Defending Pressurized Water Reactors Against Stealthy Cyber Attacks

by

Jacob Aaron Katz Farber

B.S. in Mechanical Engineering

University of Rochester, 2011

Submitted to the Graduate Faculty of

the Swanson School of Engineering in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2021

UNIVERSITY OF PITTSBURGH

SWANSON SCHOOL OF ENGINEERING

This dissertation was presented

by

Jacob Aaron Katz Farber

It was defended on

December 8, 2020

and approved by

Daniel G. Cole, Ph.D., Associate Professor

Department of Mechanical Engineering and Materials Science

William W. Clark, Ph.D., Professor

Department of Mechanical Engineering and Materials Science

Jeffrey S. Vipperman, Ph.D., Professor

Department of Mechanical Engineering and Materials Science

Department of Bioengineering

Daniel Mossé, Ph.D., Professor

Department of Computer Science

Shannon Eggers, Ph.D., Nuclear Cyber Security Specialist

Idaho National Laboratory

Dissertation Director: Daniel G. Cole, Ph.D., Associate Professor

Department of Mechanical Engineering and Materials Science

Defending Pressurized Water Reactors Against Stealthy Cyber Attacks

Jacob Aaron Katz Farber, PhD

University of Pittsburgh, 2021

The goal of this research is to improve cyber security for nuclear power plants (NPPs) by addressing the following questions: How might an attacker attack NPPs? And how can automated defenses defend against those attacks?

To answer these questions, this research takes a system theoretic approach. This means that we use dynamic system models to describe the physical aspects of both system behavior and any attacker influences on the system. The primary advantage to this approach is that it can make use of extensive system theory to understand the interactions between the system and potential attackers.

For the first question, this work focuses on extending zero-dynamics attacks to nonlinear systems. If a system has zero dynamics, it means there exists some non-zero control input that can cause a non-zero state, but results in zero measurable output by virtue of the system's characteristics. Zero-dynamics attacks take advantage of these zero dynamics. These attacks require an attacker to have knowledge of the system dynamics, but are consequently harder to detect than other strategies. As a result, it's important to know about system vulnerabilities that result from possible zero-dynamics attacks.

For the second question, this work focuses on using state-estimation techniques. But unlike previous works that assume the attack can be detected, this work cannot make that assumption because zero-dynamics attacks require that the system go through some type of transient. This work develops methods for calculating an optimal transient that balances safety with accuracy, and develops decision rules to detect attacks in the presence of noise.

Table of Contents

List of Tables

List of Figures

Preface

I would like to thank my wife, Courtney, for her constant support throughout this process. I would also like to acknowledge our son, Benjamin, who was an excellent distraction from the stress of finishing a dissertation.

Finally, I would like to thank the Department of Energy and Idaho National Laboratory for their financial support. This work is supported under (i) an Integrated University Program Graduate Fellowship and the U.S. Department of Energy Light Water Reactor Sustainability program, and (ii) the U.S. Department of Energy award number 226706, *Defending Against Stealthy Cyber Attacks*.

1.0 Introduction

The goal of this research is to improve cyber security for nuclear power plants (NPPs) by addressing the following questions: How might an attacker attack NPPs? And how can automated defenses defend against those attacks? Answering both of these questions can improve security and add to our understanding of how to safeguard these critical infrastructure.

Historically, cyber security has focused on information technology methods that act as a perimeter defense. These methods, including encryption, authentication, firewalls, and air gaps, aim to prevent unauthorized access and protect information. For applications that are less safety critical, these methods may suffice; however, these methods can be breached, suggesting additional defenses are required.

If the information technology methods are bypassed, attackers could target instrumentation and control (I&C) systems in nuclear power plants. These systems monitor process parameters and control components throughout the plant to ensure those parameters stay within safe limits. By targeting the I&C systems, an attacker could alter sensor measurements or activate components without the operator's knowledge. This could cause process parameters to exceed those safe limits, increasing the probability of reactor core damage.

In order to defend I&C systems from cyber attacks, we must move beyond information technology methods to consider the cyber-physical system as a whole. This is what our research addresses. In order to answer our primary research questions, we need to better understand how an attacker could physically damage critical systems, and how we can prevent that damage.

1.1 Research Objectives

The work carried out in this dissertation enables us to do the following:

1. characterize stealthy cyber vulnerabilities targeting nuclear power plants;
2. detect attacks by monitoring physical measurements; and
3. provide diagnostic information to enhance plant response.

Objective 1: Characterize stealthy cyber vulnerabilities targeting nuclear power plants. In a nuclear power plant, there are many possible cyber attack strategies. This work focuses on a particularly dangerous set of attacks called stealthy attacks, which aim to cause damage to critical systems with minimal observable changes. These attacks are dangerous because attackers could alter critical process parameters, and plant operators would be unable to distinguish the attack from nominal operating conditions. If the attack continues undetected, this could ultimately result in core damage. This first objective aims to understand what components an attacker could compromise that would result in a stealthy attack. By characterizing stealthy cyber vulnerabilities, we improve our knowledge of what theoretical attacks exist that target NPPs.

Objective 2: Detect attacks by monitoring physical measurements. As mentioned above, stealthy attacks are dangerous because if the defender does not know the attack is occurring, they cannot defend against it. Therefore, it is critical that we can detect attacks to prevent them from damaging the plant. This can be done because these attacks result in physical changes that either mask sensor measurements or inject dangerous control actions. This makes monitoring physical measurements a powerful secondary defense against cyber attacks. This research provides a defense-in-depth layer to detect cyber-physical attacks after they have been initiated by an attacker.

Objective 3: Provide diagnostic information to enhance plant response. After an attack has been detected, there needs to be a response to ensure plant safety. This could be a response from operators or automated decision algorithms. Responses can include driving critical process parameters to safe values or shutting down the plant. Regardless of who the decision maker is, they will require knowledge of the state of the plant, which could be unavailable during a successful stealthy attack. Without the correct knowledge, their actions could make the plant condition even worse. This research provides diagnostic estimates of the process parameters that can be used to safely respond to a cyber attack.

These objectives can be split into two primary tasks: (i) characterize stealthy attack vulnerabilities (Objective 1), (ii) and develop defenses against them (Objectives 2 and 3).

These tasks provide important answers to our main research questions of how attackers might attack NPPs and how automated defenses could defend against these attacks.

1.2 Research Approach

This research takes a system theoretic approach to the cyber-physical security problem. This means that we use dynamic system models to describe the physical aspects of both system behavior and any attacker influences on the system. The primary advantage to this approach is that it can make use of extensive system theory to understand the interactions between the system and potential attackers.

From a system theoretic viewpoint, we can expand on cyber-physical systems, which contain both cyber and physical components; see Figure 1 for a block diagram. The physical side contains: the plant, which is the dynamic system of interest; actuators, which influence the plant output; and sensors, which measure the plant output. The cyber side contains the controller, which uses the measured output to calculate a desired control signal, and the monitor, which includes any process monitoring that may exist.

During a cyber-physical attack, an attacker can inject signals in between the cyber and physical to cause damage. Similar to other works [1, 2, 3, 4, 5, 6], we assume the attacker can implement both input and output attacks. For input attacks, the attacker modifies the control signal so the physical input does not match the assumed value, and for output attacks, the attacker modifies the sensor output so the measured output does not accurately reflect the physical output. In practice, it may be unrealistic to assume that an attacker can gain that much access; however, this assumption represents a conservative viewpoint that can be used to find worst-case vulnerabilities.

It is important to introduce several additional concepts from system theory. A system can be defined by a mathematical model that includes states, inputs, and outputs: states represent the set of variables that completely describe the state of the system at a point in time; inputs are signals that enter the system and can modify it; and outputs are signals that leave the system and can either be measured or represent desired behaviors. The inputs

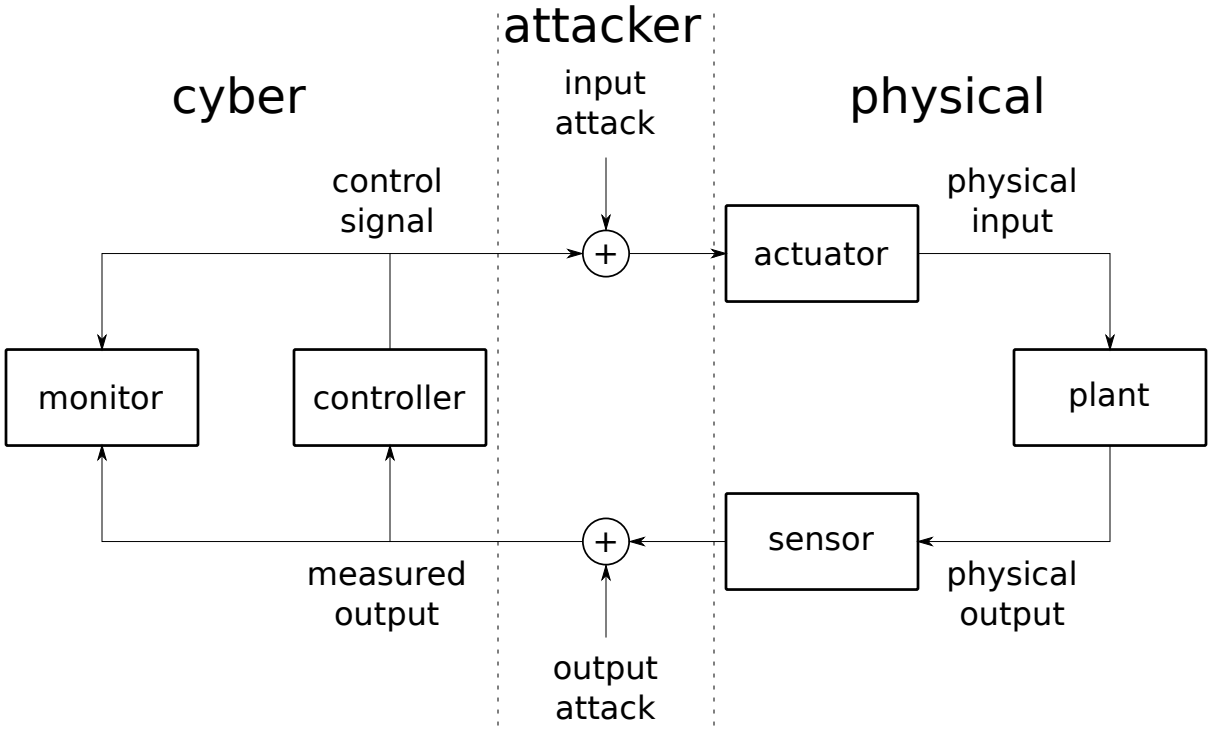


Figure 1: Block diagram of the cyber-physical system, including the attacker. This figure shows the attacker is able to inject input and output attacks between the cyber and physical components.

discussed in this work are the control signals and both attacker inputs, and the outputs discussed are the measured outputs, which include the attacker modifications in the signal. Finally, system theory includes state-estimation techniques, where the goal is to estimate the state using the available information.

For the first task of characterizing stealthy attack vulnerabilities, a system theoretic approach assumes attackers can modify control signals and sensor measurements. This means it can be used to analyze when combinations of attacked actuators and sensors could result in particularly dangerous conditions. With this approach, researchers have developed multiple strategies for how attackers might attack systems.

After studying these attack strategies, this work focuses on a specific strategy called the zero-dynamics attack. If a system has zero dynamics, it means there exists some non-zero control input that results in zero measurable output by virtue of the system's characteristics. Zero-dynamics attacks take advantage of these zero dynamics. These attacks require an attacker to have knowledge of the system dynamics, but are consequently harder to detect than other strategies. As a result, it's important to know about system vulnerabilities that result from possible zero-dynamics attacks.

Looking at previous works on zero-dynamics attacks, the primary limitation is that their approaches cannot be applied to nonlinear systems. This matters because most real systems, including within NPPs, are nonlinear and can only be approximated as linear within some small operating region. However, a successful attack will likely drive the system away from any fixed operating point, making linear methods potentially inaccurate. Therefore, our approach extends these attacks to nonlinear system dynamics using an iterative approach to solving for zero dynamics.

Using this iterative approach, we characterize vulnerabilities in the pressurizer system, which is a critical nuclear power plant subsystem. This is accomplished by analyzing all combinations of attacked actuators and sensors for unique zero-dynamics attacks that can result in system damage. These attacks are then compared using metrics that can aid designers in understanding which attacks are the most critical to defend.

For the second task of defending against attacks, a system theoretic approach uses the dynamic system models to detect attacks. The general idea is to use a model to estimate

sensor measurements and then compare the estimated values with the true values. This approach can be used to estimate the true state of the plant even if the measurements have been altered, which enables both detecting attacks and providing important diagnostic information.

Previous approaches to detecting attacks have assumed the attack can be detected based on the recorded measurement data; however, this is not the case for zero-dynamics attacks. Rather, to detect a zero-dynamics attack, the system must go through some type of transient. This work develops methods for calculating an optimal transient that balances safety with accuracy, and develops decision rules to detect attacks in the presence of noise.

These detection methods are then implemented on the pressurizer system to detect attacks and determine our estimation accuracy in the presence of noise. This is done using two different models: one physics-based and one data-driven. The physics-based model is used to both generate data and detect attacks, enabling us to demonstrate the theoretical applicability of the detection algorithm. The data-driven model is used to to detect attacks using data generated by a commercial simulator, enabling us to validate the techniques with a more accurate simulation.

1.3 Applications to the Nuclear Power Industry

This research has several important applications that help both the future fleet of advanced reactors and existing plants that require modernization:

- *Advanced Reactors and Small Modular Reactors:* The approaches developed for characterizing zero-dynamics attacks can be applied to new reactor concepts during the design phase to "design in" security. Using this work, its tools can be used to identify vulnerabilities that arise from design choices, so that plant designers can either remove them from the design or increase defenses. This means cyber security can be considered from the beginning, resulting in a more resilient and secure system.
- *Plant Modernization:* The detection methods are critical tools for the modernization of existing plants. Modernization can improve safety, increase availability, and reduce costs,

but it also makes plants more vulnerable to cyber attacks. And unlike new designs where security can be considered during the design phase, existing systems cannot be easily altered to remove vulnerabilities. To improve cyber resilience during modernization, automated detection methods should be employed.

1.4 Contributions

This research develops tools to improve cyber security for both nuclear power plants and other cyber-physical systems. These tools focus on two critical tasks: (i) characterizing stealthy cyber vulnerabilities targeting nuclear power plants, and (ii) developing defenses against these attacks.

Our main contributions to the fields of nonlinear controls and nuclear cyber security are that we:

1. extend zero-dynamics attacks to nonlinear systems through the development of an attacked system model and an algorithm;
2. develop metrics for zero-dynamics attacks to compare different attack strategies;
3. characterize zero-dynamics attacks targeting the pressurizer subsystem of pressurized water reactors;
4. develop tools to determine whether zero-dynamics attacks targeting a specific nonlinear system are detectable;
5. develop detection methods for detectable systems under zero-dynamics attacks;
6. quantify required model accuracy to be able to accurately detect attacks; and
7. implement the detection methods on the pressurizer subsystem using both a theoretical first-principles model and a data-driven model.

This work develops important analytical tools that can be implemented on a larger scale to other critical subsystems of nuclear power plants, as well as other cyber-physical systems.

1.5 Dissertation Overview

This dissertation is structured as follows. Chapter 2 describes the state of the art and limits of current practice. This sets the stage for the limitations that are extended in this work. Chapter 3 describes the system models used for the pressurizer subsystem, including both physics-based and data-driven models. The models developed in this chapter are used in both Chapters 4 and 5. Chapter 4 describes the extension of zero-dynamics attacks to nonlinear systems and the implementation on the pressurizer subsystem. Chapter 5 describes the detection methods and the implementation on the pressurizer subsystem. Finally, Chapter 6 concludes the dissertation with a recap of contributions and discussions of limitations and future work.

2.0 State of the Art and Limits of Current Practice

Having stated our research objectives, we can now discuss how other researchers have approached similar problems to the one we address. These are focused on stealthy attack strategies and attack detection methods. In addition, we address limitations of these works that must be overcome to solve our specific problem; these limitations are the foundations that justify this work.

To help illustrate the concepts in this chapter, we present some examples. Several of these examples use a simple linear spring-mass-damper (SMD) system; see Figure 2. In this system, the control input $u(t)$ is a force from an actuator, the sensor output $y(t)$ is the measured position of the spring, the attacker inputs $a_u(t)$ and $a_y(t)$ are the input and output attacks, respectively, and the states are the true position of the mass and its velocity. It is worth noting that this selection for the states is not unique, but this is a convenient choice because they represent physical quantities.

This chapter is broken into two major sections, one each on stealthy attacks and detecting attacks. For each section and each technique discussed, we present the relevant works, their limitations, and examples.

2.1 Stealthy Attacks

The first objective of this research is focused on identifying worst-case attack strategies. Based on this, limitations in existing stealthy attack strategies arise because they either cannot be applied to all systems or have known weaknesses that make them less damaging than intended. While these limitations are beneficial for defending against those strategies, more powerful strategies may exist that need to be considered.

Looking at previous works, there have been two main stealthy attack strategies that have been studied: replay attacks and zero-dynamics attacks. Replay attacks can be implemented with little knowledge of the system dynamics, making them both easier for an attacker to

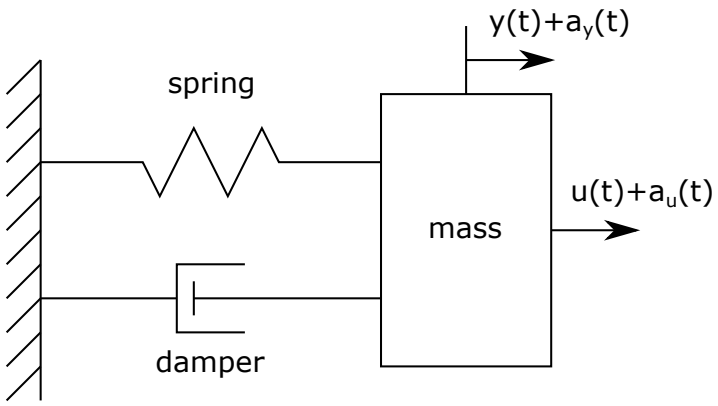


Figure 2: Image of a spring-mass-damper system. This example is used throughout this chapter.

implement and defender to detect. By contrast, zero-dynamics attacks require knowledge of the system dynamics, making them both harder for an attacker to implement and defender to detect.

2.1.1 Replay Attacks

One of the most well known stealthy attacks on a system was the Stuxnet cyber attack on the Iranian centrifuges [7, 8]. This attack was a computer worm that targeted specific types of Siemens industrial controllers. It collected real process data and then replayed that data to the controller while altering control signals to the centrifuge pumps. By implementing a stealthy attack strategy on the control systems, the worm damaged up to 1,000 centrifuges at the plant.

The Stuxnet attack strategy is called a replay attack strategy [1], which is a type of stealthy attack. For this strategy, an attacker replays recorded measurements and injects control inputs into the system. The replayed measurements mask the effects of the inputs, so the inputs can move the system into unsafe operating conditions unnoticed. These attacks can be particularly powerful against systems that run at or near steady-state operation,

including nuclear power plants, because operators do not expect to see any changes in the measurements.

The replay attack has been extended to identify the smallest set of sensors that still creates a stealthy attack using replayed measurements [2]. These attacks target system observability, which is a system property that determines whether the system state can be recreated using the available measurements. The analysis of these attacks looks at all combinations of sensors to determine when masking sensors impacts observability. This attack strategy has two advantages over generic replay attacks: first, it is simpler than a replay attack that targets all sensors because attackers may be limited to accessing a subset of sensors; and second, it ensures that an attacker masks enough to make the system unobservable.

Limitation From an attacker's perspective, the primary limitation to replay attacks is that they hide all effects, yet they do not account for the system's dynamics. Replay attacks can be detected by inserting an arbitrary signal, unknown to the attacker, into the control inputs in order to create expected transients [1, 5, 6]. Because the signal is arbitrary and unknown, the attacker cannot predict the response. As such, there will be a mismatch between the expected output and the replayed signals.

For this detection approach, the arbitrary signal has been proposed as a continuous zero mean random Gaussian control input [1, 5]. In general, the signal can be relatively small, such that an attack would be detected by statistical testing techniques. The advantage to this continuous approach is that it is easy to implement and should detect attacks fairly quickly; however, it means that the control input used to control the process is always sub-optimal.

An alternative candidate for the arbitrary signal is a periodic control input [6]. This signal has many of the same benefits of the original continuous approach, but only requires a sub-optimal control input during the periodic detection times. However, it may not detect an attack as quickly if the period does not align with then the attack starts.

Example The replay attack strategy and detection approach are demonstrated on the SMD system; see Figure 3 for the results. For simplicity, the simulation is at steady-state conditions with noise. The figure shows the state, the measured output, and the expected

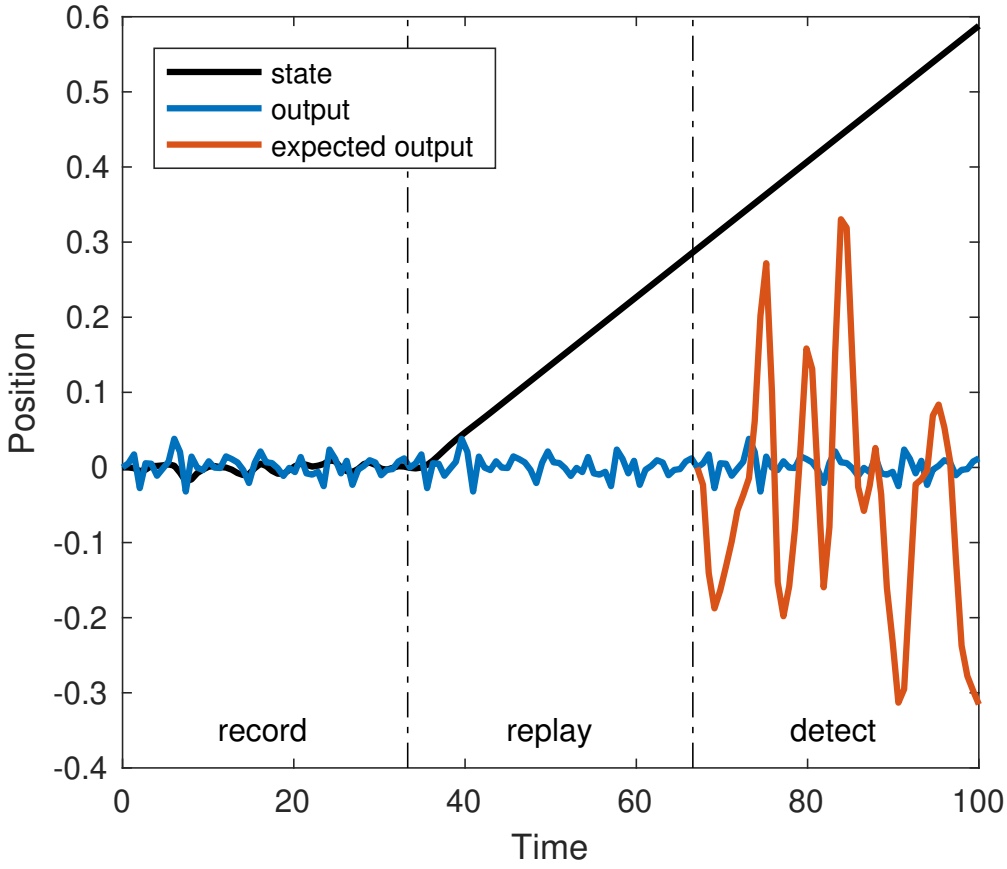


Figure 3: Results of the replay attack implemented on the SMD system. The figure shows the state, the measured output, and the expected output, and is segmented into thirds. In the first segment, the attacker only records measurement data. In the second segment, the attacker both replays the recorded measurements and adds a force onto the mass. In the third segment, the defender tries to detect the attack by adding a random force onto the spring, expecting it to respond accordingly if not under attack.

output, and is segmented into thirds. In the first segment, the attacker only records measurement data. As such, both the true position and measured position remain at the nominal condition, and the expected output is not shown because the defender is not trying to detect an attack. In the second segment, the attacker both replays the recorded measurements and adds a force onto the mass. The force causes the true position to increase even though the measured values remain nominal, resulting in a successful attack. In the third segment, the defender tries to detect the attack by adding a random force onto the spring, expecting it to respond accordingly. The true output continues to replay the recorded values, resulting in a mismatch between the output and expected output. In this way, the defender detects that the replay attack is occurring. Note that the random input shown here is comparatively large for visual purposes, but could be smaller when combined with statistical testing.

