 4.1. Steam Boiler System Diagram.

The controller is responsible for keeping the water level in the tank between preset maximum and minimum levels, while ensuring that the boiler generates a sufficient volume of steam not exceeding a preset maximum. The controller takes as inputs values from sensors that measure the water level in the 1200 liter tank. The output from the controller is control signals sent to the 4 water pumps which are simply On/Off devices.

4.2.1 Operational Modes/Safety Features

Since this steam boiler is intended for use in a nuclear power plant, allowing the system to enter a state that could damage the steam boiler, or connected equipment, is unacceptable. In order to ensure the safety of the system, the specifications for the controller call for it to switch between five different modes [104]. Four are operational

modes (*Initialization*, *Normal*, *Degraded* and *Rescue*); the fifth operational mode is *Emergency Stop*. The operation of these modes is summarized below.

Initialization mode: In this mode, the controller waits for the physical units to become ready and then ensures the starting water level in the tank is between the preset maximum and minimum required for normal operations.

Normal mode: In this mode, the controller's task is to maintain the water level in the boiler between the two "normal" levels. The controller communicates with physical units to obtain information about the states of the units (water level, etc.) and uses this information to react. The program stays in normal mode until a failure is detected, or an emergency stop signal is received.

Degraded mode: In this mode, the program tries to maintain a satisfactory water level, despite of the failure of one or more physical components.

Rescue mode: This is the mode in wherein the program attempts to maintain satisfactory water levels in the boiler, despite failure of the water level measuring unit. This indicates that it has to computationally estimate the water level taking into account boiler dynamics.

Emergency stop mode: The control program has to engage the emergency stop mode either when the vital units experience a failure, or when water levels fail to reach either of its two-limit values. This mode may also be reached upon detection of a transmission error between the program and physical units. In this mode, the control will just send a command to the physical units, to cease operation, and then terminate itself.

4.2.2 System Parameters and Variables

The value of primary concern in this system is the water level in the boiler tank (q). The various capacities and flow rates for the physical system components are shown on Table 4.1.

Table 4.1. Parameters and Variables used in the Steam Boiler System.

Parameters	**Unit**	**Description**
C = 1200	liters	Physical capacity of steam boiler
M2 = 900	liters	Maximum allowable water level
M1 = 100	liters	Minimum allowable water level
N2 = 600	liters	Maximum normal water level
N1 = 400	liters	Minimum normal water level
W = 10	liters/sec	Maximum quantity of steam at output
U1 = 10	liters/sec/sec	Maximum gradient increase of steam at output
U2 = 10	liters/sec/sec	Maximum gradient decrease of steam at output
P = 5	liters/sec	Nominal pump capacity
Variable	**Unit**	**Description**
q	liters	Current quantity of water in steam boiler
p	liters/sec	Current throughput of pumps
v	liters/sec	Current quantity of steam at output

The calculation for the water level change when the system is operating normally is defined by the following time-variant linear equation:

$$q_t = q_{t-1} + \sum_{i=1}^{4} p_i - v$$

There is also an internal model for estimating the water level in the boiler using estimated flow of the pumps.

$$q_{mt} = q_{mt-1} + \sum_{i=1}^{4} pe_i - v$$

4.3 Proposed Controller Design

As applied to this test system, the proposed controller design uses a single LSTM network for all four pumps. For each iteration of the control program, the LSTM is used to determine how many pumps should be turned on. The basic configuration of the system with the proposed controller is shown in Figure 4.2.

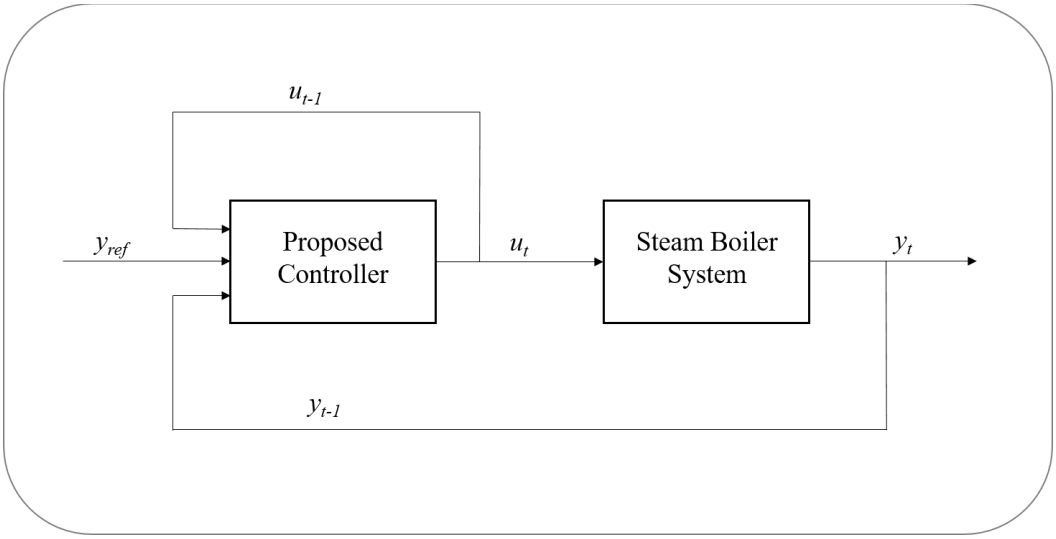

Figure 4.2. Proposed Controller with System.

For the steam boiler controller a few trial runs were conducted to identify the input sequence length that resulted in the best overall performance of the controller. In the trial runs the only parameter that was changed was the sequence length. The results are shown in Table 4.2.

Table 4.2. Change in Network Performance Correlated with Sequence Length

Length	Performance
5	97.98%
10	98.56%
15	97.76%
20	96.80%

As the sequence length was increased, the results of each trial run decreased from the maximum results that were achieved with a sequence length of ten.

Using the sequence length of ten identified in the trial runs, the next stage was to find the combination of hidden layers and nodes per layer that would maximize performance of the LSTM on the Steam Boiler system data. The results of those experiments are shown in Table 4.3.

Table 4.3. Comparison of Network Performance Correlated with Network Size

Nodes	**Hidden Layers**			
	2	**3**	**4**	**5**
768	98.42	98.22	97.44	98.46
512	98.56	98.28	98.34	98.68
256	97.78	98.42	98.26	98.06
128	97.60	97.80	98.58	95.80
64	97.08	96.34	98.22	96.32

The best performance was derived from the network with five hidden layers and 512 nodes per layer. The second best was the network with four hidden layers and 128 nodes per layer. The network size selected was the third best with two hidden layer and 512 nodes per layer. This network size was chosen because additional hidden layers significantly increase the time required to train and test the network. Since the performance difference between the top three was minimal, the smaller network was chosen.

The full LSTM used for the steam boiler controller is a six-layer network with an input layer, two hidden layers (with 512 nodes each), a fully connected 4-node layer, a dense layer (softmax layer) and an output layer with 4 nodes.

4.3.1 Inputs to the LSTM

The LSTM for the steam boiler uses a 16x10 element input matrix. Table 4.4 describes the variable values used to generate the input matrix.

Table 4.4. Input to LSTM.

Input Rows	Description	Example Variables
1	Difference between the reported water level and the reference level	(ref-q_r)
2	Difference between the modelled water level and the reference level	(ref- q_m)
3-6	Last estimated flow rate through each of the four pumps	[pe]
7-10	Last reported flow rate from each of the four pumps	[pr]
11-14	Available pump register	[1,0,1,1]
15	System Mode (Normal, Degraded, Rescue, Emergency Stop)	(11-14)
16	The sum of the estimated flow rate through each of the 4 pumps	(sum(pe))

The output of the LSTM is a single value (1 through 4) indicating how many motors should be on. Based on this decision, a switching routine in the controller then sends the appropriate signal (on or off) to each pump.

4.4 Effects of Faults and Cyber-Attacks

Before discussing how the LSTMs were trained and tested, it is necessary to show how the effects of cyber-attacks and faults on the steam boiler system components are addressed.

Recall the taxonomy of representative sensor effects described in Chapter 3, Table 3.3. Applied to the sensor of the steam boiler system, these effects are applied as described below and summarized in Table 4.5.

1. Increased Value effect - This effect is modeled as an abrupt, multiplicative function with $\breve{\alpha}$ being randomly selected from a uniform distribution with *values* on the interval (1.10, 2.10).

2. Decreased Value effect - This effect is modeled as an abrupt, multiplicative function with $\breve{\beta}$ being randomly selected from a uniform distribution with *values* on the interval (0.10, 0.80).

3. Stochastic Value effect – This effect is modeled as an abrupt, additive function with μ set to zero and σ being randomly selected from a uniform distribution with *values* on the interval (0.05, 0.55).

4. Cyclical Value effect – This effect is modeled as an abrupt, additive function with γ being randomly selected from a uniform distribution with *values* on the interval (0.10, 1.10) and δ being randomly selected from a uniform distribution with *values* on the interval (0.015, 1.015).

5. Fixed Value effect – This effect is modeled as an abrupt, function with ε being randomly selected from a uniform distribution with *values* on the interval (0.00, 20.00).

6. No Value effect – This effect is modeled as an abrupt, function with the value of y_{t+z} being set as Not-a-Number (*NaN*).

In addition to these six effects, it is also prudent to examine the performance of the controller if the quadruple tank system sensors are subjected to incipient effects. Therefore, two more effects are modeled:

7. Increased Value effect - This effect is modeled as an incipient, additive function with α being randomly selected from a uniform distribution with *values* on the interval (0.00, 1.00) and the effect being applied using the equation

$$\hat{y}_{t+z} = y_t + .025\alpha y_{t+z}$$

8. Decreased Value effect - This effect is modeled as an incipient, additive function with β being randomly selected from a uniform distribution with *values* on the interval (0.00, 1.00) and the effect being applied using the equation

$$\hat{y}_{t+z} = y_t - .05\beta y_{t+z}$$

These eight sensor effect representations are summarized in Table 4.5. The *Effect Number* and *Effect Value* fields will be used when discussing test results.

Table 4.5. Summary of Sensor Effects for the Steam Boiler System

Effect Type	Effect Number	Effect Value	Add, Mult or Replace	Abrupt or Incipient
Increased	1	(1.10, 2.10)	Mult	Abrupt
Decreased	2	(0.10, 0.80)	Mult	Abrupt
Stochastic	3	(0.05, 0.55)	Add	Abrupt
Cyclical	4	(0.10, 1.10) (0.015, 1.015)	Add	Abrupt
Fixed	5	(100, 900)	Replace	Abrupt
No Value	6	(NaN)	Replace	Abrupt
Increased	7	(0, 1.00)	Add & Mult	Incipient
Decreased	8	(0, 1.00)	Add & Mult	Incipient

There are no set rules for selecting the ranges of the effect values. In general, the low end of the range should make the effect strong enough to cause a change in the sensor. At the high end the value should not cause the effect to exceed the physical limitations of the sensor. Applied to the actuators of the quadruple-tank system, the effects are applied as described below and summarized in Table 4.6.

1. Increased Value effect – This effect is modeled as an abrupt, multiplicative function with $\hat{\alpha}$ being randomly selected from a uniform distribution with *values* on the interval (1.05, 1.50). This effect leaves the actuator controllable.

2. Decreased Value effect – This effect is modeled as an abrupt, multiplicative function with $\hat{\beta}$ being randomly selected from a uniform distribution with *values* on the interval (0.15, 0.90). This effect leaves the actuator controllable.

3. Stochastic Value effect – This effect is modeled as an abrupt, additive function with μ set to zero and σ being randomly selected from a uniform distribution with *values* on the interval (0.05, 0.55). This effect leaves the actuator controllable.

4. Cyclical Value effect – This effect is modeled as an abrupt, additive function with γ being randomly selected from a uniform distribution with *values* on the interval (0.50, 3.50) and δ being randomly selected from a uniform distribution with *values* on the interval (0.05, 0.55). This effect leaves the actuator controllable.

5. Fixed Value effect – This effect is modeled as an abrupt, function with ε being randomly selected from a uniform distribution with *values* on the interval (1.00, 4.00). This effect leaves the actuator uncontrollable.

6. No Value effect – This effect is modeled as an abrupt, function with the value of y_{t+z} being set as Not-a-Number (*NaN*). This effect leaves the actuator uncontrollable.

Two additional effects are modeled to examine the performance of the controller if the quadruple tank system sensors are subjected to incipient effects:

7. Increased Value effect – This effect is modeled as an incipient, additive function with A being randomly selected from a uniform distribution with *values* on the interval (0.00, 1.00) and the effect being applied using the equation

$$\hat{u}_{t+z} = (u_t + .0025\alpha u_{t+z})u_{t+z}$$

8. Decreased Value effect – This effect is modeled as an incipient, additive function with β being randomly selected from a uniform distribution with *values* on the interval (0.00, 1.00) and the effect being applied using the equation

$$\hat{u}_{t+z} = (u_t - .0019\beta u_{t+z})u_{t+z}$$

These eight actuator effect representations are summarized in Table 4.6. The *Effect Number* and *Effect Value* fields will be used when discussing test results.

Table 4.6. Summary of Actuator Effects for Steam Boiler System

Effect Type	Effect Number	Effect Value	Add, Mult or Replace	Abrupt or Incipient	Controllable
Increased	1	(1.05, 1.50)	Mult	Abrupt	Yes
Decreased	2	(0.10, 0.80)	Mult	Abrupt	Yes
Stochastic	3	(0.05, 0.55)	Add	Abrupt	Yes
Cyclical	4	(0.50, 3.50) (0.05, 0.55)	Add	Abrupt	Yes
Fixed	5	(1.00, 4.00)	Replace	Abrupt	No
No Value	6	(NaN)	Replace	Abrupt	No
Increased	7	(0.00, 1.00)	Add & Mult	Incipient	Yes
Decreased	8	(0.00, 1.00)	Add & Mult	Incipient	Yes

Fixed Value (#5) and the No Value (#6) effects leave the actuator uncontrollable but the steam boiler system design does include redundant actuators, so these two effects will be included in the training or testing of the controller for the steam boiler system.

As with the sensor effects, the variable range values for the actuator effects should be large enough to produce the desired effect, but so large that the actuator is driven beyond its physical capabilities.

4.5 Training the LSTM Networks

The following parameters were used to train the LSTM for this controller in MATLAB®:

Solver: ADAptive Moment estimation (ADAM)
Execution Environment: auto (defaults to GPU if one is present)
Initial Learning Rate: 0.0002
Learn Rate Schedule: piecewise
Learn Rate Drop Factor: 0.5
Learn Drop Rate: 2 (epochs)
Maximum Epochs: 10

In order to expedite the training process of the LSTM, the training inputs are compiled into "minibatches". The larger the minibatch, the quicker the training is accomplished. If the minibatch size is too large, the network will overfit to the training data and not perform well in testing or actual use. Based on the size of the training data set, the minibatch size for this application is 256 inputs.

Another important parameter in training the LSTM is the learning rate. Larger learning rates help the values of the LSTM weights to converge more quickly, but risk possibly settling in local minimum rather than the global minimum. Lower learning rates are more likely to find the global minimum but can take a very long time to get there. MATLAB® allows the designer to vary the learning rate during training. In this application,

the initial learning rate is set at 0.002 and is cut by 50% every 2 epochs. Thus, after 20 training epochs the training rate is dropped to 0.000125.

4.5.1 Generating Training Data

The data used to train the proposed controller was generated by building a computer simulation of the quadruple-tank system in MATLAB®. This test subject system was controlled by a logic statement controller that meets the control requirements when no faults or attacks are affecting the system. This controller was fed the output from the correct model so it always controlled the system correctly.

The simulation was run, iteratively introducing representative values of each of the effects of faults and cyber-attacks on each component as shown in Table 4.7.

Table 4.7. Numbers and Combinations of Data Generation Runs for Steam Boiler.

	Components				
	S	**A1**	**A2**	**Runs per Effect**	**Total Runs**
Effects	8	0	0	8	64
	0	8	0	8	64
	0	0	8	8	64
	8	8	0	1	64
	8	0	8	1	64
	0	8	8	1	64
				Total	**384**

After each component has been run with each of the different effects (8 for the sensor and 8 for each pump), the process is repeated with two components being simultaneously affected (the sensor and each pump, then 2 pumps) which totals 196 effects combinations.

Eight times as many data generation simulations were run with single component effects compared to those with two components affected. This was done to provide an approximately equal number of samples for each.

Each data generation simulation ran for 500 iterations with the component effect starting at iteration number 250. This was done to ensure the system was operating in a normal steady state the effect begins.

A total of 192,000 inputs were generated using them method described above. Each input was a 16x10 matrix.

4.5.2 Data Preprocessing

Before the data from the simulation runs can be used to train the LSTM networks there are several preprocessing steps it must go through.

The data was first be normalized to a range of $(-1,1)$. With the data normalized, 20% of the data was randomly selected and set aside for future testing. This left 153,600 data samples in the training set. Data balancing increased the training set to 162,560 samples evenly distributed across the four classification groups. The LSTMs were then trained with this data using the training parameters identified at the beginning of this section

4.6 Testing the Controller with Steam Boiler System

The linear steam boiler system was tested as a proof of concept; that a passive, data driven controller based on a deep-learning network could properly control a simple linear system in the presence of various and sometimes simultaneous cyber-attacks and faults. This system has redundant actuators and several built-in safety checks to prevent

catastrophic failure. The actuators are binary (off/on) making the control requirements comparatively simple. The testing on this system is intended to determine if a version of the proposed controller can maintain proper control of the system and if faults or attacks that would normally trigger the built-in EMERGENCY STOP mode can be countered to keep the system running rather than shutting down.

The method for testing the controller is very similar to the method for generating the training data. For all testing, the system under test (SUT) and the associated control algorithm are realized as MATLAB® scripts. Each test run is conducted by running the script for 500 iterations. Prior to each run the attack/fault effects(s) (described previously) is/are set to be activated at iteration number 250. Each of the eight sensor effects were applied on individual runs, then each of the actuator effects were applied on individual runs to two of the four pumps. This totaled 24 individual effect runs. Finally, the sensor and actuator effects were applied simultaneously on the last 192 test runs, with each run using a different combination of the different effects. In all of the test runs the value of the variable(s) were randomly selected, from the uniform range described previously, prior to the beginning of the test run.

With each test run, the performance of the steam boiler with the proposed controller was recorded against the metrics in Table 4.8.

Table 4.8. Measured Items for Steam Boiler Testing

Components Affected	How many actuators and/or sensors were simultaneously affected?
Catastrophic Failure	Did the water level ever fall below 100 liters or exceed 900 liters?
Mode 14 Triggered	Was mode 14 (Emergency Stop mode) triggered?
Proper Control Maintained	Was the liquid level in the tank maintained between 495 and 505 liters (reference level ± 1%)?
Iterations to Regain Proper Control	If proper control was not maintained, how many iterations occurred before the system was recovered back to proper control?
Iterations to Correct Solution	How many iterations occurred from the onset of the effect until the controller selected the correct solution?
Iterations with Correct Solution %	What percentage of the iterations in the test were run with the correct solution selected?

For this system, a failed test (catastrophic failure) is defined as one wherein the water level in the tank is allowed to rise above 900 liters, or drop below 100 liters. The performance of the proposed controller is also judged by its ability to keep the water level in the tank within 1% of the desired level. If the controller allows the system to go outside the ±1% range then it is also measured on how long it takes to return it to that level if it does.

The system was tested with each component being affected by one of the fault/attack effects as described previously. After a run with each component being tested with each effect, the effects were applied to two components simultaneously until all combinations of components were tested with all combinations of effects.

The results of the testing are shown in Table 4.9 below. Due to size constraints of the dissertation format, only a portion of the chart is displayed in Table 4.9. The entire chart is found in Appendix B.

Table 4.9. Test Results on Steam Boiler Using Proposed Controller.

Test Run No.	Sensor Effect	Effect Value	Pump 1 Effect	Effect Value	Pump 2 Effect	Effect Value	Components Affected	Catastrophic Failure	Mode 14 Triggered	Maintain Proper Control	Iterations to Regain Proper Control	Iterations to Correct Decision	Iterations Using Correct Solution %
1							0	0	0	1	*	0	100.00%
2	1	206.17					1	0	0	1	*	0	100.00%
3	2	0.71					1	0	0	1	*	0	100.00%
4	3	0.24					1	0	0	1	*	0	100.00%
5	4	0.50					1	0	0	1	*	0	100.00%
6	5	340.08					1	0	0	1	*	0	100.00%
7	6	*					1	0	0	1	*	0	100.00%
8	7	0.28					1	0	0	1	*	0	100.00%
9	8	0.50					1	0	0	1	*	0	100.00%
10			1	2.68			1	0	0	1	*	0	100.00%
11			2	0.74			1	0	0	1	*	0	100.00%
12			3	0.45			1	0	0	1	*	0	100.00%
13			4	0.40			1	0	0	1	*	0	99.33%
14			5	1.88			1	0	0	1	*	105	77.00%
15			6	*			1	0	0	1	*	0	100.00%
16			7	0.74			1	0	0	1	*	0	100.00%
17			8	0.10			1	0	0	1	*	0	100.00%
18					1	2.81	1	0	0	1	*	9	99.00%
19					2	0.49	1	0	0	1	*	0	100.00%
20					3	0.09	1	0	0	1	*	9	98.33%
. . .					. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
198			6	*	5	1.25	2	0	0	1	*	0	100.00%
199			6	*	6	*	2	0	0	1	*	0	100.00%
200			6	*	7	0.21	2	0	0	1	*	0	100.00%
201			6	*	8	0.12	2	0	0	1	*	0	100.00%
202			7	0.46	1	1.93	2	0	0	1	*	0	100.00%
203			7	0.22	2	0.51	2	0	0	1	*	0	100.00%
204			7	0.36	3	0.35	2	0	0	1	*	0	100.00%
205			7	0.39	4	0.26	2	0	0	1	*	0	92.67%
206			7	0.88	5	1.03	2	0	0	1	*	168	85.67%
207			7	0.64	6	*	2	0	0	1	*	0	100.00%
208			7	0.08	7	0.30	2	0	0	1	*	0	100.00%
209			7	0.86	8	0.66	2	0	0	1	*	0	100.00%
210			8	0.78	1	2.95	2	0	0	1	*	0	100.00%
211			8	0.72	2	0.16	2	0	0	1	*	0	100.00%
212			8	0.74	3	0.27	2	0	0	1	*	0	100.00%

213			8	0.93	4	0.24	2	0	0	1	*	0	99.00%
214			8	0.73	5	1.33	2	0	0	1	*	153	83.67%
215			8	0.70	6	*	2	0	0	1	*	0	100.00%
216			8	0.67	7	0.37	2	0	0	1	*	0	100.00%
217			8	0.99	8	0.64	2	0	0	1	*	0	100.00%

Of the 217 total test runs there were no catastrophic failures. In 198 cases the controller kept proper control throughout the test run. In the others, the controller regained proper control in an average of 124 iterations (median of 86). The Mode 14 (Emergency Shut Down) conditions were met in only 2 of the test runs. On average, the controller selected the correct solution in 20.95 iterations (median of zero). In total, 94.46% of all iterations were completed with the correct solution selected by the controller. The results of the test are summarized in Table 4.10.

Table 4.10. Test Result Summary for Proposed Controller

Test Runs	**Comp. Affected**	**Cat. Fail**	**Mode 14 Active**	**Proper Control**	**Iterations to Regain Proper Control**		**Iterations to Correct Model**		**Iterations Using Correct Model %**	
#	*A/S/Both*	*%*	*%*	*%*	*Median*	*Mean*	*Median*	*Mean*	*Median*	*Mean*
16	Actuator	0	0.00	93.75	250.00	250.00	0.00	10.06	99.67	94.64
8	Sensor	0	0.00	100.00	*	*	0.00	0.00	100.00	100.00
192	Both	0	1.02	88.78	78.50	114.50	0.00	50.11	100.00	94.19
Total	**-**	**0**	**0.92**	**91.24**	**86.00**	**124.05**	**0.00**	**20.95**	**100.00**	**94.46**

Statistically speaking, this controller performed perfectly on the eight tests using sensor effects. The performance on the 16 tests using actuator effects was good, but the results for the metric of how quickly it regained proper control was skewed by the fact that there was only one of the 16 tests in which proper control was maintained, and in that case the controller did not regain proper control before he test ended, hence the recorded value

of 250 iterations. With a larger pool of test results the 192 tests conducted when 2 components were being affected provides more useful results.

Using the same data summarized in the last two columns of Table 4.10., a confusion matrix of the classification results of the LSTM is shown in Table 4.11.

Table 4.11. Confusion Matrix of LSTM Classifications for Steam Boiler

	Number of Pumps Selected				
Correct Number	**5046**	**169**	**0**	**0**	1
	3817	**79904**	**243**	**0**	2
	0	**0**	**5103**	**0**	3
	0	**0**	**0**	**18468**	4
	1	2	3	4	

By way of review, model 1 is the correct choice when no component is being affected. Model 2 is the correct choice when the actuator is being affected. Model 3 is the correct choice when the sensor is being affected and model 4 is the correct choice when both are being affected.

The benefit of this testing was two-fold. First, it confirmed that a passive, data-driven, LSTM-based controller could be used to make the system under test resilient to the effects of cyber-attacks and faults. This was demonstrated by leveraging the taxonomy of effects developed in Chapter 3.

Second, this testing showed the utility of the proposed controller on a simple, linear system with redundant actuators and built in safety modes. This is a stark contrast to the test subject introduced in Chapter 5.

4.7 Chapter Summary

This chapter presented a linear system used as a proof-of-concept test subject. The proposed controller design was applied to this system. The method for generating training data using the taxonomy of effects was described. The training process was discussed and the system was tested using the various effects of faults and attacks described previously. The testing demonstrates this controller works well with the test system even in the presence of multiple simultaneous faults/attacks. Since good results were achieved on this linear system with redundant actuators, application to a more complex and challenging system is in order.

5 Test Subject: Quadruple-Tank System

5.1 Chapter Introduction

This chapter presents a detailed description of the benchmark quadruple-tank system. The application of the proposed controller to this nonlinear system is detailed. Specifics of how the taxonomy of effects is applied to the quadruple-tank system are presented. Two versions of the proposed controller are presented; single LSTM and ensemble-based). The steps for training and testing the controllers are described and the results of that testing are discussed.

5.2 Quadruple-Tank System Design

The quadruple-tank system was first introduced in 1999 as a problem for the study of multivariable control [106]. The system consists of four cylindrical tanks connected to each other through pipes as shown in Figure 5.1.

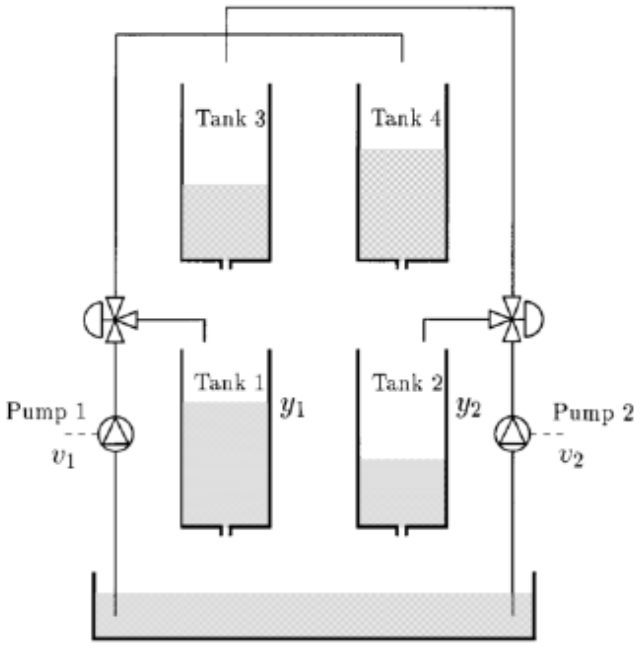

Figure 5.1. Diagram of Quadruple-Tank System [106].

The system is fed by two pumps. Specifically, tanks 1 and 4 are fed by pump 1 while tanks 2 and 3 are fed pump 2. Additionally, tank 1 is fed by the output of tank 3 and tank 2 is fed by the output of tank 4. The ratio of how much fluid flows to the upper tank or lower tank from each pump is set manually by a proportional valve at the outlet of each pump.

The function of the controller is to set the control voltage to each pump (and thereby the speed of each pump) such that the fluid level in both lower tanks is as close as possible to the preset reference level. The physical parameters of the quad-tank system [107] are shown in Table 5.1.

Table 5.1. System Parameters used in the Quadruple-Tank System.

Parameter	Description	Value
A_1, A_3	Cross-sectional area of tanks 1 and 3	28 cm^2
A_2, A_4	Cross-sectional area of tanks 2 and 4	32 cm^2
a_1, a_3	Cross-sectional area of outlet pipes 1 and 3	0.071 cm^2
a_2, a_4	Cross-sectional area of outlet pipes 2 and 4	0.057 cm2
k_c	Voltage constant	0.5 V/cm
g	Gravitational constant	981 cm/s^2

The values for these parameters were not specified in the original publication of the system. These are the values used in subsequent publications [107] and [108] which built upon the original work. The same is true for the operational parameters given in Table 5.2.

Table 5.2. Operational Parameters for the Quadruple-Tank System [107] [108].

Parameter	Description	Value
h_1^0	Initial height of liquid in tank 1	12.6 cm
h_2^0	Initial height of liquid in tank 2	13.0 cm
h_3^0	Initial height of liquid in tank 3	4.8 cm
h_4^0	Initial height of liquid in tank 4	4.9 cm
v_1^0	Initial input to pump 1	3.15 V
v_2^0	Initial input to pump2	3.15 V
k_1	Gain constant for pump 1 voltage	3.14 cm^3/Vs
k_2	Gain constant for pump 2 voltage	3.29 cm^3/Vs
γ_1	Ratio to lower tank from pump 1	0.70
γ_2	Ratio to lower tank from pump 2	0.60

The valve settings (γ_1, γ_2) determine if the majority of the water from each pump is directed to the associated lower tank or associated upper tank. The valves are set before the system runs. When the valves are set so that $(\gamma_1 + \gamma_2) < 1$ the system is said to be in a minimum phase, meaning the majority of the liquid from the pumps is going to the upper tanks. When the valves are set such that $1 \leq (\gamma_1 + \gamma_2) \leq 2$ the system is said to be in a non-minimum phase, meaning the majority of the liquid is going to the lower tanks.

5.2.1 Internal System Equations

Using these parameters, the following nonlinear equations for the rate of change in the level of fluid in each tank (where h_i is the fluid level in tank i) are presented in [106] as follows:

$$\frac{dh_1}{dt} = \frac{-a_1}{A_1}\sqrt{2gh_1} + \frac{a_3}{A_1}\sqrt{2gh_3} + \frac{\gamma_1 k_1}{A_1}v_1 \quad (1)$$

$$\frac{dh_2}{dt} = \frac{-a_2}{A_2}\sqrt{2gh_2} + \frac{a_4}{A_2}\sqrt{2gh_4} + \frac{\gamma_2 k_2}{A_2}v_2 \quad (2)$$

$$\frac{dh_3}{dt} = \frac{-a_3}{A_3}\sqrt{2gh_3} + \frac{(1-\gamma_2)k_2}{A_3}v_2 \quad (3)$$

$$\frac{dh_4}{dt} = \frac{-a_4}{A_4}\sqrt{2gh_4} + \frac{(1-\gamma_1)k_1}{A_4}v_1 \quad (4)$$

The value on the right-hand side of each equation is used to update the height of fluid in each respective tank and the output from the system (y_i) is calculated as

$$y_i = h_i k_c$$

There is no water level sensor in tank 3 or 4, therefore the water height in those tanks must be calculated if that information is required by the controller. The input to the system u_i is referenced as v_i in the equations above.