2.1.2 Linear Zero-Dynamics Attacks

The replay attack's limitation would suggest that attackers would look for other stealthy strategies to excite the system, but that result in minimal or no measured response. If such an attack is possible, the need for replay is not necessary, and the attack can remain hidden by virtue of the system's characteristics. Such attacks are called zero-dynamics attacks [3, 4].

If a system has zero dynamics, some input exists that results in zero measurable output but nonzero internal dynamics. Zero-dynamics attacks take advantage of these zero dynamics. By accounting for the system's dynamics, these attacks can still show expected transients because they add to existing signals rather than overwriting them. As such, they overcome the limitation of the replay attack, and are a pernicious attack that, if not anticipated, can be difficult to defend against.

Zero-dynamics attacks have been analyzed for linear systems represented as closed-loop transfer functions [4]. These systems can be written as

$$Y(s) = G(s)U(s) \tag{2.1}$$

where $G(s) = \frac{N(s)}{D(s)}$ is the transfer function, which is a linear mapping in the Laplace domain between the input variable $U(s)$ and output variable $Y(s)$, and $N(s)$ and $D(s)$ are matrices

of polynomials in the complex variable s. For scalar transfer functions, the roots of the polynomial $N(s)$ are called the zeros of the system, and the roots of the polynomial $D(s)$ are called the poles of the system; for matrix transfer functions, this is more complicated, and a detailed discussion can be found in [9]. Any zeros and poles that have strictly negative real parts are called minimum-phase zeros and stable poles, respectively.

In [4], transfer function matrices are analyzed for vulnerability to zero-dynamics attacks targeting (i) actuators and (ii) sensors. To accomplish this, the system is factored using coprime factorization, which rewrites the system as a ratio of stable transfer functions. Using this factorization and linear algebra, the vulnerabilities can be summarized as follows:

- an unbounded attack targeting the actuators is possible if and only if the system has non-minimum-phase zeros; and
- an unbounded attack targeting the sensors is possible if and only if the system has unstable poles.

In the above, an unbounded attack means that the attack variables can grow unbounded creating unbounded internal dynamics, but the measured variables remain zero (or below some noise threshold).

Zero-dynamics attacks have also been analyzed for linear systems described using state-space models [3]. These systems can be written as

$$\begin{aligned} \dot{x} &= Ax + B(u + a) \\ y &= Cx \end{aligned} \tag{2.2}$$

where u is the control input, y is the sensor output, x is the system state, a is the attacker input, and A, B, and C are constant system matrices.

Similar to [4], state-space systems are analyzed for vulnerability to zero-dynamics attacks [3], although this work only focuses on attacks targeting actuators. This is done using invariance principles, which are subspace techniques that look at the system geometrically. Using the linear matrices A, B, and C, the invariance techniques are used to find the subspace that the attacker can move the state in while remaining stealthy. It is important to note that despite using a different system representation, this work still requires non-minimum phase zeros to create an unbounded attack targeting the actuators.

Limitation The primary limitation to these works for our problem is that they focus on linear systems. Looking at the previous approaches, transfer functions [4] in general are limited to linear systems, and the state space methods in [3] rely on linear system theory to identify zero-dynamics attacks. This is challenging because nuclear plant systems contain nonlinear dynamics.

One way to overcome this limitation is to derive the nonlinear dynamics and then linearize them. This creates an approximate system that is valid within a small region of the state space near an equilibrium point. However, any successful attack will move the system away from an equilibrium point, making linearizations less valid; see Figure 4. As such, these linear approaches may prove less accurate and could result in incorrect conclusions about a system's zero dynamics.

Example The zero-dynamics attack strategy and detection approach for replay attacks are demonstrated on the SMD system; see Figure 5 for the results. The simulation is similar to the previous example, except is only segmented into halves. In the first segment, the attacker, assumed to have knowledge of the system dynamics, implements the zero-dynamics attack, which causes the state to increase and the measurement to remain nominal. This represents a successful attack. In the second segment, the defender again tries to detect the attack by adding a random force onto the spring. But unlike for the replay attack, here the output and expected output match, meaning the defender is unable to detect the attack using this straight-forward strategy. This attack is not detected because the attacker uses knowledge of the system dynamics to create a stealthier attack.

2.1.3 Conclusion

Of these two attack strategies, this work focuses on zero-dynamics attacks because they represent a more pernicious and dangerous class of attacks. This work extends previous works and explores the use of nonlinear generalizations of zero dynamics to the NPP problem.

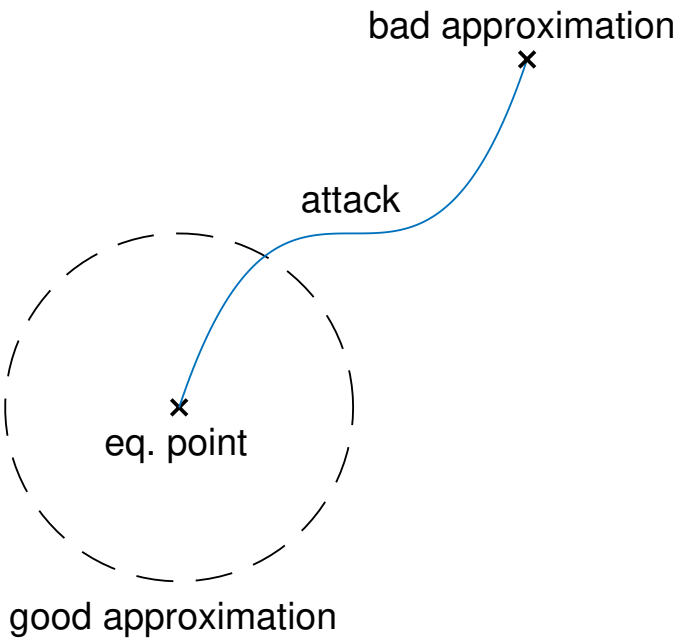


Figure 4: Sketch of a linearization approximation. Within the dashed circle, a linear system could give good results. However, outside of the region, the linear system could be highly inaccurate.

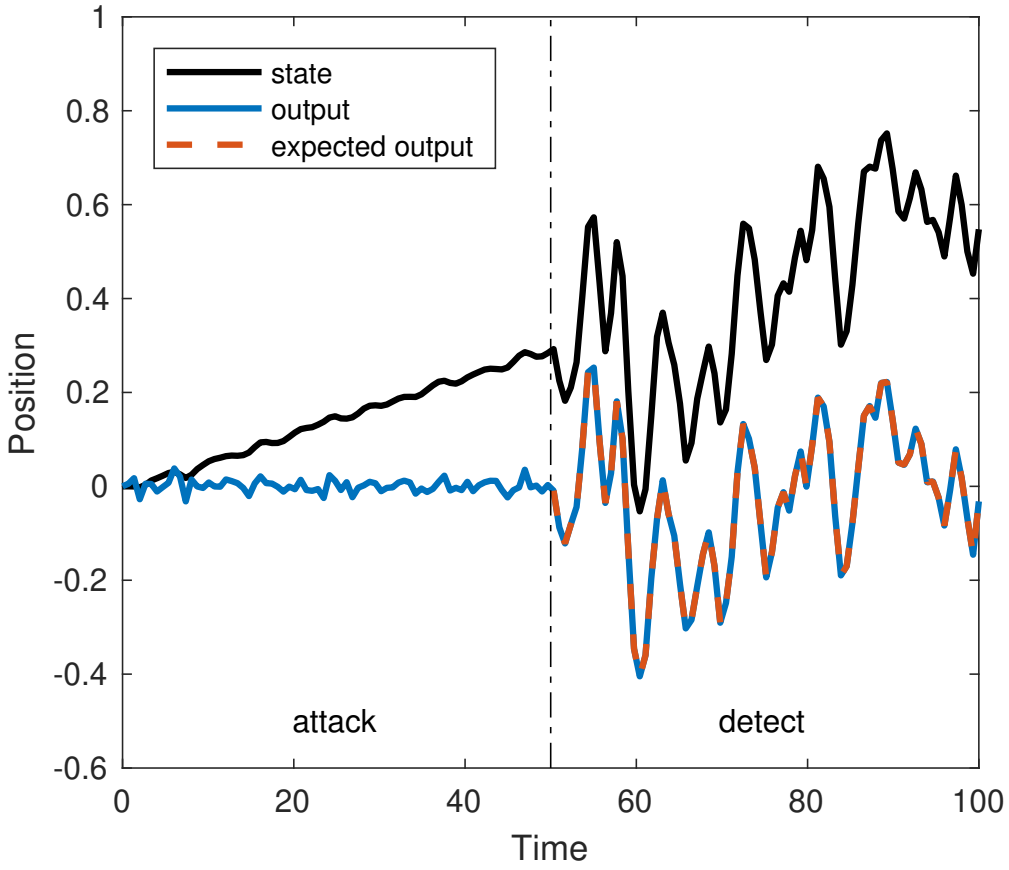


Figure 5: Results of the zero-dynamics attack implemented on the SMD system. The figure shows the state, the measured output, and the expected output, and is segmented into halves. In the first segment, the attacker, assumed to have knowledge of the system dynamics, implements the zero-dynamics attack, which causes the state to increase and the measurement to remain nominal. In the second segment, the defender again tries to detect the attack by adding a random force onto the spring, but cannot detect it because the output and expected output match.

2.2 Detection and Diagnostics

The second and third objectives of this research are focused on defending against the vulnerabilities identified during the first part of the research. While this work focuses on cyber attack detection, we also discuss some previous work on fault detection. Faults refer to abnormal operation of sensors, actuators, and devices. The goal of fault detection is to determine the current operating condition of the system. Fault detection and cyber attack detection methods share many similarities because both are focused on detecting anomalies based on monitoring time-series measurements.

We focus on two categories of methods for detection: classification methods and state-estimation methods. Classification methods often fall under the umbrella of machine learning techniques and tend to focus on how to use raw data for decision making. By contrast, state-estimation methods require system models to make decisions, although those system models could be data-driven. In addition, these methods could be combined to solve detection problems.

2.2.1 Classification Methods

Classes are labels that differentiate objects within them from other classes. Classification methods aim to accurately label unseen objects from a fixed set of classes. These methods are trained using historical data to learn the differences between the classes. Then, the classifier should identify the class of new objects by comparing them to data it has already seen. For example in fault detection, the classes might be normal or faulted operating conditions, and the methods would use measurement data to assess what the current operating conditions are. Here, we focus on a handful of works that implement classification methods on time-series data.

Within classification methods, two common algorithms are support vector machines (SVMs) and artificial neural networks (ANNs). As a very brief primer on these algorithms, both are implemented as described above, but they have differences in how they handle non-linear data. Broadly speaking, SVMs are well suited for linear or mildly nonlinear problems

and do not require extensive training sets to generalize well. By contrast, ANNs can handle extremely nonlinear problems, but require larger training sets to generalize well.

Both SVMs [10] and ANNs [11] have been implemented to detect faults in rotating bearings. These works look at vibration data from accelerometer sensors at normal and faulted conditions, where a fixed interval of data is collected periodically. For each interval, relevant statistics are extracted that summarize the data over that interval. These include root mean-squared (RMS) error, kurtosis, and maximum peak-to-peak amplitudes. Based on these values, the classifiers are trained on some fraction of the datasets and then tested using the remainder of the datasets. In these works, both methods were able to successfully diagnose the bearing conditions using vibration data.

SVMs have also been used to diagnose loss-of-coolant accidents (LOCAs) in nuclear power plants [12, 13]. These works collected simulation data from multiple sensors for LOCAs of various magnitudes and leak locations. For each simulation, the sensor values were integrated to summarize the data, and the resulting summary statistics were used to train and test the approach. Similar to the above approaches, these methods were able to successfully detect the faulted conditions.

In all of the above approaches, the authors use summary statistics of the intervals in order to handle the time-series data. But, we argue this is more appropriate for the works on rotating bearings than for the works on LOCAs. For the bearing applications, the data is highly periodic and can generally be summarized using the proposed statistics. By contrast, the LOCA data is non-periodic and depends on the initial conditions, disturbances to the system, and many other factors that make it harder to summarize succinctly for use directly with classification methods. While they were successful, their methods may have a harder time when accounting for the all the unknowns of a real plant.

Limitation The primary limitation of classification methods for some types of time-series data is that they require summary statistics to handle the large quantity of data. This approach may work well for periodic data or other data that is well suited to being summarized; however as mentioned above, methods such as integration, expected value, and RMS error may not capture differences in initial conditions, disturbances, or causality, all of which are

necessary to describe dynamic system responses. These statistics may ignore critical data or relations between the data that make them less effective or less robust.

It is important to note that any fault or cyber detection scheme should eventually have a decision rule, which has many similarities to and could be implemented with classification methods. The important distinction is how the summary statistics are created. For complicated dynamic systems, it may be important to account for the causality of inputs and outputs, rather than more simple methods. In other words, the limitation of classification methods for some time-series problems is that they may not be solely capable of solving the problems, but could be combined with other methods that account for the system dynamics. This is often accomplished using state-estimation methods, which are discussed in the next section.

One additional limitation to purely using classification methods for our problem is that the problem may not be solvable for all systems. Using only classification methods, it may be difficult or even impossible to determine whether the problem can be solved without testing classification methods on examples and hoping there are no unknown cases where it fails to work. By contrast, our more theoretical approach enables us to test a system directly to see whether it is solvable, adding insight to the problem.

Example A simple classification example is presented that differentiates between normal and faulted bearing performance. This example uses randomly generated data and is implemented using SVM classification; see Figure 6 for the results. The two variables are summary statistics of vibration data in two axes, and the blue and red data points are normal and faulted data, respectively. The line is the decision boundary, where data on a given side of the line is classified accordingly. Note that some of the training data falls on the wrong side of the boundary, and these points would be false alarms. This is typical in many real decision problems, where data is not perfectly separated.

2.2.2 State-Estimation Methods

In order to account for the system dynamics, many researchers have implemented state-estimation methods for fault and cyber detection. The system state describes its true status

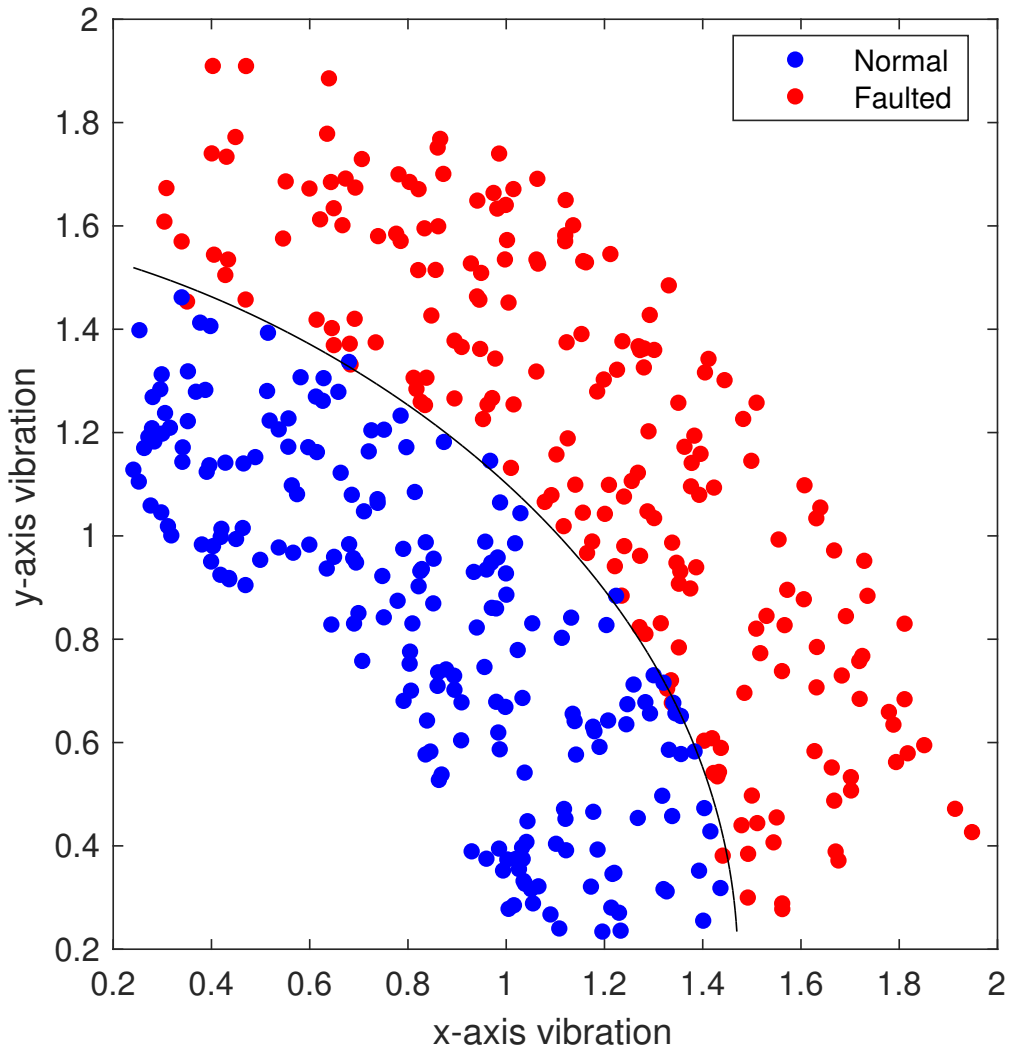


Figure 6: Simple example of an SVM classifier to detect rotating machinery faults. The two variables are summary statistics of vibration data in two dimensions, and the blue and red data points are normal and faulted data, respectively. The line is the decision boundary, where data on a given side of the line is classified accordingly.

regardless of the attack status, so an accurate state estimate can both detect the attack and provide diagnostic information. State-estimation techniques estimate the state in real-time using a system model, control inputs, and sensor outputs. The system model can be derived from first-principles or using a data-driven approach, but needs to capture the causal relationship of the system. In order to ensure accuracy, state estimators require systems be observable, which is why the replay attacks described above targeted observability. Each of the works discussed below require the system to be observable.

Works that use state-estimation techniques generally focus on either linear or nonlinear systems. Starting with linear systems, one approach uses linear observer methods [14], where an observer is a real-time state estimator that does not explicitly account for system noise. Observers can be written as linear systems using state-space representation

$$\begin{aligned} \dot{\hat{x}} &= A\hat{x} + Bu + L(\hat{y} - y) \\ \hat{y} &= C\hat{x} \end{aligned} \tag{2.3}$$

where $\hat{}$ signifies an estimate, and L is the observer gain and is a user-defined parameter. The gain compensates for the error between the true and estimated measurement, and this compensation process is called measurement feedback. In [14], a first observer is used to detect the attack. The gain is selected to drive the estimation error $\hat{x} - x$ to zero, which is typical for observer designs. From this observer, an attack is detected when the error crosses some threshold. In addition, a set of observers is used to determine the attack strategy. Through careful selection of the gains, each observer is designed so that its error will only go to zero if the attacker is attacking using a particular strategy. Through this multiple-observer approach, the observers can both detect attacks and identify the attacker strategy.

The above work [14] also makes an important assertion that is directly relevant to zero-dynamics attacks. The paper discusses theoretical limitations of detecting attacks using observers and concludes that attacks that excite zero dynamics cannot be detected for linear systems. This claim does not apply to nonlinear systems, which is the focus of this work.

If real-time detection is not needed, another approach for linear systems that can be used is optimization methods [15]. Similar to the observer method, this approach creates estimates of the system state, but does not incorporate any observer gain. Instead, it solves

an optimization problem that searches for the smallest set of attacked components that can explain the measurement data. The solution to this optimization problem provides both the attack status and the most likely attacker strategy.

State-estimation approaches have also been implemented for nonlinear systems [16, 17]. These works have used nonlinear filters, including particle filters and particle swarm optimization filters, that estimate the state for systems with noise. Similar to the observer method, these filters run in real-time and include measurement feedback. Both of these works solve the problem by considering unknown inputs as unknown states. Consider a generic nonlinear state-space system

$$\begin{aligned} \dot{x} &= f(x, u, w) \\ y &= h(x, u, w) \end{aligned} \tag{2.4}$$

where f and h are nonlinear functions, and w is some unknown input either representing a fault or attack. To solve this, the state is replaced by combining x and w into a single unknown vector. Using this approach, nonlinear filters can estimate the state and the unknown inputs simultaneously, allowing them to perform both detection and identification. In [16], the attack detection problem was examined for attacks on the GPS system of unmanned aerial vehicles. This was accomplished by creating a first-principles model and implementing both particle filters and particle swarm optimization filters. In [17], the fault detection problem was examined for LOCAs in nuclear power plants. This was accomplished by creating a data-driven model and implementing particle filters. Using these approaches, nonlinear filters can estimate the state and unknown inputs simultaneously, allowing them to perform both detection and identification.