5.3 Proposed Controller for Quadruple-Tank Test Case

As applied to the quadruple-tank test system, the proposed controller design uses one LSTM classifier network for each of the pumps as shown in Figure 5.2.

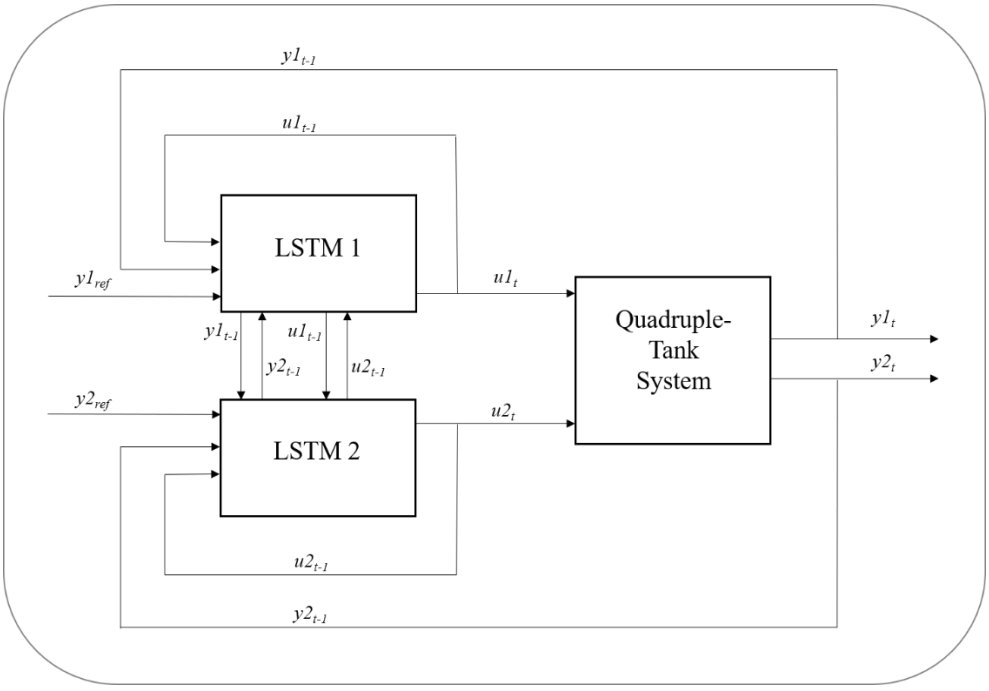

Figure 5.2. Proposed Controller with System.

5.3.1 Designing the LSTMs

In this application, the control designer determines the sequence length (time steps) of the input features. Longer sequence lengths provide more information for the network to use, but for any application there is an optimal sequence length. For the quad-tank controller a few trial runs were conducted to identify the input sequence length that resulted in the best overall performance of the controller. In the trial runs the only parameter that was changed was the sequence length. The results are shown in Table 5.3.

Table 5.3. Change in Network Performance Correlated with Sequence Length

Length	Performance
5	93.94%
10	93.38%
15	92.95%
20	92.37%
25	95.59%
30	95.93%
35	90.21%
40	90.23%

As the sequence length was increased, the results of each trial run also increased until the maximum results were achieved with a sequence length of 30. From there the results began to drop off again.

Using the sequence length of 30 identified in the trial runs, the next stage was to find the combination of hidden layers and nodes per layer that would maximize performance of the LSTM on the Quadruple-tank system data. The results of those experiments are shown in Table 5.4.

Table 5.4. Comparison of Network Performance Correlated with Network Size

Nodes	Hidden Layers								
	2	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
768	94.71	93.26	93.52	95.36	96.17	91.81	94.82	91.11	88.08
512	93.90	95.81	96.48	96.85	96.73	96.87	98.29	96.79	93.35
256	93.81	93.96	93.70	93.88	93.71	93.83	93.99	94.15	94.07
128	92.53	93.08	93.30	93.70	93.86	93.58	93.89	94.15	93.76
64	89.21	89.16	91.94	93.33	93.81	93.95	94.21	93.23	92.50

The best performance was derived from the network with eight hidden layers and 512 nodes per layer. Therefore the LSTM network is comprised of the input layer, eight hidden layers, each with 512 nodes, a fully connected 4-node layer, a softmax (dense) layer and an output layer with four output nodes – a total of 12 layers.

5.3.2 Internally Generated Models

When applied to the quadruple tank system, the proposed controller uses the LSTM to select one of several different models of the quadruple-tank system. These models are generated using the nonlinear equations described previously. Specifically, these models provide a value of the water level in each of the lower tanks. In order to better understand how the models are developed, two additional variables are introduced: y_{ir} and v_{ie}.

The variable y_{ir} is the *reported* value from the water level senor in tank i. The true value is y_i. Under normal operating conditions, $y_{ir} = y_i$, but the controller only receives y_{ir} and has no access to y_i. The variable v_{ie} is the *estimated* value of the voltage to pump i. The actual voltage to that pump is v_i in prior equations, but the controller can only estimate the pump voltage based on the last command given.

Through the remainder of this section, the discussion will refer to tank 1 and pump 1 and its associated LSTM. This done for clarity. The principles and analogous calculations also apply to tank 2 and pump 2 with its associated LSTM.

At every iteration of the control program the tank 1 sensor and pump 1 (actuator) of the quadruple-tank system could be operating normally, or the tank sensor could be affected by a fault/cyber-attack, or the pump could be affected by a fault/cyber-attack or both could be affected. For each of these four conditions a different internal model is generated. The role of the LSTM is to assess the state of the system and select one of the four models to use for the controller. The derivation of each model is described below.

Model 1: Uses the reported value of the water level sensor and is the correct model to use only when the system is operating free of any attacks or faults. Therefore, the modeled value $y_{1m} = y_{1r}$.

Model 2: Generates a value y_i under the assumption that the associated pump is experiencing an attack or fault. This means the estimated pump voltage v_{1e} is assumed to be incorrect and a new value for the modelled pump voltage needs to be calculated. This is done using equation (2) to solve for h_4

$$h_4 = \frac{\left[\frac{A_4}{a_2}\left(\frac{dh_2}{dt} + \frac{a_2}{A_2}\sqrt{2gh_2} - \frac{\gamma_2 k_2}{A_2}v_2 e\right)\right]^2}{2g}$$

The value for dh_4 can be calculated and a model value for the voltage to pump 1 (v_{1m}) is calculated by solving equation (4) for v_{1m}

$$v_{1m} = \frac{A_4\frac{dh_4}{dt} + a_4\sqrt{2gh_4}}{(1-\gamma_1)k_1}$$

This v_{1m} then replaces v_1 in equation (1) and a new dh_1 is calculated

$$\frac{dh_{1m}}{dt} = \frac{-a_1}{A_1}\sqrt{2gh_1} + \frac{a_3}{A_1}\sqrt{2gh_3} + \frac{\gamma_1 k_1}{A_1} v_1 m$$

This newly derived value of dh_1 is used to calculate the modelled value of y_{1m}.

Model 3: Generates a value y_i under the assumption that the water level sensor is experiencing an attack or fault. Since the reported value of y_{1r} would be influenced by the attack or fault, a new value is calculated using equation (1).

$$\frac{dh_{1m}}{dt} = \frac{-a_1}{A_1}\sqrt{2gh_1} + \frac{a_3}{A_1}\sqrt{2gh_3} + \frac{\gamma_1 k_1}{A_1} v_1 e$$

which is then used to calculate h_{1m} and

$$y_{1m} = h_{1m} k_c$$

Model 4: Generates a value y_{im} under the assumption that both the pump and the water level sensor are experiencing an attack and/or fault. To generate a modeled value for tank sensor 1 (y_{1m}) neither y_{1r} nor v_{1e} can be used. The model for tank 1 is then derived by first calculating the value of h_4 using equation (2).$h_4 = \frac{\left[\frac{A_4}{a_2}\left(\frac{dh_2}{dt} + \frac{a_2}{A_2}\sqrt{2gh_2} - \frac{\gamma_2 k_2}{A_2} v_2 e\right)\right]^2}{2g}$

The value for dh_4 can be calculated and a model value for v_1 is calculated by solving equation (4) for v_{1m}:

$$v_{1m} = \frac{A_4 \frac{dh_4}{dt} + a_4\sqrt{2gh_4}}{(1-\gamma_1)k_1}$$

This new value for v_{1m} is then used to calculate dh_1 using equation (1)

$$\frac{dh_{1m}}{dt} = \frac{-a_1}{A_1}\sqrt{2gh_1} + \frac{a_3}{A_1}\sqrt{2gh_3} + \frac{\gamma_1 k_1}{A_1} v_{1m}$$

y_{1m} can be calculated

$$y_{1m} = h_{1m}\, k_c$$

The same steps are followed to generate models related to the tank 2 sensor and pump 2 for use by the associated LSTM network.

5.3.3 Inputs to the LSTM

Both LSTM classifiers in the controller use the same input matrix, consisting of 27 rows and 30 columns. The 30 columns represent the consecutive time iterations while the 27 rows contain the different feature values as listed in Table 5.5.

Table 5.5. Inputs to LSTM Networks.

Input Rows	Description	Example Variables
1-2	Reported values of each sensor	y_{1r}, y_{2r}
3-4	Estimated voltage to each pump	v_{1e}, v_{2e}
5-7	Error estimated by the output of models	y_{2m}-y_{ref}, y_{3m}-y_{ref}, y_{4m}-y_{ref}
8-10	Difference in the estimated output between models	y_{2m}-y_{3m}, y_{2m}-y_{4m}, y_{3m}-y_{4m}
11-13	Absolute value of difference in change in the estimated output of models	$\lvert\Delta y_{2m}$-$\Delta y_{3m}\rvert$, $\lvert\Delta y_{2m}$-$\Delta y_{4m}\rvert$, $\lvert\Delta y_{3m}$-$\Delta y_{4m}\rvert$
14-16	Change in the estimated output of models	Δy_{2m}, Δy_{3m}, Δy_{4m}
17-19	Value of the model outputs	y_{2m}, y_{3m}, y_{4m}
20-27	Binary logic statements comparing the mean squared error between samples of the models	MSE($y_{2m}(t\text{-}19{:}t)$-$y_{4m}(t\text{-}19{:}t)$)>0.000138 AND MSE($y_{3m}(t\text{-}19{:}t)$- $y_{4m}(t\text{-}19{:}t)$) > 0.000125

A comment on the last entry of Table 5.5. As the controller runs, the last 20 values of the output of each model ($y_{im}(t-19:t)$) are stored in an array variable. In the binary logic statements, the mean squared difference of comparing each array pairwise is taken compared to a set value. If both parts of this statement are true, this statement provides a 1

to the input matrix. While these statements alone cannot provide sufficient input to control the system, they do help in discriminating between two very similar conditions.

5.4 Effects of Faults and Cyber-Attacks

Before discussing how the LSTMs were trained and tested, it is necessary to show how the effects of cyber-attacks and faults on the quad-tank system components are addressed. Recall the taxonomy of representative sensor effects described in Chapter 3 which is repeated here in Tables 5.6.

Table 5.6. Fundamental Effects on Sensors

	Sensor Effect Descriptions		
1	Increased Value	$\hat{y}_{t+z} = \alpha + y_{t+z}\ OR\ \breve{\alpha}(y_{t+z})$	Where $\alpha > 0$ and $\breve{\alpha} > 1$
2	Decreased Value	$\hat{y}_{t+z} = \beta + y_{t+z}\ OR\ \breve{\beta}(y_{t+z})$	Where $\beta < 0$ and $0 < \breve{\beta} < 1$
3	Stochastic Value	$\hat{y}_{t+z} = y_{t+z} + \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(z-\mu)^2}{2\sigma}}$	Where μ is mean and σ is the standard deviation of additive Gaussian noise
4	Cyclical Value	$\hat{y}_{t+z} = y_{t+z} + \gamma\ sin(\delta\pi z)$	Where γ determines the magnitude of an additive cyclical effect and δ determines the frequency
5	Fixed Value	$\hat{y}_{t+z} = \varepsilon$	Where ε is the fixed value
6	No Value	$\hat{y}_{t+z} = \{\}$	

Applied to the sensors of the quadruple-tank system, these effects are applied as described below and summarized in Table 5.7.

1. Increased Value effect – This effect is modeled as an abrupt, multiplicative function with $\hat{\alpha}$ being randomly selected from a uniform distribution with *values* on the interval (1.00, 3.00).

2. Decreased Value effect – This effect is modeled as an abrupt, multiplicative function with $\hat{\beta}$ being randomly selected from a uniform distribution with *values* on the interval (0.15, 0.90).

3. Stochastic Value effect – This effect is modeled as an abrupt, additive function with μ set to zero and σ being randomly selected from a uniform distribution with *values* on the interval (0.05, 0.55).

4. Cyclical Value effect – This effect is modeled as an abrupt, additive function with γ being randomly selected from a uniform distribution with *values* on the interval (0.10, 1.10) and δ being randomly selected from a uniform distribution with *values* on the interval (0.015, 1.015).

5. Fixed Value effect – This effect is modeled as an abrupt, function with ε being randomly selected from a uniform distribution with *values* on the interval (0.00, 20.00).

6. No Value effect - This effect is modeled as an abrupt, function with the value of y_{t+z} being set as Not-a-Number (*NaN*).

Two additional effects are modeled to examine the performance of the controller if the quadruple tank system sensors are subjected to incipient effects.

7. Increased Value effect – This effect is modeled as an incipient, additive and multiplicative function with α being randomly selected from a uniform distribution with *values* on the interval (0.00, 1.00) and the effect being applied using the equation

$$\hat{y}_{t+z} = y_t + .005\alpha y_{t+z}$$

8. Decreased Value effect – This effect is modeled as an incipient, additive and multiplicative function with β being randomly selected from a uniform distribution with *values* on the interval (0.0, 1.0) and the effect being applied using the equation

$$\hat{y}_{t+z} = y_t - .005\beta y_{t+z}$$

These eight sensor effect models are summarized in Table 5.7. The *Effect Number* and *Effect Value* fields will be used when discussing test results. The six representative actuator effects are described in Table 5.8.

Table 5.7. Summary of Sensor Effects for Quadruple-Tank System

Effect Type	Effect Number	Effect Value	Add, Mult, Replace	Abrupt or Incipient
Increased	1	(1.00, 3.00)	Mult	Abrupt
Decreased	2	(0.15, 0.90)	Mult	Abrupt
Stochastic	3	(0.05, 0.55)	Add	Abrupt
Cyclical	4	(0.10, 1.10) (0.015, 1.015)	Add	Abrupt
Fixed	5	(0.00, 20.00)	Replace	Abrupt
No Value	6	(NaN)	Replace	Abrupt
Increased	7	(0.00, 1.00)	Add & Mult	Incipient
Decreased	8	(0.00, 1.00)	Add & Mult	Incipient

Table 5.8. Fundamental Effects on Actuators

	Actuator Effect Descriptions		
1	Increased Value	$\hat{u}_{t+z} = \alpha + u_{t+z} \;\; OR \;\; \breve{\alpha}(u_{t+z})$	Where $\alpha > 0$ and $\breve{\alpha} > 1$
2	Decreased Value	$\hat{u}_{t+z} = \beta + u_{t+z} \;\; OR \;\; \breve{\beta}(u_{t+z})$	Where $\beta < 0$ and $0 < \breve{\beta} < 1$
3	Stochastic Value	$\hat{u}_{t+z} = u_{t+z} + \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(z-\mu)^2}{2\sigma}}$	Where μ is mean and σ is the standard deviation of Gaussian noise
4	Cyclical Value	$\hat{u}_{t+z} = u_{t+z} + \gamma \, sin(\delta \pi z)$	Where γ sets the magnitude of a cyclical effect and δ determines the frequency
5	Fixed Value	$\hat{u}_{t+z} = \varepsilon$	Where ε is the fixed value
6	No Value	$\hat{u}_{t+z} = \{\,\}$	

Applied to the actuators of the quadruple-tank system, these effects are applied as described below and summarized in Table 5.9.

1. Increased Value effect – This effect is modeled as an abrupt, multiplicative function with $\hat{\alpha}$ being randomly selected from a uniform distribution with *values* on the interval (1.05, 1.50). This effect leaves the actuator controllable.

2. Decreased Value effect – This effect is modeled as an abrupt, multiplicative function with $\hat{\beta}$ being randomly selected from a uniform distribution with *values* on the interval (0.15, 0.90). This effect leaves the actuator controllable.

3. Stochastic Value effect – This effect is modeled as an abrupt, additive function with μ set to zero and σ being randomly selected from a uniform distribution with *values* on the interval (0.05, 0.55). This effect leaves the actuator controllable.

4. Cyclical Value effect – This effect is modeled as an abrupt, additive function with γ being randomly selected from a uniform distribution with *values* on the interval (0.50, 3.50) and δ being randomly selected from a uniform distribution with *values* on the interval (0.05, 0.55). This effect leaves the actuator controllable.

5. Fixed Value effect – This effect is modeled as an abrupt, function with ε being randomly selected from a uniform distribution with *values* on the interval (0.00, 10.00). This effect leaves the actuator uncontrollable.

6. No Value effect – This effect is modeled as an abrupt, function with the value of y_{t+z} being set as Not-a-Number (*NaN*). This effect leaves the actuator uncontrollable.

As stated previously, two additional effects are also modelled.

7. Increased Value effect – This effect is modeled as an incipient, additive and multiplicative function with α being randomly selected from a uniform distribution with *values* on the interval (0.00, 1.00) and the effect being applied using the equation

$$\hat{u}_{t+z} = (u_t + .005\alpha u_{t+z})\frac{2u_{t+z}}{3}$$

8. Decreased Value effect – This effect is modeled as an incipient, additive and multiplicative function with β being randomly selected from a uniform distribution with *values* on the interval (0.00, 1.00) and the effect being applied using the equation

$$\hat{u}_{t+z} = (u_t - .005\beta u_{t+z})\frac{2u_{t+z}}{3}$$

These eight actuator effect models are summarized in Table 5.9.

Table 5.9. Summary of Actuator Effects for Quadruple-Tank System

Effect Type	Effect Number	Effect Value	Add, Mult or Replace	Abrupt or Incipient	Controllable
Increased	1	(1.05, 1.50)	Mult	Abrupt	Yes
Decreased	2	(0.15, 0.90)	Mult	Abrupt	Yes
Stochastic	3	(0.05, 0.55)	Add	Abrupt	Yes
Cyclical	4	(0.50, 3.50) (0.05, 0.55)	Add	Abrupt	Yes
Fixed	5	(0.00, 10.00)	Replace	Abrupt	No
No Value	6	(NaN)	Replace	Abrupt	No
Increased	7	(0.00, 1.00)	Add & Mult	Incipient	Yes
Decreased	8	(0.00, 1.00)	Add & Mult	Incipient	Yes

The *Effect Number* and *Effect Value* fields will be used when discussing test results. Since the Fixed Value (#5) and the No Value (#6) effects leave the actuator uncontrollable and the quadruple-tank system design does not include redundant actuators, these two effects will not be included in the training or testing of the controller for the quadruple tank system.

5.5 Training the LSTM Networks

The following parameters were used to train the LSTM for this controller in MATLAB®:

Solver: ADAptive Moment estimation (ADAM)
Execution Environment: auto (defaults to GPU if one is present)
Initial Learning Rate: 0.0002
Learn Rate Schedule: piecewise
Learn Rate Drop Factor: 0.5
Learn Drop Rate: 2 (epochs)
Maximum Epochs: 20

In order to expedite the training process of the LSTM, the training inputs are compiled into "minibatches". The larger the minibatch, the quicker the training is accomplished. If the minibatch size is too large, the network will overfit to the training data and not perform well in testing or actual use. Based on the size of the training data set, the minibatch size for this application is 512 inputs.

Another important parameter in training the LSTM is the learning rate. Larger learning rates help the values of the LSTM weights to converge more quickly, but risk possibly settling in local minimum rather than the global minimum. Lower learning rates are more likely to find the global minimum but can take a very long time to get there. MATLAB® allows the designer to vary the learning rate during training. In this application, the initial learning rate is set at 0.002 and is cut by 50% every 2 epochs. Thus, after 20 training epochs the training rate is dropped to 0.000125.

5.5.1 Generating Training Data

The data used to train the proposed controller was generated by building a computer simulation of the quadruple-tank system in MATLAB®. This test subject system was controlled by a PID controller which was fed the output from the correct model so it always controlled the system correctly.

Table 5.10. Numbers and Combinations of Data Generation for Quad-Tank System.

	Components			
	A	**S**	**Runs per Effect**	**Total Runs**
Effects	6	0	20	120
	0	8	15	120
	6	8	5	240
	Total			**480**

The simulation was run, iteratively introducing representative values of each of the effects of faults and cyber-attacks on each component. After each component has been run with each of the different effects (8 for the sensor and 6 for the actuator), the process is repeated with both components being simultaneously affected (48 different combinations).

Four times as many data generation simulations were run with actuator effects, and three times as many with sensor effects, compared to those with two components affected. This was done to provide an approximately equal number of samples for single component effects as multiple component effects.

Each data generation simulation ran for 500 iterations with the component effect starting at iteration number 250. This was done to ensure the system was operating in a normal steady state the effect begins.

A total of 240,000 inputs were generated using them method described above. Each input was a 27x30 matrix.

5.5.2 Data Preprocessing

Before the data from the simulation runs can be used to train the LSTM networks there are several preprocessing steps it must go through.

The data must first be normalized to a range of $(-1,1)$. With the data normalized, 20% of the data was randomly selected and set aside for future testing. This left 198,000 data samples in the training set.

A key step in data preprocessing for LSTMs is data balancing. If a significant portion of the data has a correct classification of “1” and only a comparative few with correct classification of 2, 3 or 4, the LSTM will learn it can get good results by simply classifying all inputs as “1”. Therefore, the data is balanced by identifying which

classification group was most represented in the data set and duplicating data samples from each of the other classification groups until all four groups had the same number of data samples. This step increased the training set to 344,000 samples evenly distributed across the four classification groups.

The LSTMs were then trained with this data using the training parameters identified at the beginning of this section

5.6 Testing Methodology for Proposed Controller

The quadruple-tank system poses some interesting and difficult control challenges. Testing with this non-linear system is intended to find the limits of both the proposed and the comparison controllers in the presence of the various cyber-attacks and component faults. Unlike the steam boiler, discussed in Chapter 4, this system has no redundant components nor built-in safety features. The actuators are variable speed pumps which make the control requirements much more complex.

The method for testing the controller is very similar to the method for generating the training data. For all testing, the system under test (SUT) and the associated control algorithm are realized as MATLAB® scripts. The tests are performed on an Intel® Core i7 CPU platform with 32 GB of RAM running Windows 10. Each test run is conducted by running the script for 500 iterations. Prior to each run the attack/fault effects(s) (described previously) is/are set to be activated at iteration number 250. Each of the eight sensor effects were applied on individual runs, then each of the six actuator effects were applied on individual runs. Finally, the sensor and actuator effects were applied simultaneously on the last 48 test runs, with each run using a different combination of the different effects. In

all of the test runs the value of the variable(s) were randomly selected, from the uniform range described previously, prior to the beginning of the test run.

5.7 Test Results

A total of 63 test runs were completed for the test; 1 with no effects applied, 8 with a sensor effect applied, 6 with an actuator effect applied and 48 with both a sensor and an actuator effect applied. For ease in understanding the complete results of the testing as presented in Table 5.11., the column headers are described below.

Test Run Number – Sequential number of the specific test run

Pump 1 Effect – Actuator effect applied in the test run. The effect numbers refer to the second column of Table 5.9.

Effect Value – Refers to the randomly selected variable value for that effect on that test run. The range of possible actuator effect values are given in column 3 of Table 5.9.

Sensor 1 Effect - Sensor effect applied in the test run. The effect numbers refer to the second column of Table 5.7.

Effect Value – Refers to the randomly selected variable value for that effect on that test run. The range of possible sensor effect values are given in column 3 of Table 5.7.

Components Affected – Total number of sensors and/or actuators affected in the test run.

Catastrophic Failure - Boolean indicator is 1 if the actual water level in tank 1 dropped to 0 or exceeded 25 cm (maximum height of the tank). Otherwise it is 0.

Maintain Proper Control – Boolean indicator is 1 if the actual water level in tank 1 did not vary by more than ±1% from the reference value after the effect was initiated. Otherwise it is 0.

Iterations to Regain Proper Control – If the previous column entry in the test run is a 0, this entry shows how many iterations occurred from the time the water level in tank 1 went outside the ±1% until it was brought back into that range.

Iterations to Correct Model – Measure of how many controller program iterations occurred from the onset of the effect(s) until the controller selected the correct model.

Iterations Using Correct Model (%) – Measure of the percentage of iterations during the test run in which the correct model was selected by the controller.

Complete results of the test runs are shown in Table 5.11.

Table 5.11. Test Results of Proposed Controller.

Test Run No.	Pump 1 Effect	Effect Value	Sensor 1 Effect	Effect Value	Components Affected	Catastrophic Failure	Maintain Proper Control?	Iterations to Regain Proper Control	Iterations to Correct Model	Iterations Using Correct Model %
1			0		0	0	1		0	100.00%
2			1	2.96	1	0	1		1	99.67%
3			2	0.56	1	0	1		1	99.67%
4			3	0.48	1	0	1		1	99.67%
5			4	0.71	1	0	1		1	99.67%
6			5	15.72	1	0	1		1	99.67%
7			6	*	1	0	1		1	99.67%
8			7	0.05	1	0	1		18	94.00%
9			8	0.82	1	0	1		7	97.67%
10	1	1.44			1	0	0	1	2	99.33%
11	2	0.73			1	0	1		2	99.33%
12	3	4.48			1	0	1		2	99.33%
13	4	2.47			1	0	1		3	99.00%
14	7	0.62			1	0	0	1	3	99.00%
15	8	0.87			1	0	0	1	3	99.00%
16	1	1.14	1	2.21	2	0	1		4	98.67%
17	1	1.50	2	0.18	2	0	0	3	3	99.00%
18	1	1.11	3	1.16	2	0	1		4	98.67%
19	1	1.43	4	0.69	2	0	0	2	2	99.33%
20	1	1.45	5	11.75	2	0	0	2	2	99.33%
21	1	1.20	6	*	2	0	0	1	4	98.67%

22	1	1.50	7	0.21	2	0	0	2	2	99.33%
23	1	1.18	8	0.83	2	0	1		3	99.00%
24	2	0.30	1	2.53	2	0	0	30	3	99.00%
25	2	0.61	2	0.87	2	0	0	10	2	99.33%
26	2	0.53	3	2.44	2	0	0	15	2	99.33%
27	2	0.52	4	0.35	2	0	0	15	2	99.33%
28	2	0.23	5	3.07	2	0	0	36	3	99.00%
29	2	0.80	6	*	2	0	0	2	4	98.67%
30	2	0.69	7	0.01	2	0	0	4	2	99.33%
31	2	0.76	8	0.25	2	0	0	2	3	99.00%
32	3	1.08	1	2.31	2	0	0	1	3	99.00%
33	3	4.91	2	0.67	2	0	1		4	98.67%
34	3	1.88	3	2.40	2	0	1		4	98.67%
35	3	2.45	4	0.92	2	0	0	2	3	99.00%
36	3	1.23	5	15.41	2	0	1		5	98.33%
37	3	1.86	6	*	2	0	0	1	4	98.67%
38	3	4.36	7	0.13	2	0	1		4	98.67%
39	3	4.50	8	0.73	2	0	0	1	3	99.00%
40	4	1.77	1	2.21	2	0	0	1	4	98.67%
41	4	2.79	2	0.89	2	0	1		4	98.67%
42	4	0.10	3	2.57	2	0	0	3	5	98.33%
43	4	2.96	4	0.96	2	0	1		10	96.67%
44	4	0.84	5	13.72	2	0	1		250	16.33%
45	4	2.48	6	*	2	0	1		17	94.33%
46	4	0.99	7	0.70	2	0	1		6	98.00%
47	4	1.84	8	0.97	2	0	1		4	98.67%
48	7	0.62	1	2.15	2	0	0	3	4	98.67%
49	7	0.65	2	0.25	2	0	0	3	4	98.67%
50	7	0.08	3	2.12	2	0	0	2	3	99.00%
51	7	0.84	4	0.03	2	0	0	2	3	99.00%
52	7	0.54	5	6.75	2	0	0	3	4	98.67%
53	7	0.80	6	*	2	0	0	3	4	98.67%
54	7	0.76	7	0.44	2	0	0	2	3	99.00%
55	7	0.87	8	0.82	2	0	0	1	2	99.33%
56	8	0.55	1	1.33	2	0	0	2	3	99.00%
57	8	0.63	2	0.83	2	0	0	3	4	98.67%
58	8	0.34	3	1.82	2	0	0	2	3	99.00%
59	8	0.10	4	0.26	2	0	0	3	4	98.67%
60	8	0.49	5	0.03	2	0	0	3	4	98.67%
61	8	0.27	6	*	2	0	0	3	4	98.67%
62	8	0.44	7	0.72	2	0	0	2	3	99.00%
63	8	0.82	8	0.24	2	0	0	1	2	99.33%

Of the 63 test runs, there was only a single test (#44) in which the proposed controller did not maintain or regain proper control of the system after it was affected. The test results are summarized in Table 5.12.

Table 5.12. Test Result Summary for Proposed Controller

Test Runs	**Component Affected**	**Cat. Fail**	**Proper Control**	**Iterations to Regain Proper Control**		**Iterations to Correct Model**		**Iterations Using Correct Model %**	
#	*A/S/Both*	*%*	*%*	*Mean*	*Median*	*Mean*	*Median*	*Mean*	*Median*
6	Actuator	0	50.00	1.00	1.00	2.50	2.50	99.00	99.00
8	Sensor	0	100.00	*	*	3.88	1.00	98.67	98.67
48	Both	0	27.08	4.29	2.00	7.80	3.00	97.37	98.50
Total	**-**	**0**	**39.68**	**4.58**	**2.00**	**7.54**	**3.00**	**97.48**	**99.00**

Using the same data summarized in the last two columns of Table 5.12., a confusion matrix of the classification results of the LSTM is shown in Table 5.13.

Table 5.13. Confusion Matrix of LSTM Classifications in Proposed Controller

	Model Selected				
Correct Model	**3338**	**0**	**0**	**0**	1
	10	**1491**	**3**	**2**	2
	30	**0**	**1977**	**1**	3
	74	**0**	**356**	**11618**	4
	1	2	3	4	

By way of review, model 1 is the correct choice when no component is being affected. Model 2 is the correct choice when the actuator is being affected. Model 3 is the correct choice when the sensor is being affected and model 4 is the correct choice when

both are being affected. These test results are discussed and analyzed in more detail in Chapter 6.

5.8 Ensemble Variant of Network-Based Controller

To this point the discussion of the design, training and testing of the proposed controller has considered a single LSTM network assigned to each actuator in the system under control. As discussed in Chapter 3, there are circumstances in which an ensemble of smaller LSTM networks may be preferable to a single larger network.

For use as a controller for the quadruple-tank system, we found an ensemble of four smaller LSTM networks (2 hidden layers with 512 nodes each), operated slightly faster a single larger network. It is also possible to extract out a small increase in classification accuracy.