Another type of state-estimation technique has been used to detect faults in nuclear plant sensors called the multivariate state estimation technique [18, 19]. This technique uses historical data from normal operating conditions to create a data-driven regression model of the system dynamics. Using the model in real-time, it predicts current values for a set of sensor measurements based on the previous measured values. At each time step, it calculates the error between the predicted values and the measured values, which should follow a known statistical distribution under normal operation. Finally, it analyzes the sequence of errors

using statistical testing techniques to determine when the system diverges from normal operation, enabling it to detect faults.

Limitation The primary limitation to these state-estimation works is that they assume the system under attack is observable and then detect attacks based on that assumption. However, this is not guaranteed during zero-dynamics attacks targeting nonlinear systems. One major difference between linear and nonlinear systems regarding observability is that nonlinear observability can depend on the control actions inserted into the system. As we discuss more in Chapter 4, nonlinear systems under zero-dynamics attack require a non-zero control input to be observable. Therefore, additional work is needed to be able to detect zero-dynamics attacks.

Example State-estimation techniques for attack detection are demonstrated on the SMD system; see Figure 7 for the results. As mentioned above, zero-dynamics attacks are not detectable for linear systems, so we present an attack example where the system is observable under attack. For this example, the attacker has access to the position sensor but not the actuator. They attack the system by slowly injecting a ramp input into the sensor and the controller compensates by trying to return the mass back to the nominal position. This can be seen in the top figure, where the output remains zero because of the controller, but the true state increases as the attack magnitude increases. However from the bottom plot, this also has a resulting increase in the actuator force that is needed to keep the mass at the new position.

For this example, a Kalman filter, which is a stochastic state-estimator for linear systems, is used to estimate the state under attack. From the top plot, both the state and state estimate are shown to match, meaning the method effectively detected that the state does not match the output and an attack could be declared.

2.2.3 Other Methods

We also want to mention other methods of defending against attacks that do not fall into the above categories. The first method proposes modifying the system dynamics to reduce the possibility of zero-dynamics attacks [3]. This approach can increase the number

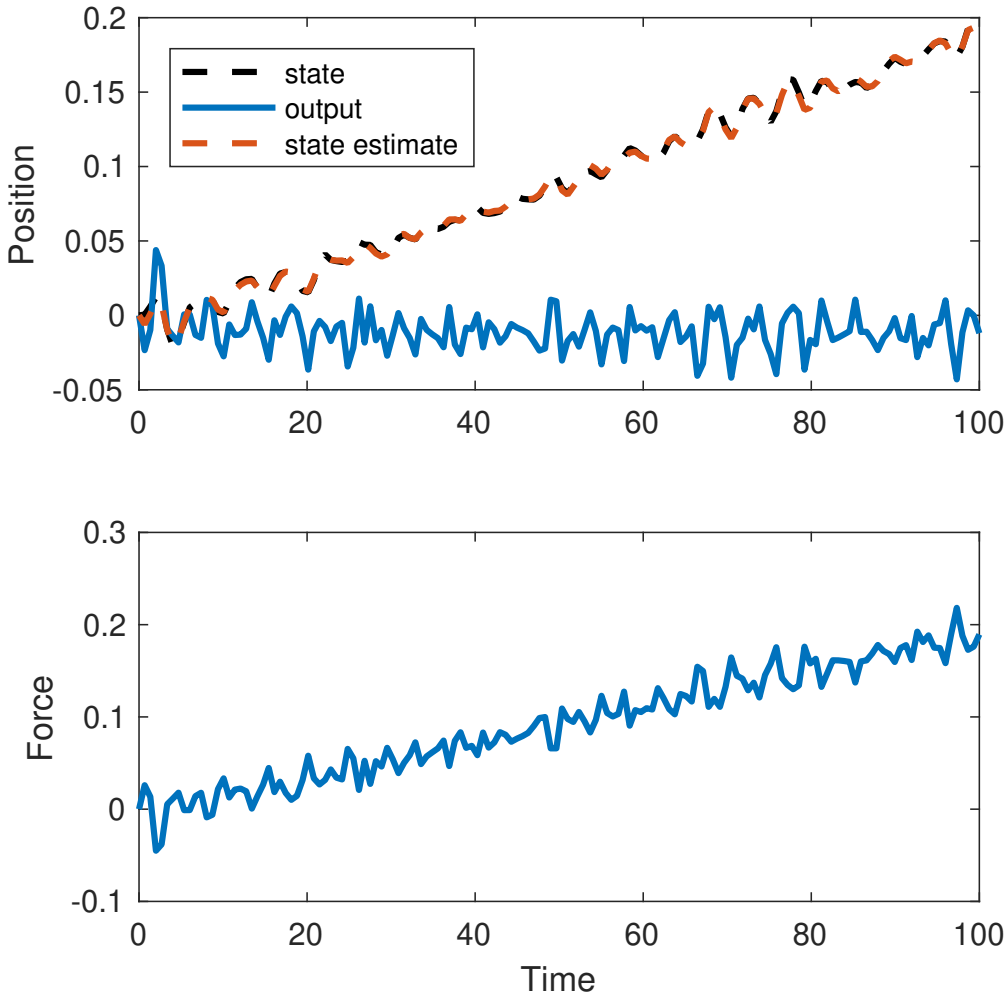


Figure 7: Results of the state estimator on the SMD system. The figure shows the state, the measured output, and the state estimate. For this example, the attacker has access to the position sensor but not the actuator. They attack the system by slowly injecting a ramp input into the sensor and the controller compensates by trying to return the mass back to the nominal position. But, the state estimator accurately estimates the state, meaning the attack is detected.

of signals an attacker needs to compromise in order to carry out a dangerous attack, making attacks more difficult to implement.

Limitation For new systems still in the design phase, this is an important step to design cyber security into the system. This is why our work analyzing nonlinear systems for vulnerability to zero-dynamics attacks is important. However, existing systems cannot be easily altered to remove vulnerabilities. To improve cyber resilience for these existing systems, additional methods should be employed.

The second method proposes encrypting the input and output signals using a modulation matrix [20]. This approach would ruin an attacker's perfect knowledge of the system, making it more difficult to implement zero-dynamics attacks that require such knowledge. Rather than proposing a limitation to this approach, we believe that this is worth implementing to help prevent attacks. However, it can be implemented alongside detection methods, because it is unlikely any single method is attack-proof.

2.2.4 Conclusion

This work focuses on using state-estimation techniques to detect the zero-dynamics attacks that are developed in the first part of this research. This can add an additional layer of defense to protect existing systems from the consequence of stealthy attacks.

2.3 Chapter Summary

To summarize this chapter, our work first focuses on identifying zero-dynamics attacks targeting nuclear power plants. The primary limitation that we overcome is extending existing methods to work with nonlinear system dynamics. Then, our work focuses on developing state-estimation methods to detect the nonlinear zero-dynamics attacks. The primary limitation that we overcome is focusing on observability as a function of control actions rather than as an assumed global property.

3.0 System Modeling

This work focuses on pressurized water reactors (PWRs). A PWR is a type of water-cooled nuclear power plant that operates at pressures well above atmospheric pressure, ensuring the water remains a liquid at high temperatures. This plant contains separate, closed, thermal-hydraulic primary and secondary loops. The primary loop generates heat and transfers it to the secondary loop, and the secondary loop uses the heat to generate steam, which is used in a Rankine cycle similar to other power plants. In general, PWRs have multiple primary and secondary loops connected to a single reactor pressure vessel.

The main components in the primary loop are the reactor pressure vessel, one or more steam generators, one or more reactor coolant pumps, and the pressurizer; see Figure 8 for a diagram. The reactor pressure vessel contains fuel rods, in which fission reactions generate heat. To remove the heat, water coolant flows through the vessel, transferring heat from the fuel rods and out of the vessel. In addition, the vessel contains control rods, which absorb neutrons in order to control the fission process.

After leaving the reactor pressure vessel, the coolant is pumped into the steam generators. These contain U-tube heat exchangers that transfer heat from the primary to the secondary loop. While the secondary loop is also pressurized, it is at a lower pressure, so water inside the steam generators can boil and produce steam. This steam then runs through a series of turbines to produce electricity, similar to other power plants.

The pressurizer is a saturated system that controls the pressure and coolant inventory using pressure and level feedback controllers. The pressure controller controls pressure by either adjusting the heater output or spraying cooler water. These actions either create or condense steam, resulting in pressure changes. In addition, the spray system has a bypass line that flows continuously at a low flow rate. The level controller controls liquid level by adjusting the net flow to the primary loop from a support system called the chemical and volume control system (CVCS). This pushes liquid up through the surge line into the pressurizer. There is also flow through the surge line from changes in the primary side temperature as the density changes. This research is implemented on the pressurizer system; see Figure 9 for a more detailed diagram of the pressurizer.

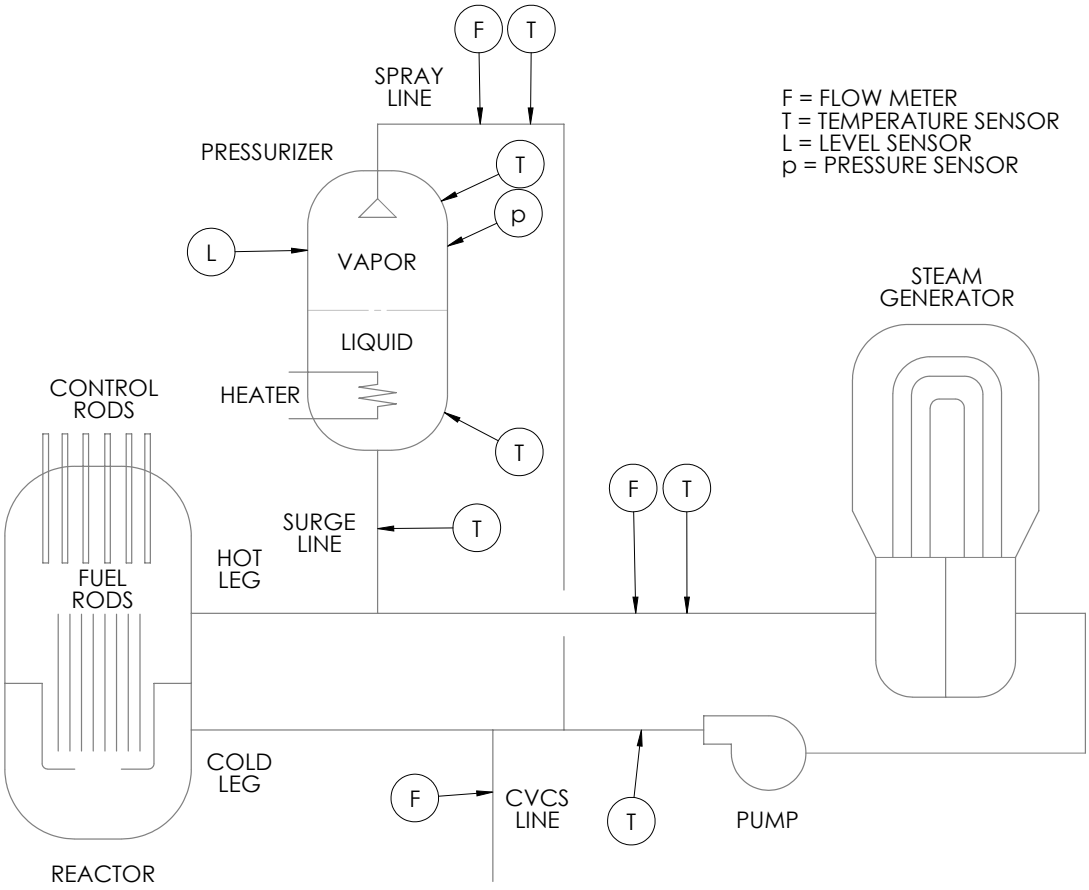


Figure 8: Schematic of the primary loop including sensor types and locations. This work focuses on the pressurizer subsystem.

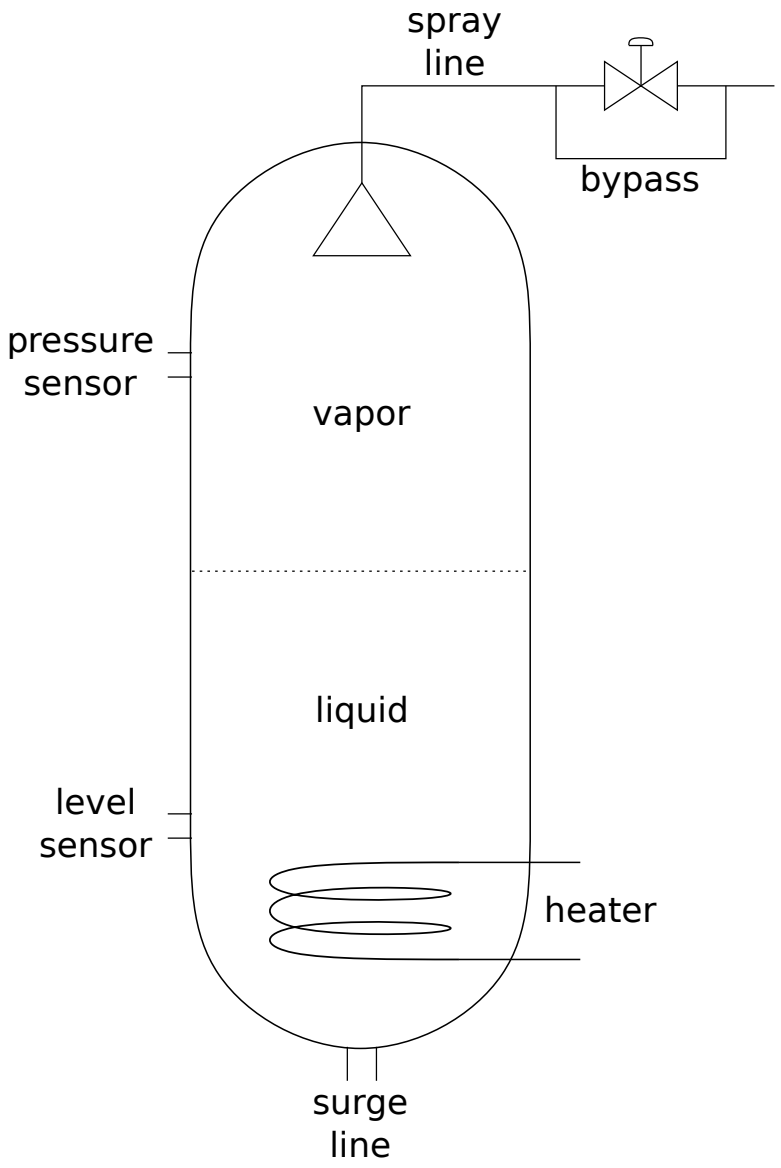


Figure 9: Detailed schematic of the pressurizer system. The pressure is controlled by the heater and spray flow. The level is controlled by the surge line flow.

We require dynamic models of the pressurizer system and develop both physics-based and data-driven models. As a brief primer on these types, physics-based methods use first-principles to develop models that describe the dynamics. They should be accurate as long any underlying assumptions hold; however, highly accurate models could require significant modeling effort. Data-driven methods rely on information contained in large quantities of historical data to develop models. These methods can be more accurate with less effort than a physics-based model; however, they may not generalize well far from the training data.

The data-driven model requires data, but both models use data in their development. The physics-based model is derived using first-principles but has a few unknown parameters. These values are estimated using system identification techniques, which use data. The data-driven model also uses system identification techniques, but uses them to estimate a fully unknown set of model parameters.

These models are used for two purposes: (i) to characterize zero-dynamics attacks, and (ii) to detect and diagnose zero-dynamics attacks. For characterizing attacks, the physics-based model is used because it captures the system dynamics over a large span of possible operating conditions. For detecting and diagnosing attacks, both models are used. First, the physics-based model is used to develop the methods and demonstrate their theoretical effectiveness. Second, the data-driven model is used to validate the methods using data from a commercial simulator. This model type is used because the physics-based model developed here is unable to achieve the necessary accuracy for the detection methods.

In addition to these models, we use system identification to identify the controller dynamics. These would normally be accessible if we were implementing our approach on a real plant. For this work, they are derived so the simulations have reasonable controller parameters, but are less important than the pressurizer dynamics.

All system data is collected using a commercial PWR simulator developed by GSE Systems. The plant simulated is a 970 MWe PWR, and the simulator uses the RETACT thermal hydraulics package, which can model non-homogeneous and non-equilibrium conditions. In addition, it has a model accuracy that is compliant with ANS-3.5, which is the American Nuclear Society's standard for "Nuclear Power Plant Simulators for Use in Operator Training and Examination" [21].

This chapter is structured as follows:

- the physics-based model is derived;
- the data-driven model structure is discussed;
- the system identification process for both models is described;
- the results of the system identification process for both models are shown and compared; and
- the results of the system identification process for the controllers are shown.

.

3.1 Physics-Based Pressurizer Model

The pressurizer dynamics are derived assuming the liquid and steam are both saturated. This means that the liquid and vapor temperatures are equal and are directly related to the pressure. The primary advantage to this model, often called an equilibrium model, is that it is significantly simpler than a non-equilibrium model that allows for subcooled liquid or superheated steam.

We believe this assumption is valid to use for a few reasons:

1. The spray system has a bypass line that flows continuously at a low flow rate. As a result, the heater has a nonzero steady-state output to compensate for the loss of energy. This energy loss in the vapor and energy gain in the liquid push both phases towards saturated conditions, making it harder for either to escape equilibrium conditions.
2. The primary cause of non-equilibrium conditions is a large influx of colder water into the pressurizer from the surge line. From an attacker's viewpoint, this research is focused on an attacker whose goal is to remain stealthy during the attack. As the attacker inputs get larger, they are more likely to be discovered by encountering other unknown dynamics. This suggests an attacker may instead choose to implement slower but longer attacks. From a defender's viewpoint, staying saturated is also beneficial because the dynamics are easier to model. Both of these mean that large influxes of cold water are likely to be avoided.

Table 1: State, input, and output variable conventions.

Variable	Description	Units
x_1	pressure	MPa
x_2	level	%
u_1	heater power	kW
u_2	surge flow	$\mathrm{kg\,s^{-1}}$
y_1	pressure sensor	MPa
y_2	level sensor	%

3. As mentioned, there is a spray bypass line that continuously pushes mass into the pressurizer from a spray accumulator. To compensate for this flow in, there must be an equal flow out of the primary loop into the accumulator. Therefore, for there to be any influx of colder water into the pressurizer, the flow rate entering the primary loop plus any expansion upward from changing primary loop temperatures must be greater than the spray bypass flow rate.

Another result of the nonzero steady-state heater output is that the heater can both increase and decrease the pressure. Therefore, the model can be simplified further by assuming the spray rate is just the bypass line and the pressure is purely controlled by the heater.

Based on the assumptions above, we can define the states, inputs, and outputs of the state-space model: the states are the pressure and liquid level; the control inputs are the heater and the surge line mass-flow rates; and the sensor outputs are the pressure and liquid level; see Table 1 for a summary of these variables.

The dynamics are derived using conservation of mass and volume on a single control volume that contains both the liquid and vapor phases [22]

$$\dot{m}_L + \dot{m}_V = W_{SU} + W_{SP} \tag{3.1}$$

$$\dot{E}_L + \dot{E}_V - V\dot{p} = \dot{Q}_{SU} + \dot{Q}_{SP} + \dot{Q}_H - \dot{Q}_W \tag{3.2}$$

where m is mass, E is energy, dot notation signifies $\frac{\mathrm{d}}{\mathrm{d}t}$, the L and V subscripts refer to the liquid and vapor phases, W_{SU} is the surge line mass-flow rate, W_{SP} is the spray line mass-flow rate and is equal to a constant, $\dot{Q}_{SU} = W_{SU}h_{SU}$, $\dot{Q}_{SP} = W_{SP}h_{SP}$, h is enthalpy, $\dot{Q}_H$ is the heater power, and $\dot{Q}_W$ is the environmental heat loss through the wall and is equal to a constant.

These conservation equations can be transformed into equations of the state variables using the chain rule, conservation of volume, and water properties. First, the derivatives of mass and energy can be written as

$$\dot{m} = \dot{\rho}V + \rho\dot{V} \tag{3.3}$$

$$\dot{E} = \dot{\rho}Vh + \rho\dot{V}h + \rho V\dot{h} \tag{3.4}$$

where ρ is density, and V is volume. Second, the derivatives of density and enthalpy for saturated phases can be written as

$$\dot{\rho} = \frac{\partial\rho}{\partial p}\dot{p} \tag{3.5}$$

$$\dot{h} = \frac{\partial h}{\partial p}\dot{p} \tag{3.6}$$

where p is pressure. Third, conservation of volume results in

$$V_L + V_V = V_{\text{Total}} \tag{3.7}$$

$$\dot{V}_L = A\dot{L} = -\dot{V}_V \tag{3.8}$$

where $V = AH$, A is the cross-sectional area, H is the height of the phase volume, and L specifically is the height of the liquid volume. Finally, the density and enthalpy can be approximated as linear functions of pressure

$$\rho(p) = c_1 p + c_2 \tag{3.9}$$

$$h(p) = c_3 p + c_4 \tag{3.10}$$

where the constants were determined using linear regression of data available from the International Association on Properties of Water and Steam Industrial Formulation 1997 [23]. Using these substitutions, Equations 3.1 and 3.2 can be written as a control-affine state-space

model. The control-affine designation is important later for characterizing the nonlinear zero-dynamics attacks.

As mentioned earlier, the surge line mass-flow rate is calculated as the sum of the net flow from the CVCS and the change from the primary side density. The net flow from the CVCS is directly measurable using sensor data. The flow from the primary side can be calculated using conservation of mass and the chain rule on the primary side

$$W_{SU,PL} = -\dot{\rho}_{PL} V_{PL} \tag{3.11}$$

where the negative is used because decreasing mass in the primary side results in increasing mass in the pressurizer, and the volume is assumed constant. This results in a total surge line flow $W_{SU} = W_{SU,PL} + W_{SU,CVCS}$.

In this model, there are a few dimensional parameters that are unknown. These parameters are the total volume of the pressurizer and the volume of the primary loop. Given real plant specifications, they could be approximated using schematics. In this work, they are estimated using system identification methods, discussed later, which should result in a more accurate model.

3.2 Data-Driven Pressurizer Model

The data-driven model captures the dynamics used for detection. The explanation of these dynamics is discussed in Chapter 5, but briefly they correspond to a large outsurge of liquid through the surge line with constant heater input. This transient maintains saturated conditions in the pressurizer, so the equilibrium assumption holds for the data-driven model. In order to describe these dynamics using a data-driven model, we select the inputs, outputs, number of states, and model structure.