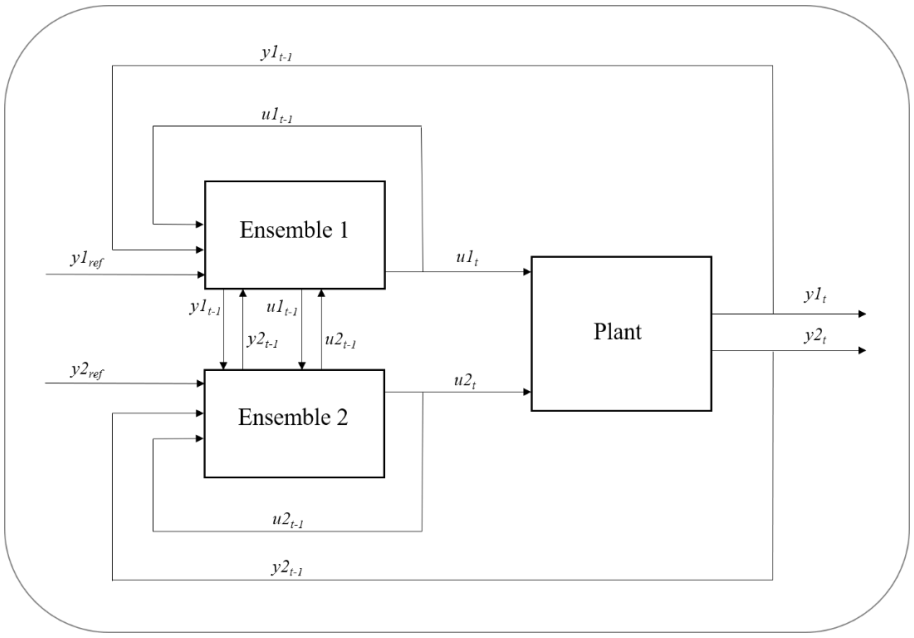

Figure 5.3. Quadruple Tank System with Ensemble-Based Controller

From a design perspective, the differences between a controller with ensembles vs single networks are comparatively minor. Figure 5.3 depicts the quadruple-tank system (plant) with two ensembles of LSTM networks as opposed to the two single networks previously shown in Figure 5.1.

Within the controller, the logic flow in an ensemble-based controller is very similar to that of single network-based controller.

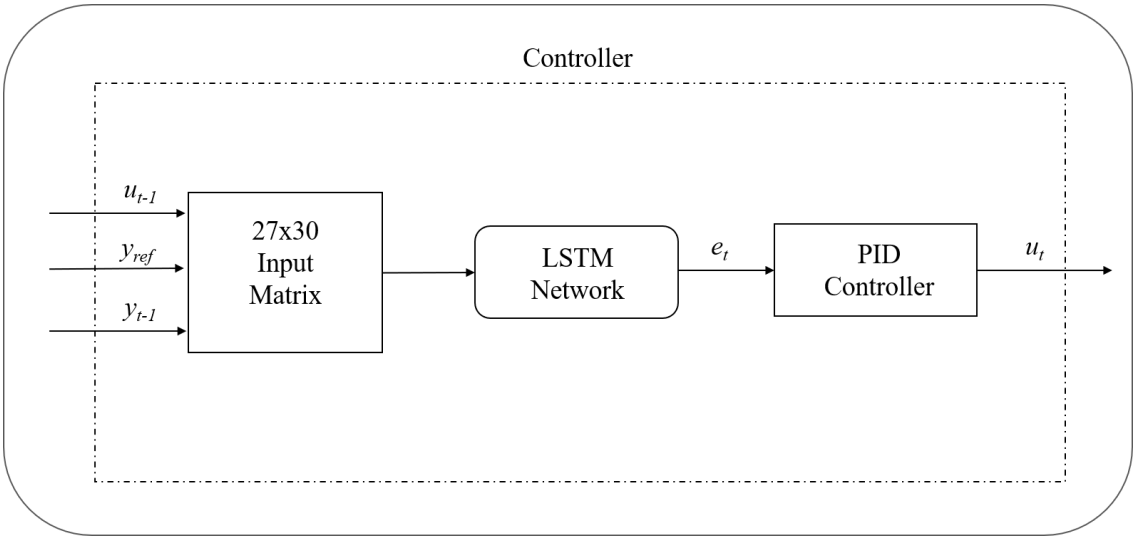

Figure 5.4. Logic Flow through a Single Network-Based Controller

Comparing the flow shown in Figure 5.4 to that shown in Figure 5.5, the ensemble requires a resolver to take the output from the four networks in the ensemble and provide a single output to the PID controller.

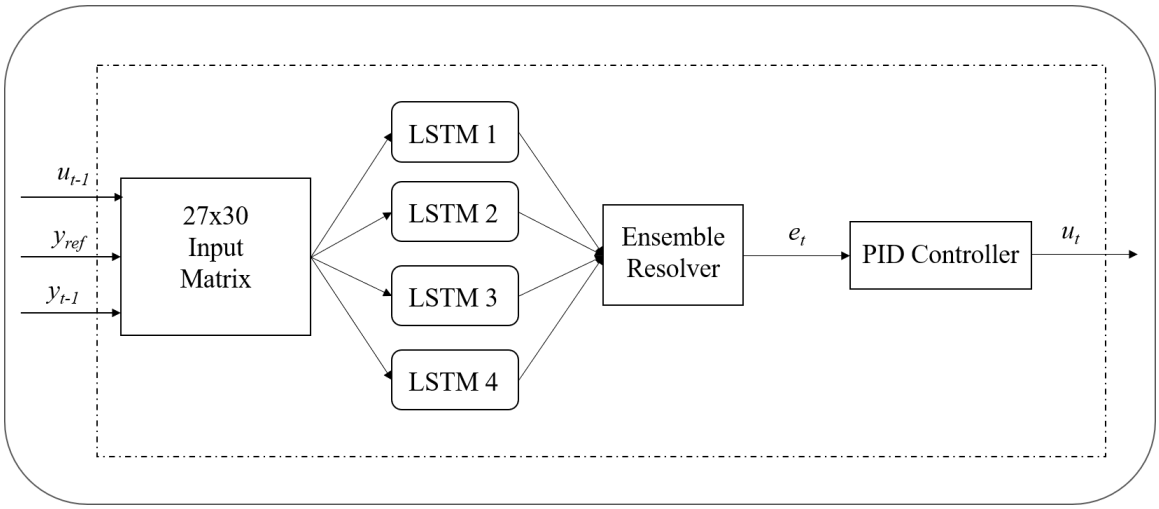

Figure 5.5. Logic Flow within an Ensemble-Based Controller.

To train the LSTMs in the ensemble the training data was divided into fourths (86,000 samples each) so that each LSTM would be trained with different data than the others.

After the LSTMs were each trained, the various options for resolving the LSTM outputs to a single solution were also trained and tested. As discussed in Chapter 3, the options considered were a majority vote (and a unanimous vote variation), a kNN network, Multiple Linear Regression and MATLAB's built-in ensemble Bagging/Boosting program. The results of testing each of these is summarized in Table 5.14.

Table 5.14. Test Results for Ensemble Resolution Options.

Method	**Accuracy**
Majority Vote	93.82%
Unanimous Vote	95.24%
k Nearest Neighbor	94.93%
Multiple Linear Regression	91.97%
Bagging/Boosting	97.43%

It should be noted that the accuracy score of the Majority Vote and Unanimous Vote methods reflected above are based only on the number inputs for which there was a majority or unanimous input to select. For those inputs that did not have a majority or unanimous answer to select, no model was selected. This can be illustrated by the results shown in the confusion matrix for the Unanimous Vote method in Table 5.15.

Table 5.15. Confusion Matrix of Unanimous Vote Ensemble Resolver.

		Model Selected				
Correct Model	**0**	**2259**	**0**	**0**	**0**	1
	11	**8**	**986**	**0**	**0**	2
	13	**16**	**0**	**1177**	**0**	3
	1447	**124**	**265**	**7**	**4187**	4
	none	1	2	3	4	

In Table 5.15, the “none” column shows the times there was no unanimous input for the resolver to select. In these cases another means of deciding a single result is required.

If all of the LSTMs in an ensemble select the same model, the output of that model is provided to the PID controller. If a unanimous decision is not reached, then a bagging/boosting classifier is activated to make the selection. The confusion matrix for the Bagging/Boosting algorithm is presented in Table 5.16.

Table 5.16. Confusion Matrix of Bagging/Boosting Ensemble Resolver.

	Model Selected				
Correct Model	**2259**	**0**	**0**	**0**	1
	9	**995**	**1**	**0**	2
	18	**0**	**1188**	**0**	3
	40	**123**	**0**	**5788**	4
	1	2	3	4	

Given that the Bagging/Boosting algorithms showed the best accuracy of the five methods considered, it would be reasonable to simply use it as the ensemble resolver. However, in this application, combining the Unanimous Vote and Bagging Boosting algorithms provided slightly improved accuracy over the Bagging/Boosting algorithm alone. The confusion matrix for two combined is shown in Table 5.17.

Table 5.17. Confusion Matrix for Unanimous Vote with Bagging/Boosting Resolver.

	Model Selected				
Correct Model	**2259**	**0**	**0**	**0**	1
	11	**993**	**1**	**0**	2
	16	**0**	**1190**	**0**	3
	46	**112**	**66**	**5806**	4
	1	2	3	4	

The two methods combined provide a classification accuracy of 97.60% which is slightly better than the 97.43% accuracy of the Bagging/Boosting algorithm alone.

5.8.1 Testing the Ensemble-Based Controller

The testing for the ensemble-based controller was conducted exactly as the testing for the single network-based controller. Complete results of the test runs is shown in Table 5.18, and a summary is provided in Table 5.19.

Table 5.18. Test Results of Proposed Controller Using Ensembles.

Test Run No.	Pump 1 Effect	Effect Value	Sensor 1 Effect	Effect Value	Components Affected	Catastrophic Failure	Maintain Proper Control?	Iterations to Regain Proper Control	Iterations to Correct Model	Iterations Using Correct Model %
1					0	0	1		0	100.00%
2			1	2.96	1	0	1		2	99.33%
3			2	0.56	1	0	1		3	99.00%
4			3	0.48	1	0	1		3	99.00%
5			4	0.71	1	0	1		3	99.00%
6			5	15.72	1	0	1		3	99.00%
7			6	*	1	0	1		3	99.00%
8			7	0.05	1	0	1		20	93.33%
9			8	0.82	1	0	1		9	97.00%
10	1	1.44			1	0		1	2	99.33%
11	2	0.73			1	0	1		2	99.33%
12	3	4.48			1	0	1		5	98.33%
13	4	2.47			1	0	1		5	98.33%
14	7	0.62			1	0		1	5	98.33%
15	8	0.87			1	0		1	5	98.33%
16	1	1.14	1	2.21	2	0	1		1	99.67%
17	1	1.50	2	0.18	2	0		2	1	99.67%
18	1	1.11	3	1.16	2	0	1		1	99.67%
19	1	1.43	4	0.69	2	0		2	1	99.67%
20	1	1.45	5	11.75	2	0		2	1	99.67%
21	1	1.20	6	*	2	0	1		1	99.67%
22	1	1.50	7	0.21	2	0		16	13	95.67%
23	1	1.18	8	0.83	2	0		14	14	95.33%
24	2	0.30	1	2.53	2	0		30	1	99.67%
25	2	0.61	2	0.87	2	0		10	1	99.67%
26	2	0.53	3	2.44	2	0		15	1	99.67%
27	2	0.52	4	0.35	2	0		15	1	99.67%
28	2	0.23	5	3.07	2	0		36	1	99.67%
29	2	0.80	6	*	2	0	1		1	99.67%

30	2	0.69	7	0.01	2	0		23	23	92.33%
31	2	0.76	8	0.25	2	0		23	24	92.00%
32	3	1.08	1	2.31	2	0	1		1	99.67%
33	3	4.91	2	0.67	2	0	1		1	99.67%
34	3	1.88	3	2.40	2	0	1		1	99.67%
35	3	2.45	4	0.92	2	0	1		1	99.67%
36	3	1.23	5	15.41	2	0	1		1	99.67%
37	3	1.86	6	*	2	0	1		1	99.67%
38	3	4.36	7	0.13	2	0		13	15	95.00%
39	3	4.50	8	0.73	2	0		14	14	95.33%
40	4	1.77	1	2.21	2	0		2	1	99.67%
41	4	2.79	2	0.89	2	0	1		1	99.67%
42	4	0.10	3	2.57	2	0		3	2	99.33%
43	4	2.96	4	0.96	2	0	1		1	99.67%
44	4	0.84	5	13.72	2	0	1		1	99.67%
45	4	2.48	6	*	2	0	1		1	99.67%
46	4	0.99	7	0.70	2	0	1		6	98.00%
47	4	1.84	8	0.97	2	0		3	14	95.33%
48	7	0.62	1	2.15	2	0		1	1	99.67%
49	7	0.65	2	0.25	2	0		1	1	99.67%
50	7	0.08	3	2.12	2	0		1	1	99.67%
51	7	0.84	4	0.03	2	0		1	1	99.67%
52	7	0.54	5	6.75	2	0		1	1	99.67%
53	7	0.80	6	*	2	0		1	1	99.67%
54	7	0.76	7	0.44	2	0		15	15	95.00%
55	7	0.87	8	0.82	2	0		1	2	99.33%
56	8	0.55	1	1.33	2	0		1	1	99.67%
57	8	0.63	2	0.83	2	0		1	1	99.67%
58	8	0.34	3	1.82	2	0		1	1	99.67%
59	8	0.10	4	0.26	2	0		1	1	99.67%
60	8	0.49	5	0.03	2	0		1	1	99.67%
61	8	0.27	6	*	2	0		1	1	99.67%
62	8	0.44	7	0.72	2	0		14	15	95.00%
63	8	0.82	8	0.24	2	0		1	2	99.33%

Table 5.19. LSTM Ensemble Test Result Summary.

Test Runs	**Component Affected**	**Cat. Fail**	**Proper Control**	**Iterations to Regain Proper Control**		**Iterations to Correct Model**		**Iterations Using Correct Model %**	
#	*A/S/Both*	*%*	*%*	*Mean*	*Median*	*Mean*	*Median*	*Mean*	*Median*
6	Actuators	0	50.00	1.00	1.00	4.00	5.00	98.67	98.33
8	Sensors	0	100.00	*	*	5.75	3.00	98.08	99.00
48	Both	0	31.25	8.06	2.00	4.04	1.00	98.65	99.67
Total	**-**	**0**	**56.25**	**7.47**	**2.00**	**4.19**	**1.00**	**98.60**	**99.67**

Using the same data summarized in the last two columns of Table 5.19 a confusion matrix of the classification results of the LSTM ensemble is shown in Table 5.20.

Table 5.20. Confusion Matrix of LSTM Ensemble Classifications.

	Model Selected				
Correct Model	**3338**	**0**	**0**	**0**	1
	12	**1482**	**0**	**12**	2
	30	**0**	**1962**	**16**	3
	71	**123**	**0**	**11854**	4
	1	2	3	4	

Comparing the summarized test results in Table 5.12 to those in 5.19, we see the ensemble-based controller showed slightly improved performance over the single LSTM-based controller. The ensemble base controller maintained proper control of the system during the tests in 56.25% of the runs where the single LSTM controller did so in only 39.68% of the tests. Additionally, the single LSTM controller had a mean correct model selection rate of 97.48% while the ensemble-based controller's mean selection rate was 98.60%. These are just two highlights in the comparison of these two controllers. The performance of these controllers will be compared in more depth in Chapter 6.

5.9 Computational Overhead

Parameters

The computational parameters of an LSTM network are weights (input and recurrent) at each hidden layer of the network as well as the biases at each cell. The number

of parameters is determined by the number inputs to the layer, the number of outputs from the layer as well as the hidden state (h) and cell state (c) described in section 3.5.1. There are four different sets of weights that are used in LSTM operations. Therefore, the number of parameters at a layer with m inputs and n outputs is calculated as

$$4n(m+n+1)$$

Consider the single large LSTM used to control the quadruple-tank system. The input to that network has 27 nodes (one for each input feature), each of the eight hidden layers contains 512 nodes, there is a softmax layer and 4 output nodes.

This means the first hidden layer has $4\times512(27+512+1)=1,105,920$ parameters. The second through the eighth hidden layers each have $4\times512(512+512+1)=2,099,200$ parameters. This makes a total of 15,800,320 parameters for the network.

In comparison, the ensemble of smaller networks was made up of four LSTMs each with two hidden layers of 512 nodes each. Each of those networks had the same 1,105,920 parameters for the first hidden layer, 2,099,200 parameters in the second layer. As a result, each of the smaller networks had 3,205,120 parameters for a total of 12,820,480 parameters in the ensemble.

Memory Requirements

Given the number of parameters for both the large LSTM and the LSTM ensemble, we can estimate the memory requirements for each. Assuming 64-bit processing and assuming each of the parameters in the LSTM networks is a double precision floating point number, each parameter will require 8 bytes of memory. From this, the large LSTM is estimated to be 128MB and the ensemble of LSTMs is estimated to be 103MB. Recall that

as applied to the quadruple-tank system, the controller would employ two LSTMs, one for each actuator. This doubles the estimated memory requirement for each.

During operation, the controller program needs to capture and store past values of input features. With a segment length of 30 time steps and 27 features, this temporal data and the basic controller algorithm can bring the minimum memory requirement up to 258MB for the large LSTM controller and 208MB for the LSTM ensemble controller. These estimates all assume the controller program runs on a CPU. If the controller computer is equipped with a GPU, the memory requirements could easily jump to over 1.5 GB because of the wider vector paths used.

The amount of RAM available on current industrial computers (process controllers) is discussed later in this section.

Multiply and Accumulate Operations

To better appreciate the computational capability required to operate a large LSTM network, it is useful to consider the Multiply and Accumulate (MAC) operations required in its use. In [109], the authors describe how the number of MAC operations in an LSTM are estimated. In short, matrix to vector multiplications dominate the number of computations made in an LSTM as well as the number of parameters. The number of MAC operations in a layer of an LSTM with m inputs and n outputs is given by

$$sn[4(n+m)+3]$$

where s is the segment length of the input features.

Thus, the large LSTM network used in this dissertation performs 33,162,240 MAC operations in the first layer and 62,960,640 in the second through eighth layers. This is a total of 473,886,720 MAC operations at each iteration of the program.

Similarly, in the ensemble of smaller networks each LSTM would execute the 32,486,400 MAC operations in the first layer and the 62,960,640 operations in the second layer. This makes a total of 96,122,880 operations per network or 384,491,520 operations in the entire ensemble. The computational complexity of the LSTM is then $O(n^2+n)$.

As configured for the quadruple-tank system, each actuator requires its own LSTM in the proposed controller design. Therefore, the large LSTM controller for the quadruple-tank system uses two networks and would execute approximately 950 million MAC operations per iteration of the control algorithm. In the case of the LSTM ensemble controller, two ensembles would be used and the control algorithm would execute over 760 million MAC operations per iteration.

This large number of MAC operations per iteration of the control program brings to mind the question of what the sampling rate of the controller can be in this application. Across the industrial controls community, the lower bound for sampling rates varies depending on the type of process under control. The general rules of thumb are; 1 second or faster for flow loops (processes) and 1-2 seconds for pressure processes. These processes react quickly to changes in controller output. Temperature control processes react more slowly to controller outputs so sampling every 2-5 seconds is the standard for those processes.

Modern CPUs, implementing combinational logic, are capable completing a MAC operation every cycle. Clock speed and the number of cores of the control computer CPU then drive the best possible sampling rate of the controller.

Industrial Computers

A brief survey of industrial computers used for process control was conducted. Computers from three different sources were considered, using four different models from each source as samples. The survey revealed that the industrial computers currently available all use modern, Intel® multi-core processors with clock speeds ranging from just under 2 GHz to just under 4 GHz. The available RAM amounts range from 4 GB to 64 GB. Table 5.21 summarizes these findings.

Table 5.21. Industrial Computer Processor Cores, Clock Speeds and RAM.

Brand	**Allen-Bradley by Rockwell Automation**			
Model	VersaView 5400	6181P	6181P Advanced	6177R
Processor	Atom E3845	Core i3 - 4102E	Core i7 - 4700EQ	Core i5 - 2500
Cores	4	2	4	4
Speed	1.91 GHz	1.6 GHz	2.4 GHz	3.1 GHz
Max RAM	4 GB	16 GB	16 GB	32 GB
Brand	**Industrial Computers Limited**			
Model	IPPC 1602	IPPC 1501	IPPC 1503	IPPC 2107
Processor	Celeron - J1900	Core i3 - 3120ME	Core i7 - 3540M	Core i5 - 6300U
Cores	4	2	2	2
Speed	2.42 GHz	2.4 GHz	2.9 GHz	2.40 GHz
Max RAM	4 GB	8 GB	8 GB	16 GB
Brand	**CONTEC**			
Model	VPC 1600	VPC 3000	VPC 3100-G	VPC 5000-G
Processor	Celeron G1820TE	Core i3 - 4570S	Core i7 - 8700	Xeon E-2278GE
Cores	2	4	6	8
Speed	2.2 GHz	2.9 GHz	3.2 GHz	3.3 GHz
Max RAM	4 GB	8 GB	32 GB	64 GB

From the information shown in Table 5.21, it appears any of the industrial computers listed would have sufficient RAM to run the proposed controller algorithms. Using the CPU specifications identified in Table 5.21 the plots shown in Figure 5.6 were generated.

Figure 5.6 shows the relationship between processor clock speed and maximum sampling rate when running a control algorithm that executes 1 billion MAC operations per iteration.

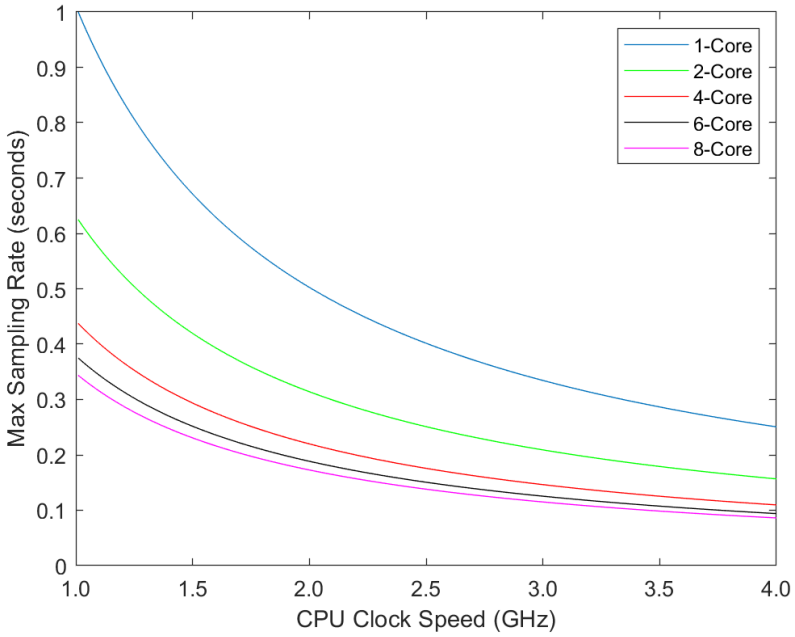

Figure 5.6. CPU Clock Speed vs. Sampling Rates.

When considering the effect of multiple CPU cores on the possible sampling rate for the plots in Figure 5.6, Amdahl's law was applied with the assumption that 75% of the controller code could be run in parallel.

Any of the CPU core/clock speed combinations in Table 5.21 would allow a sampling rate of 1 second or less using a control algorithm similar to the proposed controller with the large LSTM networks. Of course, if a significantly faster sampling rate (e.g., 0.25 seconds) is required, a CPU with a higher clock speed and/or more cores would be required to reach the desired sampling rate.

5.10 Chapter Summary

In this chapter, the benchmark, nonlinear quadruple-tank system was described and used as a test subject. The specific design of the proposed controller as applied to this system was also presented in detail. The utility of the effects taxonomy was demonstrated. The specifics of testing the system and controller were discussed and results presented. In addition, the design implementation and testing of an LSTM ensemble-based controller were also presented. Finally, the computational overhead requirements for using the proposed controller were discussed.

6 Reference Controller, Comparative Test Results Analysis and Stuxnet-Like Attack Testing

6.1 Chapter Introduction

This chapter describes a previously published, active fault tolerant controller [12] which was applied to the same quadruple-tank system described in Chapter 5. The controller is tested using the same tests applied to the proposed controller in the previous chapter and the results of that testing are described. This chapter presents a discussion on the results of testing the proposed controller and the comparison controller with 63 different effects on a sensor and/or a pump of the quadruple-tank system as described in Chapter 5. Both versions of the proposed controller and the comparison controller are tested while the system is subjected to simulated Stuxnet-like attacks. The results of that testing are compared and discussed as well.

6.2 Comparison Controller Design

In [12] the authors presented a reconfigurable control strategy for linear continuous time invariant systems. This controller was selected because it embodies the more common characteristics of current FTC; it is active, model-driven and incorporates an observer as part of the solution. In addition, it is intended to achieve some of the same control requirements as the controller proposed in this dissertation. Namely, to maintain proper control of the system in the presence of both sensor and actuator faults whether they occur one at a time or simultaneously.

As described in section 2.2.1, a reconfigurable controller uses a fault detection and identification mechanism to trigger reaction in the controller. Figure 6.1 shows the generic function of a reconfigurable controller.

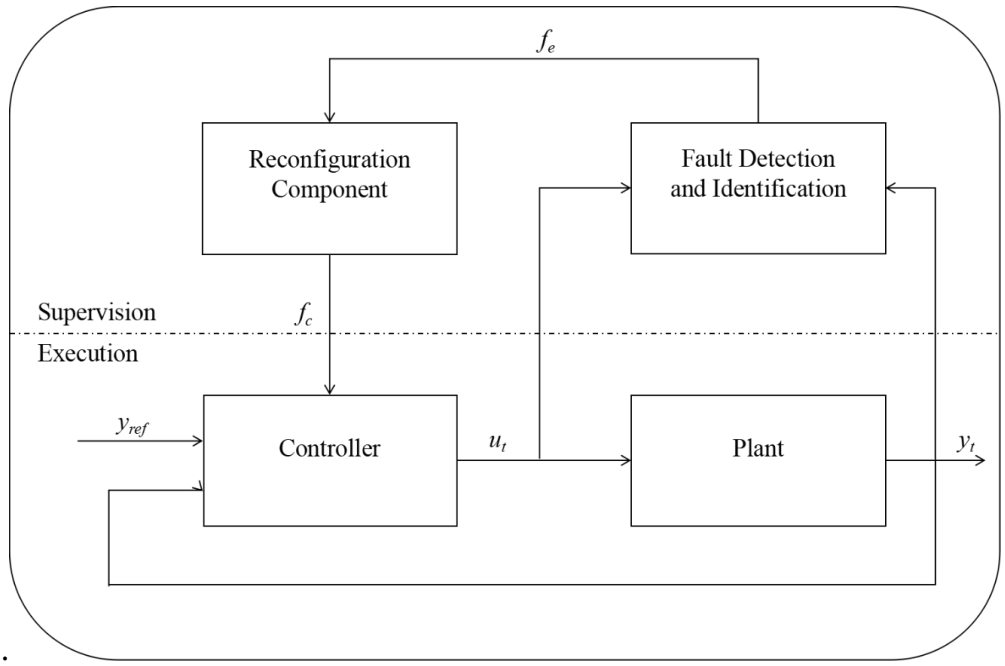

Figure 6.1. Reconfigurable Control System Diagram.

In general terms, when the fault detection and identification mechanism detects and identifies a particular fault, it generates a fault estimate (f_e) for the reconfiguration mechanism. In response, the reconfiguration mechanism generates a correction function (f_c) for the controller. Depending on the specific design, the correction function may cause a change in the controller itself so it can adapt to the fault, or it may modify one or more the signals into the controller (y_t) or out of the controller (u_t). This latter approach is sometimes referred to as "fault hiding" as the controller itself does not need to be aware of the fault to function normally. This is the approach used in the comparison reference controller [12].

The authors of [12] apply their active, reconfigurable controller to the benchmark quadruple-tank system described in Chapter 5. They utilize linearized state-space equations to model the system to be controlled. In control theory, such state space models represent

a system using a series of first-order differential equations and an algebraic output equation. In general form, the state equations are derived as

$$\dot{x}_n = f_n(x_1, x_2, \ldots x_n, u_1, u_2, \ldots u_m)$$

and the output equation is derived as

$$y_p = h_p(x_1, x_2, \ldots x_n, u_1, u_2, \ldots u_m)$$

where x_i is the state space, u_i is the input from the controller and y_i is the system output. If the system of equations are linear they can be rewritten in the general form

$$\dot{x} = Ax + Bu$$

$$y = Cx$$

Where A, B, and C are the state, input, and output matrices, respectively. Using this nomenclature, the nominal system with feedback controller diagram shown in Figure 6.2 is reconfigured to the new, but equivalent, diagram shown in Figure 6.3. This form will be used in the remainder of this system description.

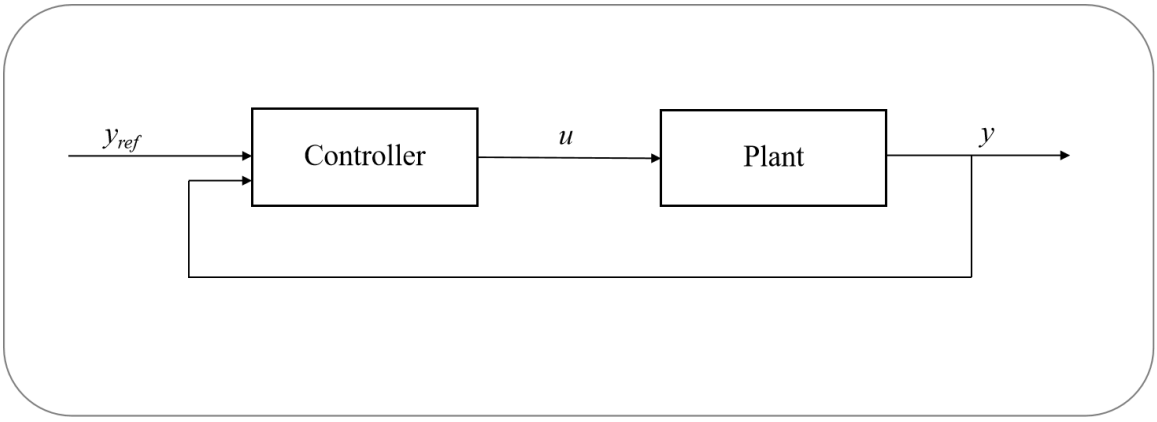

Figure 6.2. Nominal System with Feedback Control.

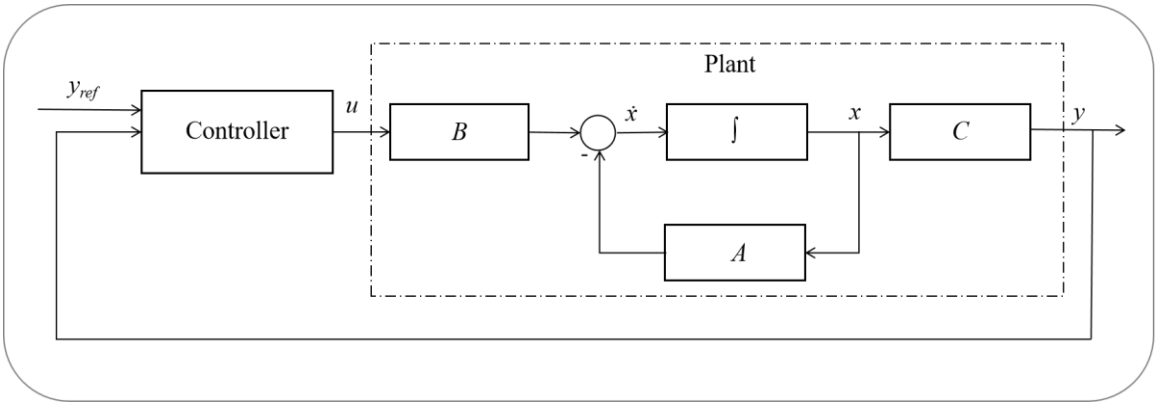

Figure 6.3. Equivalent System with Feedback Control.

The reference system uses a classical proportional-integral (PI) controller for each of the two pumps in the quadruple-tank system. Recall the nonlinear equations for the rate of change in the level of fluid in each tank in the quadruple-tank system previously presented in section 5.2.1:

$$\frac{dh_1}{dt} = \frac{-a_1}{A_1}\sqrt{2gh_1} + \frac{a_3}{A_1}\sqrt{2gh_3} + \frac{\gamma_1 k_1}{A_1} v_1 \quad (1)$$

$$\frac{dh_2}{dt} = \frac{-a_2}{A_2}\sqrt{2gh_2} + \frac{a_4}{A_2}\sqrt{2gh_4} + \frac{\gamma_2 k_2}{A_2} v_2 \quad (2)$$

$$\frac{dh_3}{dt} = \frac{-a_3}{A_3}\sqrt{2gh_3} + \frac{(1-\gamma_2) k_2}{A_3} v_2 \quad (3)$$

$$\frac{dh_4}{dt} = \frac{-a_4}{A_4}\sqrt{2gh_4} + \frac{(1-\gamma_1) k_1}{A_4} v_1 \quad (4)$$

Using equations (1-4), the system can be represented by linearized the state equations presented in [12].

$$\frac{dx}{dt} = \begin{bmatrix} \frac{-1}{T_1} & 0 & \frac{A_3}{A_1 T_3} & 0 \\ 0 & \frac{-1}{T_2} & 0 & \frac{A_4}{A_2 T_4} \\ 0 & 0 & \frac{-1}{T_3} & 0 \\ 0 & 0 & 0 & \frac{-1}{T_4} \end{bmatrix} x + \begin{bmatrix} \frac{\gamma_1 k_1}{A_1} & 0 \\ 0 & \frac{\gamma_2 k_2}{A_2} \\ 0 & \frac{(1-\gamma_2) k_2}{A_3} \\ \frac{(1-\gamma_2) k_1}{A_4} & 0 \end{bmatrix} u$$

which has the form $\dot{x} = Ax + Bu$. The output equation is given by $y = Cx$:

$$y = \begin{bmatrix} K_c & 0 & 0 & 0 \\ 0 & K_c & 0 & 0 \end{bmatrix} x$$

where the time constants in A are given by:

$$T_i = -\frac{A_i}{a_i}\sqrt{\frac{2h_i^o}{g}}, \qquad \text{where i} = 1, \ldots, 4$$

The fault hiding reconfigurable control approach is based on the idea of placing a reconfiguration block between the nominal controller and the faulty plant at reconfiguration time as shown in Figure 6.4.

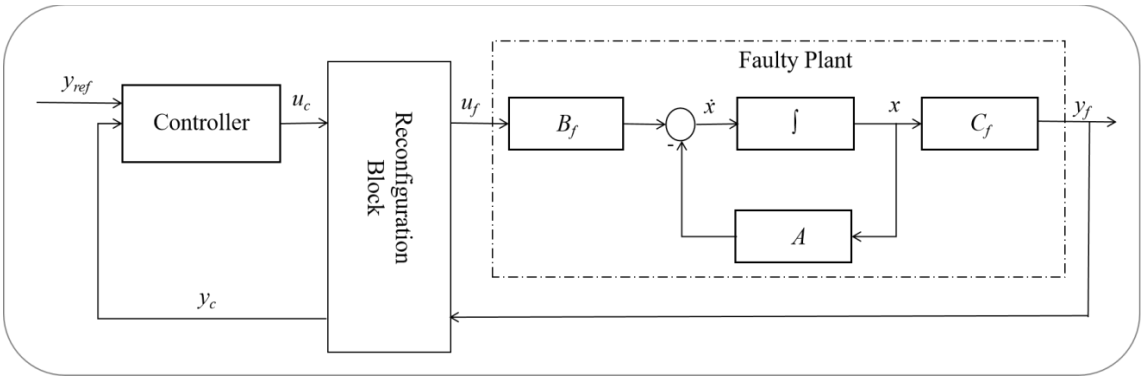

Figure 6.4. Fault Hiding with Reconfiguration Block.

In Figure 6.4 the output signal from a faulty sensor (y_f) is reconfigured to a correct output signal (y_c) for the nominal controller. The correct input from the nominal controller

(u_c) is reconfigured to a signal (u_f) that will compensate for the faulty actuator in the plant. The reconfiguration block hides the fault from the controller and helps the faulty plant to operate properly. The reconfiguration block contains a virtual sensor and a virtual actuator. In the event of a sensor fault, the controller activates the virtual sensor as shown in Figure 6.5.

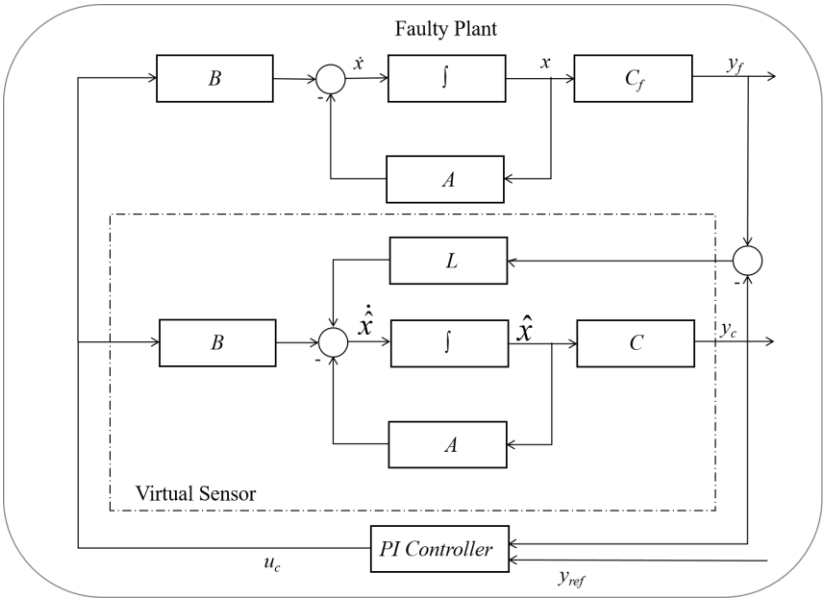

Figure 6.5. Faulty System with Virtual Sensor [12].

The derivation of the virtual sensor is similar to the design of a state observer (as described in section 2.2.1). The derivation is given by:

$$\dot{\hat{x}} = A\hat{x} + Bu_c + L(y_f - C\hat{x})$$

The parameter L is chosen so that all poles are within the design set C_g:

$$\sigma(A - LC_f) \subset C_g$$

This resulting block is the virtual sensor and is actually a Luenberger observer with output matrix C. In the event of an actuator fault, the virtual actuator uses the input signal meant

for the nominal process and transforms it into a signal useful for controlling the faulty plant as shown in Figure 6.6.

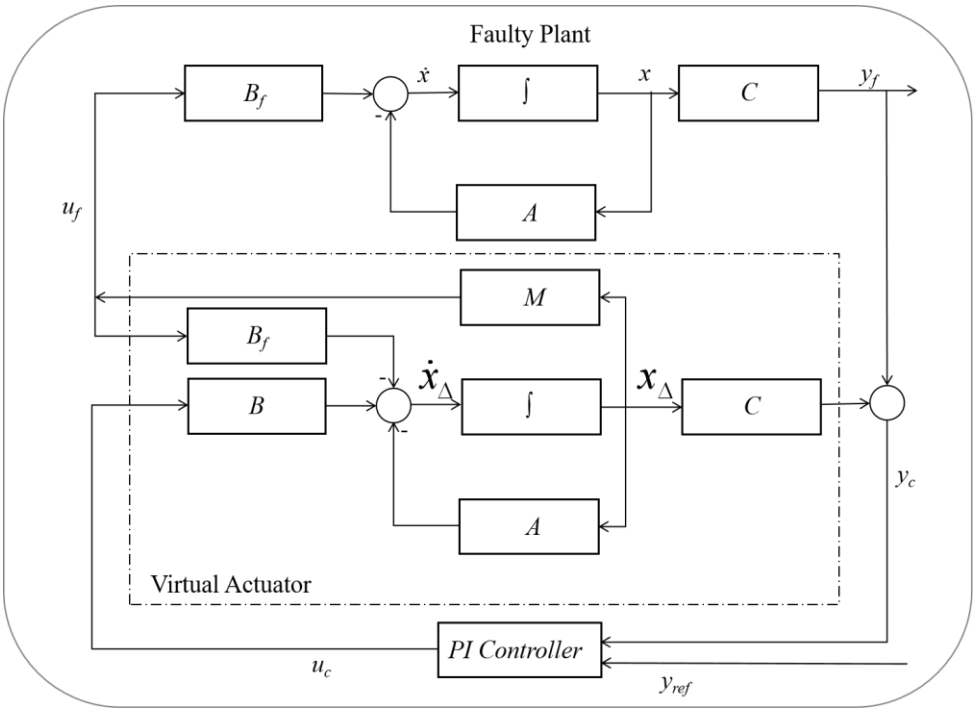

Figure 6.6. Faulty Plant with Virtual Actuator [12].

As defined by the state-space model, the virtual actuator is described as:

$$\dot{x}_\Delta = (A - B_f M)x_\Delta + Bu_c$$

$$y_c = y_f + Cx_\Delta$$

$$u_f = Mx_\Delta$$

where M is chosen so that all poles are within the designated set C_g where

$$\sigma(A - B_f M) \subset C_g$$

When a fault occurs in both a sensor and an actuator at the same time, the reconfiguration block is realized by the interconnection of a virtual sensor and a virtual actuator as shown in Figure 6.7.

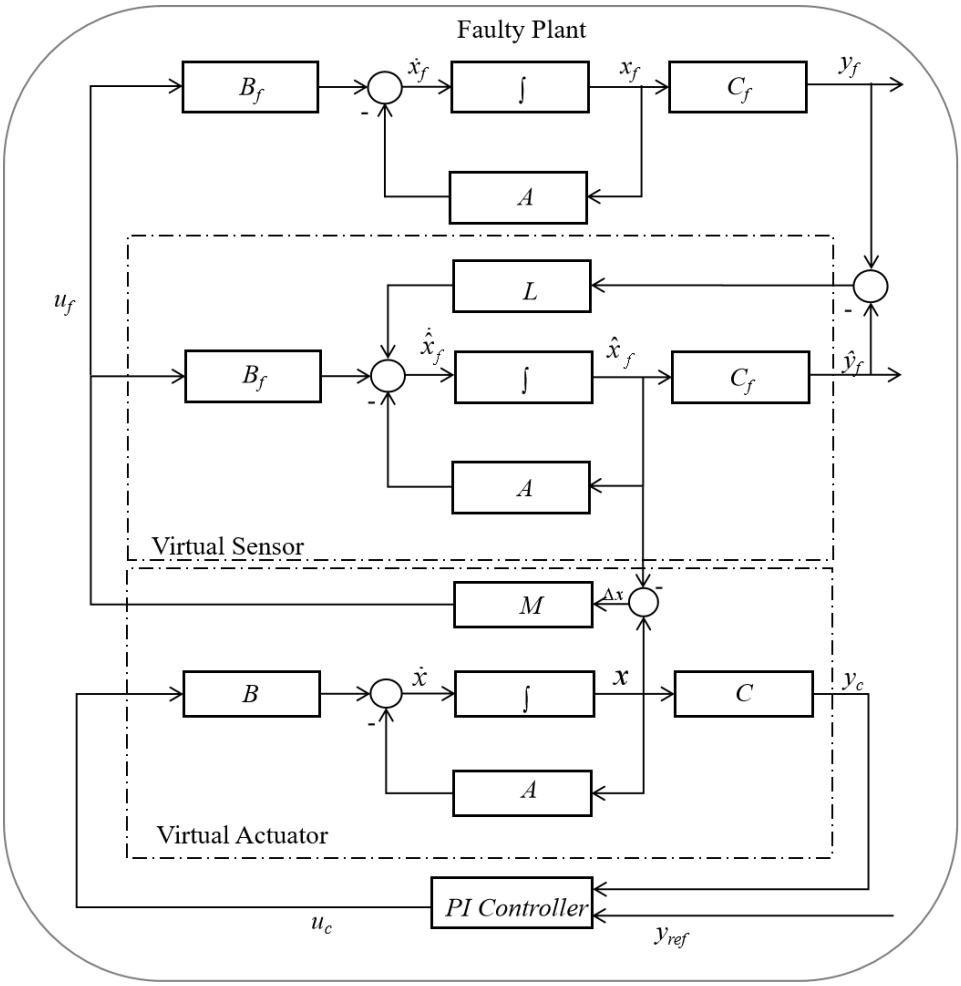

Figure 6.7. Virtual Sensor and Actuator with Faulty Plant [12].

As described previously, the virtual sensor is an observer for the state of the faulty plant with an output error injection and the virtual actuator contains a reference model of the nominal plant with feedback of the difference between the reference state and the observed state, as well as feed through of the control input. In this configuration the state space model equations are:

$$\dot{x} = Ax + Bu_c$$

$$y_c = Cx$$

$$u_f = M(x - \hat{x}_f)$$

The authors of [12] use a simple fault detection mechanism. In the case of the virtual sensor, a fault is indicated if $C \neq C_f$. For the virtual actuator, a fault is indicated if $B \neq B_f$. Both the virtual actuator and virtual sensor are engaged when $C \neq C_f$ and $B \neq B_f$. This simplistic approach works fine in a sterile simulation environment, but does not allow for the possibility of minor signal noise or minor fluctuation in these parameters that easily occur when any change occurs in the system. In order to avoid excessive false positives, we implement this comparison controller using a typical residual generator [11]. The residual generator is simply sliding windows which tracks the sum of the difference between C and C_f and between B and B_f. The window sums the difference over the last twenty iterations. This type of residual generator is commonly used in active FTC systems.

6.3 Comparison Controller Testing

For all testing, the system under test (SUT) and the associated control algorithm are realized as MATLAB® scripts. The tests are performed on an Intel® Core i7 CPU platform with 32 GB of RAM running Windows 10. Each test run is conducted by running the script for 500 iterations. Prior to each run the attack/fault effects(s) (described previously) is/are set to be activated at iteration number 250. The type of effect and magnitude of the effects were identical to those used in testing the proposed controller.

The various effects are shown in Table 6.1 and Table 6.2. The *Effect Number* and *Effect Value* fields will be used when discussing test results. Since the Fixed Value (#5) and the No Value (#6) effects leave the actuator uncontrollable and the quadruple-tank system design does not include redundant actuators, these two effects were not included in the training or testing of the controller for the quadruple tank system.

Table 6.1. Summary of Sensor Effects Used in Testing the Comparison Controller.

Effect Type	Effect Number	Effect Value	Add, Mult or Replace	Abrupt or Incipient
Increased	1	(1.00, 3.00)	Mult	Abrupt
Decreased	2	(0.15, 0.90)	Mult	Abrupt
Stochastic	3	(0.05, 0.55)	Add	Abrupt
Cyclical	4	(0.10, 1.10) (0.015, 1.015)	Add	Abrupt
Fixed	5	(0.00, 20.00)	Replace	Abrupt
No Value	6	(NaN)	Replace	Abrupt
Increased	7	(0.00, 1.00)	Add & Mult	Incipient
Decreased	8	(0.00, 1.00)	Add & Mult	Incipient

Table 6.2. Summary of Actuator Effects Used in Testing the Comparison Controller.

Effect Type	Effect Number	Effect Value	Add, Mult or Replace	Abrupt or Incipient	Controllable
Increased	1	(1.05, 1.50)	Mult	Abrupt	Yes
Decreased	2	(0.15, 0.90)	Mult	Abrupt	Yes
Stochastic	3	(0.05, 0.55)	Add	Abrupt	Yes
Cyclical	4	(0.50, 3.50) (0.05, 0.55)	Add	Abrupt	Yes
Fixed	5	(0.00, 10.00)	Replace	Abrupt	No
No Value	6	(NaN)	Replace	Abrupt	No
Increased	7	(0.00, 1.00)	Add & Mult	Incipient	Yes
Decreased	8	(0.00, 1.00)	Add & Mult	Incipient	Yes

In each test run the onset of the effect is timed to begin after the system is in a steady state condition (iteration #250). This is done to ensure an “apples to apples” comparison of the different controllers and to allow clarity in identifying how the system is controlled by each controller and how long it takes for the controller to return the system to the steady state condition. The tests performed on this controller were identical to those performed on the proposed controller in Chapter 5, and each test run used the same effect variable values as were used to test the proposed controller and the ensemble variant of that controller.

6.3.1 Comparison Controller Test Results

A total of 63 test runs were completed for the test; 1 with no effects applied, 8 with a sensor effect applied, 6 with an actuator effect applied and 48 with both a sensor and an actuator effect applied. The complete results of the testing are presented in Table 6.3. The column headers are identical to those used in the tests in Chapter 5.

Table 6.3. Results of Tests on Comparison Controller.

Test Run No.	Pump 1 Effect	Effect Value	Sensor 1 Effect	Effect Value	Components Affected	Catastrophic Failure	Maintain Proper Control?	Iterations to Regain Proper Control	Iterations to Correct Model	Iterations Using Correct Model %
1					0	0	1		0	100.00%
2			1	2.96	1	0	0	250	1	99.67%
3			2	0.56	1	0	0	28	3	99.00%
4			3	0.48	1	0	0	46	inf	16.33%
5			4	0.71	1	0	0	18	inf	16.33%
6			5	15.72	1	0	0	22	2	99.33%
7			6	*	1	0	1		1	99.67%
8			7	0.05	1	0	0	49	77	74.33%
9			8	0.82	1	0	0	75	44	85.33%
10	1	1.44			1	0	1		19	93.67%
11	2	0.73			1	0	1		16	94.67%
12	3	4.48			1	0	1		21	93.00%
13	4	2.47			1	0	1		24	92.00%
14	7	0.62			1	0	1		13	95.67%
15	8	0.87			1	0	1		12	96.00%
16	1	1.14	1	2.21	2	0	0	250	22	92.67%
17	1	1.50	2	0.18	2	0	0	250	2	99.33%
18	1	1.11	3	1.16	2	0	0	35	13	95.67%
19	1	1.43	4	0.69	2	0	0	250	inf	16.33%
20	1	1.45	5	11.75	2	0	0	13	11	96.33%
21	1	1.20	6	*	2	0	0	222	24	92.00%
22	1	1.50	7	0.21	2	0	0	40	42	86.00%
23	1	1.18	8	0.83	2	0	0	250	26	91.33%
24	2	0.30	1	2.53	2	0	0	221	7	97.67%
25	2	0.61	2	0.87	2	0	0	23	8	97.33%
26	2	0.53	3	2.44	2	0	0	119	15	95.00%
27	2	0.52	4	0.35	2	0	0	1	25	91.67%

28	2	0.23	5	3.07	2	0	0	1	3	99.00%
29	2	0.80	6	*	2	0	0	219	24	92.00%
30	2	0.69	7	0.00	2	0	0	250	211	29.67%
31	2	0.76	8	0.25	2	0	0	29	41	86.33%
32	3	1.08	1	2.31	2	0	0	217	14	95.33%
33	3	4.91	2	0.67	2	0	0	228	4	98.67%
34	3	1.88	3	2.40	2	0	0	16	19	93.67%
35	3	2.45	4	0.92	2	0	0	20	22	92.67%
36	3	1.23	5	15.41	2	0	0	3	10	96.67%
37	3	1.86	6	*	2	0	0	228	17	94.33%
38	3	4.36	7	0.13	2	0	0	250	49	83.67%
39	3	4.50	8	0.73	2	0	0	250	28	90.67%
40	4	1.77	1	2.21	2	0	0	3	12	96.00%
41	4	2.79	2	0.89	2	0	0	24	10	96.67%
42	4	0.10	3	2.57	2	0	0	9	13	95.67%
43	4	2.96	4	0.96	2	0	0	4	23	92.33%
44	4	0.84	5	13.72	2	0	0	17	12	96.00%
45	4	2.48	6	*	2	0	1		13	95.67%
46	4	0.99	7	0.70	2	0	0	1	29	90.33%
47	4	1.84	8	0.97	2	0	0	40	26	91.33%
48	7	0.62	1	2.15	2	0	0	218	14	95.33%
49	7	0.65	2	0.25	2	0	0	11	3	99.00%
50	7	0.08	3	2.12	2	0	0	250	24	92.00%
51	7	0.84	4	0.03	2	0	0	16	15	95.00%
52	7	0.54	5	6.75	2	0	0	5	4	98.67%
53	7	0.80	6	*	2	0	0	218	14	95.33%
54	7	0.76	7	0.44	2	0	0	250	32	89.33%
55	7	0.87	8	0.82	2	0	0	17	30	90.00%
56	8	0.55	1	1.33	2	0	0	177	15	95.00%
57	8	0.63	2	0.83	2	0	0	19	6	98.00%
58	8	0.34	3	1.82	2	0	0	189	15	95.00%
59	8	0.10	4	0.26	2	0	0	2	32	89.33%
60	8	0.49	5	0.03	2	0	0	225	14	95.33%
61	8	0.27	6	*	2	0	0	224	14	95.33%
62	8	0.44	7	0.72	2	0	0	250	28	90.67%
63	8	0.82	8	0.24	2	0	0	250	44	85.33%

Reviewing the results of these tests, the value of the taxonomy of effects (from Chapter 3) becomes more apparent. The comparison controller performed well when the effects were presented as abrupt, multiplicative increases or decreases to the sensor and/or actuator. In the article presenting this active, fault tolerant controller [12], these were the

only type of tests performed. In this research, the broader range effects identified conditions that posed a challenge to the comparison controller as well as highlighting a weakness of the active FTC; the inherent delay in responding to a fault or cyber-attack that is the result of using a dedicated detection/identification mechanism. The test results are summarized in Table 6.4.

Table 6.4. Test Result Summary for Comparison Controller.

Test Runs	**Component Affected**	**Cat. Fail**	**Proper Control**	**Iterations to Regain Proper Control**		**Iterations to Correct Detection**		**Iterations Using Correct Detection %**	
#	*A/S/Both*	%	%	*Mean*	*Median*	*Mean*	*Median*	*Mean*	*Median*
6	Actuators	0	100.00	*	*	17.50	17.50	94.17	94.17
8	Sensors	0	12.50	69.71	46.00	16.00	1.50	73.75	92.17
48	Both	0	2.08	123.49	119.00	22.48	15.00	90.76	95.00
Total		**0**	**14.29**	**116.52**	**47.50**	**20.83**	**15.00**	**89.07**	**95.00**

Using the same data summarized in the last two columns of Table 6.4, a confusion matrix of the detection/identification results of the comparison controller is shown in Table 6.5. By way of review, fault type 1 is the correct choice when no component is being affected. Fault type 2 is the correct choice when the actuator is being affected. Fault type 3 is the correct choice when the sensor is being affected and fault type 4 is the correct choice when both are being affected.

Table 6.5. Confusion Matrix of Fault Detection in Comparison Controller.

	Fault Type Detected				
Correct Fault Type	**3338**	**0**	**0**	**0**	1
	105	**1401**	**0**	**0**	2
	624	**4**	**1378**	**2**	3
	1284	**27**	**19**	**10718**	4
	1	2	3	4	

6.4 Discussion of Test Results for All Three Controllers

The quadruple-tank test subject was subjected to 63 different tests in which a system sensor and/or a system pump was subjected to the different effects of a cyber-attack or fault. The tests were conducted on the system with the comparison controller and with the proposed controller (both single LSTM and ensemble variants) separately, but concurrently, so that the exact same effects were applied to each. The summary results of these tests are provided for comparison across all three controllers in Table 6.6.

Table 6.6. Summary of Test Results on Quadruple-Tanks System.

Controller	**Cat. Fail**	**Proper Control Maintained**	**Iterations to Regain Proper Control**		**Iterations to Correct Detection or Model Selection**		**Iterations Using Correct Detection or Model %**	
	%	%	*Mean*	*Median*	*Mean*	*Median*	*Mean*	*Median*
Comparison	0	14.29	116.52	47.50	20.83	15.00	89.07	95.00
Proposed (Ensemble)	0	56.25	7.47	2.00	4.19	1.00	98.60	99.67
Proposed (Single LSTM)	0	39.68	4.58	2.00	7.54	3.00	97.48	99.00

In no test did any controller allow a catastrophic failure of the system. Failure is defined as allowing a tank to run dry or to overfill. Most of the results summarized in Table 6.6 are based on the performance of the system under control, and allow a ready comparison of how well the three controllers perform their intended function. Whether the priority is maintaining proper control or regaining that control after it is lost, the single LSTM controller performed better on these tests than the comparative controller and the

ensemble based controller did somewhat better than the single LSTM controller, making it the best of the three from a controller performance perspective.

6.4.1 Statistical Analysis of Test Results for All Three Controllers

In the area of active fault tolerant controls, "fault detection" is an ubiquitous term and is usually paired with other terms like "identification", "diagnosis" or "recognition". These terms all refer to the act of determining what type of fault has been detected. In the case of the comparison controller described in this chapter, that determination identifies the fault as being associated with a sensor, with an actuator or with both. In other words. It is performing classification of the fault. Since the proposed controller is based on a classification tool (LSTM) it is appropriate to compare both controllers as classifiers.

Precision and recall are both popular metrics for evaluating classifier performance. Precision is the percentage that the classifier correctly predicts positive when making a decision. More specifically, precision is the number of correctly identified positive examples divided by the total number of examples that are classified as positive.

As presented in [110], given an $n \times n$ confusion matrix m, like the one shown in Table 6.5, the precision measurement P for the i^{th} class is given by

$$P_i = \frac{m_{ii}}{\sum_{x=1}^{n} m_{ix}}$$

Recall is the percentage of positives correctly identified out of all the existing positives; it is the number of correctly classified positive examples divided by the total number of true positive examples in the test set. The Recall measurement R for the i^{th} class is given by

$$R_i = \frac{m_{ii}}{\sum_{x=1}^{n} m_{xi}}$$

Precision and recall are often achieved at the expense of the other, i.e., high precision is achieved at the expense of recall and vice versa. An ideal classifier would have both high recall and high precision.

The F-measure, sometimes called the F-statistic or F1-score, is the harmonic measure of precision and recall in a single measurement. The F1-score ranges from 0 to 1, with a measure of 1 being a classifier perfectly capturing both precision and recall. The F1-Score (harmonic mean) for the i^{th} class is calculated using the precision and recall for that class:

$$F1_i = \frac{2P_iR_i}{P_i + R_i}$$

With these three scores, an overall F1-score for the entire classifier can be calculated.

There are three such overall F1-scores; Macro F1, Weighted F1 and Micro F1. The Macro F1 ($\mathcal{F}1$) is simply the arithmetic mean of the individual class hormonic means:

$$\mathcal{F}1 = \frac{1}{n}\sum_{x=1}^{n}\frac{2P_xR_x}{P_x + R_x}$$

When averaging the Macro F1, equal weight is given to each class even if the distribution of data across those classes is unbalanced. Such is the case with the tests conducted on the various controllers for the quadruple-tank system. This can cause the Macro F1 to skewed by an over represented class on which the classifier performs poorly or exceptionally well.

To counter this, a Weighted F1 ($WF1$) score includes the number of samples in each class (s_x), known as the "support" to give equal weigh to each class.

$$WF1 = \frac{\sum_{x=1}^{n}\frac{2P_xR_x}{P_x + R_x}s_x}{\sum_{x=1}^{n}s_x}$$

As with the Macro F1 there are also weighted averages for the Precision and Recall.

The final overall F1-score is the Micro-F1. Rather than being an average of F1-scores, the Micro-F1 provides an overall accuracy for the classifier. It is simply the proportion of correctly classified samples to all the samples. Looking back at the example confusion matrix, the Micro-F1 is calculated as

$$\mathbb{F}1 = \frac{\sum_{x=1}^{n} m_{xx}}{\sum_{y=1}^{n}\sum_{x=1}^{n} m_{xy}}$$

The micro-precision and micro-recall scores are calculated in the exact same manner, so *micro-F1 = micro-precision = micro-recall = accuracy*. For reference, all of the scores discussed are presented in Table 6.7 through 6.9 for the comparison controller, ensemble controller and LSTM controller respectively.

From the scores for the comparison controller (Table 6.7.) the impact of the unbalanced precision and recall scores for classes 1 and 3 can be observed in the respective F1-score for those two classes. Relying solely on the macro-F1 score as an overall assessment of the classification performance would underrate the performance with a score of 0.8682. Using the weighted-F1 improves the overall assessment, but the micro-F1 score appears to be the best overall measure of the comparative controller's classification ability.

The ensemble and single LSTM classifiers were more consistent across their overall scores. For both of these the micro-F1-score is the preferred score. With scores of 0.9860 and 0.9748 respectively, the ensemble and LSTM controllers demonstrate they are more accurate classifiers than the comparison controller. With a slightly higher accuracy score, and the results discussed in section 6.3.1, the ensemble-based controller is shown to be the more accurate classifier as well as the better overall controller.

Table 6.7. Precision, Recall and F1-Score Analysis for Comparison Controller.

Class	Precision	Recall	F1-Score	Support
1	0.6238	1.0000	0.7683	3338
2	0.9784	0.9303	0.9537	1506
3	0.9864	0.6863	0.8094	2008
4	0.9998	0.8896	0.9415	12048
Macro F1	0.8971	0.8765	**0.8682**	
Weighted F1	0.9303	0.8907	**0.8979**	
Micro F1	0.8907	0.8907	**0.8907**	

Table 6.8. Precision, Recall and F1-Score Analysis for Ensemble-Based Controller.

Class	Precision	Recall	F1-Score	Support
1	0.9673	1.0000	0.9833	3338
2	0.9234	0.9841	0.9527	1506
3	1.0000	0.9771	0.9884	2008
4	0.9976	0.9839	0.9907	12048
Macro F1	0.9721	0.9863	**0.9788**	
Weighted F1	0.9866	0.9860	**0.9862**	
Micro F1	0.9860	0.9860	**0.9860**	

Table 6.9. Precision, Recall and F1-Score Analysis for LSTM-Based Controller.

Class	Precision	Recall	F1-Score	Support
1	0.9670	1.0000	0.9832	3338
2	1.0000	0.9900	0.9950	1506
3	0.8463	0.9846	0.9102	2008
4	0.9997	0.9643	0.9817	12048
Macro F1	0.9533	0.9847	**0.9675**	
Weighted F1	0.9777	0.9748	**0.9754**	
Micro F1	0.9748	0.9748	**0.9748**	

6.5 Testing with Stuxnet-Like Attacks

In the introductory section of this dissertation there is a reference to the Stuxnet virus attack and the damage it caused. The Stuxnet virus was able to work undetected because it not only was able to change the speed of the Iranian centrifuge motors, it also caused the motor speed sensor to report to the human operators that the centrifuge was operating within normal parameters.

6.5.1 Stuxnet-Like Test Design

Using the effects of the actual Stuxnet attack as an example, the comparison controller, the ensemble controller and the proposed controller design were tested with five different Stuxnet-like attacks on the quad-tank system. In all five tests the sensor value was locked in at the reference amount of 10 cm. The pump was then affected to abruptly rise in the first test and abruptly drop in the second test. The third test applied an oscillation to the pump voltage which caused the pump speed to oscillate. The other two tests were conducted applying an incipient effect to the pump; gradually increasing it in the fourth test and gradually decreasing it in the fifth test.