For this application, we already have a physics-based model to base some of these selections on. The inputs and outputs are similar to before: the inputs are still the heater power and surge flow, and the outputs are still the pressure and level measurements. The only change is the surge flow is broken into two inputs. In a PWR, the total surge flow is

a combination of the charging flow, which is the flow entering the primary loop, and the letdown flow, which is the flow leaving the primary loop. In the data-driven model, these two inputs are separated, which was done because it improved model performance.

In addition to inputs and outputs, the number of states needs to be specified. Again to match the physics-based model, two states are used for this data-driven model. For simplicity, these states are the true pressure and level. This selection is able to sufficiently capture the dynamics, such that additional states are not needed.

Data-driven models need some type of model structure, which is the nonlinear function type. There are many generic model structures that can be used, and we considered polynomial functions and artificial neural networks (ANNs). ANNs, previously mentioned in Chapter 2, are able to handle highly nonlinear dynamics, but can require very large datasets to generalize well. Polynomial functions can still handle nonlinear dynamics and require smaller datasets to generalize well. Therefore, polynomial functions are a good place to start; after implementing them, we did not believe the additional nonlinear capabilities of ANNs were necessary.

A discrete-time polynomial state-space model can be written

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + E\zeta_k \\ y_k &= Cx_k + Du_k + F\zeta_k \end{aligned} \tag{3.12}$$

where A, B, E, C, D, and F are unknown system matrices that are estimated using system identification, and ζ_k is a vector of monomials made from the components of x_k and u_k. For example, if using a second degree polynomial model with two states x_1 and x_2 and an input u_1, then $\zeta = \begin{bmatrix} x_1^2 & x_1x_2 & x_1u_1 & x_2^2 & x_2u_1 & u_1^2 \end{bmatrix}^T$, where the time subscript k is dropped for notational convenience. For this work, the states are the pressure and level, so $C = I$, and $D = F = 0$.

The vector of monomials must be specified to implement the model. This was done by separating the data into training and testing sets. The training data is used to calculate the unknown parameters for a given model and the test set is used to determine the polynomial degrees. The optimal set of monomials is selected as the set that best generalizes to the test set.

When implementing the model, the states and inputs are normalized around an equilibrium condition. Each state and input is redefined by subtracting off the steady-state values for nominal conditions.

3.3 System Identification

In both the physics-based and data-driven models, there are unknown parameters. These parameters are estimated using system identification techniques, which determine the values that best match observation data according to some optimality criteria. For this work, the parameters are estimated using maximum likelihood estimation, which optimizes the total likelihood of a set of noisy observations.

These techniques compare measurement data to model estimates at discrete time steps. Both of the models can be written using the discrete-time nonlinear representation

$$\begin{aligned} x_{k+1} &= f(x_k, u_k, \theta) \\ y_k &= Cx_k + v_k \end{aligned} \tag{3.13}$$

where θ is the unknown parameter vector, and $v_k \sim \mathcal{N}(0, R)$ is zero-mean Gaussian measurement noise with covariance equal to R. Based on this model, the estimated measurements as a function of a given θ can be written as

$$\begin{aligned} \hat{x}_{k+1} &= f(\hat{x}_k, u_k, \theta) \\ \hat{y}_k &= C\hat{x}_k \end{aligned} \tag{3.14}$$

where $\hat{}$ signifies an estimated value.

This estimated model is used to calculate the total likelihood of a sequence of measurements. The likelihood of observing a single measurement for a given parameter vector $p(y_k|Y_{k-1}, \theta)$ is normally distributed $\mathcal{N}(\hat{y}_k, R)$. The likelihood of observing a series of measurements $Y_T = \{y_0, \ldots, y_T\}$ is

$$p(Y_T|\theta) = \prod_{k=0}^{T} p(y_k|Y_{k-1}, \theta) \tag{3.15}$$

For numerical reasons, it is common to use the natural logarithm of the likelihood function. This creates a sum of log-likelihood values instead of a product, but still has the same maximum location. This results in

$$\begin{aligned}\log p(Y_T|\theta) &= \sum_{k=0}^{T} \log p(y_k|Y_{k-1},\theta) \\ &= \sum_{k=0}^{T} \log\left(\frac{1}{(2\pi)^k|R|}\exp\left(-\frac{1}{2}\tilde{y}_k^T R^{-1}\tilde{y}_k\right)\right) \\ &\propto -\sum_{k=0}^{T} \tilde{y}_k^T R^{-1}\tilde{y}_k \end{aligned} \tag{3.16}$$

where the second line is the probability density function of the multivariate Gaussian distribution, and $\tilde{y}_k = \hat{y}_k - y_k$ is the output estimation error. In words, the log-likelihood is proportional to the negative of the weighted least-squares error using the inverse of the covariance matrix as the weight. As such, minimizing the weighted least-squares error provides the same solution and is easier to calculate.

The optimization problem can be written

$$\theta^* = \arg\min_{\theta} \quad V(\theta) \tag{3.17}$$

where $V(\theta) = \sum_{k=0}^{T} \tilde{y}_k^T R^{-1}\tilde{y}_k$ is the objective function, and θ^* is the optimal parameter estimate. The problem is solved using gradient descent methods, which iteratively calculate new estimates based on the gradient of the objective function

$$\theta_{n+1} = \theta_n - \gamma \boldsymbol{\nabla} V(\theta_n) \tag{3.18}$$

where n is the iteration number, γ is a step size, and the gradient is defined with respect to the unknown parameter vector $\boldsymbol{\nabla} V(\theta_n) = \frac{dV}{d\theta}\big|_{\theta=\theta_n}$. This algorithm is implemented differently for the physics-based and data-driven models.

3.3.1 Physics-Based Model

The gradient descent optimization for the physics-based model is implemented by numerically approximating the gradient. This is done using the finite difference approach, which estimates the gradient at a given point by evaluating the objective at nearby points. Based on that numerical gradient, the next candidate parameter vector is calculated as outlined above.

This numerical approach is taken for the physics-based model because it is easy to calculate. The reason for this is that the objective function has already been defined, so no additional derivations are necessary to implement it. The primary challenge with this numerical approach is that it requires significantly more function evaluations, and that gets exponentially worse as the number of variables increases. But, the physics-based model has few unknown parameters, so the dimensionality is not a major problem. In addition, the data does not need to describe a large range of conditions, meaning it can be a small dataset. This makes function evaluations extremely fast, so the potential challenges of this numerical approach are not prohibitive for the physics-based model.

3.3.2 Data-Driven Model

Compared with the physics-based model, the data-driven model has many unknowns and a much larger dataset because the entire model structure must be estimated. This means it is worth the extra effort to calculate the gradient directly. For the discrete-time polynomial model, there exists a closed-form method of solving for the gradient, which is described here.

First, we introduce a few notational elements that are used to calculate the gradient:

$$\zeta_k' = \frac{\partial \zeta_k}{\partial x_k} \tag{3.19}$$

$$I_{ij}^{m\times n} = \begin{bmatrix} 0 & \dots & 0 & \dots & 0 \\ \vdots & & \vdots & & \vdots \\ 0 & \dots & 1 & \dots & 0 \\ \vdots & & \vdots & & \vdots \\ 0 & \dots & 0 & \dots & 0 \end{bmatrix} \begin{matrix} \\ \\ \text{i} \\ \\ \\ \end{matrix} \quad \begin{matrix} \\ \\ \text{j} \\ \\ \\ \end{matrix} \tag{3.20}$$

where $I_{ij}^{m\times n} \in \mathbb{R}^{m\times n}$ is a zero matrix with a one at entry (i, j).

The gradient of the objective function is calculated as

$$\frac{dV}{d\theta} = 2\sum_{k=0}^{T} \left(\frac{d\tilde{y}_k}{d\theta}\right)^T R^{-1}\tilde{y}_k \tag{3.21}$$

where

$$\frac{d\tilde{y}_k}{d\theta} = \frac{d\hat{y}_k}{d\theta} \tag{3.22}$$

can be calculated as the solution to a dynamic system. For each entry (i, j) of the three unknown system matrices A, B, and E, this is written

$$J_k^{Xij} = \frac{\partial \hat{y}_k}{\partial X_{ij}} \tag{3.23}$$

with $X \in \{A, B, E\}$. The dynamic system that solves for this is [24]

$$x_{k+1}^{Aij} = I_{ij}^{n_x\times n_x}\hat{x}_k + (A + E\zeta_k')x_k^{Aij} \tag{3.24}$$

$$J_k^{Aij} = Cx_k^{Aij} \tag{3.25}$$

$$x_{k+1}^{Bij} = I_{ij}^{n_x\times n_u}u_k + (A + E\zeta_k')x_k^{Bij} \tag{3.26}$$

$$J_k^{Bij} = Cx_k^{Bij} \tag{3.27}$$

$$x_{k+1}^{Eij} = I_{ij}^{n_x\times n_z}\zeta_k + (A + E\zeta_k')x_k^{Eij} \tag{3.28}$$

$$J_k^{Eij} = Cx_k^{Eij} \tag{3.29}$$

where n_x, n_u, and n_z are the cardinalities of x, u, and ζ, respectively. The solution to this dynamic system is directly used to calculate the gradient of the objective function. For the data-driven model, this is much faster and more efficient than numerically calculating the gradient.

In order to implement this system identification process, the coefficients are first calculated for the linear model using the above process. This means finding the optimal values of A and B, with $E = 0$. Once these optimal values have been estimated, they can be used as the initial condition for further optimization with $E_0 = 0$.

3.4 Results

The system identification techniques are implemented for both the physics-based and data-driven models. These techniques use data collected from the GSE simulator to estimate parameters. After system identification has been completed, the models are fully defined and can be used in later chapters.

To make the data more representative of real plant data, additive Gaussian white noise is added to the sensor measurements. The standard deviations of these noise distributions is 0.1 % of the full sensor range, as defined by the simulator, resulting in noise standard deviations of 4.8×10^{-3} MPa and 1.0×10^{-1} % for the pressure and level sensors, respectively.

3.4.1 Results for Physics-Based Model

For the physics-based model, there are only a few parameters that need to be estimated. This means we can use a relatively small dataset to identify these unknowns. As mentioned, the relevant signals are the pressure, level, heater power, and surge flow; see Figure 10 for a plot of these variables, not including noise.

Using these inputs and outputs and including noise, the optimal unknown parameters are calculated. The optimal values should enable the estimated output to track the simulator data; see Figure 11 for a plot of the pressure and level from the simulator compared with

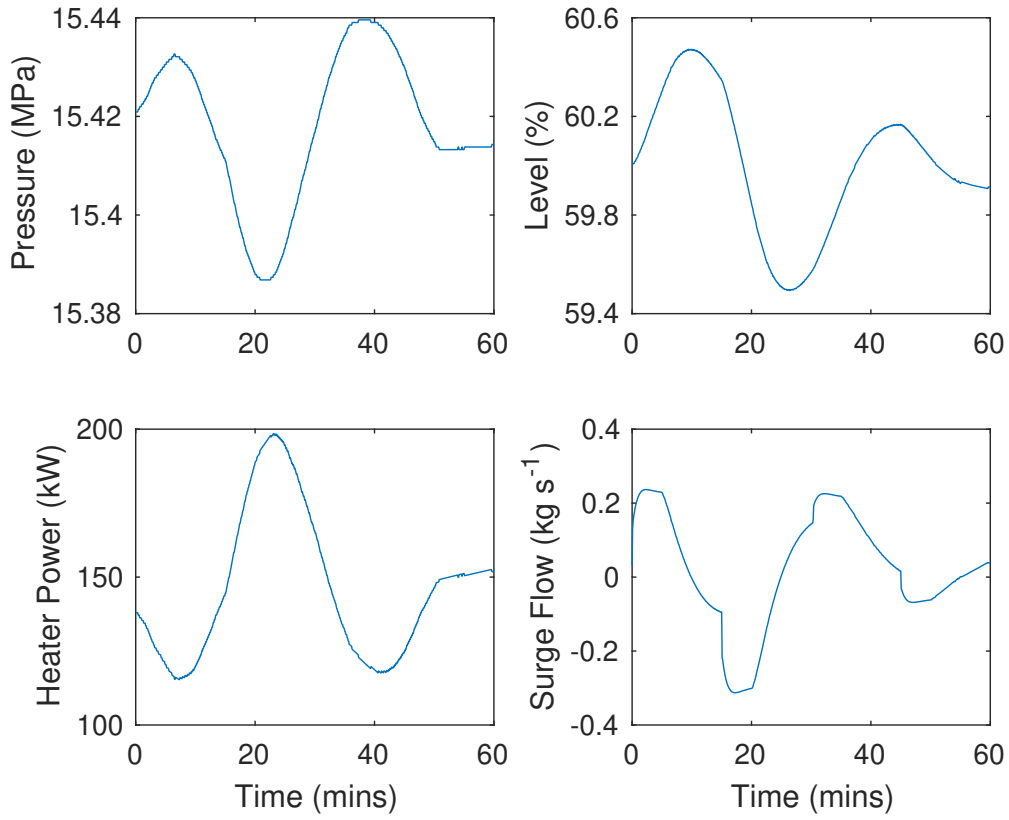


Figure 10: Data used to estimate parameters for the physics-based model. The top two plots are output variables, and the bottom two plots are input variables.

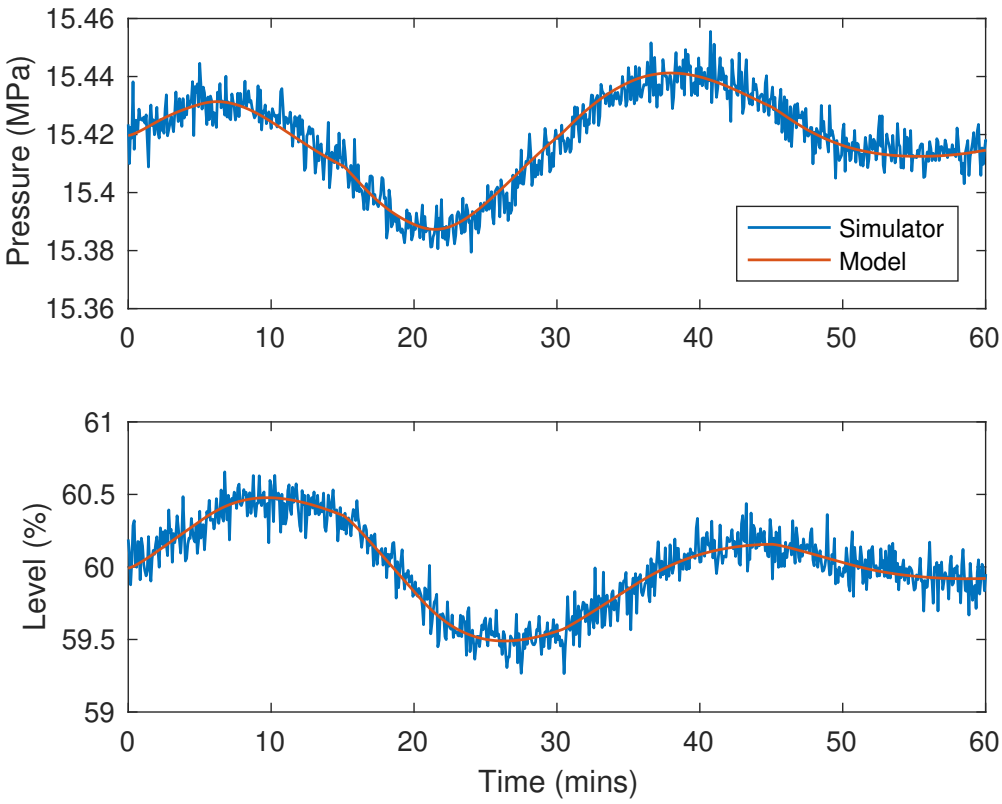


Figure 11: Comparison of the pressure and level data between the commercial simulator and physics-based model. The simulator data includes noise, but the model data does not because it is the optimal model output that fits the noisy data.

our optimal model. From this plot, the model captures the simulator data well, suggesting that our assumptions hold for this dataset.

3.4.2 Results for Data-Driven Model

For the data-driven model, there are many unknowns that need to be estimated, but for a very specific transient type. The data collected includes multiple runs of the detection transient from varying initial conditions. The relevant signals are shown for multiple runs starting from different conditions; see Figure 12.

Using these inputs and outputs and including noise, the optimal polynomial model is calculated. This is done using the training dataset as mentioned previously. The optimal model should enable the estimated output to track the simulator data; see Figure 13 for a plot of the pressure and level from the simulator compared with our optimal model. Note that this simulation comes from the test set, so was not used to directly estimate the model parameters. From this plot, the model captures the simulator data well.

3.4.3 Comparing Physics-Based and Data-Driven Models

In order to validate the detection methods using simulator data, the model must be very accurate. This level of accuracy is quantified in more detail in Chapter 5. Here, the accuracy of the two models is discussed using data from the detection transients. Note that we expect the data-driven model to be more accurate because it was designed specifically to detect this type of transient.

For the comparison, the dataset used is the same as shown for the data-driven results. In addition, the dataset does not include noise to more easily see the accuracy difference; see Figure 14 for the results from the two models compared with the simulator data. As expected, the data-driven model captures the dynamics extremely accurately. By contrast, the physics-based model captures the general trend and is pretty accurate given that is has never seen this transient before.

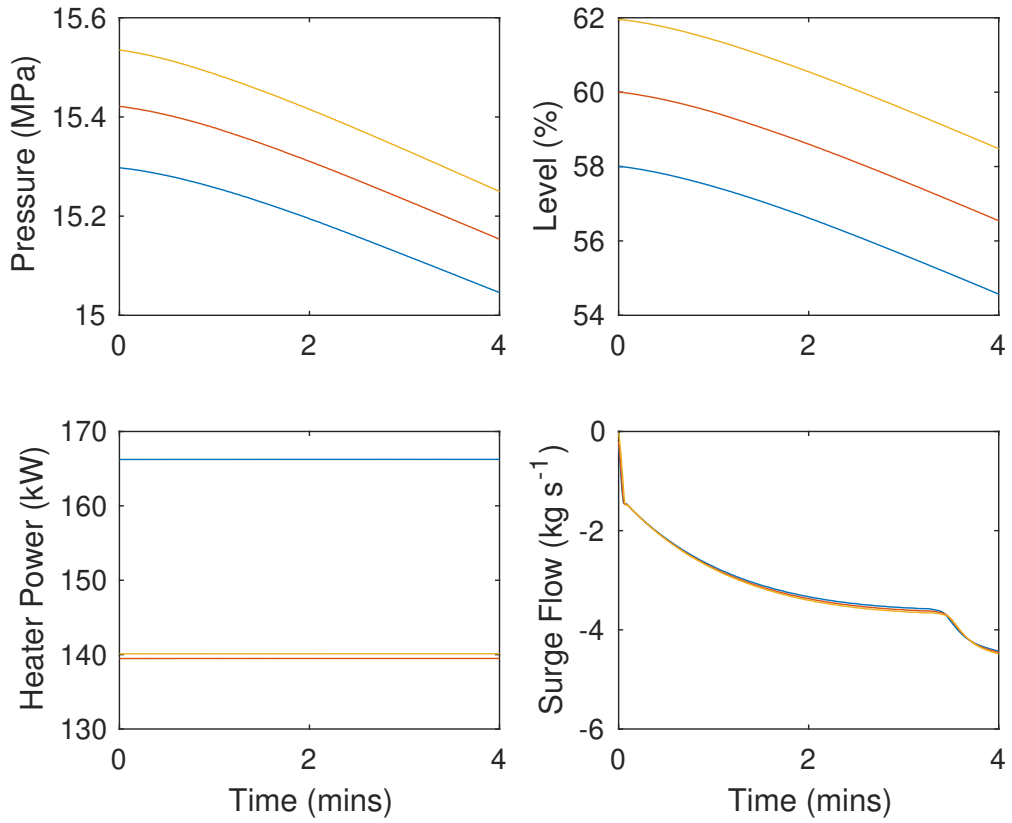


Figure 12: A sample of the data used to estimate parameters for the data-driven model. The top two plots are output variables, and the bottom two plots are input variables. In the four plots, the colors each correspond to a different dataset, meaning three datasets are shown here.

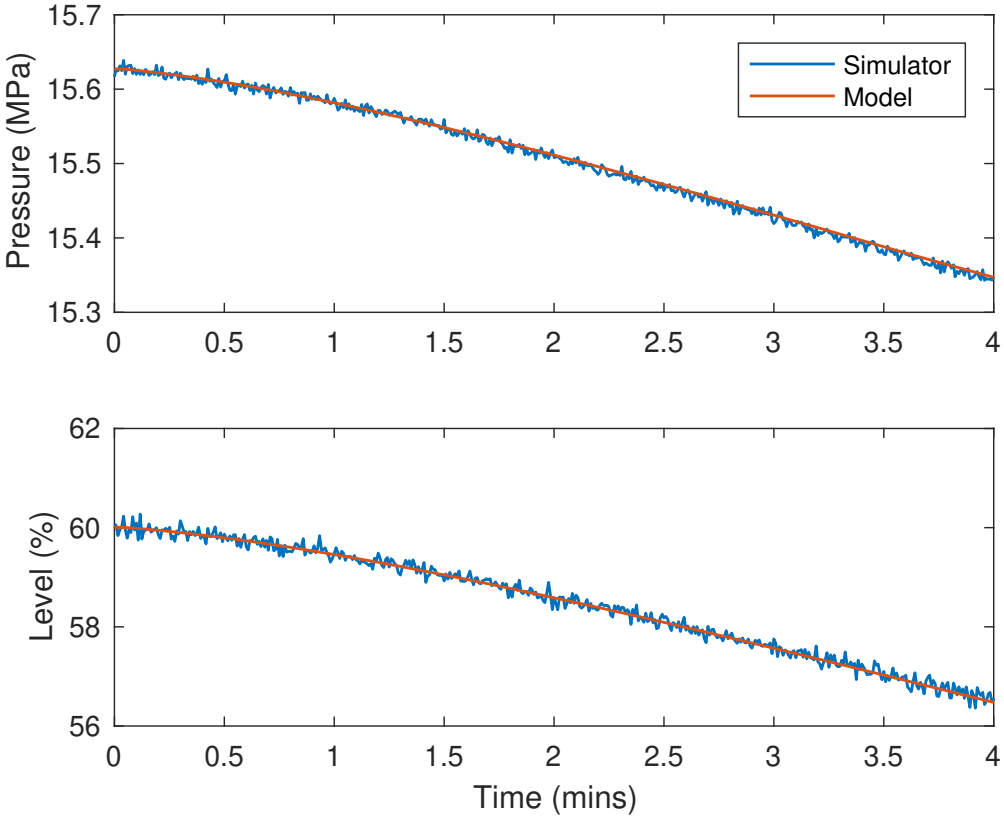


Figure 13: Comparison of the pressure and level data between the commercial simulator and data-driven model. This plot shows just one of the datasets for reference. The simulator data includes noise, but the model data does not because it is the optimal model output that fits the noisy data.