In all five cases the attack occurred after the system had reached steady-state operations and the response of the respective systems was monitored over a 1,000 iteration window to allow the controllers more time to recover from the effects of the attack. All five attack scenarios were run four times with changes to the randomly selected effect variable values in the actuator effects. All three controllers were subjected to the exact same tests.

6.5.2 Stuxnet-Like Test Results

To better visualize the differences in how these controllers handled the Stuxnet-like attacks, the following figures are grouped by test. The first group of three figures are from each of the three controllers showing performance during the first test and so on with each subsequent test. Each figure has 2 panels. The panel on the left depicts the water level in the tank (solid blue line) and the reported level from the tank sensor (dashed red line). The panel on the right depicts the voltage to the pump. The Stuxnet-like attack affects the pump voltage while the controller attempts to counter that effect. The quality of control in each case is represented by how close the controller can keep the actual water level to the desired (reference) level.

Figures 6.8 through 6.10 depict examples of the test #1 performance of the comparison controller, the ensemble controller and the single LSTM controller respectively. Comparing the plots of the first Stuxnet-like attack test, we see the comparison controller never did recognize this as an attack on the sensor and the actuator. As a result, it never responded to it. Both the Ensemble controller and the LSTM controller quickly regained control of the system.

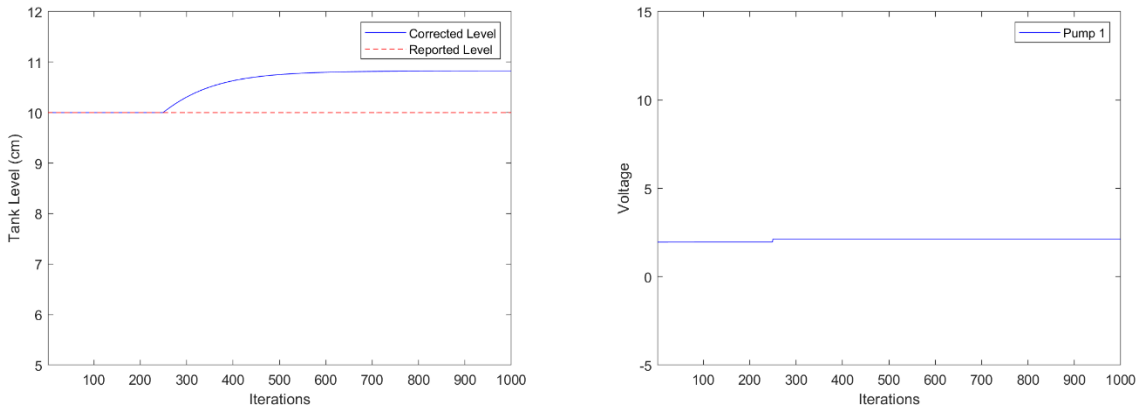

Figure 6.8. System Response to Stuxnet-like Attack #1 with Comparison Controller.

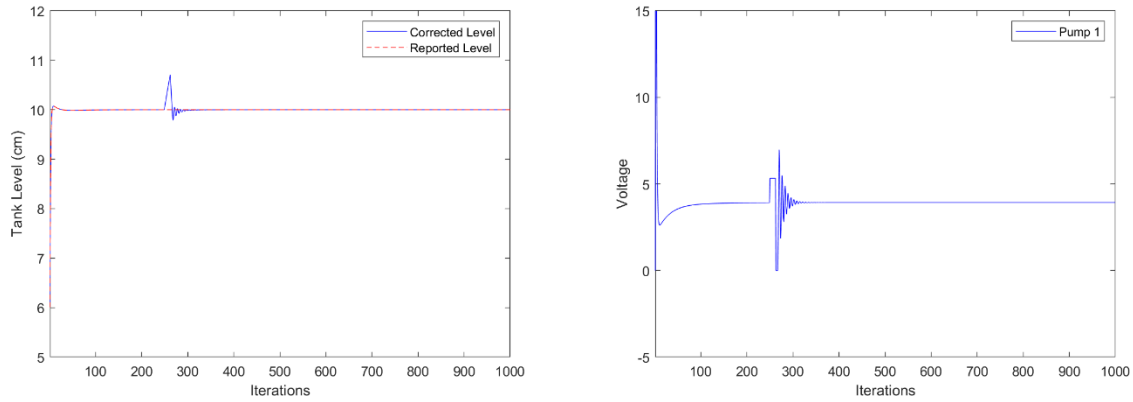

Figure 6.9. System Response to Stuxnet-like Attack #1 with LSTM Ensemble Controller.

Figures 6.11. through 6.13. depict examples of the test #2 performance of the comparison controller, the ensemble controller and the single LSTM controller respectively.

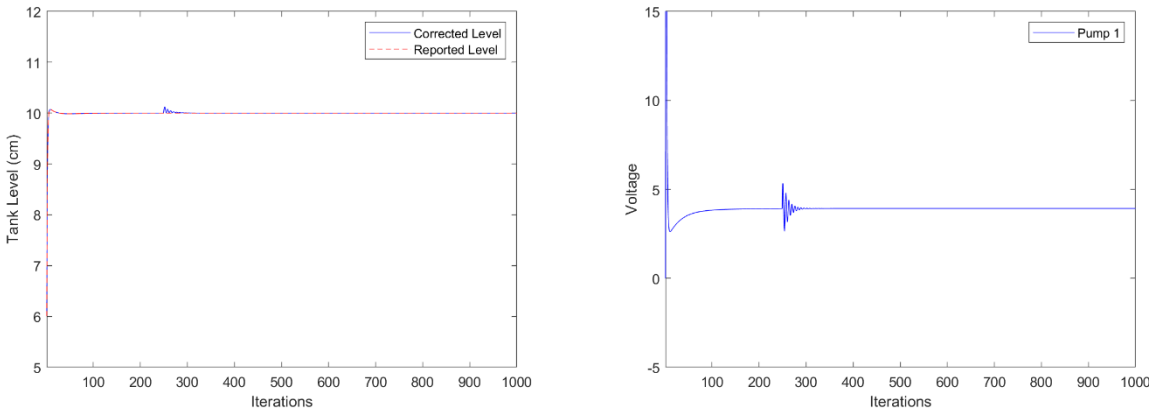

Figure 6.10. System Response to Stuxnet-like Attack #1 with Proposed Controller.

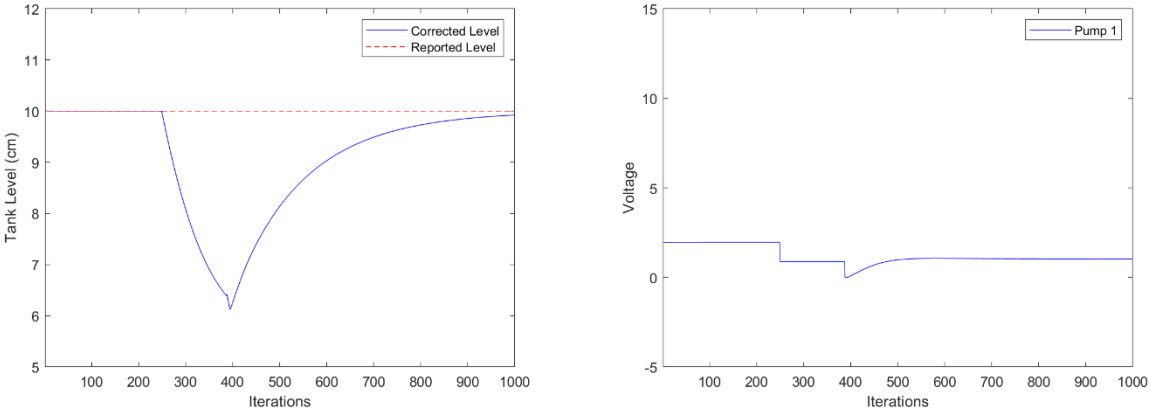

Figure 6.11. System Response to Stuxnet-like Attack #2 with Comparison Controller.

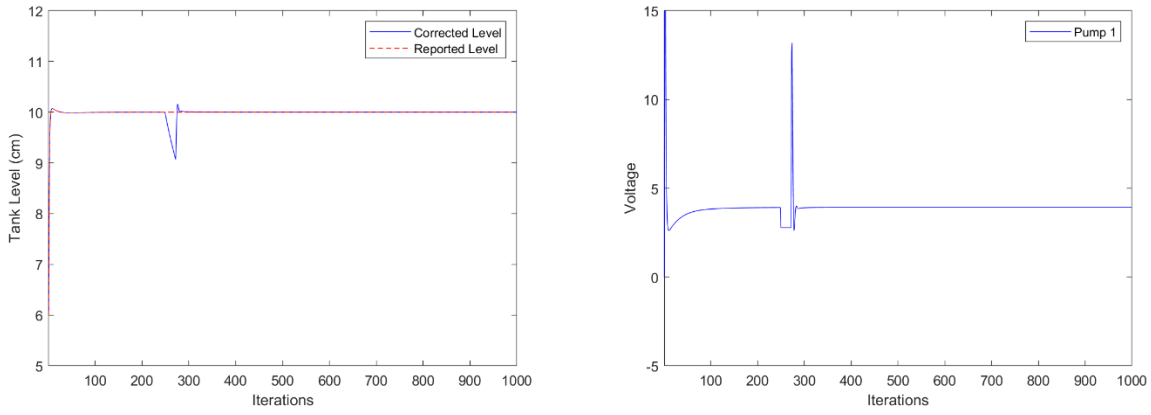

Figure 6.12. System Response to Stuxnet-like Attack #2 with LSTM Ensemble Controller

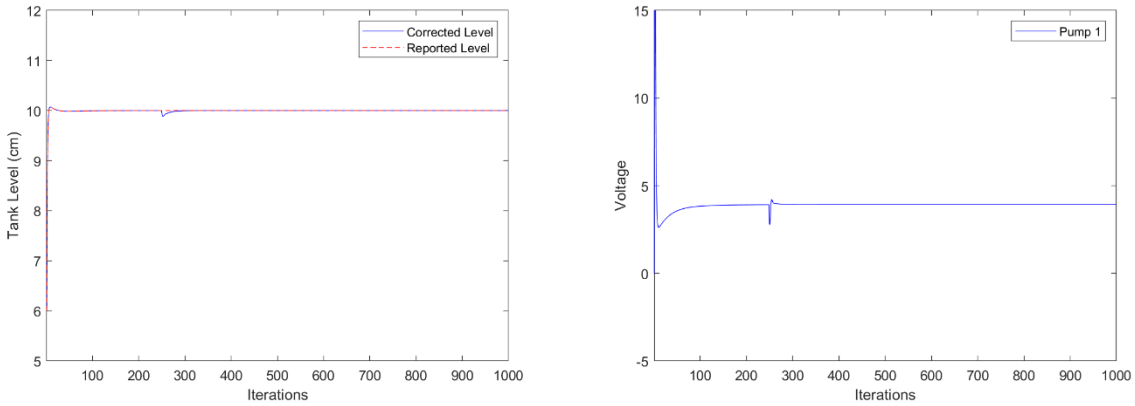

Figure 6.13. System Response to Stuxnet-like Attack #2 with Proposed Controller.

In the second attack, the comparison controller did detect and identify the attack correctly and was able to regain proper control of the system before the test ended. The ensemble controller caused a spike in pump voltage before bringing the system under control and the LSTM controller regained proper control very quickly.

Figures 6.14 through 6.16 depict examples of the test #3 performance of the comparison controller, the ensemble controller and the single LSTM controller respectively.

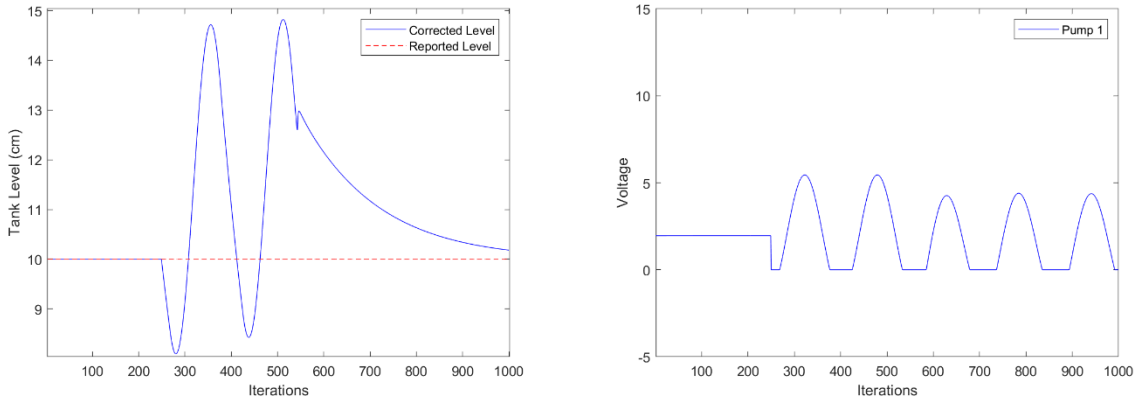

Figure 6.14. System Response to Stuxnet-like Attack #3 with Comparison Controller.

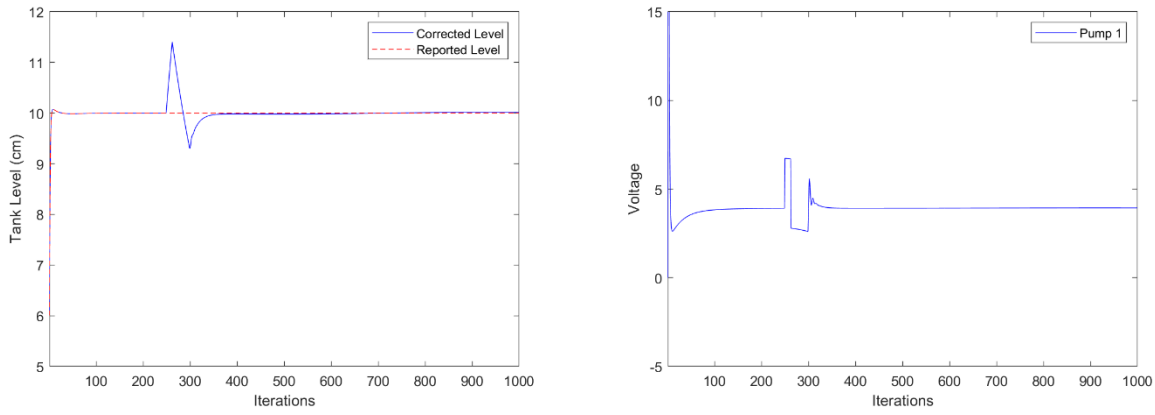

Figure 6.15. System Response to Stuxnet-like Attack #3 with LSTM Ensemble Controller.

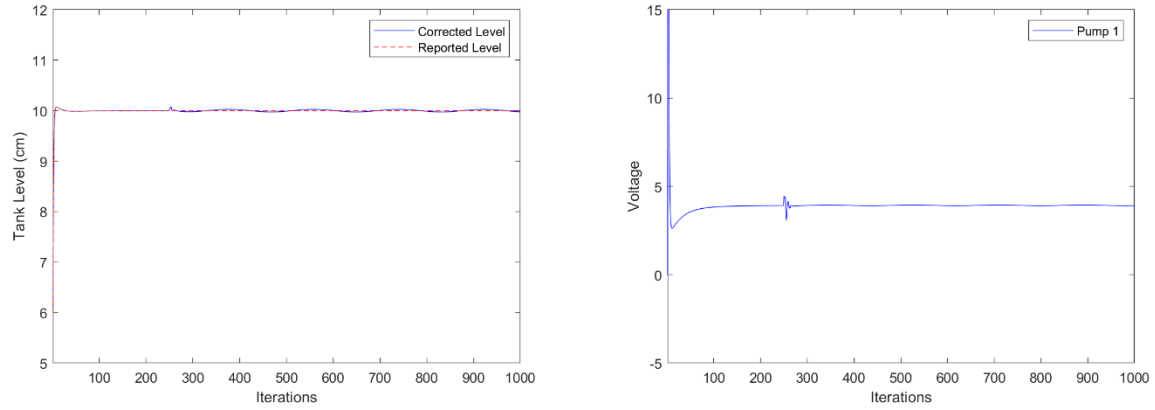

Figure 6.16. System Response to Stuxnet-like Attack #3 with Proposed Controller.

The third Stuxnet-like attack is the most similar to the actual Stuxnet attack in which the Iranian centrifuges were subjected large swings in motor speed. In this test the comparison controller did eventually detect the attack at iteration 542. The ensemble controller regained control after a full cycle of the oscillation and the LSTM controller did so almost immediately.

Figures 6.17 through 6.19 depict examples of the test #4 performance of the comparison controller, the ensemble controller and the single LSTM controller respectively.

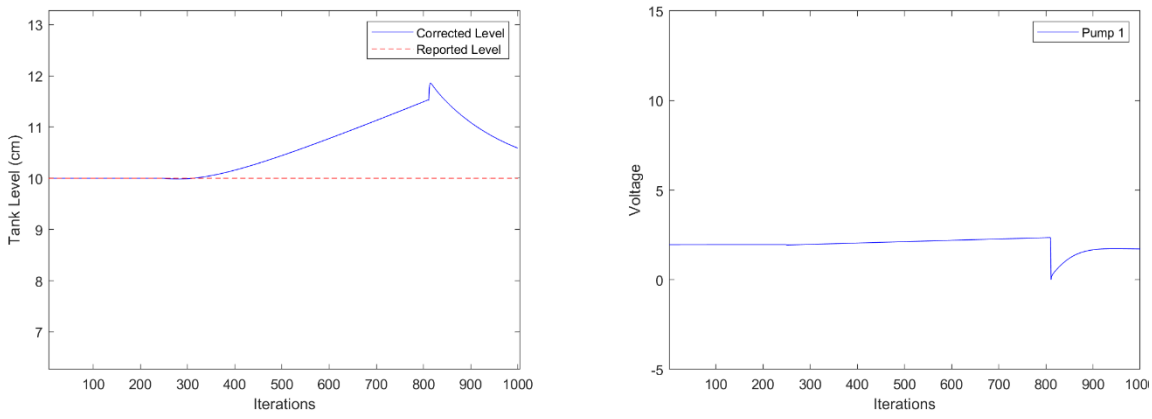

Figure 6.17. System Response to Stuxnet-like Attack #4 with Comparison Controller.

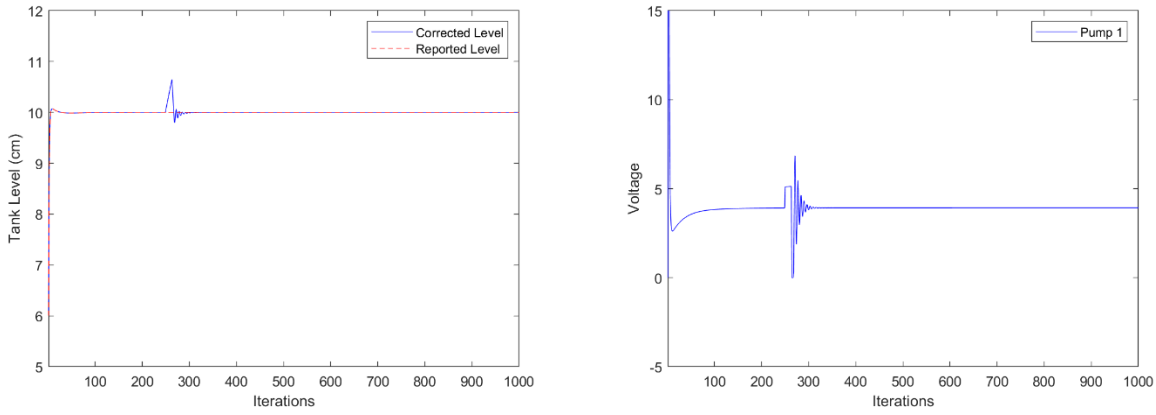

Figure 6.18. System Response to Stuxnet-like Attack #4 with LSTM Ensemble Controller.

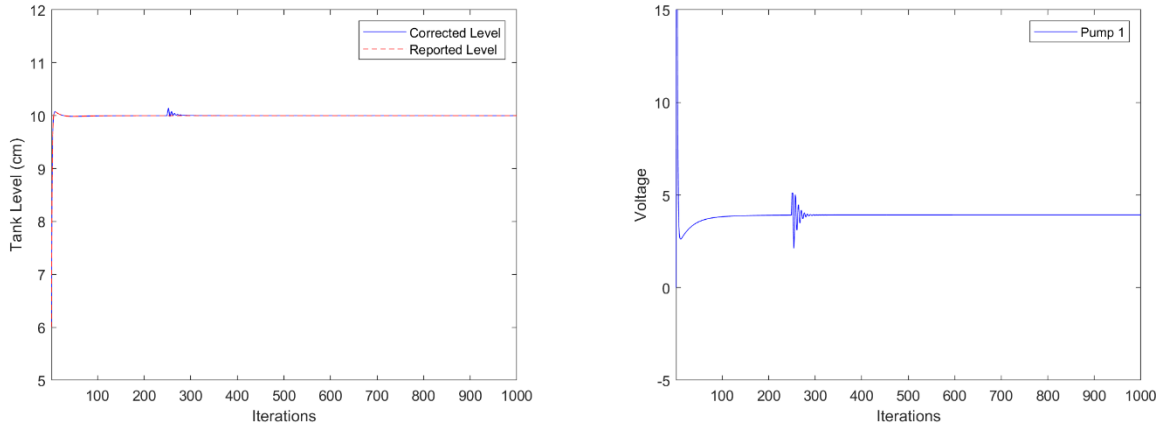

Figure 6.19. System Response to Stuxnet-like Attack #4 with Proposed Controller.

In the fourth test, the comparison controller did eventually detect the attack. This case illustrates the challenge to active fault tolerant controllers posed be incipient effects of cyber-attacks and faults. Because the onset of the effect was gradual, it took the controller over 560 iterations to detect it. The other controllers handled the effect readily. Figures 6.20 through 6.22 depict examples of the test #5 performance of the comparison controller, the ensemble controller and the single LSTM controller respectively.

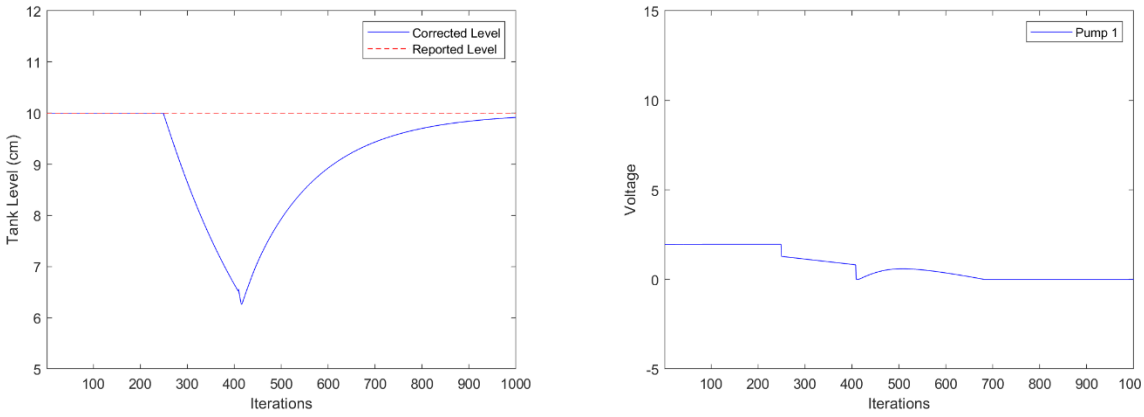

Figure 6.20. System Response to Stuxnet-like Attack #5 with Comparison Controller

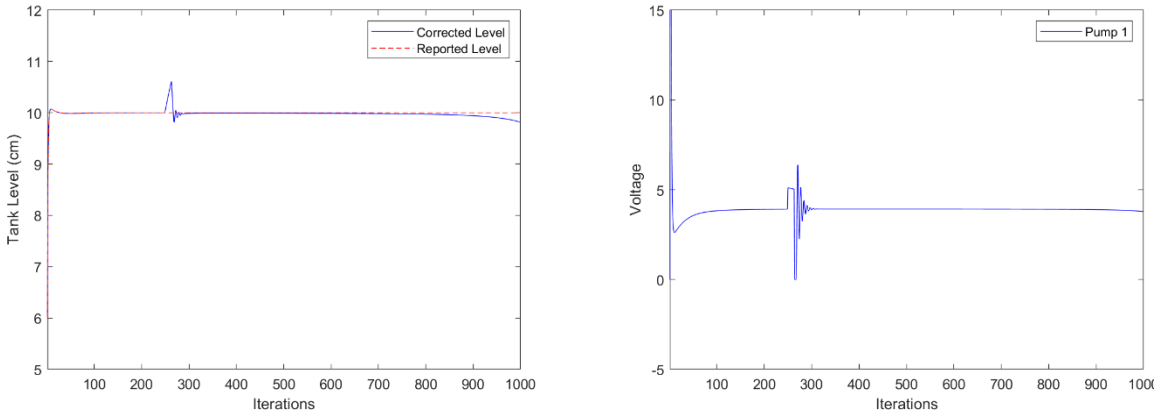

Figure 6.21. System Response to Stuxnet-like Attack #5 with LSTM Ensemble Controller.

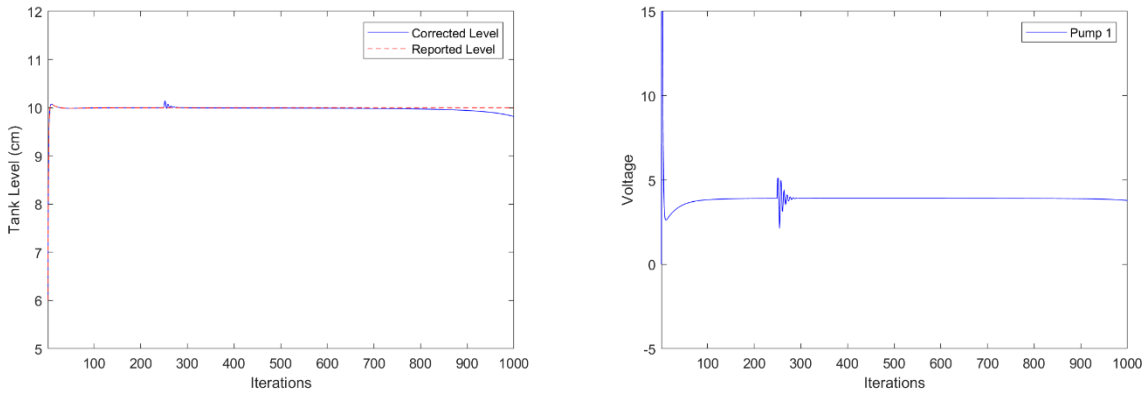

Figure 6.22. System Response to Stuxnet-like Attack #5 with Proposed Controller.

Similar to the fourth test, the fifth attack was eventually detected by the comparison controller while the other controllers were able regain or maintain proper control throughout the test.

The figures shown above allow us to visualize how the effects from the taxonomy influence the system and how the controller behaves in the presence of those effects. The numerical results of these tests allow us to better measure and analyze their performance.

The results on the test metrics for the comparison controller are presented in Table 6.10. In this test series there were no catastrophic failures. Of the 20 Stuxnet tests the comparison controller detected and identified the attack as applying to both the sensor and the actuator in 17. Those that were not detected are indicated by the term “inf” in that column of the chart. Of those it did detect, it took between 138 to 587 iterations of the control program to do so. Another indication that the comparison controller struggled to perform under these conditions is the time required for the controller to bring the system back into proper control which started at 623 iterations on the low end and there were 10

tests in which the controller did not regain proper control before the test ended (indicated by 750 iterations in that column).

The same tests were run against the ensemble LSTM controller. The results are presented in Table 6.11.

Table 6.10. Results of Stuxnet-Like Attack on Quadruple-Tank System with the Comparison Controller.

Test Run No.	Pump Effect	Effect Value	Sensor Effect	Effect Value	Components Affected	Catastrophic Failure	Maintain Proper Control	Iterations to Regain Proper Control	Iterations to Correct Detection	Iterations Using Correct Detection %
1	1	1.36	5	10	2	0	0	739	179	77.63%
2	1	1.12	5	10	2	0	0	750	407	49.13%
3	1	1.10	5	10	2	0	0	750	587	26.63%
4	1	1.09	5	10	2	0	0	750	inf	6.13%
5	2	0.71	5	10	2	0	0	696	194	75.75%
6	2	0.77	5	10	2	0	0	701	223	72.13%
7	2	0.87	5	10	2	0	0	750	366	54.25%
8	2	0.45	5	10	2	0	0	706	138	82.75%
9	4	0.28	5	10	2	0	0	750	inf	6.13%
10	4	0.32	5	10	2	0	0	750	293	63.38%
11	4	0.05	5	10	2	0	0	723	103	87.13%
12	4	0.18	5	10	2	0	0	623	107	86.63%
13	7	0.23	5	10	2	0	0	750	561	29.88%
14	7	0.08	5	10	2	0	0	750	inf	6.13%
15	7	0.82	5	10	2	0	0	750	314	60.75%
16	7	0.43	5	10	2	0	0	750	411	48.63%
17	8	0.91	5	10	2	0	0	721	159	80.13%
18	8	0.44	5	10	2	0	0	710	166	79.25%
19	8	0.87	5	10	2	0	0	719	159	80.13%
20	8	0.91	5	10	2	0	0	720	159	80.13%

Table 6.11. Results of Stuxnet-like Attacks on the Quadruple-Tank System with the LSTM Ensemble Controller.

Test Run No.	Pump Effect	Effect Value	Sensor Effect	Effect Value	Components Affected	Catastrophic Failure	Maintain Proper Control	Iterations to Regain Proper	Iterations to Correct Model	Iterations Using Correct Model %
1	1	1.36	5	10	2	0	0	15	13	98.38%
2	1	1.12	5	10	2	0	0	12	16	98.00%
3	1	1.10	5	10	2	0	0	10	16	98.00%
4	1	1.09	5	10	2	0	0	9	16	98.00%
5	2	0.71	5	10	2	0	0	23	23	97.13%
6	2	0.77	5	10	2	0	0	23	24	97.00%
7	2	0.87	5	10	2	0	0	21	24	97.00%
8	2	0.45	5	10	2	0	0	28	24	97.00%
9	4	0.28	5	10	2	0	0	11	16	98.00%
10	4	0.32	5	10	2	0	0	141	24	97.00%
11	4	0.05	5	10	2	0	0	34	13	98.38%
12	4	0.18	5	10	2	0	0	28	23	97.13%
13	7	0.23	5	10	2	0	0	15	14	98.25%
13	7	0.08	5	10	2	0	0	15	14	98.25%
15	7	0.82	5	10	2	0	0	15	14	98.25%
16	7	0.43	5	10	2	0	0	15	14	98.25%
17	8	0.91	5	10	2	0	0	15	14	98.25%
18	8	0.44	5	10	2	0	0	15	14	98.25%
19	8	0.87	5	10	2	0	0	15	14	98.25%
20	8	0.91	5	10	2	0	0	15	14	98.25%

The LSTM ensemble controller selected the correct model in each of the 20 tests. The speed at which it did so ranged from as few as 13 iterations and up to 24 iterations. The controller also regained proper control of the system in every test, requiring from 9 to 141 iterations to do so. In all 20 test cases the disruption to the system was minimal as the LSTM ensemble controller using ensembles readily handled the Stuxnet-like attacks.

The same tests were also run against the proposed (single LSTM) controller as summarized in Table 6.12. The proposed controller was able to maintain control of the system during the simulated Stuxnet-like attacks. In five of the tests, the water level in the tank was maintained within the ±1% range during the entire test. In the other tests, it was quickly brought back into proper control with the worst case being test #8 in which 20 iterations were required to do so. In all cases, the controller selected the correct model in only 2 to 4 iterations.