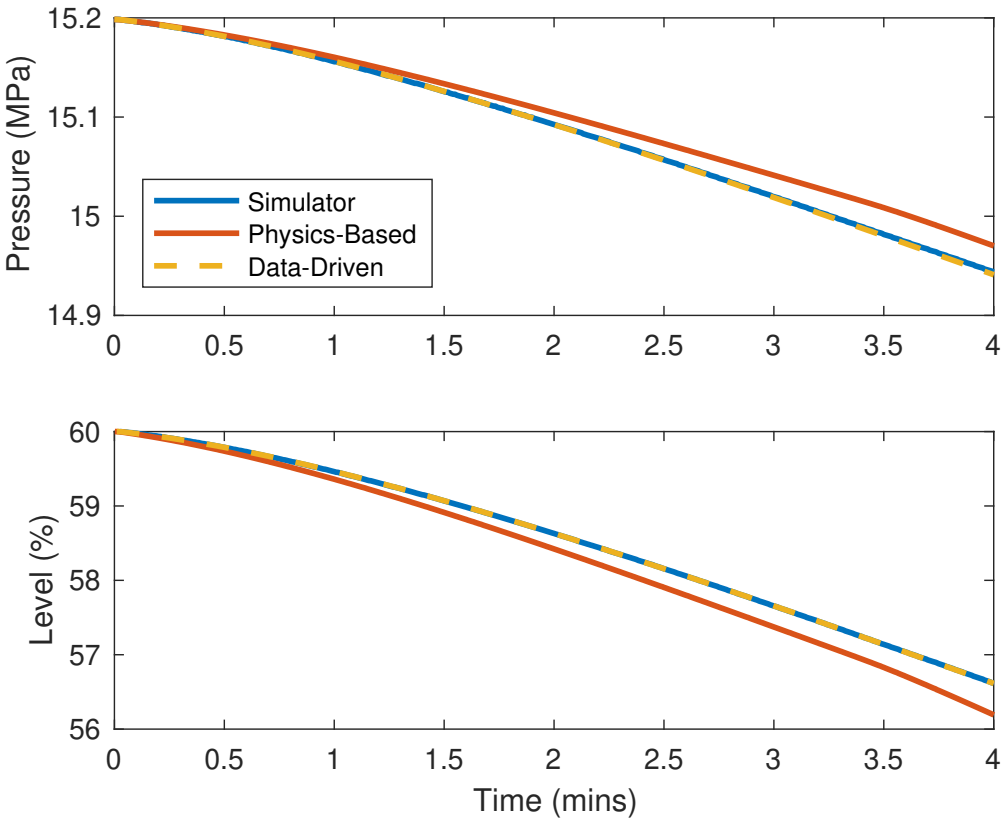


Figure 14: Comparison of the pressure and level data between the physics-based and data-driven models. This plot shows the simulator data and the estimates from both the physics-based and data-driven models. From the plot, the data-driven model is an excellent match for the simulator data. By contrast, the physics-based model captures the phenomena, but does not perform as well.

3.5 Controller Models

For later simulations, a realistic controller model is helpful. The pressurizer uses classic proportional integral (PI) controllers. These have the transfer function

$$C(s) = \frac{u(s)}{e(s)} = k_P + \frac{k_I}{s} \tag{3.30}$$

and state-space representation

$$\begin{aligned} \dot{x}_c &= e \\ u &= k_I x_c + k_P e \end{aligned} \tag{3.31}$$

where $e = r - y_s$. For the pressure controller, the reference pressure is constant $r_{\text{pressure}} = 15.42\,\text{MPa}$. For the level controller, the reference level is a function of the average reactor coolant system temperature $r_{\text{level}} = 25 + 1.1(T_{\text{RCS}} - 291.7\,^\circ\text{C})$.

The gains for these controllers are estimated using the commercial simulator in order to get realistic values for an actual plant. This is done using Matlab's built-in `ssest` function, which estimates an unknown linear state-space model based on data; see Figures 15 and 16 for the results of this controller system identification.

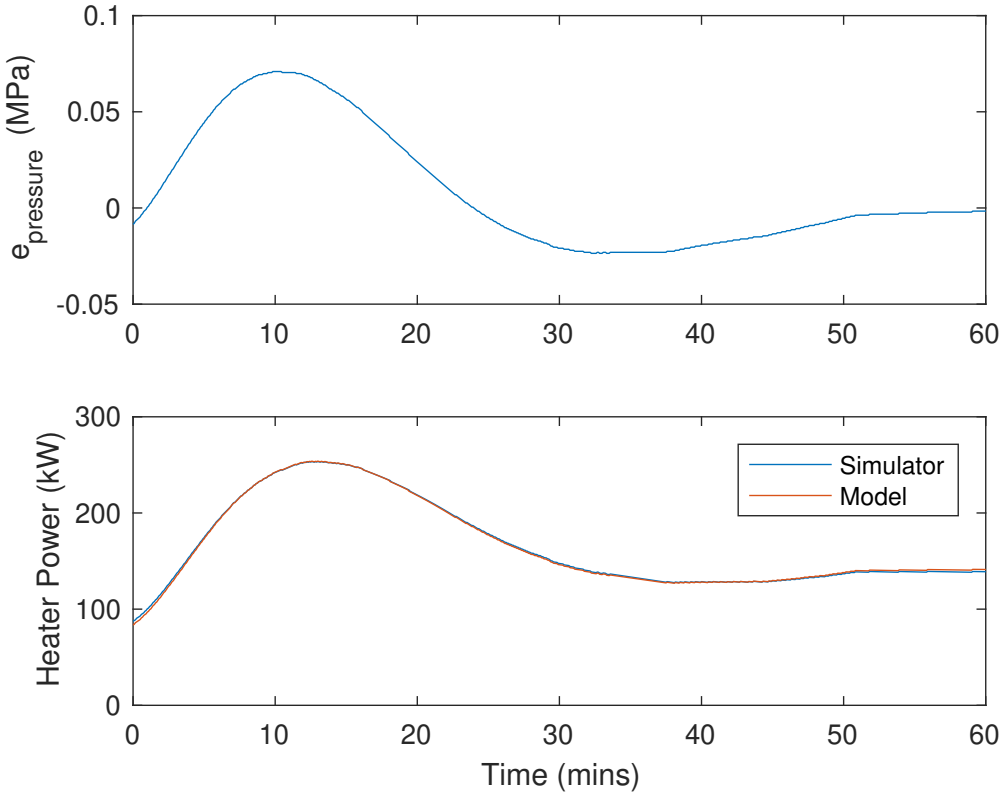


Figure 15: Comparison of the heater output data and the PI model. The top plot shows the error signal between the true pressure and setpoint, and the bottom plot shows both the simulator data and the model estimates.

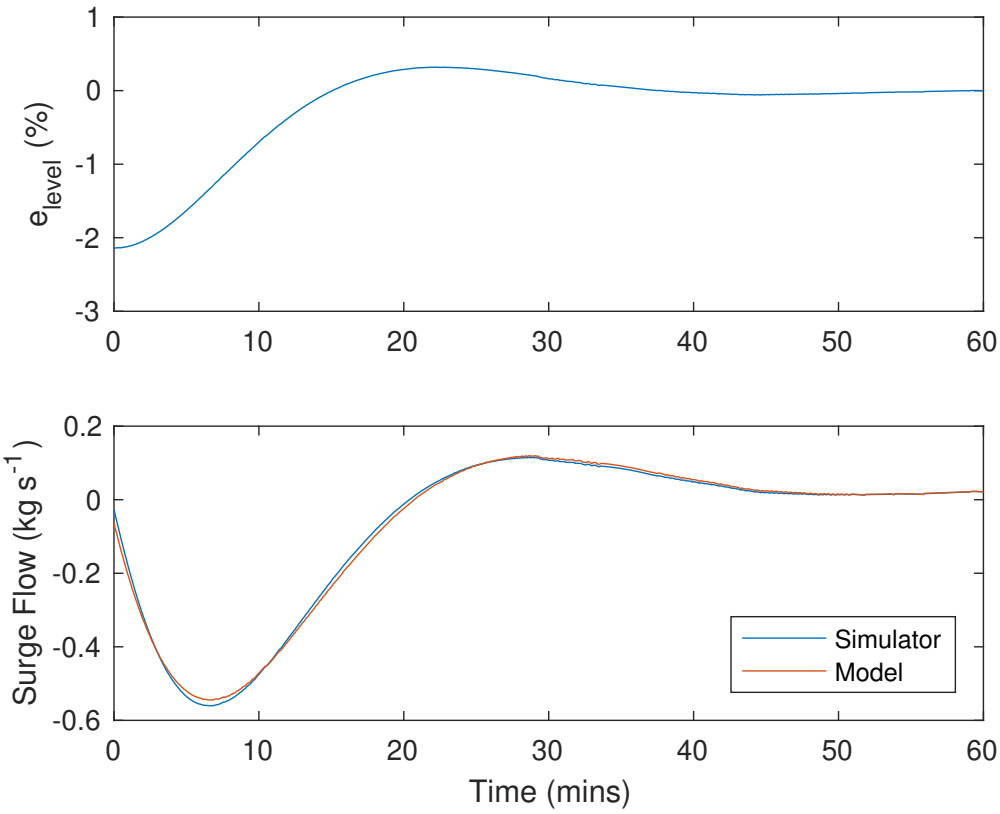


Figure 16: Comparison of the surge flow data and the PI model. The top plot shows the error signal between the true level and setpoint, and the bottom plot shows both the simulator data and the model estimates.

4.0 Characterizing Nonlinear Zero-Dynamics Attacks

The first primary task in this research is to characterize stealthy cyber-attack strategies targeting nuclear power plants. The two main stealthy attack strategies previously researched are replay attacks and zero-dynamics attacks [1, 2, 4]. With the goal of identifying worst-case scenarios, this effort is focused on zero-dynamics attacks. While this has been done for linear systems, linear approaches may prove less accurate and could result in incorrect conclusions about a system's zero dynamics. In order to complete this task for a plant's nonlinear system dynamics, this research extends zero-dynamics attacks to nonlinear systems.

Our approach works directly with the nonlinear dynamics to investigate zero-dynamics attacks. First, the nonlinear attacked system is defined, where an attacker can modify both control signals and measurement signals. Then, invariant subspace techniques are used to identify the largest submanifold in which an attacker can maintain zero output. This is done using an iterative algorithm that determines whether a nontrivial solution exists. If it exists, then the zero dynamics can be calculated directly from the equations in the algorithm.

This approach is then implemented on the pressurizer subsystem. The algorithm is used to evaluate all combinations of attackable signals. These are then classified by the existence of nontrivial zero dynamics and the stability of the equilibrium condition when nontrivial zero-dynamics exist. They are compared using a damage metric for how long it would take for an attacker to reach an undesirable system state. Finally, the stability of the attack is analyzed, which differs from the stability of the equilibrium condition. The distinction is discussed later.

This chapter is structured as follows:

- the state-space model under attack is described;
- an algorithm for solving the zero-dynamics attacks targeting nonlinear dynamics is presented;
- the results of implementing the algorithm on the pressurizer system are discussed; and
- the stability analysis on the attack is presented.

4.1 State-Space Model Under Attack

This work considers nonlinear control-affine systems of the form

$$\begin{aligned} \dot{x}_s &= f_s(x_s) + g_s(x_s)u \\ y_s &= h_s(x_s) \end{aligned} \tag{4.1}$$

where $x_s \in U \subset \mathbb{R}^n$ is the system state, $u \in \mathbb{R}^q$ is the system input, $y_s \in \mathbb{R}^p$ is the system output, $f_s(\cdot)$ and $g_s(\cdot)$ are nonlinear functions that describe the system dynamics, and $h_s(\cdot)$ is a nonlinear function that describes the output. In addition, $y_s = 0$ and $u = 0$ define the system's equilibrium point, which can be accomplished by subtracting off the reference value if needed.

In order to incorporate the attacker into the model, we assume they can modify some combination of inputs and outputs. In the model, this means rewriting the system input and output to include the attacker. For the input attack, this can be done simply as $u \Rightarrow u + a_u$, where a_u is the input attack signal. For the output attack, this is somewhat more complicated. In order to use existing nonlinear zero-dynamics tools, the attacked system model should also be in control-affine form. This means the output should be purely a function of the state. Therefore, we define an attack state so that $y_s \Rightarrow y_s + x_a$, with $\dot{x}_a = a_y$, where a_y is the output attack signal. This mathematical transformation should have no influence on the system's zero dynamics [25]. The result is the attacked system model

$$\begin{aligned} \dot{x}_s &= f_s(x_s) + g_s(x_s)(u + a_u) \\ \dot{x}_a &= a_y \\ y_s &= h_s(x_s) - x_a \end{aligned} \tag{4.2}$$

Finally, we assume the system has some closed-loop control law. For the pressurizer system, this is a simple linear controller. However, to keep things general, a nonlinear controller can be used

$$\begin{aligned} \dot{x}_c &= f_c(x_c, y_s) \\ u &= h_c(x_c, y_s) \end{aligned} \tag{4.3}$$

where $x_c \in \mathbb{R}^c$ is the controller state if the controller has dynamics.

The final result is a closed-loop system under attack. The combined state can be written $x = \begin{bmatrix} x_s^T & x_a^T & x_c^T \end{bmatrix}^T$, and the combined output is simply the system output $y = y_s$. More explicity, we get the combined system dynamics

$$\dot{x} = \begin{bmatrix} f_s(x_s) + g_s(x_s)h_c(x_c, h_s(x_s) - x_a) \\ 0 \\ f_c(x_c, h_s(x_s) - x_a) \end{bmatrix} + \begin{bmatrix} g_s(x_s) & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} a_u \\ a_y \end{bmatrix} \tag{4.4}$$

and the combined system output

$$y = h_s(x_s) - x_a \tag{4.5}$$

Together, this results in the attacked nonlinear control-affine system

$$\begin{aligned} \dot{x} &= f(x) + g(x)a \\ y &= h(x) \end{aligned} \tag{4.6}$$

where a is the combined attacker input signal. With a slight abuse of notation, this new attacked system has q inputs and p outputs.

4.2 An Algorithm for Calculating Zero Dynamics

Zero dynamics exist when the output can be set to and remain zero using a suitable choice of initial state and input. If they exist, the *zero dynamics* refers to the nonzero system dynamics that maintain zero output. In solving for them, there are two critical pieces that must be accounted for. First, there is the subspace on which zero dynamics exists, called the *output-zeroing submanifold*. Second, there is the corresponding input that maintains zero output, called the *output-zeroing input*.

Zero dynamics can be more rigorously defined using invariance concepts:

Definition. A submanifold is *locally invariant* at x_0 in a neighborhood U_0 of x_0 if all points that belong to the submanifold remain in the submanifold after a specified transformation.

Definition. A submanifold is *locally controlled invariant* if there are control actions that can make the submanifold locally invariant.

For the control-affine system of Equation 4.6, a submanifold M is locally controlled invariant at x_0 in a neighborhood U_0 of x_0 if there always exists an input a^* such that the vector $f^* = f(x) + g(x)a^*$ is tangent to M for all $x \in M$. Using these invariance definitions:

Definition. *Zero dynamics* exist if there exists a submanifold M of U containing the initial state x_0 that

1. has zero output everywhere on M, and
2. is locally controlled invariant at x_0 in a neighborhood U_0 of x_0.

If such a submanifold exists, then the *zero dynamics* are the dynamics f^*.

To create a more concrete understanding of these concepts, we will use an example from orbital dynamics. For a moon on a stable orbit around a planet, the orbital path represents an invariant set. If the motion of the moon is described by $\dot{x} = f(x)$, this means that $f(x)$ is always tangent to the path, ensuring that the moon stays on the orbital path; see Figure 17 for a diagram of this invariant orbit. Now, for a satellite on an imperfect orbit around a planet with thrusters that maintain the orbit, the orbital path represents a controlled invariant set. If the dynamics are represented by $\dot{x} = f(x) + g(x)u$, then the satellite without using the thrusters would stray off the path. However, the satellite can remain on the orbital path because there always exists a thruster input u^* that keeps $f(x)+g(x)u^*$ tangent to the orbit; see Figure 18 for a diagram of this controlled invariant orbit[1].

The zero-dynamics problem is related to the controlled invariant set from the satellite orbit. If we can measure the distance between the satellite and the orbit, then the orbit is the submanifold with zero output, and the thrusters are the control inputs. For the satellite orbit problem, zero-dynamics exist if there always exists a thruster input that keeps the satellite on the orbit.

The definition above of zero dynamics does not necessarily result in a unique submanifold. For example, the output at the equilibrium condition x_{eq} is by definition zero, resulting in the

[1]With thanks to *Made by Made*, *Markus*, and *Prettycons* from the Noun Project for graphics.

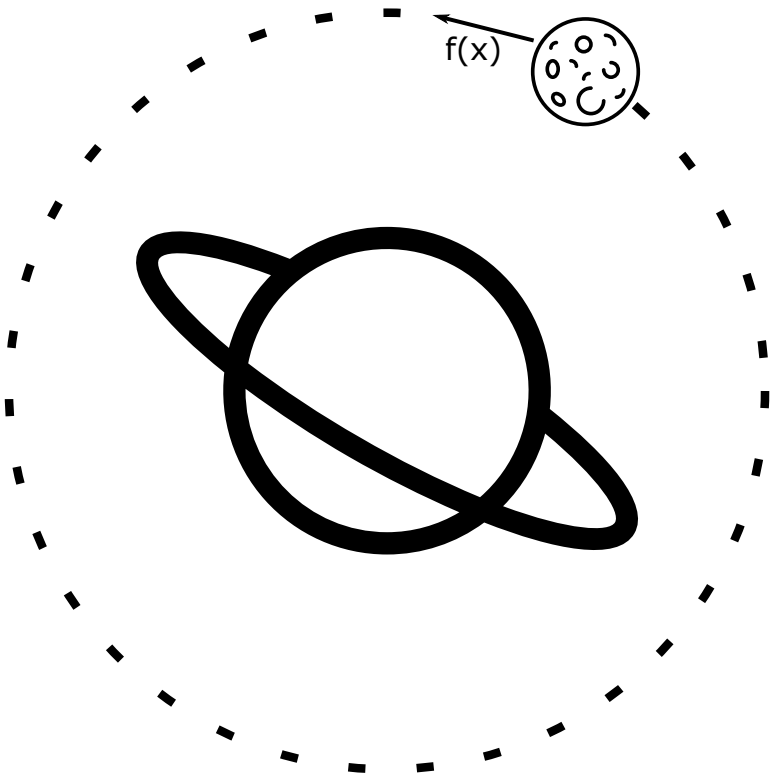


Figure 17: Diagram of an invariant orbit. The planet is in the center and the moon orbits around it along the dashed curve. The orbital path represents an invariant set because its dynamics, described by $f(x)$, are always tangent to the path, ensuring that the moon stays on the orbital path.

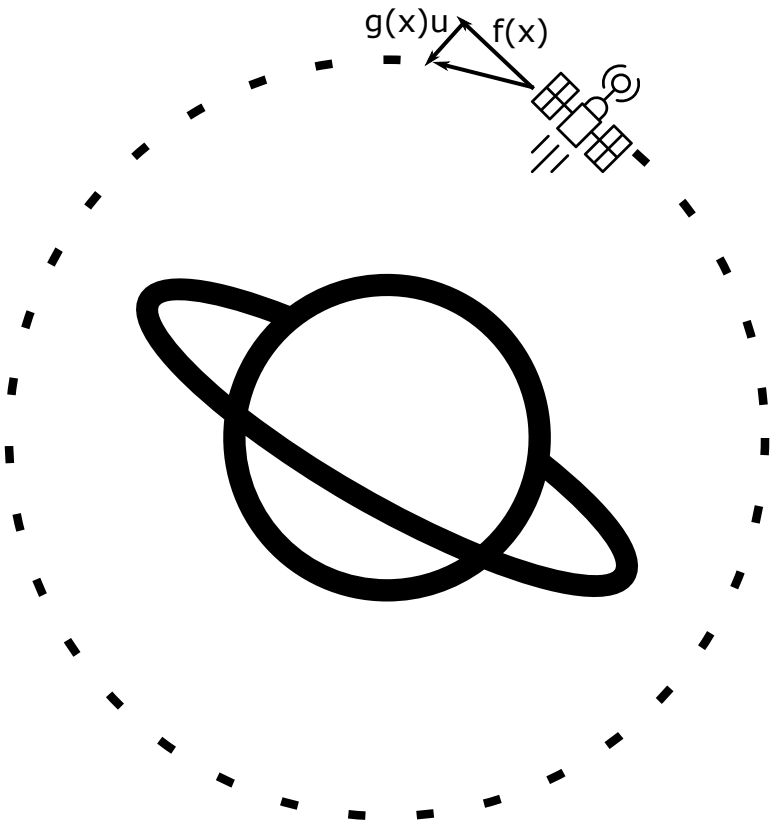


Figure 18: Diagram of a controlled invariant orbit. The planet is in the center and the satellite orbits around it along the dashed curve. The orbital path represents a controlled invariant set because its dynamics, described by $f(x) + g(x)u$, can always be made tangent to the path by control actions u, ensuring that the satellite stays on the orbital path.

Algorithm 1 This algorithm finds the maximal output-zeroing submanifold.

```
define constraints
define candidate
while isNotTrivial and isNotInvariant do
    calculate additional constraints
    update constraints with additional constraints
    update candidate using constraints
end while
```

trivial output-zeroing submanifold $M = \{x_{eq}\}$. In solving for the submanifold, the objective becomes to calculate the maximal output-zeroing submanifold, M^*.

From this definition, there are three pieces to solve for: (1) the maximal output-zeroing submanifold M^*, (2) the corresponding output-zeroing input a^*, and (3) the resulting zero dynamics f^*.

4.2.1 Maximal Output-Zeroing Submanifold

The maximal output-zeroing submanifold is calculated using the iterative algorithm found in [26]; refer to that for additional details and examples. Each iteration starts with a constraint function that defines a candidate submanifold. This candidate submanifold is then analyzed to determine whether it is either trivial or locally controlled invariant. If it is either, the algorithm converges and the candidate submanifold is the maximal output-zeroing submanifold. Otherwise, the constraint function is updated with an additional constraint, defining a new candidate submanifold. This process continues until it converges. See Algorithm 1 for an overview of this process.

Step 0: Define the initial constraint function A constraint function $\Phi_k(x) = 0$ is used to define a candidate submanifold M_k. The initial constraint function $\Phi_0 = h(x)$ is used to guarantee its corresponding initial candidate submanifold M_0 has zero output everywhere on M_0, fulfilling the first requirement of the zero dynamics definition. In addition, all future

constraint functions Φ_{k+1} contain Φ_k, such that $M_{k+1} \subset M_k$. This ensures that all future candidate submanifolds also fulfill this first requirement of the zero-dynamics definition.

Step 1: Define the candidate submanifold Using the current constraint function $\Phi_k(x)$, the candidate submanifold is defined as

$$M_k = \{x \in U_k : \Phi_k(x) = 0\} \tag{4.7}$$

where U_k is a neighborhood containing x_0 that ensures M_k is a smooth submanifold. If the candidate submanifold is the trivial solution, the algorithm converges with $M_k = M^*$. Otherwise, continue to Step 2.

Step 2: Test for controlled invariance Given the candidate submanifold M_k, it is necessary to check whether it is locally controlled invariant. This can be done by calculating whether

$$L_f\Phi_k(x) + L_g\Phi_k(x)a = 0 \tag{4.8}$$

is solvable in a for all $x \in M_k$, where $L_f\Phi_k(x) = \frac{\partial \Phi_k}{\partial x} f(x)$ and $L_g\Phi_k(x) = \frac{\partial \Phi_k}{\partial x} g(x)$ are Lie derivatives. If it is solvable, then M_k fulfills the second requirement of the zero dynamics definition, and the algorithm converges with $M_k = M^*$. Otherwise, continue to Step 3.

Step 3: Calculate additional constraints If Equation 4.8 is not solvable in a for all $x \in M_k$, then the constraint function needs to be updated. This is done by calculating the set of all $x \in M_k$ that the equation is solvable for. This new set can be written as an additional constraint $\phi_k(x) = 0$, with updated constraint $\Phi_{k+1} = \begin{bmatrix} \Phi_k^T & \phi_k^T \end{bmatrix}^T$. At the end of Step 3, increment k and return to Step 1. ■

Steps 2 and 3 above may not be simple to calculate. In order to implement them, we have divided the problem into three cases (as a reminder, q is the number of inputs, and p is the number of outputs).

Case 1: $q = p$ For this case, we assume that all rows of Equation 4.8 are linearly independent for all $x \in M_k$. If there are linearly dependent rows, then zero dynamics could exist with fewer than $q = p$ inputs; see Case 2.

We can test whether Equation 4.8 is solvable in a for all $x \in M_k$ by looking at the rank of $L_g\Phi_k(x)$. If this matrix is full rank on M_k, then the equation is solvable. This completes Step 2.

In order to calculate additional constraints, there is a difference between single-input, single-output (SISO) and multi-input, multi-output (MIMO) systems. For SISO systems, this can be done manually without any special tricks. For MIMO systems, calculate a smooth matrix of functions $R_k(x)$ such that

$$R_k(x)L_g\Phi_k(x) = 0 \tag{4.9}$$

With such a function $R_k(x)$, Equation 4.8 is solvable in a if and only if x satisfies

$$R_k(x)L_f\Phi_k(x) = 0 \tag{4.10}$$

This becomes our additional constraint

$$\phi_k(x) = R_k(x)L_f\Phi_k(x) \tag{4.11}$$

In this algorithm, ϕ_{k-1} is contained in ϕ_k, so the updated constraint function is rewritten as $\Phi_{k+1} = \begin{bmatrix}\Phi_0^T & \phi_k^T\end{bmatrix}^T$. Note also that R_{k-1} is contained in R_k. This completes Step 3.

Case 2: $q < p$ This is the overconstrained case where there are fewer inputs than outputs. In order for there to be nontrivial zero dynamics, there must be rows of Equation 4.8 that are linearly dependent for all $x \in M_k$. As a result, this is the least likely case to have nontrivial zero dynamics.

A general strategy for approaching this system configuration is to use the equations from Case 1 to solve all combinations of inputs and outputs with equal numbers of each. This will result in multiple maximal output-zeroing submanifolds and multiple output-zeroing inputs. At the end, a solution exists on the intersection of the maximal output-zeroing submanifolds if there is a common output-zeroing input that solves Equation 4.8 for all the combinations without contradiction.

Case 3: $q > p$ This is the underconstrained case where there are more inputs than outputs. This means that there are more degrees of freedom than constraints, making this the most likely case to have nontrivial zero dynamics.

Similar to Case 2, a general strategy for approaching this system configuration is to use the equations from Case 1 to solve all combinations of inputs and outputs with equal numbers of each. For these combinations, all remaining inputs can be attacker prescribed. However in contrast to Case 2, here the attacker can choose any of the combinations.

4.2.2 Output-Zeroing Input

If the calculation of the maximal output-zeroing submanifold results in a nontrivial solution, then there is a corresponding output-zeroing input. This is solved for by solving Equation 4.8, which must have a solution because it is a requirement for the candidate submanifold to be locally controlled invariant. The result is the output-zeroing input $a^* = a^*(x)$ that corresponds to the maximal output-zeroing submanifold M^*.

4.2.3 Zero Dynamics

The final step is to calculate the resulting zero dynamics from the output-zeroing input. This is easily done by calculating the state trajectory starting at x_0 with dynamics calculated by

$$f^* = f(x) + g(x)a^*(x) \tag{4.12}$$

Once the zero dynamics are calculated, they can be checked for stability and other metrics.

4.2.4 Example Problem

For the pressurizer system, the dynamics are too complicated to work out on paper. Therefore to demonstrate the algorithm for calculating zero-dynamics, we go through a single

iteration of the following example system with two inputs, two outputs, and five states [26]

$$\begin{aligned} \dot{x} &= \overbrace{\begin{bmatrix} x_2 \\ x_4 \\ x_1 x_4 \\ x_5 \\ x_3 \end{bmatrix}}^{f(x)} + \overbrace{\begin{bmatrix} 1 & 0 \\ x_3 & x_2 \\ 0 & 1 \\ x_5 & x_2 \\ 1 & 1 \end{bmatrix}}^{g(x)} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \\ y &= \underbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}_{h(x)} \end{aligned} \tag{4.13}$$

Note that we are calculating the zero dynamics of the system rather than the zero-dynamics attacks; however, the algorithm is the same.

This system follows Case 1 from the algorithm with an equal number of inputs and outputs. Starting at Step 0, the initial constraint function is $\Phi_0 = h(x)$, and at Step 1, the candidate submanifold is

$$M_0 = \{x \in \mathbb{R}^5 : x_1 = x_2 = 0\} \tag{4.14}$$

Moving to Step 2, the Lie derivatives are

$$L_f \Phi_0(x) = \begin{bmatrix} x_2 \\ x_4 \end{bmatrix} \tag{4.15}$$

$$L_g \Phi_0(x) = \begin{bmatrix} 1 & 0 \\ x_3 & x_2 \end{bmatrix} \tag{4.16}$$

which do not pass the test for controlled invariance because $\text{rank}(L_g \Phi_0(x)) = 1$ for $x \in M_0$. This means Step 3 is needed. An $R_0(x)$ is calculated as

$$R_0(x) = \begin{bmatrix} -x_3 & 1 \end{bmatrix} \tag{4.17}$$

resulting in the additional constraint

$$\phi_0 = x_4 - x_2 x_3 \tag{4.18}$$

that gets concatenated to the initial constraint function. Returning to Step 1, the new candidate submanifold is

$$M_1 = \{x \in \mathbb{R}^5 : x_1 = x_2 = x_4 = 0\} \tag{4.19}$$

This same process continues until $\text{rank}(L_g\Phi_2(x)) = 2$ with the maximal output-zeroing submanifold, output-zeroing input, and zero dynamics calculated as

$$M^* = \{x \in \mathbb{R}^5 : x_1 = x_2 = x_4 = x_5 = 0\} \tag{4.20}$$

$$u^* = \begin{bmatrix} 0 \\ -x_3 \end{bmatrix} \tag{4.21}$$

$$f^* = \begin{bmatrix} 0 \\ 0 \\ -x_3 \\ 0 \\ 0 \end{bmatrix} \tag{4.22}$$

4.3 Attacks Targeting the Pressurizer

We implement the zero dynamics algorithm on the pressurizer model using all possible attack sets. These attack sets are the permutations of the maximal attack set, $S_{\max} = \{a_{y1}, a_{y2}, a_{u1}, a_{u2}\}$. Using this combinatorial approach, there are 15 attack sets to be analyzed, not including the null set.

Some of the combinations are trivial based on the form of the pressurizer output function. Including the attack, this is the linear function $y = h(x) = x - x_a$. This is used to calculate the initial candidate submanifold $M_0 = \{x \in U : h(x) = 0\}$. Without any output attack, x_a is nonexistent, resulting in the trivial solution $M_0 = \{0\}$. This means that any attack set that does not contain at least a_{y1} or a_{y2} is guaranteed to be trivial and can be ignored.

Of the remaining attack sets, some sets have identical output-zeroing submanifolds and inputs. These are not reported, resulting in eight attack sets with distinct zero dynamics.

Table 2: Summary of the zero-dynamics attacks targeting the pressure.

Target = Pressure

Set Number	Attack Set	Stability	Damage Time (hrs)
S_1	a_{y1}, a_{u1}	stable	284
S_2	a_{y1}, a_{u2}	asymptotically stable	∞
S_3	a_{y1}, a_{u1}, a_{u2}	unstable	84
S_4	a_{y1}, a_{y2}, a_{u1}	unstable	76
S_5	$a_{y1}, a_{y2}, a_{u1}, a_{u2}$	unstable	13

Table 3: Summary of the zero-dynamics attacks targeting the level.

Target = Level

Set Number	Attack Set	Stability	Damage Time (hrs)
S_6	a_{y2}	stable	387
S_7	a_{y2}, a_{u1}, a_{u2}	unstable	13
S_8	a_{y1}, a_{y2}, a_{u2}	unstable	12
S_5	$a_{y1}, a_{y2}, a_{u1}, a_{u2}$	unstable	11

These sets are discussed in terms of the system stability under attack and the time it takes to achieve some damaging outcome. The results, further defined below, are shown in Tables 2 and 3.

4.3.1 Stability

For the eight distinct attack sets, the zero dynamics could be categorized into asymptotically stable, stable, and unstable attack dynamics [27]:

1. *Asymptotically stable* means that the state moves from non-equilibrium points towards the equilibrium point. For these dynamics, the attacker cannot steer the state far from the equilibrium point.
2. *Stable* means that the state will always stay within some bounded region of the equilibrium point. For these dynamics, the attacker may be able to keep the state off of the equilibrium point, but they cannot steer the state unbounded.
3. *Unstable* means that the state can go unbounded away from the equilibrium point. For these dynamics, the attacker can move the state far from the equilibrium point.

Below, we discuss one example attack set for each of these stability types. In the resulting plots, the input and input-attack magnitudes are normalized. The heater output is normalized using the steady-state magnitude at nominal operating conditions, and the surge flow is normalized using the spray flow magnitude. In addition, the plotted state and output values are the true values rather than the difference between their steady-state values.

Asymptotically Stable There is one attack set targeting the pressure that results in asymptotically stable zero dynamics. This set is $S_2 = \{a_{y1}, a_{u2}\}$, with the maximal output-zeroing submanifold equal to $M^* = \{x \in U : x_2 = 0\}$. Based on this submanifold, we might expect that the attacker can steer x_1; however, this is not the case. In reality, this attack can only start at a non-equilibrium initial condition $x_0 \in M^*$, and the attacker can maintain zero output while the state slowly decays back to equilibrium conditions; see Figure 19 for a simulation of this attack. For this attack, the attacker may not even be able to get to the non-equilibrium initial condition, making this a poor attack strategy.

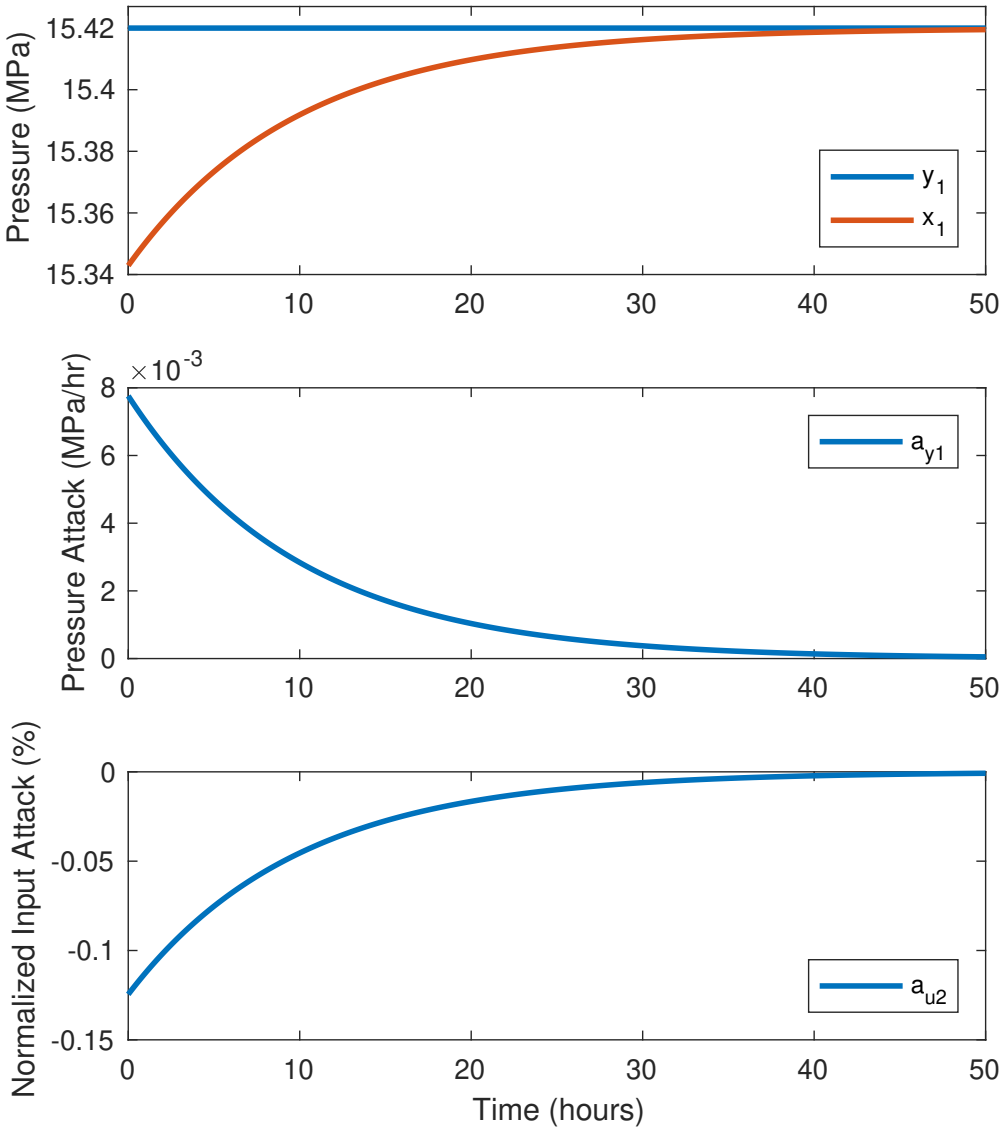


Figure 19: Simulation results of the asymptotically stable attack on the pressure. The top plot shows both the measured and true pressure. The measured pressure remains at the nominal value, while the true pressure returns from some nonzero initial condition to the the nominal value. The bottom two plots show the required attacker signals to achieve zero output.

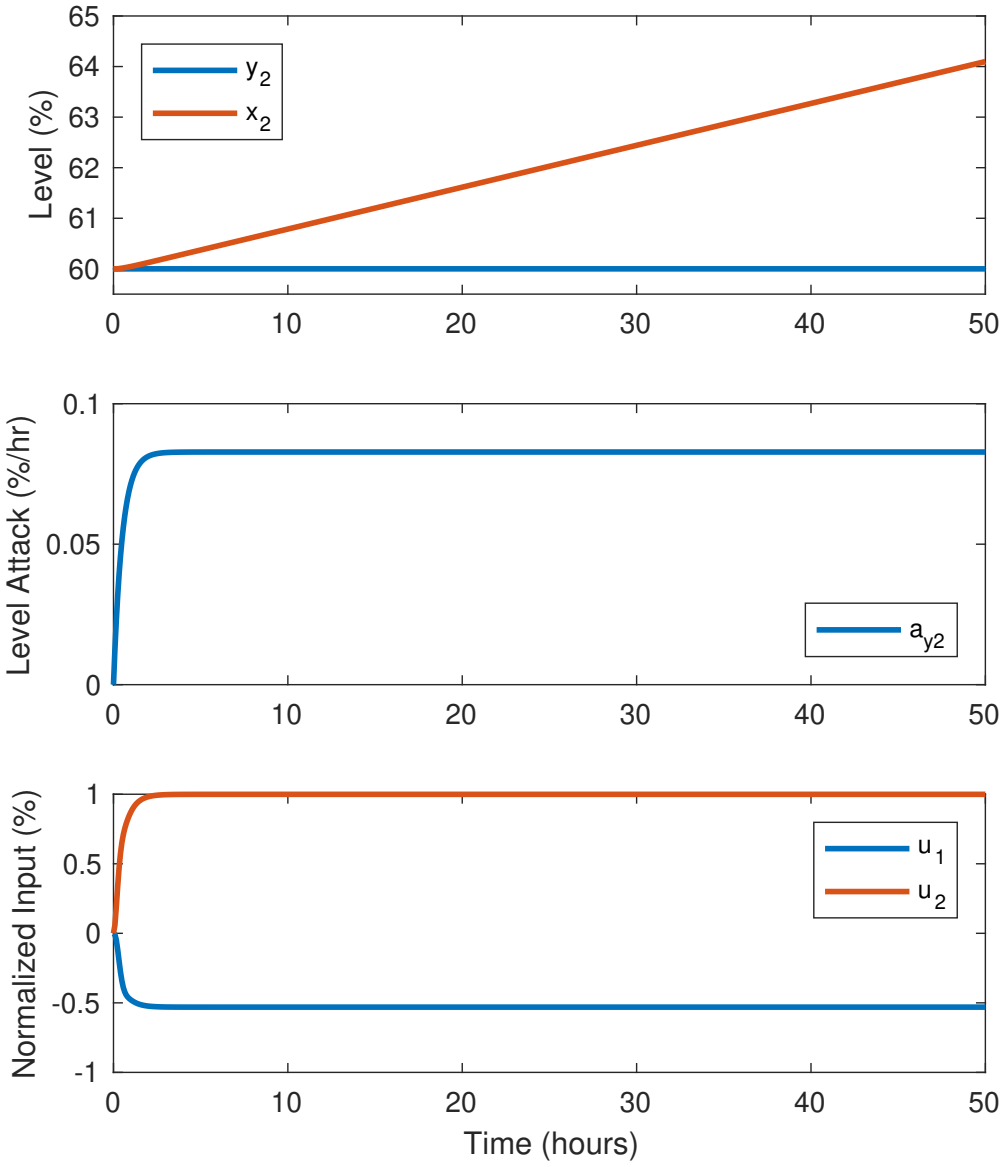


Figure 20: Simulation results of the stable attack on the level that is made unstable when the zero-output constraint is relaxed slightly. The top plot shows both the measured and true level. The measured level remains at the nominal value, while the true level increases uncontrolled. The middle plot shows the attacker signal to achieve zero measured level. The bottom plot shows the nonzero inputs that are required to maintain zero output, but are kept to less than 1 % on a normalized scale.

Stable There are two attack sets, one each targeting the pressure and level, that result in stable zero dynamics. This section discusses $S_6 = \{a_{y2}\}$, with the maximal output-zeroing submanifold equal to $M^* = \{x \in U : x_1 = 0\}$. Similar to the asymptotically stable case, this attack must start at some non-equilibrium initial condition $x_0 \in M^*$, but here the resulting zero dynamics are $\dot{x}_2^* = 0$. This means that the state does not decay back to the equilibrium condition. In other words, every state with $x_1 = 0$ is a stable zero dynamics attack, but the attacker has no way of moving through the possible attacked states while remaining on the output-zeroing submanifold.

This attack can become unstable if the zero dynamics constraints are relaxed. If the attacker inserts an impulse into a_{y2}, there will be a corresponding step change in y_2. This step will then decay to zero from the controller, which will use nonzero control signals. This attack can be extended by inserting a step into a_{y2}, which will, after a small transient, result in zero y_2, nonzero control signals, and unbounded x_2; see Figure 20 for a simulation of this attack.

Unstable The remaining attack sets, targeting a combination of pressure and level, all result in unstable zero dynamics. The primary reason these are unstable is that the attacker has access to more attack inputs than there are measurable outputs, which is the undercon-strained case. In each of these scenarios, the attacker can steer the state within the maximal output-zeroing submanifold. As an example, this section discusses $S_7 = \{a_{y2}, a_{u1}, a_{u2}\}$ with the maximal output-zeroing submanifold equal to $M^* = \{x \in U : x_1 = 0\}$. In this case, the attacker can start at the equilibrium point and move the pressure unbounded; see Figure 21 for a simulation of this attack. With no measurable changes, the attacker can quickly steer the pressure into dangerous territory.