Table 6.12. Results of Stuxnet-like Attacks on the Quadruple-Tank System with the Proposed Controller (Single LSTM).

Test Run No.	Pump Effect	Effect Value	Sensor Effect	Effect Value	Components Affected	Catastrophic Failure	Maintain Proper Control	Iterations to Regain Proper Control	Iterations to Correct Model	Iterations Using Correct Model %
1	1	1.36	5	10	2	0	0	2	2	99.75%
2	1	1.12	5	10	2	0	1	*	4	99.50%
3	1	1.10	5	10	2	0	1	*	4	99.50%
4	1	1.09	5	10	2	0	1	*	4	99.50%
5	2	0.71	5	10	2	0	0	3	2	99.75%
6	2	0.77	5	10	2	0	0	2	3	99.63%
7	2	0.87	5	10	2	0	1	*	3	99.63%
8	2	0.45	5	10	2	0	0	20	2	99.75%
9	4	0.28	5	10	2	0	1	*	4	99.50%
10	4	0.32	5	10	2	0	0	17	2	99.75%
11	4	0.05	5	10	2	0	0	12	2	99.75%
12	4	0.18	5	10	2	0	0	3	2	99.75%
13	7	0.23	5	10	2	0	0	2	3	99.63%
14	7	0.08	5	10	2	0	0	2	3	99.63%
15	7	0.82	5	10	2	0	0	2	3	99.63%
16	7	0.43	5	10	2	0	0	2	3	99.63%
17	8	0.91	5	10	2	0	0	2	3	99.63%
18	8	0.44	5	10	2	0	0	2	3	99.63%
19	8	0.87	5	10	2	0	0	2	3	99.63%
20	8	0.91	5	10	2	0	0	2	3	99.63%

The tests results for all three controllers are summarized in Table 6.13. Using the test data above, the confusion matrices for the three controllers are presented in Tables 6.14 through 6.16. By design, the Stuxnet attack is difficult to detect because the sensor replay attack portion hides what the malware does to the actuators. The large number of false negatives shown in Table 6.14 are consistent with expectations.

Table 6.13. Summary of Test Results for Stuxnet-Like Attacks.

Controller	Cat. Fail	Proper Control Maintained	Iterations to Regain Proper Control		Iterations to Correct Detection or Model Selection		Iterations Using Correct Detection or Model %	
	%	*%*	*Mean*	*Median*	*Mean*	*Median*	*Mean*	*Median*
Comparison	0	0.00	727.90	744.50	266.24	194.00	57.63	67.75
Proposed (Ensemble)	0	0.00	23.75	15.00	17.20	15.00	97.85	98.13
Proposed (Single LSTM)	0	25.00	5.00	2.00	2.90	3.00	99.64	99.63

Table 6.14. Confusion Matrix of Stuxnet Attack Detection in Comparison Controller.

	Fault Type Selected				
Correct Fault Type	**980**	**0**	**0**	**0**	1
	0	**0**	**0**	**0**	2
	0	**0**	**0**	**0**	3
	6776	**0**	**3**	**8241**	4
	1	2	3	4	

Table 6.15. Confusion Matrix of Stuxnet Attack Classification of Ensemble Controller.

	Model Selected				
Correct Model	**980**	**0**	**0**	**0**	1
	0	**0**	**0**	**0**	2
	0	**0**	**0**	**0**	3
	58	**268**	**0**	**14676**	4
	1	2	3	4	

The ensemble controller fared much better in correctly selecting the correct model to use during the Stuxnet-like attacks. The 268 cases where model 2 was mistakenly selected over model 4, indicate the effects of the attack on the system led the controller to make decisions as if only the actuator was being affected.

Table 6.16. Confusion Matrix of Stuxnet Attack Classification of LSTM Controller.

	Model Selected				
Correct Model	**980**	**0**	**0**	**0**	1
	0	**0**	**0**	**0**	2
	0	**0**	**0**	**0**	3
	58	**0**	**0**	**14962**	4
	1	2	3	4	

The LSTM controller did not have the same problem as the ensemble controller. Based on the data in Table 6.12 the 58 misclassifications shown here are fully attributable to the 2 or 3 iterations that elapsed in every test between the onset of the attack and the controller selecting the correct model. Based on the information in these confusion matrices, we use the procedures described earlier in the chapter to calculate the micro-F1 score for each controller. Those scores are shown in Table 6.17

Table 6.17. Micro F1 Score for Controllers

Controller	**Micro F1 Score**
Comparison	0.5763
Ensemble	0.9785
LSTM	0.9964

Based on the Stuxnet test results discussed thus far, it is not surprising that the comparison controller has the lowest accuracy of the three when working with the Stuxnet-like attack data. That controller's reliance on detection of an attack specifically designed to avoid detection poses a significant challenge to this and other active fault tolerant controllers.

In the general test section, the ensemble-based controller showed the best accuracy score for dealing with the array and combinations of effects on the system components. Clearly the ensemble-based approach works very well for general purpose classification of the various effects. The single larger LSTM-based controller showed remarkable accuracy in working with the system inputs and interactions specific to the Stuxnet-like effects. This would appear to support the idea that a large deep-learning network can learn the subtle

relationships and interactions between the various input provided to the controller when working with a complex, non-linear system like the quadruple tank system.

6.6 Conclusions

The active comparison controller performed reasonably well in these tests. It is a state-of-the-art, fault tolerant controller which encompasses some of the preferred methods of fault detection and model switching. While it did take a while to detect and respond to the Stuxnet-like attack, the fact that it did so at all is to the designers' credit.

In comparison, both versions of the proposed passive, deep learning-based controller did not suffer from the detection delays demonstrated in the active comparison controller. This confirms one of the touted benefits of passive over active FTC; namely not relying on a detection mechanism to initiate the fault tolerant behavior. Both variants of the proposed controller showed improved performance over the comparison controller as measured by every metric of the testing. This not only showed the efficacy of this controller design, but also the value of the taxonomy representing the various effects of faults and cyber-attacks in training and testing as was done in this research effort.

6.7 Chapter Summary

This chapter reviewed the comparison controller and its application to the quadruple-tank system. The specific design of the comparison controller was presented and the testing approach for the system was described. Results of that testing were also discussed.

The results of testing performed on the quadruple-tank system using the comparison controller and proposed controllers (as described in the previous chapters) has been

presented and discussed with analysis of their performance as controllers and as classifiers. In addition, all three controllers were tested against 20 Stuxnet-like attacks. The wide variety of variables and possible combinations of conditions demonstrated the value of the effects taxonomy as a basis for building useful training data as well as comprehensive test scenarios. The test results support the validity of the approach and show the efficacy of a passive controller design for dealing with multiple simultaneous cyber-attacks and/or faults in an ICS.

7 Conclusions and Recommendations

7.1 Chapter Overview

This chapter summarizes the research documented in this dissertation. The original research questions are revisited and the contributions of the research are presented along with avenues of future work as it relates to this research effort.

7.2 Conclusions of Research

This dissertation presents a deep learning-based, passive controller designed to keep the system under control operating properly, even when multiple components of the system are experiencing the effects of faults and/or attacks. The various effects of these faults and attacks are described and enumerated in a taxonomy so that they can be considered in controller design and used in the testing.

7.2.1 Research Questions Revisited

Question 1. What are the physical layer effects in Cyber Physical System that could result from a component fault or cyber-attack?

Answer 1. In this work we have presented the development and implementation of a unique taxonomy of effects. This taxonomy addresses the possible effects a fault or cyber-attack could have on the physical components (sensors and actuators) of an industrial control system. It provides the building blocks for representing the wide variety of physical layer effects when designing, training and/or testing control systems. By focusing on the physical layer effects, the particular type of cyber-attack or how it occurred become irrelevant.

Question 2. How could a passive Fault Tolerant Controller use deep learning to maintain operations, and how could it be realized?

Answer 2. The controller proposed in this dissertation is a passive Fault Tolerant Controller. The heart of the controller is a Long Short-Term Memory (LSTM) network. By using this deep-learning network, control decisions are made by holistically considering the entire state of the system under control and selecting the next command signal. The command signal is chosen from options generated by internal models of the system. Since no special action is required if an attack or fault is in effect, functioning in the presence of an attack, or fault, are all in the normal course of operations for the controller.

There are two variants of this controller presented. One uses a single large LSTM to make the control decision, while the other uses an ensemble of smaller LSTMs to accomplish the same goal.

The taxonomy of effects described in the answer to Question 1 is pivotal to the realization of this controller design. The LSTM requires large amounts of training data that is representative of the input data that will be available to the network during operations. With the development of the taxonomy, we can represent the various effects of cyber-attacks and faults the system could experience and generate large amounts of data with good variation. This made it possible to realize the passive FTC with restricting it to particular types of faults or attack effects.

Question 3. Will the approach work for both linear and nonlinear systems?

Answer 3. As demonstrated in this research effort, the approach is equally adept at controlling both linear and non-linear systems. The deep-learning network can separate out the differences in the inputs (which are sometimes very subtle) and readily handle the

complexity of a non-linear system. In published literature, some designers have modelled the non-linear system under control as a piecewise linear system to allow their controller to work with it. Because the proposed controller in this dissertation is so capable, there was no need to represent the non-linear system as piecewise linear.

Question 4. What benefit(s) does passive FTC with deep learning provide compared to an active FTC?

Answer 4. The primary benefits of passive FTC, in comparison to active, are:

1. Simpler design. The passive FTC takes input from the system under control and sends a control command at every iteration of the control program regardless of any faults or cyber-attacks on the system. The active controller relies on a dedicated fault detection/identification mechanism to trigger it into reconfiguring the controller or to calculate alternative solutions. These mechanisms add complexity to the active controller design.
2. Minimization of delays. Must take the extra steps described above, there is an inherent delay from the time a fault or attack occurs until the active controller can take corrective action. Depending on the severity of the effect of the fault or attack, this delay could result in serious negative impact to the system to the system. Passive controllers do not suffer from these delays.
3. Resilience to a broad array of effects. One of the greatest perceived limitations of passive controllers is that some knowledge of the faults or attacks the controller will face must me known a priori and passive controllers are, therefore, limited in their utility. By basing this controller design on a deep-learning network, the application of the controller is not limited to certain types

of faults or effects that fall within limited parameters. The deep-learning can be trained to handle any conceivable effect from a fault or cyber-attack. Again, the development of the taxonomy of effects was key to making this so.

7.2.2 Contributions of Research

The primary contribution of this research effort is the novel design of an ICS controller unlike others in current research as follows:

1. The proposed controller is unique in that it uses one or more deep-learning networks and does not require a fault or attack detection/identification function to operate properly. It is a passive design that considers the entire state of the system and decides how to control it based on that information, regardless of either the presence or absence of faults and/or attacks. This deep learning-based approach to passive FTC is not found elsewhere in controls literature.

2. This research challenges the notion that passive controllers cannot be broadly applied because some knowledge of the types of faults and attacks the system may face is required, *a priori*. This dissertation presents a unique taxonomy of effects that provides the building blocks to represent any effect of a fault or cyber-attack on an ICS. This is done by addressing those effects at how they are manifested at the physical component level of the system.

3. The controller presented in this dissertation is not restricted in application to linear or piecewise linear systems. It is also not limited to the type of fault or to a particular component (sensor or actuator) fault. This makes the proposed controller exceptional among passive fault tolerant controllers.

4. The results of the testing demonstrate the deep-learning based passive controller can perform at least as well as its active FTC counterparts. The controller's ability to handle a Stuxnet-like attack on the system shows one of the primary advantages of the proposed controller over the active FTC designs which must rely on the fault detection/identification function in order to operate.

7.3 Recommendations for Future Research

The research presented here can be further explored in a number of different directions.

1. Both example controllers described and tested in this work use a classification type LSTM network to select from among the possible modeled commands to control the system. Using a regression-type deep-learning network to determine the output signal to the controller may allow this to be a purely data-drive (model-free) design. Early efforts to use such a network this in this work were unsuccessful, but do not eliminate the possibility of future success.

2. The use of an ensemble of deep-learning networks requires significant computational overhead, as described in section 5.9. Further work focused on reducing the computational demand of the ensemble or refining a single deep-learning network to perform at the same (or higher) level as the ensemble presented here would be a marked improvement.

3. The application and evaluation of this method to real world systems in a laboratory, or range environments constitutes a logical reason for further continuation of this effort. As pointed out in [111], there is a great need for facilities that incorporate real-

world industrial control systems and processes to provide training on the effects cyber-initiated actions have on physical systems.

7.4 Summary

This chapter has revisited the original research questions and summarized how they were addressed in the dissertation. The contributions of this research in passive fault tolerant control using deep-learning methods have been discussed. Along with the pivotal role played by the developed taxonomy of effects in making this controller design feasible. Additionally, areas for potential further research have also been identified.

8 Bibliography

[1] Bundesampt fur Sicherheit in der Informationstechnik, "Die Lage der IT-Sicherheit in Deutchland," Bundesampt fur Sicherheit in der Informationstechnik, 2014.

[2] K. Zetter, "A Cyberattack Has Caused Confirmed Physical Damage for the Second Time Ever," *Wired Online,* 8 January 2015.

[3] M. Abrams and J. Weiss, "Malicious Control System Cyber Scurity Attack Case Study - Maroochy Water Services," NIST Computer Security Resource Center, Gaithersburg, MD., 2008.

[4] D. E. Sanger, "Utilities Cautioned About Potential for a Cyberattack After Ukraine's," New York Times, New York, NY., 2016.

[5] United States Government Accounting Office, "DEFENSE INFRASTRUCTURE: Management Issues Requiring Attention in Utility Privatization," GAO, Washington D.C., 2005.

[6] Cyber Security Research Alliance, "Designed-In Cyber Security for Cyber-Physical Systems," Cyber Security Research Alliance, Gaithersburg, MD, 2013.

[7] V. M. Igure, S. A. Laughter and R. D. Williams, "Security issues in SCADA networks," *Computers & Security,* vol. 25, no. 6, pp. 498-506, 2006.

[8] D. H. Ryu, H. J. Kim and K. Um, "Reducing security vulnerabilities for critical infrastructure," *Journal of Loss Prevention in the Process Industries,* vol. 22, pp. 1020-1024, 2009.

[9] E. Byers and J. Lowe, "The Myths and Facts behind Cyber Security Risks for Industrial Control Systems," in *Proceedings of the Verband der Elektrotechnik, Elektronik und Informationstechnik (VDE) Congress*, Frankfurt am Main, Germany, 2004.

[10] J. Jiang and X. Yu, "Fault-tolerant control systems: A comparative study between active and passive approaches," *Annual Reviews in Control,* vol. 36, no. 1, pp. 60-72, 2012.

[11] J. F. Combita, J. Giraldo, A. A. Cardenas and N. Quijano, "Response and Reconfiguration of Cyber-Physical Control Systems: A survey," in *IEEE 2nd Colombian Conference on Automatic Control (CCAC)*, Manizales, Colombia, 2015.

[12] I. Hameed, E. Elmadbouly and M. 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.

[13] M. Blanke, C. W. Frei, F. Kraus, R. J. Patton and M. Staroswiecki, "What is Fault-Tolerant Control?," in *Proceedings of the IFAC Fault Detection, Supervision and Safety for Technical Processes*, Budapest, Hungary, 2000.

[14] Y. M. Zhang and J. Jiang, "Issues on integration of fault diagnosis and reconfigurable control in active fault-tolerant control systems," in *IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes*, Beijing, China, 2006.

[15] S. M. Tabatabaeipour, *Fault Diagnosis and Fault Tolerant Control of Hybrid Systems*, Aalborg, Denmark: Aalborg University, 2010.

[16] M. Staroswiecki, "Actuator faults and the linear quadratic control problem," in *Proceedings of the 42nd IEEE International Conference on Decision & Control*, Maui, HI., 2003.

[17] R. Veillette, "Reliable linear-quadratic state-feedback control," *Automatica*, vol. 31, no. 1, pp. 137-144, 1995.

[18] F. Liao, J. Wang and G. Yang, "Reliable robust flight tracking control: an LMI approach," *IEEE Transactions on Control Systems Technology*, vol. 10, no. 1, pp. 76-89, 2002.

[19] G. Yang, J. Wang and Y. Soh, "Reliable H-infinity controller design for linear systems," *Automatica*, vol. 37, no. 5, pp. 717-725, 2001.

[20] G. H. Yang, J. Lam and J. Wang, "Reliable H-infinity control for affine nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 43, no. 8, pp. 1112-1117, 1998.

[21] G. H. Yang, J. Wang and Y. Soh, "Reliable guaranteed cost control for uncertain nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 45, no. 11, pp. 2188-2192, 2000.

[22] R. Wang, M. Liu and J. Zhao, "Reliable H∞ control for a class of switched nonlinear systems with actuator failures," *Nonlinear Analysis: Hybrid Systems*, vol. 1, no. 2, pp. 317-325, 2007.

[23] G. Feng, "Controller design and analysis of uncertain piecewise-linear systems," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 49, no. 2, pp. 224-232, 2002.

[24] J. Zhang and W. Tang, "Output feedback H∞ control for uncertain piecewise linear systems," *Journal of Dynamical and Control Systems*, vol. 14, no. 1, pp. 121-144, 2008.

[25] G. H. Yang, S. Y. Zhang, J. Lam and J. L. Wang, "Reliable control using redundant controllers," *IEEE Transactions on Automatic Control*, vol. 43, no. 11, pp. 1588-1593, 1998.

[26] A. Escada and M. Boukhnifer, "Experimental second order sliding mode fault tolerant control for moment gyroscope system with sensor fault," in *International Conference on Control, Decision and Information Technology (CoDIT'14)*, Metz, France, 2014.

[27] A. Ghodebane, M. Saad, C. Hobeika, J. F. Boland and C. and Thibeault, "Design of a Tolerant Flight Control System in Response to Multiple Actuator Control Signal Faults Induced by Cosmic Rays," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 52, no. 2, pp. 681-697, 2016.

[28] M. Polas and A. Fekih, "A Multi-Gain Sliding Mode Based Controller for the Pitch Angle Control of a Civil Aircraft," in *IEEE South Eastern Symposium on System Theory*, Tyler, TX., 2010.

[29] R. Ghazali, M. Rahmat and A. Hashim, "Performance Comparison between Sliding Mode Control with PID Sliding Surface and PID Controller for an Electro-hydraulic Positioning System," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 1, no. 4, pp. 447-452, 2011.

[30] S. Spurgeon, "Sliding mode control: a tutorial," in *2014 European Control Conference (ECC)*, Strasbourg, France, 2014.

[31] A. Fekih and P. Pilla, "A Passive Fault Tolerant Control Strategy for the uncertain MIMO Aircraft Model F-18," in *38th Southeastern Symposium on System Theory*, Macon, GA., 2007.

[32] U. Vaidya and M. Fardad, "On optimal sensor placement for mitigation of vulnerabilities to cyberattacks in large-scale networks," in *Proceedings of the 2013 European Control Conference (ECC)*, Zurich, Switzerland, 2013.

[33] O. Vukovic, K. Sou, G. Dan and H. Sandberg, "Network-aware mitigation of data integrity attacks on power system state estimation," *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 6, pp. 1108-1118, 2012.

[34] Y. Mo and B. Sinopoli, "Secure estimation in the presence of integrity attacks," *IEEE Transactions on Automatic Control*, vol. 60, no. 4, pp. 1145-1151, 2015.

[35] Y. Yang, Y.-J. Wang, J.-Q. Qiu and Y.-R. Niu, "Robust H∞ Passive Fault-Tolerant Control for Uncertain Singular Systems," in *Proceedings of the 2011 International Conference on Machine Learning and Cybernetics*, Guilin, China, 2011.

[36] A. Fekih, "Fault Tolerant Control Design for Complex Systems: Current Advances and Open research Problems," in *IEEE International Conference on Industrial Technology (ICIT)*, Seville, Spain, 2015.

[37] M. Buciakowski, M. Witczak and J. Korbicz, "A quadratic boundedness approach to fault tolerant control for nonlinear system," *IFAC PapersOnLine*, vol. 50, no. 1, pp. 2101-2106, 2017.

[38] N. Bedioui, R. Houimli and M. Besbes, "Adaptive Observer Design for Actuator and Sensor Faults Estimations: VTOL Air Craft System Application," in *Proceedings of the 2019 IEEE International Conference on Advanced Systems and Emergent Technologies (IC_ASET)*, Hammamet, Tunisia, 2019.

[39] L. Xiao, Z. Meng, X. Huang and L. Ma, "Adaptive Observer based Fault Tolerant Control for Aircraft Engine with Sensors and Actuators Faults," in *Proceedings of the 38th Chinese Control Conference*, Guangzhou, China, 2019.

[40] Z. Qu, B. Dahhou and G. Roux, "Fault Tolerant control system based on subspace predictive control and multiple model predictive control," in *IEEE Proceedings of the 3rd Conference on Control and Fault-Tolerant Systems (SysTol)*, Barcelona, Spain, 2016.

[41] S. Narasimhan and G. Biswas, "Model-based diagnosis of hybrid systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part A,* vol. 37, no. 3, pp. 348-361, 2007.

[42] H. Yang, V. Cocquempot and B. Jiang, "Robust fault tolerant tracking control with application to hybrid nonlinear systems," *IET Control Theory and Applications,* vol. 3, no. 2, pp. 211-224, 2011.

[43] A. R. Kodakkadan, V. Reppa and S. Olaru, "Switching-stable control mechanism in the presence of guaranteed detectable sensor faults," in *3rd Conference on Control and Fault-Tolerant Systems (SysTol),* Barcelona, Spain, 2016.

[44] A. Arici and T. Kara, "Model Reference Adaptive Control of a Quadruple Tank Process with Actuator Faults," in *Proceedings of the 10th International Conference On Electrical And Electronics Engineering,* Bursa, Turkey, 2017.

[45] D. Xin and G.-H. Yang, "An adaptive approach to state feedback tracking control of systems with actuator failures," in *Proceedings of the American Control Conference,* New York, NY., 2007.

[46] G. Tao, S. Chen and S. M. Joshi, "An adaptive actuator failure compensation controller using output feedback," *IEEE Transactions on Automatic Control,* vol. 47, no. 3, pp. 506-511, 2002.

[47] D. B. Doman and A. D. Ngo, "Dynamic inversion-based adaptive reconfigurable control of the X-33 on ascent," *Journal of Guidance, Control, and Dynamics,* vol. 25, no. 2, pp. 275-284, 2002.

[48] T. Hartmann, F. Fouquet, J. Klein, G. Nain and Y. Le Traon, "Reactive security for smart grids using models@run. time-based simulation and reasoning," in *Smart Grid Security,* New York, NY., Springer, 2014, pp. 139-153.

[49] A. A. Cárdenas, S. Amin, Z. S. Lin, Y. L. Huang, C. Y. Huang and S. Sastry, "Attacks against process control systems: Risk assessment detection and response," in *Proceedings of the 6th ACM Symposium on Information Computer and Communications Security (ASIACCS '11),* Hong Kong, China, 2011.

[50] L. Sha, "Using simplicity to control complexity," *IEEE Software,* vol. 18, no. 4, pp. 20-28, 2001.

[51] Q. Chen and S. Abdelwahed, "A model-based approach to self-protection in scada systems," in *Proceedings of the 9th International Workshop on Feedback Computing,* Philadelphia, PA., 2014.

[52] S. Sridhar and M. Govindarasu, "Model-based attack detection and mitigation for automatic generation control," *IEEE Transactions on Smart Grid,* vol. 5, no. 2, pp. 580-591, 2014.

[53] R. Isermann, "Fault diagnosis of machines via parameter estimation and knowledge processing–tutorial paper," *Automatica,* vol. 29, no. 4, pp. 815-835, 1993.

[54] Z. S. Hou and Z. Wang, "From Model-Based to Data-Driven Control: Survey, Classification and Perspective," *Information Sciences,* vol. 235, pp. 3-35, 2013.

[55] Z. S. Hou and J. X. Xu, "On data-driven control theory: the state of the art and perspective," *Acta Automatica Sinica,* vol. 35, no. 6, pp. 650-667, 2009.

[56] B. Huang and R. Kadali, Dynamic Modeling, Predictive Control and Performance Monitoring: A Data-Driven Subspace Approach, London, UK: Springer, 2008.

[57] W. Ma, X. Li, J. Wang, B. Wang and H. Lin, "Data-driven Unfalsification based Robust Fault Tolerant Control," in *Proceedings of the 35th Chinese Control Conference,* Chengdu, China, 2016.

[58] H. Izadi, B. Gordon and Y. Zhang, "A Data-Driven Fault Tolerant Model Predictive Control with Fault Identification," in *Proceedings of 2010 Conference on Control and Fault Tolerant Systems*, Nice, France, 2010.

[59] J. Wang and G. Yang, "Data-Driven Output-Feedback Fault-Tolerant Compensation Control for Digital PID Control Systems With Unknown Dynamics," *IEEE Transactions On Industrial Electronics,* vol. 63, no. 11, pp. 7029-7039, 2016.

[60] Z. Li and G.-H. Yang, "Data-driven adaptive fault-tolerant control for a class of multiple-input–multiple-output linear discrete-time system," *IET Control Theory Applications,* vol. 11, no. 16, pp. 2824-2833, 2017.

[61] L. Meneghetti, M. Terzi, S. Del Favero, G. Susto and C. Cobelli, "Data-Driven Anomaly Recognition for Unsupervised Model-Free Fault Detection in Artificial Pancreas," *IEEE Transactions on Control Systems Technology,* vol. 8, no. 1, pp. 33-47, 2020.

[62] H. Xie, C. Hu, Y. Zhang, X. Yang and Y. Li, "Comparison of Two Performance Optimization Approaches for Data-driven Design of Fault-Tolerant Control Systems," in *IEEE 27th International Symposium on Industrial Electronics (ISIE)*, Cairns, Australia, 2018.

[63] W. Kwong, K. Passino, E. Laukonen and S. Yurkovich, "Expert supervision of fuzzy learning systems for fault tolerant aircraft control," *Proceedings of the IEEE,* vol. 83, no. 3, pp. 466-483, 1995.

[64] K. Passino and S. Yurkovich, Fuzzy Control, Boston, MA.: Addison-Wesley Longman, 1997.

[65] M. Holmes and A. Ray, "Fuzzy damage-mitigating control of a fossil power plant," *IEEE Transactions on Control Systems Technology,* vol. 9, no. 1, pp. 140-147, 2001.

[66] Q. Song and Y.-D. Song, "Data-Based Fault-Tolerant Control of High-Speed Trains with Traction Braking Notch Nonlinearities and Actuator Failures," *IEEE Transactions on Neural Networks,* vol. 2, no. 12, pp. 2250-2261, 2011.

[67] H. Patel and V. Shah, "Passive Fault-Tolerant Tracking for Nonlinear System with intermittent Fault and Time Delay," *IFAC PapersOnLine,* vol. 52, no. 11, pp. 200-205, 2019.

[68] F. Bastiani and I. Chen, "The role of artificial Intelligence in fault-tolerant process-control systems," in *Proceedings of the 1st international conference on Industrial and engineering applications of artificial intelligence and expert systems*, Tullahoma, TN., 1988.

[69] D. Yu, T. Chang and D. Yu, "Adaptive neural model-based fault tolerant control for multi-variable processes," *Engineering Applications of Artificial Intelligence*, vol. 18, no. 4, pp. 393-411, 2005.

[70] M. Polyarpu and A. Vemuri, "Learning Approach to Fault Tolerant Control: an Overview," in *Proceedings of the 1998 IEEE ISIC/CIRA/ISAS Joint Conference*, Gaithersburg, MD., 1998.

[71] P. Ballé, M. Fischera, D. Fussel, O. Nells and R. Isermann, "Integrated control, diagnosis and reconfiguration of a heat exchanger," *IEEE Control Systems Magazine*, vol. 18, no. 3, pp. 52-63, 1998.

[72] X. Y. Zhang, H.-Y. Su and J. Chu, "Adaptive Sliding Mode-Like Fuzzy Logic Control for High-Order Nonlinear Systems," in *Proceedings of the IEEE International Symposium on Intelligent Control*, Houston, TX., 2003.

[73] B. R. Marquez, A. G. Loukianov and E. N. Sanchez, "Hierarchical Intelligent Sliding Mode Control: Application to Stepper Motors," in *Proceeding of the 2004 American Control Conference*, Boston, MA., 2004.

[74] P. Schroder, A. Chipperfield, P. Fleming and N. Grum, "Fault tolerant control of active magnetic bearings," in *IEEE International Symposium on Industrial Electronics*, Pretoria, South Africa, 1998.

[75] E. Sugawara, M. Fukushi and S. Horiguchi, "Fault Tolerant Multi-layer Neural Networks with GA Training," in *The 18th IEEE International Symposium on Defect and Fault Tolerance in VLSI systems*, Boston, MA., 2003.

[76] A. Ashari, M. Yazdanpanah and A. Sedigh, "Reconfigurable sliding-mode control design using genetic algorithms and eigenstructure assignment," in *Proceedings of the International Conference on Control and Automation*, Budapest, Hungary, 2005.

[77] F. Lv, C. Wen, Z. Bao and M. Liu, "Fault diagnosis based on deep learning," in *Proceedings of the IEEE 2016 American Control Conference (ACC)*, Boston, Mass., 2016.

[78] B. Eroglu, C. Sahin, B. Yuksek, N. K. Ure and G. Inalhan, "Deep Recurrent and Convolutional Networks for Accelerated Fault Tolerant Adaptive Flight Control Under Severe Failures," in *Proceedings of the 2018 Annual American Control Conference (ACC)*, Milwaukee, WI., 2018.

[79] K.-P. Lee, B.-H. Wu and S.-L. Peng, "Deep-learning-based fault detection and diagnosis of air-handling units," *Building and Environment*, vol. 157, pp. 24-33, 2019.

[80] R. Iqbal, T. Maniak, F. Doctor and C. Karyotis, "Fault Detection and Isolation in Industrial Processes Using Deep Learning Approaches," *IEEE Transactions On Industrial Informatics*, vol. 15, no. 5, pp. 3077-3084, 2019.

[81] W. Liu and Z. Hu, "Aero-engine Sensor Fault Diagnosis Based on Convolutional Neural Network," in *Proceedings of the 2019 Chinese Control And Decision Conference (CCDC)*, Nanchang, China, 2019.

[82] L. Gao, D. Li, D. Li and H. Yin, "An Improved LSTM Based Sensor Fault Diagnosis Strategy for the Air-cooled Chiller System," in *Proceedings of the 38th Chinese Control Conference*, Guangzhou, China, 2019.

[83] F. Aznar, M. Pujol and R. Rizo, "Obtaining fault tolerance avoidance behavior using deep reinforcement learning," *Neurocomputing*, vol. 345, pp. 77-91, 2019.

[84] D.-T. Hoang and H.-J. Kang, "A Survey on Deep Learning based Bearing Fault Diagnosis," *Neurocomputing*, vol. 335, pp. 327-335, 2019.

[85] P. F. Odgaard, J. Stoutstrup and M. Kinnaert, "Fault Tolerant Control of Wind Turbines - a benchmark model," in *Proceedings of the 7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, Barcelona, Spain, 2009.

[86] X. Liu and Y.-D. Song, "Robust adaptive fault-tolerant control of dynamic systems with floating nonlinearities and fading actuators," in *Proceedings of the 2011 Chinese Control Conference*, Yantai, China, 2011.

[87] V. Gomathi, S. Muthumari, V. M. Nivedita and P. Vaishnavi, "Fault diagnosis in a Quadruple tank system using Kalman Filter," Proceedings of the 2017 International Conference on Computation of Power, Energy, Information and Communication (ICCPEIC), Melmaruvathur, Tamilnadu, India, 2017.