4.3.2 Damage Time

In order to compare the attack sets, we calculate the time it would take to reach some undesirable system state, called the damage time. The undesirable states have been chosen as values that would normally result in an automatic plant shutdown. For pressure, this has been chosen as 12.86 MPa, where nominal pressure is 15.42 MPa; this is undesirable because

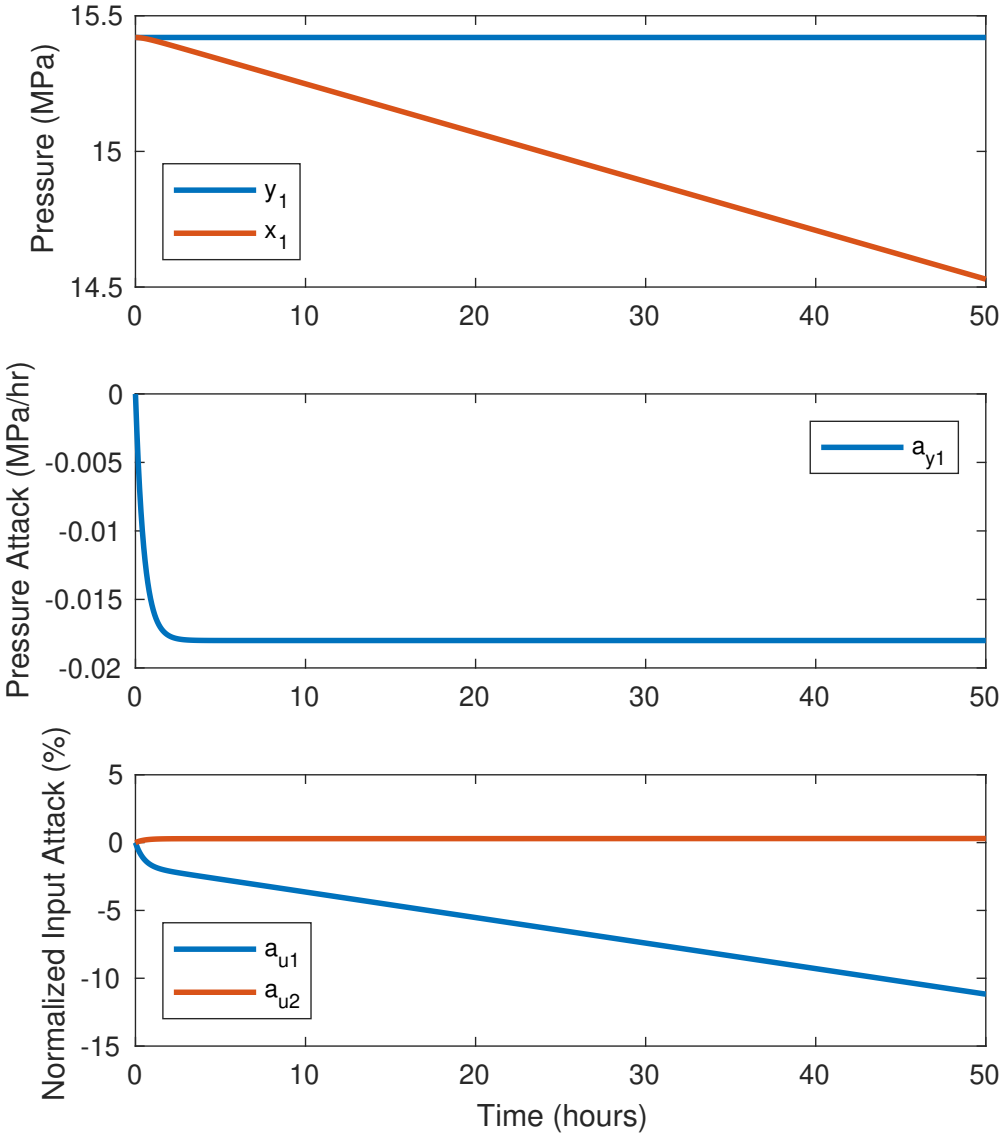


Figure 21: Simulation results of the unstable attack on the pressure. The top plot shows both the measured and true pressure. The measured pressure remains at the nominal value, while the true pressure decreases uncontrolled. The bottom two plots show the required attacker signals to achieve zero output.

low pressures could result in local boiling near the fuel source. For level, this has been chosen as 92 %, where nominal level is 60 %; this is undesirable because if the pressurizer fills completely with liquid, a large surge into the pressurizer could result in a pressure spike.

The damage times are calculated as constrained optimization problems. For the stable attack sets, the constraints are that the measurable inputs have to be below 1 % using the normalized scales. This low value has been selected because it could be mistaken for model uncertainty or sensor drift. For the unstable attack sets, the constraints are that the attack inputs have to be below 30 % using the normalized scales. This value has been limited because if these inputs get too large, they could either excite other dynamics or could saturate the inputs, both of which might be detectable.

These damage times can be used to compare each of the attacks targeting pressure and level; see Tables 2 and 3. Starting with the attacks targeting pressure, the slowest attack is the stable attack, which takes almost twelve days to reach the damage state. However, for a plant that runs at steady-state conditions for months continuously, an attack could continue undetected for that length of time. As the attacker gets more access, this time frame reduces to three days, and finally to half a day as they get access to everything.

Looking at the results for the attacks targeting level, the slowest attack is again the stable attack, which takes roughly sixteen days to reach the damage state. Once the attacker gets access to three or more inputs, this time reduces to roughly half a day, regardless of which inputs they have access to.

Based on the damage times, it is clear that the the unstable attacks are the most dangerous attacks targeting the pressurizer. However, they also might be the least likely attacks due to the increased number of signals required to implement them. By contrast, the stable attacks, particularly S_6 targeting the level, require fewer attack inputs and can still reach the same damage states. This makes them in theory easier to implement and therefore a potentially bigger threat to actual plants.

4.4 Local Stability of Output-Zeroing Submanifold

We previously discussed stability of the equilibrium point, which determines whether the zero-dynamics attack could steer the state away from the nominal equilibrium point. However, it is also important to determine the stability of the output-zeroing submanifold. Stability of the output-zeroing submanifold provides information on what happens if a successful attack is perturbed slightly off the output-zeroing submanifold. This scenario is essentially guaranteed for systems with noise and disturbances.

Similar to the previous stability discussion, the output-zeroing submanifold can be any of asymptotically stable, stable, or unstable. And, these types have similar meanings to those previously defined, except that they refer to the entire output-zeroing submanifold. But while an attacker wants the nominal equilibrium point to be unstable so they can damage the system, they want the output-zeroing submanifold to be asymptotically stable so that the attack returns to the output-zeroing submanifold in the presence of disturbances. If it is just stable or unstable, the state would remain off the output-zeroing submanifold, making it easy to detect.

This problem can be looked at directly using the nonlinear dynamics or using linearized dynamics. We briefly discuss the approach that uses the nonlinear dynamics directly, but also discuss why this approach is particularly difficult for this problem. Instead, we solve it using linearization methods.

4.4.1 Challenges of Implementing Nonlinear Lyapunov Methods

For nonlinear systems of the form $\dot{x} = f(x)$, the most rigorous method of proving stability of some equilibrium point $x_{eq} = 0$ is using Lyapunov methods. To show local asymptotic stability, the user must find a positive definite Lyapunov function $V(x) > 0$ whose derivative is negative definite $\dot{V}(x) < 0$. This function is often thought of as an energy function, where the energy dissipates along all trajectories towards the equilibrium point. Other types of stability can be shown by either adding or modifying these constraints.

One very common Lyapunov function form is to use the quadratic form $V(x) = \frac{1}{2}x^T Px$, where P is a positive definite symmetric matrix. For this form and for constant P, $\dot{V}(x) = f(x)^T Px$. The user then needs to find some positive-definite matrix P that guarantees this is negative definite over the state space. It is worth noting that such a matrix P may not exist.

The challenge with implementing this method for the current problem is that we have an invariant equilibrium set rather than an equilibrium point. Specifically, we want to demonstrate that the maximal output-zeroing submanifold is asymptotically stable, which means demonstrating that the controlled invariant submanifold of x that maps to $y = 0$ is asymptotically stable. To do this using the above method, the Lyapunov function could be of the form $V(x) = \frac{1}{2}\Phi^*(x)^T P\Phi^*(x)$, where $\Phi^*(x)$ is the final constraint function from the algorithm above; however, showing that $\dot{V}(x)$ is negative everywhere except M^* is significantly more challenging than showing it for a single equilibrium point.

4.4.2 Linearized Output Stability

Rather than implementing nonlinear Lyapunov methods, we instead linearize the system. Linearization is done about an equilibrium point and enables us to look at the stability of the output-zeroing submanifold in the vicinity of that equilibrium point. However, this approach only allows us to look at a single equilibrium point at a time, and we want to look at a set of equilibrium points. To get around this problem, we look at the stability around a finite set of equilibrium points and use matrix perturbation theory to approximate the stability over the set of points between the finite set.

In order to linearize the system, we start with the more general attacked system dynamics

$$\begin{aligned} \dot{x} &= f(x, a) \\ y &= h(x) \end{aligned} \tag{4.23}$$

with equilibrium point $(\overline{x}, \overline{a})$ defined by $f(\overline{x}, \overline{a}) = 0$. Near the equilibrium point, this system

can be approximated by a linear system of the form

$$\begin{aligned} \dot{x} &= Ax + Ba \\ y &= Cx \end{aligned} \tag{4.24}$$

The linear matrices can be derived using Taylor's series expansion. Starting with a change of variables

$$\delta_x = x - \overline{x} \tag{4.25}$$

$$\delta_a = a - \overline{a} \tag{4.26}$$

$$\delta_y = y - h(\overline{x}) \tag{4.27}$$

the dynamics can be rewritten

$$\begin{aligned} \dot{\delta}_x &= f(\overline{x} + \delta_x, \overline{a} + \delta_a) \\ \delta_y &= h(\overline{x} + \delta_x) - h(\overline{x}) \end{aligned} \tag{4.28}$$

The first-order Taylor's series approximation of these equations results in

$$\begin{aligned} \dot{\delta}_x &= f(\overline{x}, \overline{a}) + \left.\frac{\partial f}{\partial x}\right|_{(\overline{x},\overline{a})} \delta_x + \left.\frac{\partial f}{\partial a}\right|_{(\overline{x},\overline{a})} \delta_a \\ &= \left.\frac{\partial f}{\partial x}\right|_{(\overline{x},\overline{a})} \delta_x + \left.\frac{\partial f}{\partial a}\right|_{(\overline{x},\overline{a})} \delta_a \\ \delta_y &= \left.\frac{\partial h}{\partial x}\right|_{(\overline{x},\overline{a})} \delta_x \end{aligned} \tag{4.29}$$

With a slight abuse of notation, we continue to use x, a, and y, and the linearized matrices are defined as

$$A = \left.\frac{\partial f}{\partial x}\right|_{(\overline{x},\overline{a})} \tag{4.30}$$

$$B = \left.\frac{\partial f}{\partial a}\right|_{(\overline{x},\overline{a})} \tag{4.31}$$

$$C = \left.\frac{\partial h}{\partial x}\right|_{(\overline{x},\overline{a})} \tag{4.32}$$

For this stability analysis, there are a few structural components of the matrices that we take advantage of. First, the state matrix can be written as

$$Ax = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} \begin{bmatrix} x_s \\ x_a \\ x_c \end{bmatrix} \tag{4.33}$$

with $A_{21} = A_{22} = A_{23} = 0$ because they are for the attack state $\dot{x}_a = a_y$. Second, the output matrix can be written as

$$Cx = \begin{bmatrix} C_{11} & C_{12} & C_{13} \end{bmatrix} \begin{bmatrix} x_s \\ x_a \\ x_c \end{bmatrix} \tag{4.34}$$

with $C_{13} = 0$. In addition, to get a closed-form solution, it is helpful to limit the analysis to systems with equal numbers of outputs and system states x_s, resulting in full rank C_{11} matrices. This subset of systems includes the pressurizer system. This means that without loss of generality, $C_{11} = I$. If not, a suitable linear transformation can be found that achieves this.

Based on our assumption of a full rank C_{11} matrix, it is possible for this linear system to become affine under attack. This occurs for the the underconstrained case with more inputs than outputs. For this case, one or more of the inputs is user-prescribed. For constant prescribed values, this results in $\dot{x} = Ax + Ba + b$, where b is a constant vector. Since this dynamic equation is the more general case, it will be used for the remainder of the analysis.

In order to calculate the zero-dynamics for this linear system, we use the nonlinear algorithm of Section 4.2. Using the algorithm, systems that pass the controlled invariance test on the first iteration, which includes the pressurizer, result in an analytical solution for the stability analysis. Based on this assumption, the controlled invariance test can be written as

$$\dot{y} = CANx + CBa + Cb = 0 \tag{4.35}$$

where Nx equals x evaluated at the solution of $Cx = 0$. For example, if

$$C = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \tag{4.36}$$

then $Nx = \begin{bmatrix} x_3 & 0 & x_3 \end{bmatrix}^T$. It is important to note that for the relevant systems with $C_{11} = I$, Nx can be written such that it is only a function of the attacker states. This makes sense to do as the attacker is guaranteed to have access to these variables. To pass the controlled invariance test on the first iteration, CB must be full rank. For systems with this property, we can then calculate the output-zeroing input as

$$a^* = -(CB)^{-1}(CANx + Cb) \tag{4.37}$$

which results in the zero-dynamics

$$\dot{x} = (A - B(CB)^{-1}CAN)x + (I - B(CB)^{-1}C)b \tag{4.38}$$

The goal now is to analyze the local stability of the output-zeroing submanifold for the above zero-dynamics. This is accomplished by looking at a system with state variable z defined by

$$z = \begin{bmatrix} y \\ x_c \end{bmatrix} \tag{4.39}$$

$$\dot{z} = \begin{bmatrix} C\dot{x} \\ A_{31}y + A_{33}x_c \end{bmatrix} \tag{4.40}$$

$$= \begin{bmatrix} CA(I - N)x \\ A_{31}y + A_{33}x_c \end{bmatrix} \tag{4.41}$$

where $\dot{x_c}$ is the control law and can be simplified as above for $C_{11} = I$. This can be further simplified by noting that $(I - N)x = \begin{bmatrix} y^T & 0^T & x_c^T \end{bmatrix}^T$, where the 0 is a vector of appropriate size. Using the structural components of A and C noted above, this simplifies to

$$CA(I - N)x = \begin{bmatrix} A_{11} & A_{13} \end{bmatrix} z \tag{4.42}$$

As a result, the dynamics for our system z simplify to

$$\dot{z} = \begin{bmatrix} A_{11} & A_{13} \\ A_{31} & A_{33} \end{bmatrix} z \tag{4.43}$$

This system is identical to the original system without attacker states. In other words, the stability of the output-zeroing submanifold for the attacked system is the same as the stability of the equilibrium point for the unattacked system. Therefore, we can instead show stability for this unattacked system.

In order to analyze the stability of the unattacked system, we look at the eigenvalues of the matrix

$$A_z = \begin{bmatrix} A_{11} & A_{13} \\ A_{31} & A_{33} \end{bmatrix} \tag{4.44}$$

If all of the eigenvalues λ_i of the matrix A_z satisfy $\text{Real}(\lambda_i) < 0$, then the origin is asymptotically stable. This condition will also be written as the eigenvalues are in the left-half plane (LHP).

For the set of possible attacks, the nearest equilibrium point could be anywhere in the state space. Therefore, we want to look at the stability near all these points. In order to help do this, we introduce a theorem from matrix perturbation theory.

Theorem (Gershgorin Circle Theorem [28])**.** If $X^{-1}AX = D+F$ where $D = \text{diag}(d_1, \ldots, d_n)$ is a diagonal matrix, and F has zeros on the diagonal, then

$$\lambda(A) \subset \bigcup_{i=1}^{n} D_i \tag{4.45}$$

where $D_i = \{z \in \mathbb{C} : |z - d_i| \leq \sum_{j=1}^{n} |f_{ij}|\}$.

This theorem provides a way to bound the eigenvalues as a matrix is perturbed from some nominal matrix.

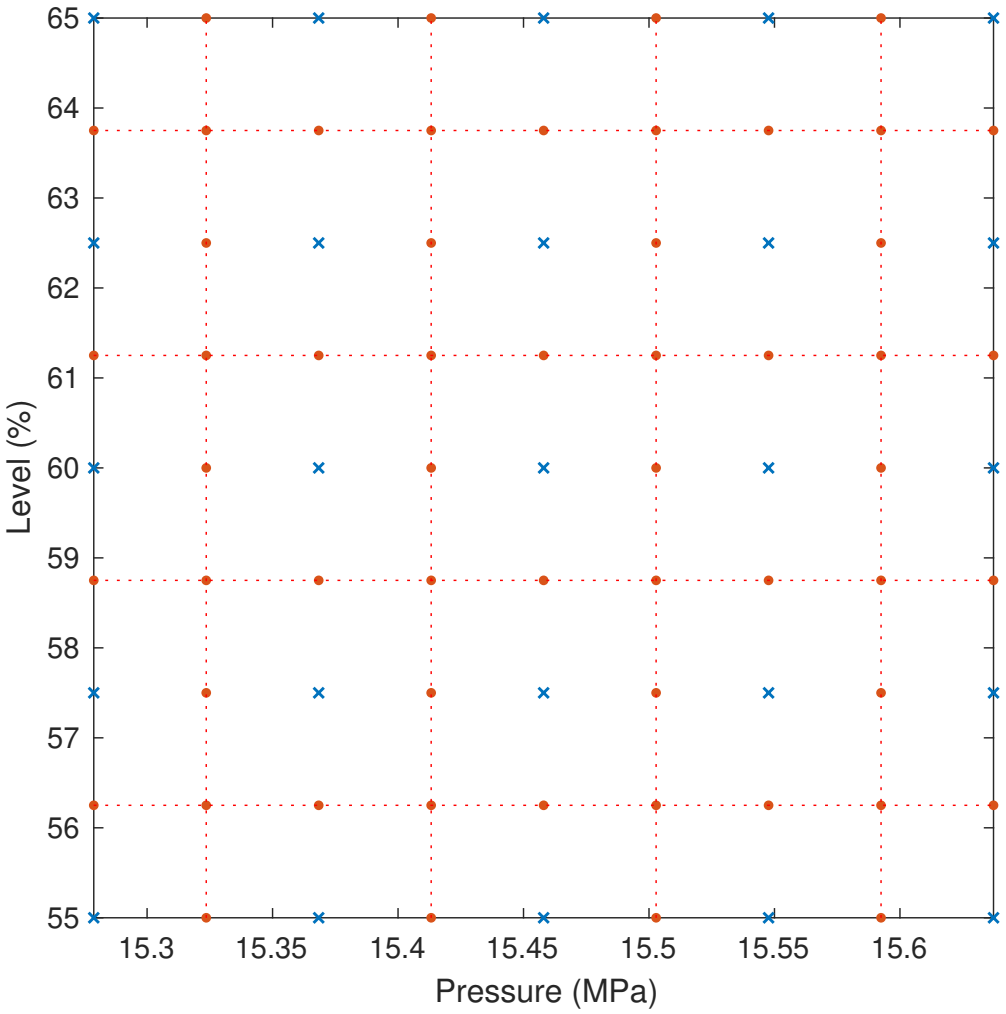


Figure 22: The discretization approach used for the stability analysis. The blue exes are the nominal points, and the red dots are the perturbed points. These are used in conjunction with the Gershgorin Circle Theorem.

In order to use this theorem, we use the following procedure:

1. Discretize the state space into a finite number of rectangles. The centers of the points are called nominal points and the edges are called perturbed points. The perturbed points represent the furthest the matrix can be perturbed from the nominal point within a given rectangle. An example can be seen in Figure 22, where the blue exes are the nominal points and the red dots are the perturbed points.
2. Calculate the eigenvalues and eigenvectors at each of the nominal points. If the nominal eigenvalues are stable, continue.
3. Calculate bounds for the eigenvalues within the rectangle. At the nominal points, the A matrix can be diagonalized using the transformation $X^{-1}A_{\text{nom}}X$, where X is the eigenvector matrix. For systems with non-repeating eigenvalues, this will result in a diagonal matrix D with the eigenvalues along the diagonal. For each of the perturbed points, this same transformation will result in $X^{-1}AX = D + F$, with D and F defined in the Gershgorin circle theorem. The theorem is then used directly to calculate circles that contain the perturbed eigenvalues.
4. Check that all eigenvalues in the circles are in the LHP. If so, continue to the next nominal point. Otherwise, make the discretization finer.

Using this procedure, we can look at the eigenvalues of the nominal points and the Gershgorin circles produced by the perturbed points. The eigenvalues of the nominal points are shown in Figure 23. From this plot, it is clear that the nominal eigenvalues are all in the LHP. The Gershgorin circles are shown in Figure 24 for most of the perturbed points, all of which are in the LHP. However, for some values of the state x, there are repeated eigenvalues, which means those A matrices are not diagonizable. The discrete points selected get close enough to these repeated eigenvalues that they maintain large F matrices after the eigenvector transformation. Figure 25 shows the Gershgorin circles for all points, where some of the circles are significantly larger than others and do cross into the right-half plane (RHP). Note that this does not mean they are unstable, but rather the method does not work well near these points.

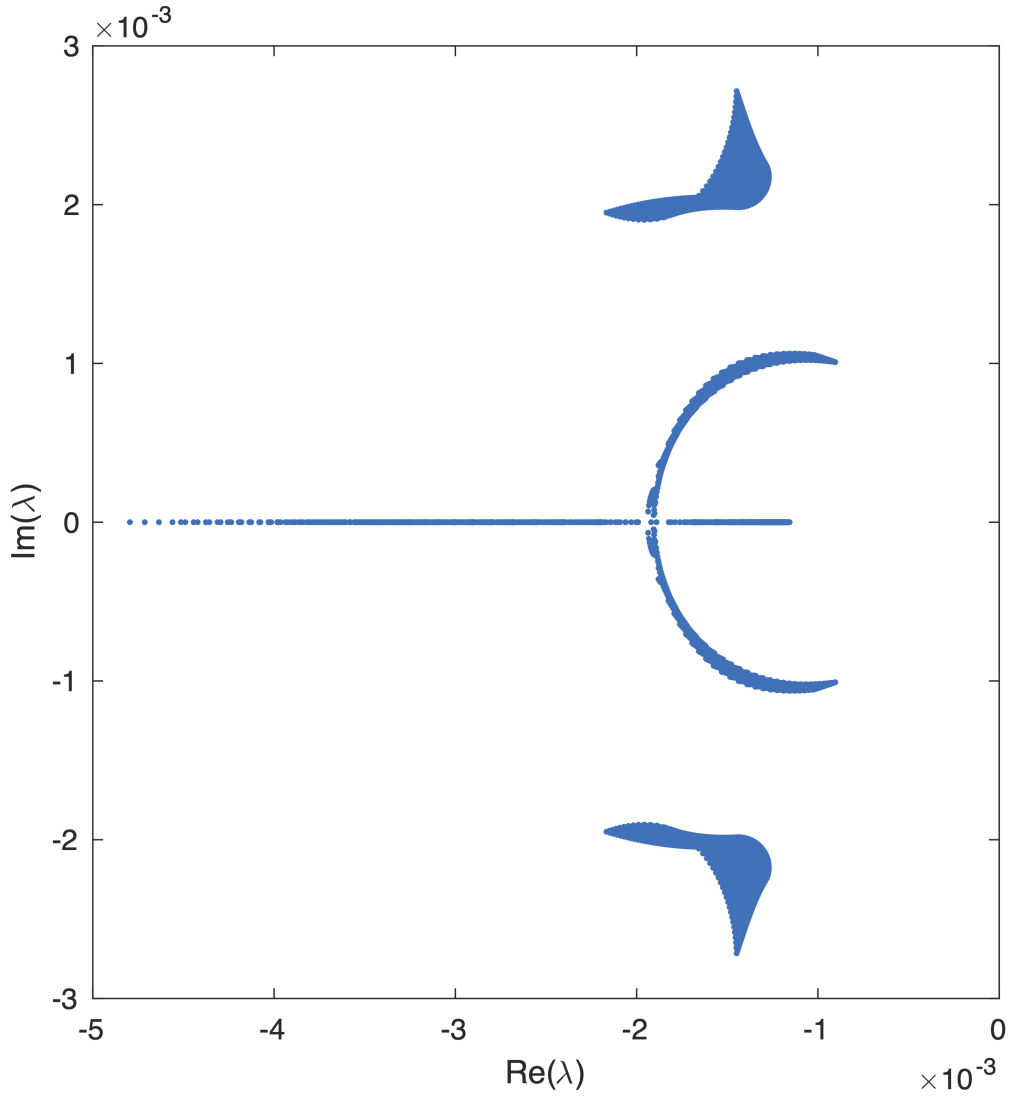


Figure 23: Plot of the real and imaginary portions of the eigenvalues at the nominal points. All these eigenvalues are in the LHP.