[88] B. Potteiger, W. Emfinger, H. Neema, X. Koutosukos and C. Tang, "Evaluating the effects of cyber-attacks on cyber physical systems using a hardware-in-the-loop simulation testbed," in *2017 Resilience Week (RWS)*, Wimington, DE., 2017.

[89] A. Humayed, J. Lin, F. Li and B. Luo, "Cyber-Physical Systems Security—A Survey," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1802-1831, 2017.

[90] I. Friedberg, K. McLaughlin, P. Smith, D. Laverty and S. Sezer, "STPA-SafeSec: Safety and security analysis for cyber-physical systems," *Journal of Information Security and Applications*, vol. 34, no. 2, pp. 183-196, 2017.

[91] D. Wardell, R. Mills, G. Peterson and M. Oxley, "A Method for Revealing and Addressing Security Vulnerabilities in Cyber-physical Systems by Modeling Malicious Agent Interactions with Formal Verification," in *Procedia Computer Science*, Redondo Beach, CA., 2016.

[92] M. Chiply, "WBDG," 21 Feb 2020. [Online]. Available: https://www.wbdg.org/resources/cybersecurity. [Accessed 12 Sept 2020].

[93] International Electrotechnical Commission, "IEC 61158-1, Ed. 2.0, Industrial communication networks - Fieldbus specifications," IEC, Zurich, 2019.

[94] I. N. Fovino, A. Carcano, M. Masera and A. Trombetta, "An Experimental investigation of malware attacks on SCADA systems," *International Journal of Critical Infrastructure Protection*, vol. 2, pp. 139-145, 2009.

[95] S. Gallagher, "ARS Technica," 25 January 2018. [Online]. Available: https://arstechnica.com/information-technology/2018/01/the-internet-of-omg-vulnerable-factory-and-power-grid-controls-on-internet/. [Accessed 20 February 2019].

[96] J. Wang, "Liquid Level Sensing Using Capacitive-to-Digital Converters," Analog, 15 April 2015. [Online]. Available: www,analog.com/en/analogue-dialogue/articles/liquid-level-sensing-using-cdcs.html. [Accessed 24 Sept 2020].

[97] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735-1780, 1997.

[98] J. Kafunah, "Vanishing and Exploding Gradient Problems," 21 May 2018. [Online]. Available: https://www.jefkine.com/general/2018/05/21/2018-05-21-vanishing-and-exploding-gradient-problems/. [Accessed 16 Nov 2019].

[99] MathWorks, "Long Short-Term Memory (LSTM)," MathWorks, undated. [Online]. Available: https://www.mathworks.com/discovery/lstm.html?s_tid=srchtitle. [Accessed 14 November 2019].

[100] MathWorks, "Help Center - Long Short-Term Memory Networks," MathWorks, undated. [Online]. Available: https://www.mathworks.com/help/deeplearning/ug/long-short-term-memory-networks.html. [Accessed 14 November 2019].

[101] M. Dwarampudi and N. Reddy, "Effects of padding on LSTMs and CNNs," 18 March 2019. [Online]. Available: https://arxiv.org/abs/1903.07288. [Accessed 2 December 2019].

[102] ELPROCUS, "The Working Principles of a PID Controller Beginners," ELPROCUS, 2020. [Online]. Available: https://www.elprocus.com/the-working-of-a-pid-controller/. [Accessed 13 Sept 2020].

[103] H. R. Bonab and F. Can, "A Theoretical Framework on the Ideal Number of Classifiers for Online Ensembles in Data Streams," in *Proceedings on the 2016 Conference on Information and Knowledge Management*, Indianapolis, IN., 2016.

[104] J.-R. Abrial, E. Boerger and H. Langmaack, "The Steam Boiler Control Specification Problem," 14 November 1996. [Online]. Available: http://www.informatik.unikiel.de/~procos/dag9523/steam-boiler-problem.ps.Z. [Accessed 22 June 2015].

[105] J.-R. Abrial, "Additional Information Concerning the Dynamic Behaviour of the Steam Boiler," 14 November 1996. [Online]. Available: http://www.informatik.uni-kiel.de/~procos/dag9523/steam-boiler-additional-info.ps.Z. [Accessed 22 June 2015].

[106] K. Johansson, A. Horch, O. Wijk and A. Hansson, "Teaching Multivariable Control Using the Quadruple-Tank Process," in *Proceedings of the 38th IEEE Conference on Decision & Control*, Phoenix, AZ., 1999.

[107] R. S. M. Malar and T. Thyagarajan, "Design of Decentralized Fuzzy Controllers for Quadruple Tank Process," *IJCSNS International Journal of Computer Science and Network Security*, vol. 8, no. 11, pp. 163-168, 2008.

[108] G. Park, C. Lee and H. Shim, "On Stealthiness of Zero-dynamics Attacks against Uncertain Nonlinear Systems: A Case study with Quadruple-tank Process," in *Proceedings of the 23rd International Symposium on Mathematical Theory of Networks and Systems*, Hong Kong, China, 2018.

[109] N. Rezk, M. Purnaprajna, T. Nordstrom and Z. Ul-Abdin, "Recurrent Neural Networks: An Embedded Computing Perspective," *IEEE Access*, vol. 8, pp. 57967-57996, 2020.

[110] J. Opitz and S. Burst, "Macro F1 and Macro F1," 20 Nov 2019. [Online]. Available: https://arxiv.org/pdf/1911.03347.pdf. [Accessed 10 Oct 2020].

[111] J. Butts and M. Glover, "How Industrial Control System Security Training is Falling Short," in *Critical Infrastructure Protection IX, 9th IFIP 11.10 International Conference, ICCIP*, Arlington, VA., 2015.

A. Appendix: Additional References

Active Fault Tolerant Controls

Principle component analysis: the work in [A1] presents an approach to process and sensor fault detection, identification, and reconstruction via principal component analysis. The principal component analysis model partitions the measurement space into a principal component subspace where normal variation occurs, and a residual subspace that faults may occupy. Both process faults and sensor faults are characterized by a direction vector, which describes the behavior of the fault. Fault reconstruction is accomplished by sliding the sample vector as close as possible to the principal component subspace. When the actual fault is assumed, the maximum reduction in the squared prediction error is achieved. A fault-identification index is defined in terms of the reconstructed squared prediction error.

State Space Subspace approach to fault detection is presented in [A2] where a method is proposed to detect, isolate, and compensate sensor degradation. A numerical algorithm for subspace state space system identification is used to track the changes of the time constants and gains of the sensor and the monitored system. Without requiring redundant sensors and measurement devices, this method utilizes the fact that sensor readings depict dynamic characteristics of the sensors as well as those of the monitored system. The method is verified in angular sensor degradation detection using high-fidelity simulations of an automotive electronic throttle system. This method is applied to linear systems only. This method is not model-based, but uses input and output data. The significance of this distinction is discussed in section 2.2.2.

Control Allocation [A3], [A4], [A5], [A6] takes on the problem of generating a desired control effort and distributing it among multiple (usually redundant) actuators. This method is popular in the field of aircraft control design. Control allocation was used for multiple fault detection in [A7]. This work requires the addition of redundant smart actuators which are networked together.

In systems based on **game theory**, the game between the attacker and defender can be simultaneous (e.g., minimax) [A8] in which each side chooses an action based on the assumed strategy of their adversary, or sequential (e.g., Stackelberg game) [A9] where one side responds with an action based on the adversaries last action.

[A1] R. Dunia and S. J. Qin, “Joint diagnosis of process and sensor faults using principal component analysis,” *Control Eng. Pract.*, vol. 6, no. 4, pp. 457–469, Apr. 1998.

[A2] L. Jiang, D. Djurdjanovic, and J. Ni, “A new method for sensor degradation detection, isolation and compensation in linear systems,” in *Proc.ASME IMECE,* Nov. 2007, vol. 9, pp. 1089–1101.

[A3] Y. Luo, A. Serrani, S. Yurkovich, "Model predictive dynamic control allocation scheme for reentry vehicles", *Journal of Guidance Control and Dynamics*, vol. 30, no. 1, pp. 100-113, 2007.

[A4] Q. Wang, L. Qing, C. Wong, S. Tingyan, "Robust control allocation method in the presence of control effector failure", *Proc. of IEEE Int. Conf. on Information Automation*, pp. 660-664, 2014.

[A5] Y. Zhang, C.A. Rabbath, C.Y. Su, "Reconfigurable control allocation applied to an aircraft benchmark model", *Proc. of American Control Conference*, pp. 1052-1057, 2008.

[A6] D. Chen and G. Liu, "An adaptive control allocation algorithm for nonlinear vehicles with parameter uncertainty," *2017 11th Asian Control Conference (ASCC)*, Gold Coast, QLD, 2017, pp. 982-987.
doi: 10.1109/ASCC.2017.8287304

[A7] Inseok Yang and Dongik Lee, "Networked Fault-Tolerant Control Allocation for Multiple Actuator Failures", *Mathematical Problems in Engineering*, Volume 2015, Article ID 249293, 14 pages
http://dx.doi.org/10.1155/2015/249293

[A8] A. Farraj, E. Hammad, A. A. Daoud, D. Kundur, "A game-theoretic control approach to mitigate cyber switching attacks in smart grid systems", *Proceedings of the IEEE Smart Grid Communications*, pp. 958-963, 2014.

[A9] Q. Zhu, T. Basar, "Game-theoretic methods for robustness security and resilience of cyberphysical control systems: Games-in-games principle for optimal cross-layer resilient control systems", *IEEE Control Systems*, vol. 35, no. 1, pp. 46-65, 2015.

[A10] T. Baúar, P. Bernard, H∞ Optimal Control and Related Minimax Design Problems, Birkhäuser, 1995.

[A11] Y. Yuan, F. Sun, H. Liu, "Resilient control of cyber-physical systems against intelligent attacker: a hierarchal stackelberg game approach", *To appear on International Journal of Systems Science*, 2015.

[A12] A. Barth, B. Rubinstein, M. Sundararajan, J. Mitchell, D. Song, P. Bartlett, "A learning-based approach to reactive security", *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 4, pp. 482-493, July 2012.

[A13] D. Shelar, S. Amin, "Analyzing vulnerability of electricity distribution networks to der disruptions", *American Control Conference (ACC) 2015*, pp. 2461-2468, 2015.

[A14] P. Frank and X. Ding. Survey of robust residual generation and evaluation methods in observer-based fault detection systems. *Journal of Process Control*, 7(6):403–424, 1997.

[A15] M.E. Romero, M.M. Seron, "Sensor fault tolerant control of doubly fed induction machines using multiple switching mechanisms", *Proceedings of Australian Control Conference*, pp. 191-197, 2011.

[A16] S. Aberkane, J.C. Ponsart, D. Sauter, "Robust output feedback stochastic stabilization of active fault tolerant control systems", *Proc. of IEEE Inter. Symposium on Intelligent Control*, pp. 1561-1566, 2005.

[A17] S. Aberkane, D. Sauter, J.C. Ponsart, D. Theilliol, "H∞ stochastic stabilization of active fault tolerant control systems: convex approach", *Proc. of IEEE European Control Conference on Decision and Control*, pp. 3783-3788, 2005.

[A18] A. Fekih, S. Seelem, "A Fault Tolerant Control Design for Automatic Steering Control of Ground Vehicles", *Proceedings of IEEE International Conference on Control Applications*, pp. 1491-1496, 2012.

[A19] B.J. Bacon, A.J. Ostroff, S.M. Joshi, "Reconfigurable NDI controller using inertial sensor failure detection and isolation", *IEEE Transactions on Aerospace and Electrical Systems*, vol. 37, no. 4, pp. 1373-1383, 2001.

[A20] Y. Bo, Y. Jun, "Parameter space approach for active fault tolerant control design", *Proc. of Intl. Conference on Intelligent Computation Technology and Automation*, vol. 1, pp. 409-412, 2008.

[A21] J.D. Boskovic, R.K. Mehra, S.H. Yu, "A decentralized scheme for accommodation of multiple simultaneous actuator failures", *Proc. of American Control Conference*, pp. 5098-5103, 2002.

[A22] J. Chen, R.J. Patton, Z. Chen, "Active fault tolerant flight control systems design using the linear matrix inequality method", *Trans. of Institute of the Measurement and Control*, vol. 21, pp. 77-84, 1999.

[A23] S. Chen, G. Tao, S.M. Joshi, "On matching conditions for adaptive state tracking control of systems with actuator failures", *IEEE Transactions on Automatic Control*, vol. 47, no. 3, pp. 473-478, 2002.

[A24] D.R. Espinoza-Trejo, D.U. Campos-Delgado, "Active fault-tolerant scheme for variable speed drives under actuator and sensor faults", *Proceedings of IEEE International Conference on Control Applications*, pp. 474-479, 2008.

[A25] S. Ganguly, A. Marcos, G. Balas, "Reconfigurable LPV control design for Boeing pp.747-100/200 longitudinal Axis", *Proc. of American Control Conference*, pp. 3612-3617, 2002.

[A26] S. Kanev, M. Verhaegen, "A bank of reconfigurable LQG controllers for linear system subject to failures", *Proc. of Conf. on Decision and Control*, pp. 3684-3689, 2000.

[A27] G. Liu, D. Wang, Y. Li, "Active fault-tolerant control with actuation reconfiguration", *IEEE Trans. on Aerospace and Electronic Systems* 40(3), pp. 1110-1117, 2004.

[A28] Y.W. Kim, G. Rizzoni, V.I. Utkin, "Developing a fault tolerant power train control system by integrating design of control and diagnostics*", International. Journal of Guidance. Control and Dynamics*, vol. 11, no. 11, pp. 1095-1114, 2001.

[A29] M. Mahmoud, J. Jiang, Y.M. Zhang, "Stabilization of active fault tolerant control systems with imperfect fault detection and diagnosis", *Journal of Stochastic Analysis and Applications*, vol. 21, no. 3, pp. 673-701, 2003.

[A30] M. Maki, J. Jiang, K.A. Hagino, "Stability guaranteed active fault-tolerant control against actuator failures", *International Journal of Robust and Nonlinear Control*, vol. 14, no. 2, pp. 1061-1077, 2004.

[A31] H. Niemann, "Fault tolerant control based on active fault diagnosis", *Proc. of American Control Conference*, pp. 2224-2229, 2005.

[A32] G. Tao, S.M. Joshi, X. Ma, "Adaptive state feedback and tracking control of systems with actuator failures", *IEEE Trans. on Autom. Control*, vol. 46, no. 1, pp. 78-95, 2001.

[A33] Z. Weng, R.J. Patton, P. Cui, "Active fault tolerant control of a double inverted pendulum", *Proc. of the Institution of Mech. Eng. Part I: J. of Sys. & Cont. Eng*., vol. 221, no. 6, pp. 895-904, 2007.

[A34] W. Junsheng, W. Zhengxin, T. Zuohua, S. Songjiao, "An active fault tolerant controller design method for time delay systems", *Proc. of Chinese Control Conference*, pp. 422-425, 2007.

[A35] A. Fekih, Ben J. Hmida, "A self tuning adaptive approach for actuator failure compensation in aircraft systems", *IEEE Int. Conf. on Industrial Technology*, pp. 10-14, March 14-17 2010.

[A36] Y. Zhang, J. Jiang, "Integrated active fault-tolerant control using IMM approach", *IEEE Trans. on Aerospace and Electronics Systems*, vol. 37, no. 4, pp. 1221-1235, 2001.

[A37] Y. Zhang, J. Jiang, "Issues on integration of fault diagnosis and reconfigurable control in active fault-tolerant control systems", *Symposium on Fault Detection Supervision and Safety of Technical Processes*, pp. 1437-1448, 2006.

[A38] A.L. Gehin, M. Assas, M. Staroswiecki, "Structural Analysis of system reconfigurability", *Symposium on Fault Detection Supervision and Safety for Technical Processes,* pp. 292-297, 2000.

[A39] G. Bajpai, B.C. Chang, A. Lau, "Reconfiguration of Flight Control Systems for Actuator Failures", *IEEE Aerospace and Electronic Systems Mag*., vol. 16, no. 9, pp. 29-33, 2000.

[A40] L-W. Ho, G.Y. Gary, "Reconfigurable control system Design for fault diagnosis and accommodation", *Proc. of IEEE Conference on Decision and Control*, pp. 1873-1878, 2001.

[A41] Wang Dan, Wu Zhiliang, Yao Yubin, Niu Xiaobing, "An FDI Approach for Aircraft Actuator Partial Failure", *Proc. of Chinese Control Conference*, pp. 440-444, 2007.

[A42] Y. Zhang, V. S. Suresh, B. Jiang, D. Theilliol, "Reconfigurable control allocation against aircraft control effector failures", *Proc. of IEEE Int. Conference on Control Applications*, pp. 1197-1202, 2007.

[A43] H. Alawi, C. Edwards, "Fault tolerant sliding mode control design with piloted simulator evaluation", *Journal of Guidance Control and Dynamics*, vol. 31, no. 5, pp. 1186-1201, 2008.

[A44] A. Fekih, "Integrated fault tolerant flight control design with application to an F-16 aircraft" in *Journal of Control Science and Engineering*, ACTA Press, vol. 39, no. 3, pp. 159-167, 2011.

[A45] J.D. Boskovic, N. Knoebel, R.K. Mehra, "An initial study of a combined robust and adaptive control approach to guaranteed performance flight control", *Proc. of American Control Conference*, pp. 5150-5155, 2008.

[A46] D. Shin, G. Moon, Y. Kim, "Design of reconfigurable flight control system using adaptive sliding mode control: actuator fault", *Proc. of the Institute of Mechanical Engineers Part G. Journal of Aerospace Engineering*, vol. 219, pp. 321-328, 2005.

[A47] A. Fekih, "Effective fault tolerant control design for a class of nonlinear systems: Application to a class of motor control", *IET Control Theory & Applications*, vol. 2(9), pp. 762-772, 2008.

[A48] J. Shtessel, J. Buffington, S. Banda, "Tailless aircraft flight control using multiple time scale re-configurable sliding modes", *IEEE Trans. on Contr. Syst. Technology*, vol. 10, pp. 288-296, 2002.

[A49] S.R. Wells, R. A. Hess, "Multi-input/multi-output sliding mode control for a tailless fighter aircraft", *Journal of Guidance Control and Dynamics*, vol. 26, pp. 463-473, 2003.

[A50] X. Yu, Y. Li, X. Wang, K. Zhao, "An autonomous robust fault tolerant control system", *Proc. of IEEE Conf. on Information Acquisition*, pp. 1191-1196, 2006.

[A51] Hwang, Insoek, Sungwan, Kim, Kim, Youdan and Seah, Chze Eng, "A Survey of Fault Detection, Isolation, and Reconfiguration Methods", *IEEE Transactions on Control Systems Technology*, Vol. 18, No. 3, May 2010, pages 636-653.

[A52] Beard, R., Failure accommodation in linear systems through self-recognition, PhD Thesis, MIT, 1971.

[A53] R. Patton and J. Chen. Observer-based fault detection and isolation: robustness and applications. *Control Eng. Prac*., 5(5):671–682, 1997.

[A54] S. Simani, C. Fantuzzi, and R. J Patton. Model-based fault diagnosis in dynamic systems using identification techniques. *Springer Science & Business Media*, 2003.

[A55] P. Frank. Advances in observer-based fault diagnosis. In *International Conference on Fault Diagnosis: TOOLDIAG*, 1993.

[A56] H. Sneider and P. M. Frank. Observer-based supervision and fault detection in robots using nonlinear and fuzzy logic residual evaluation. *IEEE Trans. on Control Systems Technology*, 4(3):274–282, 1996.

[A57] H. Hammouri, M. Kinnaert, and E. H. El Yaagoubi. Observer-based approach to fault detection and isolation for nonlinear systems. *IEEE Transactions on Automatic Control*, 44(10):1879–1884, 1999.

[A58] S. Paoletti, A. Garulli, J. Roll, and A. Vicino. A necessary and sufficient condition for input-output realization of switched affine state space models. In *IEEE Conference on Decision and Control*, pages 935–940, 2008.

[A59] A. Abdo, S. X. Ding, J. Saijai, and W. Damlakhi. Fault detection for switched systems based on a deterministic method. In *IEEE Conference on Decision and Control* (CDC), pages 568–573, 2012.

[A60] W. Pan, Y. Yuan, H. Sandberg, J. Goncalves, and G. Stan. Online fault diagnosis for nonlinear power systems. *Automatica*, 55:27–36, 2015.

[A61] S. Campbell and R. Nikoukhah. *Auxiliary signal design for failure detection*. Princeton University Press, 2004.

[A62] J. K. Scott, R. Findeisen, R. D Braatz, and D. M. Raimondo. Input design for guaranteed fault diagnosis using zonotopes. *Automatica*, 50(6):1580–1589, 2014.

[A63] P. Rosa, C. Silvestre, J. S. Shamma, and M. Athans. Fault detection and isolation of ltv systems using set-valued observers. *In IEEE Conference on Decision and Control (CDC)*, pages 768–773, 2010.

[A64] F. Harirchi and N. Ozay. Model invalidation for switched affine systems with applications to fault and anomaly detection. *IFAC ADHS Conference*, 48(27):260–266, 2015.

[A65] F. Harirchi, Z. Luo, and N. Ozay. Model (in)validation and fault detection for systems with polynomial state-space models. *In American Control Conference (ACC)*, pages 1017–1023, July 2016.

[A66] J. Anderson and A. Papachristodoulou. On validation and invalidation of biological models. *BMC Bioinformatics*, 10(1):1, 2009.

[A67] N. Ozay, M. Sznaier, and C. Lagoa. Convex certificates for model (in)validation of switched affine systems with unknown switches. *IEEE Transactions on Automatic Control*, 59(11):2921–2932, 2014.

[A68] H. Lou and P. Si. The distinguishability of linear control systems. *Nonlinear Analysis: Hybrid Systems*, 3(1):21–38, 2009.

[A69] P. Rosa and C. Silvestre. On the distinguishability of discrete linear time-invariant dynamic systems. In *IEEE CDC-ECC*, pages 3356– 3361, 2011.

[A70] M. Babaali and M. Egerstedt. Observability of switched linear systems. In *International Workshop on Hybrid Systems: Computation and Control*, pages 48–63. Springer, 2004.

[A71] Harirchi, Farshad and Ozay, Necmiye, “Model Invalidation for Switched Affine Systems with Applications to Fault and Anomaly Detection”, *International Federation of Automatic Control (IFAC) 48-27* (2015) p260-266.

[A72] Harirchi, Farshad and Ozay, Necmiye, “Guaranteed Model-Based Fault Detection in Cyber-Physical Systems: A Model Invalidation Approach”, Cornell University Library archieve: 1609.05921v1, 19 Sep 2016, pp 1-34.

[A73] Farshad Harirchi and Sze Zheng Yong, Guaranteed Fault Detection and Isolation for Switched Affine Models, Cornell University Library archive: 1704.05947v1, 19 April 2017, pp 1-8.

[A74] Oudghiri, M.; Chadli, M. & El Hajjaji, A. (2008). Sensors Active Fault Tolerant Control For Vehicle Via Bank of Robust H∞ Observers. *17th International Federation of Automatic Control (IFAC) World Congress*, July 2008, IFAC, Seoul, Korea.

[A75] F-J. Hung, M-T. Tsai, "Fault-tolerant control for six-phase PMSM drive system via intelligent complementary sliding-mode control using TSKFNN-AMF", *IEEE Trans. on Industrial Electronics*, vol. 60, no. 12, pp. 5747-5762, 2013.

[A76] B. Genge, I. Kiss, P. Haller, "A system dynamics approach for assessing the impact of cyber attacks on critical infrastructures", *To appear on International Journal of Critical Infrastructure Protection*, 2015.

[A77] E. K. Wang, Y. Ye, X. Xu, S. Yiu, L. Hui, K. Chow, "Security issues and challenges for cyber physical system", *Proceedings of the 2010 IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber Physical and Social Computing*, pp. 733-738, 2010.

[A78] F. Pasqualetti, F. Dorfler, F. Bullo, "Attack detection and identification in cyber-physical systems", *IEEE Transactions on Automatic Control,,*, vol. 58, no. 11, pp. 2715-2729, 2013.

[A79] R. Mitchell, I.-R. Chen, "A survey of intrusion detection techniques for cyber-physical systems", *ACM Computing Surveys*, vol. 46, no. 4, pp. 55:1-55:29, Mar. 2014.

[A80] A. A. Cárdenas, S. Amin, Z.-S. Lin, Y.-L. Huang, C.-Y. Huang, S. Sastry, "Attacks against process control systems: Risk assessment detection and response", *Proceedings of the 6th ACM Symposium on Information Computer and Communications Security ser. ASIACCS '11*, pp. 355-366, 2011.

[A81] C. Barreto, A. A. Cárdenas, N. Quijano, "Controllability of dynamical systems: Threat models and reactive security" in *Decision and Game Theory for Security*, Springer, pp. 45-64, 2013.

[A82] P. Hu, H. Li, H. Fu, D. Cansever, P. Mohapatra, Dynamic defense strategy against advanced persistent threat with insiders. IEEE Xplore Digital Library. 747-755. 10.1109/INFOCOM.2015.7218444.

Data Driven Control

The work in [A88] is concerned with the problem of data-driven fault-tolerant control for multiple simultaneous sensor drift faults in variable-gain digital PID systems

with very large time constants and long dead time, which are common characteristics of process control systems. No existing data-driven residual generation method can allow building (with low computational costs) residual variables independent of the state of these systems, and meanwhile guarantee that each of the sensor faults is mapped uniquely and entirely onto the associated residual variable. To solve the aforementioned technical difficulty, a novel residual generation technique is devised via the dead time as well as the coefficients and state of the variable gain PID controller. On this basis, a methodology is developed for the purpose of the full-decoupling estimation of several sensor malfunctions from the residual variables. Finally, a resulting data-driven approach to compensate for the aforesaid faults is applied to a dual-chamber electric heating furnace (which is a typical process plant), so that the effectiveness and advantages of the proposed methods are verified by experiment.

In [A89] the same authors present a data-driven output-feedback fault-tolerant tracking control design for unknown discrete-time linear systems with stochastic measurement and process noise. With the aid of a proposed input-output data-based online iterative identification algorithm, this study presents a fault detection strategy, an approximate dynamic programming approach to performance optimization, and a method for the achievement of tracking control via a pre-filter. The resulting fault-tolerant control (FTC) scheme is applied to a practical DC servo motor system with load fluctuations and nonlinearity. Finally, the experimental test demonstrates that the proposed FTC strategy (including a methodology for compensating load variations and nonlinearity) has better applicability and practicability than the one given by the previous work, where a load needs to be held constant and it is required that the motor run in a nearly linear small working region.

In [A86] the authors focus on the data-driven model-free adaptive fault detection and estimation (FDE) and fault-tolerant control (FTC) problems for multi-input multi-output (MIMO) discrete-time systems with unknown sensor faults. First, the initial systems are transformed into a novel data-based model with only one unknown parameter. Second, a fault estimator is established to detect the sensor faults. Noting that a time-varying residual threshold is developed to determine whether the sensor faults occur or not. Then, the unknown faults are approximated based on the approximation capability of a generalized fuzzy hyperbolic model and the FTC approaches are reconstructed by applying the optimality criterion. Given the use of a data-based model and the fuzzy hyperbolic model, it questionable if this is really a data-driven method. This method is restricted to single sensor faults.

The same authors presented similar work in [A87] using but used artificial neural networks to apply the method to a Multiple-Input Single-Output (MISO) electromechanical system.

The authors of [A85] propose a data driven fault tolerant strategy for nonlinear systems. The core of the proposed strategy is a just-in-time learning (JITL) based soft sensor. When a sensor fault is detected, the value of soft sensor is adopted as the redundancy instead of the real faulty sensor value. Due to the complexity of mechanism model for nonlinear system, a kind of just-in-time learning method is employed for the soft sensor. The indexes are predicted by the just-in-time learning method using only historical sample data. Instead of considering the system global model as normal the just-in-time

learning methods only consider the approximate system at the current time. Two nonlinear systems, a numerical one and a benchmark of a wastewater treatment system, are employed for experiments. Only single sensor faults are considered in this work. There was discussion of tackling multiple faults.

[A84] H. Y. Yang, S. H. Lee, and M. G. Na, "Monitoring and uncertainty analysis of feedwater flow rate using data-based modeling methods," *IEEE Trans. Nucl. Sci.*, vol. 56, no. 4, pp. 2426–2433, Aug. 2009.

[A85] H. Yu, Y. Jiang and S. Yin, "A Data Driven Sensor Fault Tolerant Scheme for Nonlinear Systems", *Proceedings of the 43rd Annual Conference of the Industrial Electronics Society (IECON)*, 2017, pp 7055-7060.

[A86] L. Liu, Z. Wang and H. Zhang, "Data-Based Adaptive Fault Estimation and Fault-Tolerant Control for MIMO Model-Free Systems Using Generalized Fuzzy Hyperbolic Model", *IEEE Transactions on Fuzzy Systems*, Vol. 26, No. 6, 2018, pp 3191-3205.

[A87] L. Liu, Z. Wang and H. Zhang, "Echo State Networks Based Data-Driven Adaptive Fault Tolerant Control With Its Application to Electromechanical System", *IEEE/ASME Transactions on Mechatronics*, Vol. 23, No. 3, 2018, pp 1372-1382.

[A88] J-S Wang and G-H Yang, "Data-Driven Approach to Accommodating Multiple Simultaneous Sensor Faults in Variable-Gain PID Systems", *IEEE Transactions on Industrial Electronics*, Vol. 66, No. 4, 2019, pp 3117-3126.

[A89] J-S Wang and G-H Yang, "Data-Driven Output-Feedback Fault-Tolerant Tracking Control Method and Its Application to a DC Servo System", *IEEE/ASME Transactions on Mechatronics*, Vol. 24, No. 3, 2019, pp 1186-1196

[A90] H. Luo, S. Yin, T. Liu and A.Q. Khan, "A Data-Driven Realization of the Control Performance-Oriented Process Monitoring System", *IEEE Transactions on Industrial Electronics*, Vol 67, No 1, Jan 2020. pp 521-530.

[A91] Luis E. Garza Castanon and Adriana Vargas Martinez (2009). *Artificial Intelligence Methods in Fault Tolerant Control, Automation Control - Theory and Practice*, A D Rodi (Ed.), ISBN: 978-953-307-039-1,

[A92] M. Chen, R. Mei, "Actuator fault tolerant control for a class of nonlinear systems using neural networks", Proc. of IEEE Int. *Conf. on Control & Automation*, pp. 101-106, 2014.

[A93] P. Baldi, P. Castaldi, N. Mimmo, S. Simani, "Satellite attitude active FTC based on Geometric Approach and RBF Neural Network", *Proc. of Conf. on Control and Fault-Tolerant Systems*, pp. 667-673, 2013.

[A94] J. Frolik, M. Abdelrahman, and P. Kandasamy, "A confidence-based approach to self-validation, fusion and reconstruction of quasi-redundant sensor data," *IEEE Trans. Instrum. Meas.*, vol. 50, no. 6, pp. 1761–1768, Dec. 2001.

[A95] S. Alag, A. M. Aggogino, and M. Morjaria, "A methodology for intelligent sensor measurement, validation, fusion, and fault detection for equipment monitoring and diagnostics," *AI EDAM*, vol. 15, no. 4, pp. 307–320, Sep. 2001.

Artificial Intelligence in Active FTC

The work in [A95] presents a comprehensive methodology based on a hybrid system of AI and statistical techniques. The methodology is designed for monitoring complex equipment systems, which validates the sensor data, associates a degree of validity with each measurement, isolates faulty sensors, estimates the actual values despite faulty measurements, and detects incipient sensor failures. The system consists of two Bayesian networks. The first is used to provide a list of potentially faulty sensors, while the second is used to isolate the fault.

The methodology consists of four steps: redundancy creation, state prediction, sensor measurement validation and fusion, and fault detection through residue change detection. These steps use the information obtained by looking at a sensor individually, information from the sensor as part of a group of sensors, and the immediate history of the process that is being monitored. Using a Kalman Filter, the methodology can detect multiple sensor failures, both abrupt as well as incipient. It can also detect subtle sensor failures such as drift in calibration and degradation of the sensor. The methodology is applied to data from a gas turbine power plant.

The authors of [A97] use adaptive fuzzy models to detect and identify a fault, thus providing fault tolerant control for a nonlinear system. The nonlinear fuzzy inference system is based on the Takagi-Sugeno fuzzy models. This approach is limited to detecting and identifying a single fault at a time. They later apply this method to a rail vehicle motor controller in [A98].

An unspecified nonlinear system is controlled by a neural network in [*90]. Using the residual signal generated by the fault detection module another neural network-based control loop is introduced to compensate for the fault(s) detected.

Likewise, in [A96] genetic programming is used to increase the convergence rate of the fault detection and identification routine applied to an evaporation station at a sugar factory. This approach is specifically geared toward nonlinear systems.

In [A94-A96] the authors propose using an artificial neural network to compensate for the unknown dynamics of the fault(s). The controller uses a sliding mode control signal to keep the system performance in an acceptable range until the ANN can determine the best response to the detected fault. A multiple-fault case is also addressed.

A Fault Detection and Identification (FDI) routine based on a wavelet neural net is propose in [A102]. The network is trained using data from simulated fault scenarios. The approach is designed to detect and identify only a single fault at a time.

The authors of [A103] and [A104] use component level models to generate system data. Like [A97] a Takagi-Sugeno fuzzy model is used to capture the structure of the system under control. Finally fuzzy/neural control is used in conjunction with a learning algorithm to capture the dynamics caused by faults. This approach is applied to jet engine turbine control.

In [A105] the authors use the flexibility of fuzzy logic to select from among several models for the best fit to the conditions of the system when one or more faults occur.

In [A106-A108] the authors propose a neural network-based learning approach. The Controller references a Dynamic Model bank when a fault is detected. If the bank contains the appropriate model for the fault in question, that model is used. Otherwise the supervisor module changes the initial conditions of the algorithm to keep system performance within acceptable bounds. The controller is demonstrated on a nonlinear system.

In [A109] the controller architecture uses a neural controller aiding an existing conventional controller. The neural controller uses a feedback error learning mechanism and employs a dynamic Radial Basis Function neural network called Extended Minimal Resource Allocating Network (EMRAN), which uses only on-line learning and does not need a priori training. The conventional controller is designed using a classical design approach to achieve the desired autonomous landing profile with tight touchdown dispersions. The system is demonstrated in cases with single and double faults.

The authors of [A110] demonstrate an application of neural estimators for detection and identification of faults in sensors and actuators in flight controls. The actuator faults are addressed using nonlinear dynamic inversion. The sensor faults are addressed by replacing the sensor outputs with neural estimates computed during fault detection and identification.

In [A111] a robust fault tolerant fuzzy control problem for continuous-time nonlinear systems with time delay and sensor faults is addressed. The Takagi-Sugeno fuzzy model is employed to represent a nonlinear system. Using a sensor fault model and fuzzy state observer, the authors develop a fuzzy output feedback controller. The fuzzy control system is reliable in the sense of that asymptotic stability is achieved not only when all sensor components are operating properly, but also in the presence of some component failures.

In [A112] a neural network state observer is trained by the actual nonlinear control system. From the residual difference between outputs of actual system and neural network observer, the fault of control system is detected and determined. Fault tolerant control is realized by using a compensation controller to maintain the stability and performance of the system. As an example of the application, a tracking control problem for the speed and azimuth of a mobile robot driven by two independent wheels is addressed by using the controller.

The authors of [A113] use an RBF neural network to compensate for the system faults and disturbances using model reference adaptive control technology. The Lypanov stability theory is used to tune the weights of the neural network.

In [A114] the authors use an iterative learning algorithm combined with an unknown input observer. The controller is applied to a satellite attitude control system.

The work in [A115] uses a RBF neural net along with reference models for adaptive control applied to wind turbines.

The work in [A116] introduces a fault tolerant controller design for nonlinear unknown systems with multiple actuators and bounded disturbance. The controller consists of an adaptive learning-based control law and a switching function mechanism. The adaptive control law is implemented by a two-layer neural network and the switching function is designed to automatically search for the correct switching vector to turn off the unknown faulty actuator if there is one. The stability of the system output under the occurrence of actuator failure is proved through standard Lyapunov approach, while the other signals are guaranteed to be bounded. The proposed controller design is implemented on a simulation example using a continuous stirred tank reactor.

Finally, the authors of [A117] propose an adaptive, active fault tolerant control for a class of nonlinear systems with unknown actuator faults. The actuator fault is assumed to have no traditional affine appearance of the system state variables and control input. A radial basis function neural network (NN) is be used in the design of the fault tolerant controller. The authors claim their fault-tolerant control scheme can minimize the time delay between fault occurrence and accommodation and reduce the adverse effect on system performance.

[A96] M. Witczak, A. Obuchowicz, J. Korbicz, "Genetic programming based approaches to identification and fault diagnosis of nonlinear dynamic systems", *Int. Journal of Control,* vol. 75, no. 13, pp. 1012-1031, 2002.

[A97] Lopez-Toribio, C. J., Patton, R. J., and Uppal, F. J. "Artificial Intelligence Approaches to Fault Diagnosis", *International Journal of Applied mathematics and Computer Science*, Vol. 9, No. 3, pp. 471-518, 1999.

[A98] Lopez-Toribio, C. J., Patton, R. J., and Daley, S. "Takagi-Sugeno fuzzy fault tolerant control of an induction motor", *Neural Computing and Applications,* Vol. 19, No. I1, pp. 19–28, 2000.

[A99] G.G. Yen & L. W. Ho, "Intelligent Fault Tolerant Control Using Artificial Neural Networks", *Proceedings of the IEEE-INNS-ENNS International Conference on Neural Networks (IJCNN)* Vol 1, 2000, pp 266-271.

[A100] G.G. Yen & L. W. Ho, "Fault Tolerant Control: An Intelligent Sliding Mode Control Strategy", *Proceedings of the 2000 American Control Conference*, 2000 ,pp 4204-4208.

[A101] Ho, L.-W., and Yen, G. G., "Reconfigurable control system design for fault diagnosis and accommodation", *International Journal of Neural Systems,* Vol. 12, No. 6, pp. 497–520, 2002.

[A102] N.S. Clements, B. S. Heck and G. Vachtsevanos, "Component Based Modeling and Fault Tolerant Control of Complex Systems", Proceedings of the 19th Digital Avionics Systems Conference (DASC), 2000 Vol 2, pp 6.F.4-1 – 6.F.4-4

[A103] Diao, Y. & Passino, K., "Stable fault-tolerant adaptive fuzzy/neural control for turbine engine", *IEEE Transactions on Control Systems Technology*, Vol. 9, No. 3, (May 2001) 494-509, ISSN: 1063-6536.

[A104] Diao,Y. & Passino, K., "Intelligent fault-tolerant control using adaptive and learning methods", *Control Engineering Practice*, Vol. 10, N. 8, (August 2002) 801-817, ISSN: 0967-0661.

[A105] Ichtev, A., "Multiple fuzzy models for fault tolerant control", *International Journal of Fuzzy Systems*, Vol. 5, No. 1, pp. 31–40, 2003.

[A106] P. G. DeLima & G.G. Yen, "Addressing to Online Adaptive Controller Malfunction in Fault Tolerant Control", *Proceedings of the 2004 IEEE International Symposium on Intelligent Control*, 2004, pp 1279-1284.

[A107] Yen, G. & DeLima, P., "An Integrated Fault Tolerant Control Framework Using Adaptive Critic Design", *International Joint Conference on Neural Networks*, Vol. 5, pp. 2983-2988, ISBN: 0-7803-9048-2.

[A108] G.G. Yen, P.G. DeLima, "Improving the performance of globalized dual heuristic programming for fault-tolerant control through an online learning Supervisor", *IEEE Trans. on Automation Science and Engineering*, vol. 2, no. 2, pp. 121-131, 2005.

[A109] Pashilkar,A.; Sundararajan, N.; Saratchandran, P., "A Fault-tolerant Neural Aided Controller for Aircraft Auto-landing", *Aerospace Science and Technology*, Vol. 10, pp. 49-61.

[A110] Perhinschi, M.; Napolitano, M.; Campa, G., Fravolini, M.; & Seanor, B., "Integration of Sensor and Actuator Failure Detection, Identification, and Accommodation Schemes within Fault Tolerant Control Laws", *Control and Intelligent Systems*, Vol. 35, No. 4, 309-318, ISSN: 1480-1752.

[A111] S-C. Tong, "Robust Fault Tolerant Fuzzy Control for Nonlinear Systems with Sensor Failures", *Proceedings of the Sixth International Conference on Machine Learning and Cybernetics*, 2007, pp 604-609.

[A112] Z. Li, "Fault Diagnosis and Fault Tolerant Control of Mobile Robot Based on Neural Networks", *Proceedings of the Eighth International Conference on Machine Learning and Cybernetics*, 2009, pp 1077-1081.

[A113] Z. Chuan, W. Yifei, C. Qingwei, "Adaptive fault-tolerant control for a class of networked control system with random time delay based on neural network", *Proc. of Chinese Cont. Conf*., pp. 4217-4222, 2011.

[A114] Q. Jia, Y. Guan, Y. Zhang, Y. Jiang and Y. Shen, "Active fault-tolerant control for satellite system via learning unknown input observer", *Proceedings of the 10th World Congress on Intelligent Control and Automation*, Beijing, 2012, pp. 2965-2967. doi: 10.1109/WCICA.2012.6358378

[A115] L-L. Fan, Y-D. Song, "Neuro-adaptive model-reference fault-tolerant control with application to wind turbines", *IET Control Theory & Applications*, vol. 6, no. 4, pp. 475-486, 2012.

[A116] Q. Yang, B. Liu & Z. Yu, "Adaptive Learning based Fault Tolerant Control for Uncertain Nonlinear Systems", *Proceedings of the 2012 International Conference on Machine Learning and Cybernetics*, 2012, pp 1418-1423.

[A117] Shen, B. Jiang, P. Shi & C-C Lim, "Novel Neural Networks-Based Fault Tolerant Control Scheme With Fault Alarms", *IEEE Transactions on Cybernetics*, Vol 44, No 11, 2014, pp 2190-2201.

Detecting Multiple Faults and Attacks

[A118] Krishnamoorthy, Ganesh, Ashok, Pradeepkumar and Tesar, Delbert, "Simultaneous Sensor and Process Fault Detection and Isolation in Multiple-Input-Multiple-Output Systems", *IEEE Systems Journal*, Vol. 9, No. 2, June 2015.

[A119] Li, Yumei, Voos, Holger, Darouach, Mohmed and Hua, Changchun, "An Algebraic Detection Approach for Control Systems under Multiple Stochastic Cyber attacks", *IEEE/CAA Journal of Automatica Sinica*, Vol 2, No 3, July 2015, pages 258-266.

[A120] Zhu, Jun-Wei and Yang, Guang-Hong, "Fault-tolerant control for linear systems with multiple faults and disturbances based on augmented intermediate estimator", *IET Control Theory and Applications*, Vol. 11, Issue 2, 2017, pages 164-172.

[A121] Mukkamala, S., Yendrapalli, K., Basnet, R.B. and Sung A.H., "Detecting Coordinated Distributed Multiple Attacks", *21st International Conference on Advanced Information Networking and Applications Workshops (AINAW '07),* Vol 1, 2007, pages 557-562.

[A122] Jasnetski, Artjom, et al, "High-Level Modeling and Testing of Multiple Control Faults in Digital Systems", *IEEE 19th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*, 20-22 April 2016.

[A123] Ubar, Raimund, Oyeniran, and Stephen Adeboye, “Multiple Control Fault Testing in Digital Systems with High-Level Decision Diagrams”, *IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR),* 19-21 May 2016

[A124] Paliath, Vivin and Shakarian, Paulo, “Modeling cyber attacks on Industrial Control Systems” *IEEE Conference on Intelligence and Security Informatics (ISI),* 28-30 September 2016, pages 316-318.

[A125] M. Blanke, R. Izadi-Zamanabadi, S. A. Bogh, and Z. P. Lunau. Fault tolerant control systems-a holistic view. Control Engineering Practice, 5(5):693–702, 1997.

[A126] R. J. Patton. Fault-tolerant control systems: The 1997 situation. In *3rd IFAC symposium on fault detection supervision and safety for technical processes*, volume 3, pages 1033–1054, 1997.

Reconfigurable FTC

[A127] M. Blanke, C. Frei, F. Kraus, R. J. Patton, and M. Staroswiecki. What is fault tolerant control? In *4th IFAC symposium on fault detection, supervision and safety for technical processes*, pages 40–51, 2000.

[A128] Calise, A. J., Lee, S., & Sharma, M., “Development of a reconfigurable flight control law for tailless aircraft”, *Journal of Guidance, Control, and Dynamics*, 24(5), pp. 896–902, 2001.

[A129] M. Blanke, M. Staroswiecki, and N. E. Wu. Concepts and methods in fault-tolerant control. In *American Control Conference*, volume 4, pages 2606–2620, 2001.

[A130] J. Jiang. Fault-tolerant control systems-an introductory overview. *Acta Automatica Sinica*, 31(1):161–174, 2005.

[A131] Zhang, Y. and Jiang, J. “Bibliographical review on reconfigurable fault-tolerant control systems,” *Annual Reviews in Control*, Vol.32, pp: 229–252. 2008.

[A132] Yang, H., Cocquempot, V., Jiang, B. “Optimal Fault-Tolerant Path-Tracking Control for 4WS4WD Electric Vehicle,” *IEEE Transactions on Intelligent Transportation Systems*, Vol. 11, No. 1, pp, 237-243, 2010.

[A133] R. Isermann. *Fault-diagnosis systems*. Springer Verlag, 2006.

[A134] M. Blanke, M. Kinnaert, J. Lunze, and M. Staroswiecki. *Diagnosis and Fault-Tolerant Control*. Springer-Verlag, 2006.

B. Appendix: Steam Boiler Test Results Chart

Test Run No.	Sensor Effect	Effect Value	Pump 1 Effect	Effect Value	Pump 2 Effect	Effect Value	Components Affected	Catastrophic Failure	Mode 14 Triggered	Maintain Proper Control?	Iterations to Regain Proper Control	Iterations to Correct Decision	Iterations Using Correct Solution %
1							0	0	0	1	*	0	100.00%
2	1	206.17					1	0	0	1	*	0	100.00%
3	2	0.71					1	0	0	1	*	0	100.00%
4	3	0.24					1	0	0	1	*	0	100.00%
5	4	0.50					1	0	0	1	*	0	100.00%
6	5	340.08					1	0	0	1	*	0	100.00%
7	6	*					1	0	0	1	*	0	100.00%
8	7	0.28					1	0	0	1	*	0	100.00%
9	8	0.50					1	0	0	1	*	0	100.00%
10			1	2.68			1	0	0	1	*	0	100.00%
11			2	0.74			1	0	0	1	*	0	100.00%
12			3	0.45			1	0	0	1	*	0	100.00%
13			4	0.40			1	0	0	1	*	0	99.33%
14			5	1.88			1	0	0	1	*	105	77.00%
15			6	*			1	0	0	1	*	0	100.00%
16			7	0.74			1	0	0	1	*	0	100.00%
17			8	0.10			1	0	0	1	*	0	100.00%
18					1	2.81	1	0	0	1	*	9	99.00%
19					2	0.49	1	0	0	1	*	0	100.00%
20					3	0.09	1	0	0	1	*	9	98.33%
21					4	0.54	1	0	0	0	250	9	92.67%
22					5	3.36	1	0	0	1	*	9	49.00%
23					6	*	1	0	0	1	*	0	100.00%
24					7	0.02	1	0	0	1	*	10	99.33%
25					8	0.72	1	0	0	1	*	10	99.33%
26	1	447.99	1	1.49			2	0	0	1	*	0	100.00%
27	1	297.91	2	0.61			2	0	0	1	*	0	100.00%
28	1	344.57	3	0.50			2	0	0	1	*	0	100.00%
29	1	485.93	4	0.26			2	0	0	0	250	151	92.00%
30	1	718.16	5	3.66			2	0	0	0	4	99	52.33%
31	1	140.89	6	*			2	0	0	1	*	0	100.00%
32	1	617.00	7	0.97			2	0	0	1	*	0	100.00%
33	1	376.34	8	0.50			2	0	0	1	*	0	100.00%
34	1	667.00			1	1.59	2	0	0	1	*	0	100.00%
35	1	670.79			2	0.69	2	0	0	1	*	0	100.00%
36	1	429.47			3	0.14	2	0	0	1	*	9	98.00%
37	1	672.76			4	0.34	2	0	0	0	250	9	93.33%
38	1	166.15			5	1.63	2	0	0	1	*	9	75.00%

39	1	668.67			6	*	2	0	0	1	*	0	100.00%
40	1	758.26			7	1.00	2	0	0	1	*	153	95.67%
41	1	281.74			8	0.26	2	0	0	1	*	10	99.33%
42	2	0.34	1	2.35			2	0	0	1	*	0	100.00%
43	2	0.47	2	0.66			2	0	0	1	*	0	100.00%
44	2	0.21	3	0.26			2	0	0	1	*	0	100.00%
45	2	0.68	4	0.47			2	0	0	0	250	187	92.33%
46	2	0.48	5	3.70			2	0	0	0	2	103	51.67%
47	2	0.17	6	*			2	0	0	1	*	0	100.00%
48	2	0.24	7	0.12			2	0	0	1	*	0	100.00%
49	2	0.36	8	0.32			2	0	0	1	*	0	100.00%
50	2	0.76			1	1.86	2	0	0	1	*	9	99.00%
51	2	0.14			2	0.42	2	0	1	1	*	9	99.33%
52	2	0.38			3	0.45	2	0	0	1	*	10	96.67%
53	2	0.77			4	0.11	2	0	0	1	*	10	91.00%
54	2	0.14			5	1.63	2	0	1	1	*	9	74.67%
55	2	0.49			6	*	2	0	0	1	*	0	100.00%
56	2	0.59			7	0.65	2	0	0	1	*	10	97.33%
57	2	0.73			8	0.47	2	0	0	1	*	10	99.33%
58	3	0.55	1	1.47			2	0	0	1	*	0	100.00%
59	3	0.36	2	0.37			2	0	0	1	*	0	100.00%
60	3	0.25	3	0.50			2	0	0	1	*	0	100.00%
61	3	0.18	4	0.30			2	0	0	1	*	0	99.33%
62	3	0.28	5	1.24			2	0	0	1	*	140	84.33%
63	3	0.13	6	*			2	0	0	1	*	0	100.00%
64	3	0.10	7	0.40			2	0	0	1	*	0	100.00%
65	3	0.41	8	0.01			2	0	0	1	*	0	100.00%
66	3	0.10			1	3.04	2	0	0	1	*	9	99.00%
67	3	0.38			2	0.37	2	0	0	1	*	0	100.00%
68	3	0.20			3	0.19	2	0	0	1	*	9	97.67%
69	3	0.25			4	0.48	2	0	0	0	71	10	77.00%
70	3	0.21			5	1.71	2	0	0	1	*	9	74.33%
71	3	0.23			6	*	2	0	0	1	*	0	100.00%
72	3	0.15			7	0.38	2	0	0	1	*	10	98.33%
73	3	0.30			8	0.44	2	0	0	1	*	10	99.33%
74	4	0.38	1	2.08			2	0	0	1	*	0	100.00%
75	4	0.22	2	0.35			2	0	0	1	*	0	100.00%
76	4	0.06	3	0.11			2	0	0	1	*	0	100.00%
77	4	0.09	4	0.29			2	0	0	1	*	0	99.67%
78	4	0.26	5	3.26			2	0	0	0	2	93	54.33%
79	4	0.21	6	*			2	0	0	1	*	0	100.00%
80	4	0.31	7	0.30			2	0	0	1	*	0	100.00%
81	4	0.34	8	0.17			2	0	0	1	*	0	100.00%
82	4	0.26			1	2.85	2	0	0	1	*	9	99.00%
83	4	0.08			2	0.66	2	0	0	1	*	9	99.33%
84	4	0.34			3	0.37	2	0	0	1	*	9	96.33%
85	4	0.51			4	0.50	2	0	0	0	45	9	71.67%

86	4	0.14			5	1.26	2	0	0	1	*	9	80.33%
87	4	0.32			6	*	2	0	0	1	*	0	100.00%
88	4	0.21			7	0.50	2	0	0	1	*	10	98.33%
89	4	0.52			8	0.21	2	0	0	1	*	10	99.33%
90	5	617.82	1	2.83			2	0	0	1	*	0	100.00%
91	5	806.00	2	0.13			2	0	0	1	*	0	100.00%
92	5	843.89	3	0.36			2	0	0	1	*	0	100.00%
93	5	113.62	4	0.38			2	0	0	1	*	0	99.67%
94	5	528.18	5	1.54			2	0	0	1	*	119	83.33%
95	5	225.07	6	*			2	0	0	1	*	0	100.00%
96	5	722.37	7	0.71			2	0	0	1	*	0	100.00%
97	5	130.29	8	0.07			2	0	0	1	*	0	100.00%
98	5	570.14			1	2.73	2	0	0	1	*	9	99.00%
99	5	606.40			2	0.59	2	0	0	1	*	0	100.00%
100	5	852.30			3	0.09	2	0	0	1	*	245	99.33%
101	5	298.13			4	0.43	2	0	0	0	250	10	96.33%
102	5	433.60			5	2.35	2	0	0	1	*	9	64.00%
103	5	801.63			6	*	2	0	0	1	*	0	100.00%
104	5	563.82			7	0.95	2	0	0	1	*	146	96.00%
105	5	790.12			8	0.36	2	0	0	1	*	0	100.00%
106	6	*	1	2.16			2	0	0	1	*	0	100.00%
107	6	*	2	0.79			2	0	0	1	*	0	100.00%
108	6	*	3	0.12			2	0	0	1	*	0	100.00%
109	6	*	4	0.35			2	0	0	0	6	163	80.00%
110	6	*	5	3.24			2	0	0	1	*	95	55.00%
111	6	*	6	*			2	0	0	1	*	0	100.00%
112	6	*	7	0.22			2	0	0	1	*	0	100.00%
113	6	*	8	0.89			2	0	0	1	*	0	100.00%
114	6	*			1	1.37	2	0	0	1	*	9	99.00%
115	6	*			2	0.40	2	0	0	1	*	9	99.00%
116	6	*			3	0.08	2	0	0	1	*	10	98.33%
117	6	*			4	0.08	2	0	0	0	132	10	82.67%
118	6	*			5	3.18	2	0	0	1	*	9	51.00%
119	6	*			6	*	2	0	0	1	*	0	100.00%
120	6	*			7	0.11	2	0	0	1	*	10	99.00%
121	6	*			8	0.61	2	0	0	1	*	10	99.00%
122	7	0.80	1	2.15			2	0	0	1	*	0	100.00%
123	7	0.02	2	0.10			2	0	0	1	*	0	100.00%
124	7	0.88	3	0.22			2	0	0	1	*	0	100.00%
125	7	0.11	4	0.36			2	0	0	1	*	0	99.33%
126	7	0.08	5	3.64			2	0	0	0	1	99	52.00%
127	7	0.54	6	*			2	0	0	1	*	0	100.00%
128	7	0.04	7	0.29			2	0	0	1	*	0	100.00%
129	7	0.12	8	0.81			2	0	0	1	*	0	100.00%
130	7	0.82			1	1.41	2	0	0	1	*	9	99.33%
131	7	0.99			2	0.19	2	0	0	1	*	0	100.00%
132	7	0.29			3	0.11	2	0	0	1	*	9	98.67%

133	7	0.41			4	0.51	2	0	0	0	250	9	92.00%
134	7	0.54			5	1.25	2	0	0	1	*	9	80.33%
135	7	0.59			6	*	2	0	0	1	*	0	100.00%
136	7	0.77			7	0.35	2	0	0	1	*	10	98.67%
137	7	0.55			8	0.77	2	0	0	1	*	10	99.33%
138	8	0.72	1	1.31			2	0	0	1	*	0	100.00%
139	8	0.11	2	0.33			2	0	0	1	*	0	100.00%
140	8	0.14	3	0.14			2	0	0	1	*	0	100.00%
141	8	0.16	4	0.10			2	0	0	1	*	0	100.00%
142	8	0.18	5	1.65			2	0	0	1	*	115	82.67%
143	8	0.52	6	*			2	0	0	1	*	0	100.00%
144	8	0.41	7	0.30			2	0	0	1	*	0	100.00%
145	8	0.50	8	0.90			2	0	0	1	*	0	100.00%
146	8	0.80			1	2.93	2	0	0	1	*	9	99.00%
147	8	0.90			2	0.19	2	0	0	1	*	0	100.00%
148	8	0.31			3	0.28	2	0	0	1	*	13	96.67%
149	8	0.23			4	0.37	2	0	0	0	250	9	93.33%
150	8	0.09			5	3.14	2	0	0	1	*	9	52.33%
151	8	0.25			6	*	2	0	0	1	*	0	100.00%
152	8	0.23			7	0.30	2	0	0	1	*	10	99.33%
153	8	0.10			8	0.18	2	0	0	1	*	10	99.33%
154			1	2.91	1	2.50	2	0	0	1	*	0	100.00%
155			1	2.94	2	0.49	2	0	0	1	*	0	100.00%
156			1	1.54	3	0.40	2	0	0	1	*	0	100.00%
157			1	2.15	4	0.26	2	0	0	0	250	143	91.33%
158			1	2.05	5	3.09	2	0	0	1	*	103	57.33%
159			1	1.56	6	*	2	0	0	1	*	0	100.00%
160			1	2.59	7	0.57	2	0	0	1	*	0	100.00%
161			1	1.69	8	1.00	2	0	0	1	*	0	100.00%
162			2	0.14	1	1.38	2	0	0	1	*	0	100.00%
163			2	0.24	2	0.34	2	0	0	1	*	0	100.00%
164			2	0.28	3	0.18	2	0	0	1	*	0	100.00%
165			2	0.52	4	0.33	2	0	0	1	*	0	99.00%
166			2	0.57	5	1.17	2	0	0	1	*	170	91.67%
167			2	0.32	6	*	2	0	0	1	*	0	100.00%
168			2	0.37	7	0.83	2	0	0	1	*	0	100.00%
169			2	0.21	8	0.78	2	0	0	1	*	0	100.00%
170			3	0.16	1	1.35	2	0	0	1	*	9	99.33%
171			3	0.33	2	0.26	2	0	0	1	*	0	100.00%
172			3	0.14	3	0.13	2	0	0	1	*	0	100.00%
173			3	0.16	4	0.22	2	0	0	1	*	0	99.00%
174			3	0.46	5	1.60	2	0	0	1	*	122	80.33%
175			3	0.48	6	*	2	0	0	1	*	0	100.00%
176			3	0.32	7	0.31	2	0	0	1	*	0	100.00%
177			3	0.33	8	0.02	2	0	0	1	*	0	100.00%
178			4	0.41	1	2.04	2	0	0	0	86	135	88.00%
179			4	0.38	2	0.44	2	0	0	1	*	0	99.00%

180			4	0.08	3	0.14	2	0	0	1	*	176	92.33%
181			4	0.17	4	0.45	2	0	0	1	*	0	99.00%
182			4	0.10	5	2.43	2	0	0	1	*	105	66.67%
183			4	0.49	6	*	2	0	0	1	*	0	100.00%
184			4	0.10	7	0.32	2	0	0	1	*	0	100.00%
185			4	0.22	8	0.97	2	0	0	1	*	0	99.00%
186			5	3.40	1	1.57	2	0	0	0	1	105	55.33%
187			5	2.12	2	0.32	2	0	0	1	*	112	73.33%
188			5	2.00	3	0.20	2	0	0	1	*	109	70.67%
189			5	1.73	4	0.20	2	0	0	1	*	0	99.00%
190			5	2.39	5	2.12	2	0	0	0	7	215	48.33%
191			5	3.20	6	*	2	0	0	1	*	0	100.00%
192			5	2.52	7	0.36	2	0	0	1	*	105	64.00%
193			5	2.12	8	0.73	2	0	0	1	*	110	73.00%
194			6	*	1	2.82	2	0	0	1	*	0	100.00%
195			6	*	2	0.80	2	0	0	1	*	0	100.00%
196			6	*	3	0.41	2	0	0	1	*	0	100.00%
197			6	*	4	0.23	2	0	0	1	*	0	100.00%
198			6	*	5	1.25	2	0	0	1	*	0	100.00%
199			6	*	6	*	2	0	0	1	*	0	100.00%
200			6	*	7	0.21	2	0	0	1	*	0	100.00%
201			6	*	8	0.12	2	0	0	1	*	0	100.00%
202			7	0.46	1	1.93	2	0	0	1	*	0	100.00%
203			7	0.22	2	0.51	2	0	0	1	*	0	100.00%
204			7	0.36	3	0.35	2	0	0	1	*	0	100.00%
205			7	0.39	4	0.26	2	0	0	1	*	0	92.67%
206			7	0.88	5	1.03	2	0	0	1	*	168	85.67%
207			7	0.64	6	*	2	0	0	1	*	0	100.00%
208			7	0.08	7	0.30	2	0	0	1	*	0	100.00%
209			7	0.86	8	0.66	2	0	0	1	*	0	100.00%
210			8	0.78	1	2.95	2	0	0	1	*	0	100.00%
211			8	0.72	2	0.16	2	0	0	1	*	0	100.00%
212			8	0.74	3	0.27	2	0	0	1	*	0	100.00%
213			8	0.93	4	0.24	2	0	0	1	*	0	99.00%
214			8	0.73	5	1.33	2	0	0	1	*	153	83.67%
215			8	0.70	6	*	2	0	0	1	*	0	100.00%
216			8	0.67	7	0.37	2	0	0	1	*	0	100.00%
217			8	0.99	8	0.64	2	0	0	1	*	0	100.00%

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From – To)*
23-11-2020	Dissertation	June 2013 – Dec 2020

TITLE AND SUBTITLE
Deep Learning-Based, Passive Fault Tolerant Control Facilitated by a Taxonomy of Cyber-Attack Effects

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Wardell, Dean C., Lt Col, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way, Building 640
WPAFB OH 45433-8865

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT-ENG-DS-20-D-013

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Intentionally left blank

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
DISTRUBTION STATEMENT A. APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES
This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

There have been several cyber-attacks on the cyber-physical systems (CPS) that monitor and control critical infrastructure over the last few years. The need for increased cyberspace security for these industrial control systems (ICS) has been widely discussed extensively researched. This work presents a novel controller design that does not rely on fault or attack detection. It incorporates deep learning and ensemble learning techniques to holistically consider the state of the system under control and determine which model to use for further control signals. This work also presents a taxonomy of effects for use in designing training and testing FTC. The Taxonomy is foundational to the proposed controller. The approach shows improved results over an active FTC even when the system is subjected to Stuxnet-like attack conditions.

15. SUBJECT TERMS
Cyber-Physical Systems, Cyber-Security, Ensemble Learning, Deep Learning, Fault Tolerant Control

16. SECURITY CLASSIFICATION OF: UNCLASSIFIED			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Robert F. Mills, AFIT/ENG
U	U	U	UU	213	19b. TELEPHONE NUMBER *(Include area code)* (937) 255-6565, ext 4527 (robert.mills@afit.edu)

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18