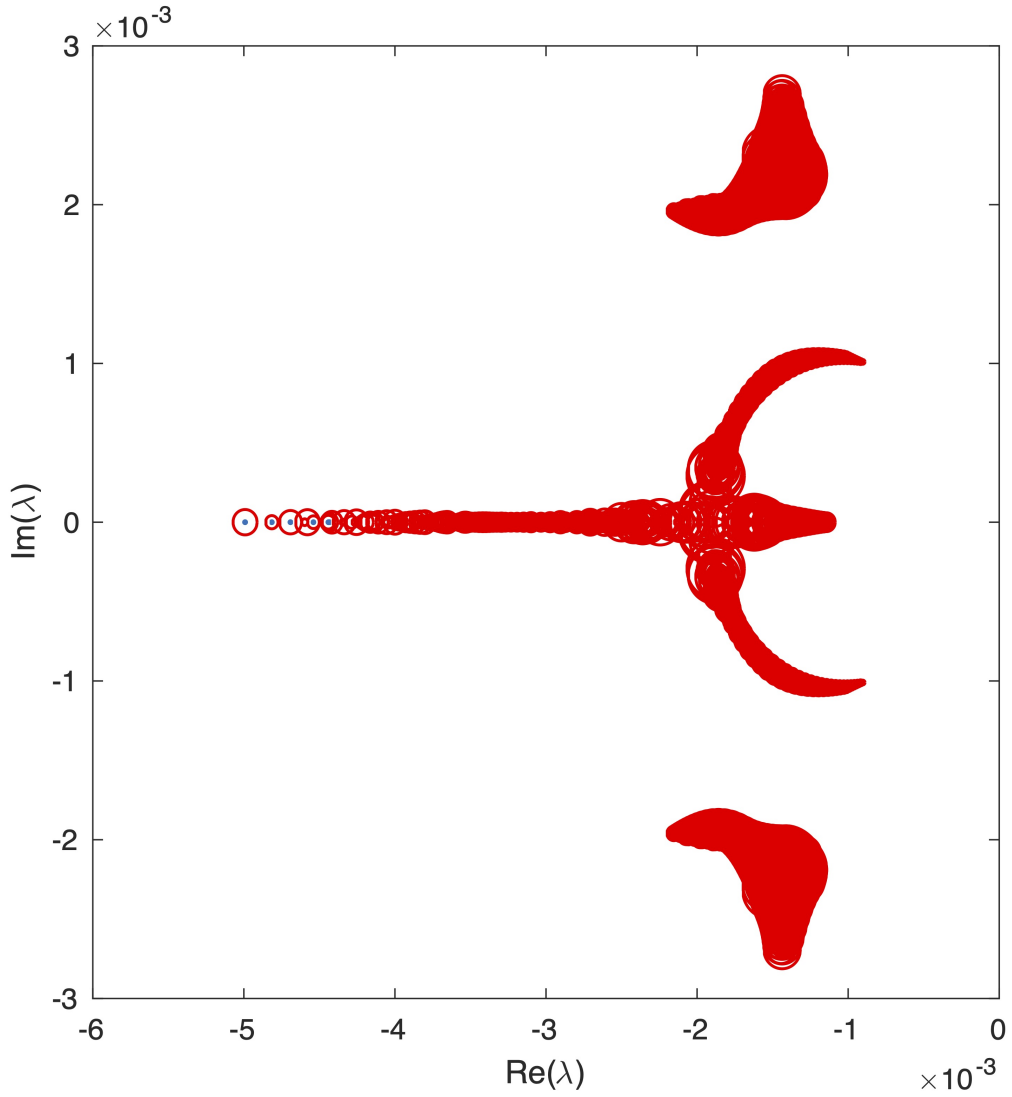


Figure 24: Plot of the real and imaginary portions of the Gershgorin circles at the perturbed points, but hides some larger circles. All of these circles are in the LHP.

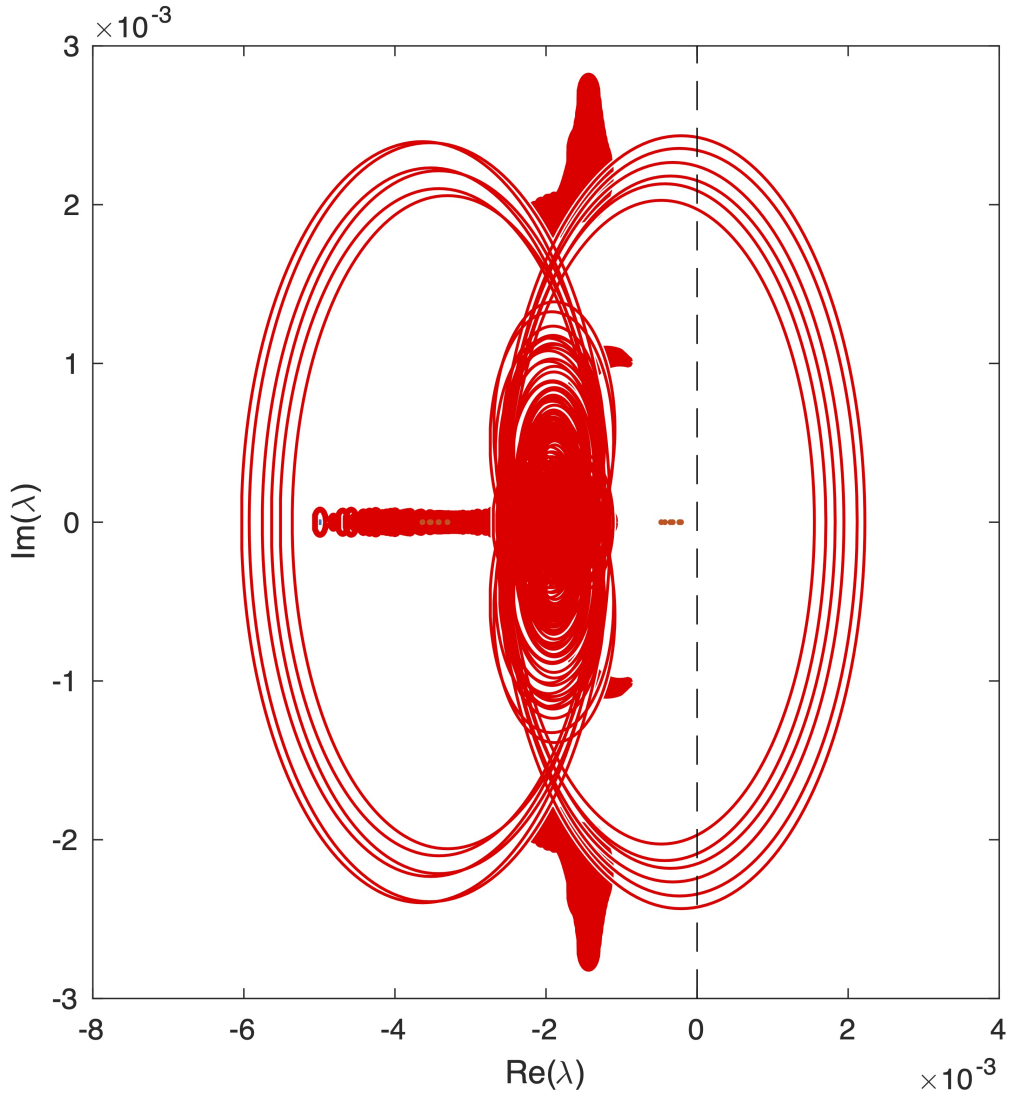


Figure 25: Plot of the real and imaginary portions of the Gershgorin circles at the perturbed points, including the larger circles. Some of the circles cross into the RHP.

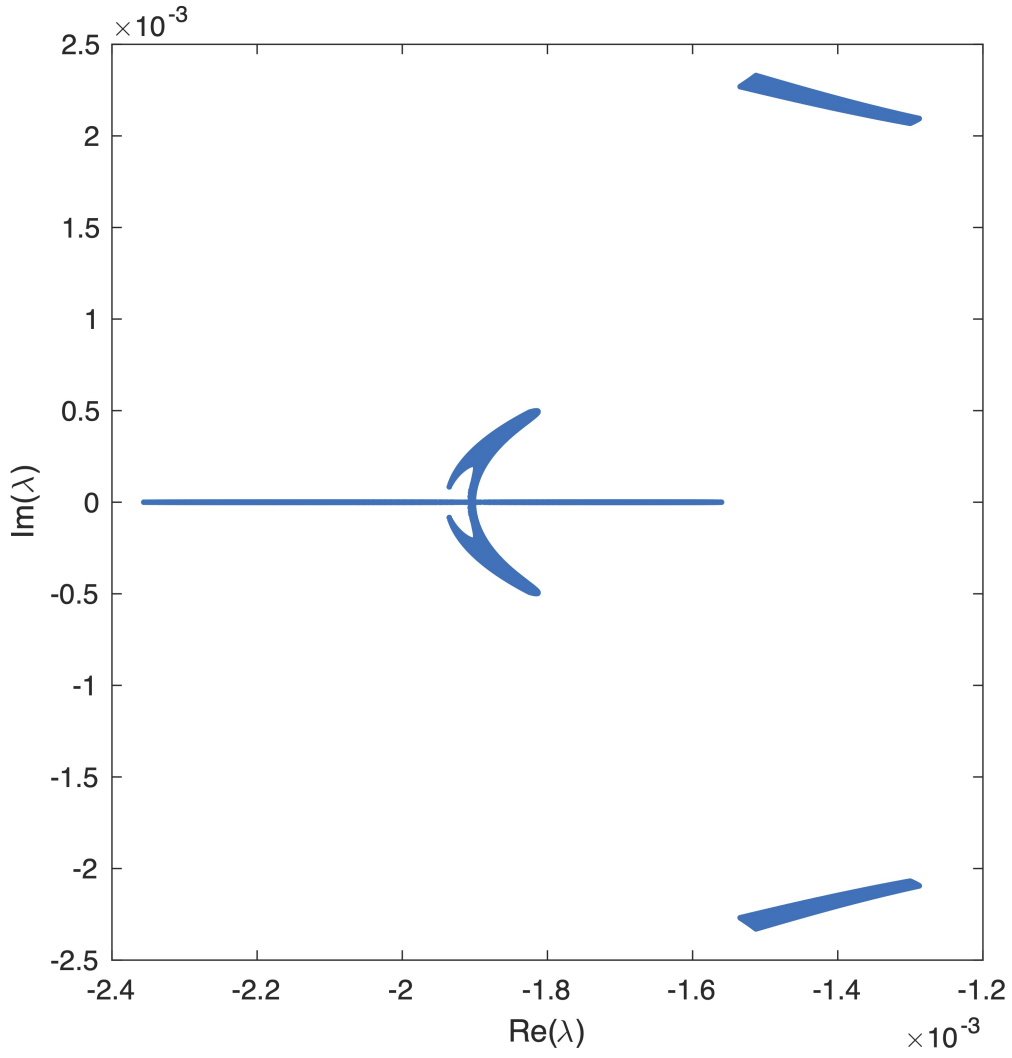


Figure 26: Plot of the real and imaginary portions of the eigenvalues at the nominal points evaluated at a finer discretization near the troublesome points. None of the points show evidence of crossing into the RHP.

For these points near repeated eigenvalues, we can look at the eigenvalues of the nominal points using a much finer discretization. This enables us to look for any evidence that they move into the RHP. The eigenvalues for this finer discretization are shown in Figure 26. Similar to the original nominal eigenvalues plot, this plot shows no signs that the eigenvalues are entering the RHP. Therefore, we conclude that the eigenvalues are most likely in the LHP for all equilibrium points. This concludes the stability analysis.

These results tell us that the output-zeroing submanifold is locally asymptotically stable. The results do not guarantee that if the state gets sufficiently far from the output-zeroing submanifold, then it will still return to the submanifold. This is due to the limitations of using linearizations. However, we expect that this is the case due to the dynamics and the properties of the controller designs. From the attacker's perspective then, this locally asymptotically stable property means that these attacks can remain stealthy in the presence of disturbances or noise.

4.5 Chapter Summary

As a reminder, our research objective relating to this chapter is to characterize stealthy cyber vulnerabilities targeting nuclear power plants. In this chapter, we accomplish the following:

- the state-space model is transformed from a control-affine model into an attack-affine model to be used for zero-dynamics analysis;
- zero-dynamics are more formally defined and the nonlinear zero-dynamics algorithm are presented;
- the algorithm is implemented on the nonlinear pressurizer system dynamics, resulting in eight unique attacks and seven that could result in system damage; and
- the output-zeroing submanifold is analyzed using local stability methods to show that the attacks targeting the pressurizer can remain stealthy.

This work completes the objective because we develop theoretical methods for characterizing

zero-dynamics attacks, which are a particularly dangerous class of zero-dynamics attacks. We then implement them on a critical subsystem of pressurizer water reactors to better understand the vulnerabilities associated with the pressurizer dynamics.

This concludes the analysis from the attacker's perspective on how an attacker could take advantage of system dynamics to attack a system. In the next chapter, we approach the cybersecurity problem from the defender's perspective and discuss how to detect zero-dynamics attacks targeting nonlinear systems.

5.0 Detecting Zero-Dynamics Attacks Targeting Nonlinear Systems

The second primary task in this research is to defend against the attacks developed in the previous chapter. The primary limitation of previous works relating to this task is that they assume the system under attack is observable and then detect attacks based on that assumption. We do not make such an assumption in this chapter. As a reminder, if this research were focused on linear systems, this would be impossible. However, it can be possible for nonlinear systems.

Our approach starts with the physics-based model and first sets up both the exact problem and an approximation. The exact problem has sufficiently many unknowns that it may be too difficult to solve. By making the approximation, the number of unknowns is reduced, making it more easily solved.

Using this approximation, we look at observability under zero-dynamics attack. One major difference between linear and nonlinear systems regarding observability is that nonlinear observability can depend on the control actions inserted into the system. For nonlinear systems under zero-dynamics attack, a non-zero control input is needed for observability. In other words, the defender must perturb the system in order to detect zero-dynamics attacks. Based on this, we develop methods to calculate an optimal input that will minimize sensitivity to noise. This is done using a quantitative measure of observability.

Once an optimal input is selected, we combine unscented Kalman filters with maximum likelihood estimation to detect and diagnose the attack. Unscented Kalman filters are state estimators that can be used for nonlinear stochastic systems. To accurately estimate the state, these filters require a reasonable initial estimate of the state; however, during a zero-dynamics attack, this is impossible. To overcome this, maximum likelihood estimation is used to select an optimal initial condition.

We test the approach on the pressurizer subsystem of an NPP. Multiple simulations are run with varying conditions and the error statistics are used to make decisions on the attack status.

Finally, we validate the approach using data from the commercial simulator. To do this, we combine the data-driven model with maximum likelihood estimation to estimate the state and detect the attack. This also involves a more detailed analysis of model accuracy, and correlates this accuracy with estimation accuracy.

This chapter is structured as follows:

- the exact and approximate problems are setup, and a test is presented to determine if the approximate problem is solvable;
- the optimization problem is presented to solve for the optimal input;
- the approach for estimating the state is presented;
- the decision rules for declaring an attack are discussed;
- the results are shown and discussed;
- the validation results are presented; and
- some discussions on the feasibility of implementing this approach offline and discussions on safety and economics are presented.

5.1 Problem Setup

This section introduces both the exact problem we want to solve and an approximation. The approximation makes the problem more tractable by constraining the unknown attacker inputs. In addition, this section discusses how to test whether the approximation is solvable for a given system, which is framed as a nonlinear observability problem.

5.1.1 Exact and Approximate Problems

We consider nonlinear dynamic systems under attack

$$\begin{aligned} \dot{x} &= f(x, u + a_u) \\ y &= h(x) + a_y \end{aligned} \tag{5.1}$$

where $x \in \mathbb{R}^n$ is the state, $u \in \mathbb{R}^q$ is the control input, $y \in \mathbb{R}^p$ is the measurement output, $a_u \in \mathbb{R}^q$ is the attacker input that modifies the control input, $a_y \in \mathbb{R}^p$ is the attacker input that modifies the measurement output, and $f(\cdot)$ and $h(\cdot)$ are nonlinear functions.

The objective of this work is to detect zero-dynamics attacks, which can be achieved by estimating the state. As a result, the exact problem is to estimate x using known input u and output y in the presence of unknown attacker inputs a_u and a_y.

In general, a defender has no knowledge of which attack inputs an attacker has access to, requiring the conservative assumption that an attacker could access all of them during an attack. Based on this assumption, an attacker would be able to implement any of several zero-dynamic attacks depending on which inputs they really have access to. In addition, they could change how the attack is implemented while trying to detect it, making the problem even more difficult. As such, it is beneficial to look for an approximation that removes the specific attack strategy from the problem.

The approximation selected uses the idea of an equilibrium state, defined as a state $\overline{x}$ with a corresponding input $\overline{u}$ such that $f(\overline{x}, \overline{u}) = 0$.

Assumption. *Under attack, the true state is always close to an equilibrium state and can be approximated as being at that equilibrium state.*

Based on this assumption, the approximate problem can be summarized as trying to estimate the nearest equilibrium state to the true state.

This approximation is advantageous because it reduces the number of unknowns by constraining the attacker inputs. Without loss of generality, we assume that $u = 0$ and $y = 0$ when not under attack. Then at equilibrium condition $(\overline{x}, \overline{u})$, a_u must equal the equilibrium input

$$a_u = \overline{u} \tag{5.2}$$

and a_y must satisfy the constraint

$$0 = h(\overline{x}) + a_y \tag{5.3}$$

These constraints can be directly substituted into (5.1).

The approximate system can be written as a perturbation from equilibrium conditions. The state and its derivative can be written

$$x = \overline{x} + \delta_x \tag{5.4}$$

$$\dot{x} = \dot{\delta}_x \tag{5.5}$$

where δ_x is the perturbation from the equilibrium state. Similarly, the total input can be written

$$u + a_u = \overline{u} + \delta_u \tag{5.6}$$

where $u = \delta_u$ is the perturbation from the equilibrium input. With these substitutions, the resulting system is

$$\begin{aligned} \dot{\delta}_x &= f(\overline{x} + \delta_x, \overline{u} + \delta_u) \\ y &= h(\overline{x} + \delta_x) - h(\overline{x}) \end{aligned} \tag{5.7}$$

Here, $\overline{u}$ is an implicit function of $\overline{x}$, so is not an additional unknown. Based on this system description, the approximate problem that we address in this work is to estimate $\overline{x}$ and δ_x using known input δ_u and output y.

5.1.2 Observability of the Approximate Problem

In the approximate problem, we want to estimate the unknown state δ_x and parameter vector $\overline{x}$ using the known input δ_u and output y. Before trying to accomplish this task, it is important to know whether the information contained in δ_u and y is sufficient to be able to estimate δ_x and $\overline{x}$. This question can be answered using the concept of observability; if a system is observable, it is possible to estimate the state using measurement data.

For nonlinear systems, observability can be a function of the control input. As such, we propose the following:

Proposition. *For a nonlinear system under zero-dynamics attack, a necessary condition for observability is some non-zero control input.*

Proof. The proof comes directly from the concept of zero dynamics. During a zero-dynamics attack, there is zero measurable output regardless of the true state. As a result, there is no information that can be used to distinguish between states. In order to get any measurable output that could possibly be used to distinguish between states, there must be some input that perturbs the system. □

We can then test a specific system to determine whether a non-zero input is sufficient to make the approximate problem solvable.

In order to apply the observability concept to our approximate problem, we recast the unknowns as an augmented state vector with augmented dynamics. The augmented state z is defined as the concatenation of the state and the unknown constant parameter vector

$$z = \begin{bmatrix} \delta_x \\ \overline{x} \end{bmatrix} \tag{5.8}$$

The augmented dynamics $\dot{z}$ are defined as the concatenation of the state dynamics and the null dynamics that describe constants

$$\dot{z} = \begin{bmatrix} \dot{\delta}_x \\ 0 \end{bmatrix} \tag{5.9}$$

With a slight abuse of notation, we reuse the function names f and h, but they now refer to the augmented system as a function of the augmented state and perturbation input

$$\begin{aligned} \dot{z} &= f(z, \delta_u) \\ y &= h(z) \end{aligned} \tag{5.10}$$

If this augmented system is observable, then the state and unknown parameter vector can be jointly estimated [29].

We now focus on how to test for observability. We want to determine whether the augmented system is locally observable, which can be more formally defined.

Definition. A system is *locally observable around a state* z_0 if within a neighborhood U of z_0 it is possible to determine z_0 from measurement data over a finite time. A system is *locally observable* if this is true for every $z_0 \in \mathbb{R}^n$ [30, 29].

Before introducing the test, the Lie derivative of h along f can be defined as

$$L_f h(z) = \frac{\partial h(z)}{\partial z} f(z, \delta_u) \tag{5.11}$$

and the i^{th} order Lie derivatives as

$$\begin{aligned} L_f^2 h(z) &= \frac{\partial L_f h(z)}{\partial z} f(z, \delta_u) \\ &\vdots \\ L_f^i h(z) &= \frac{\partial L_f^i h(z)}{\partial z} f(z, \delta_u) \end{aligned} \tag{5.12}$$

Using these derivatives, an observability matrix can be defined

$$\mathcal{O}(z, \delta_u) = \begin{bmatrix} \frac{\partial}{\partial z} h(z) \\ \frac{\partial}{\partial z} (L_f h(z)) \\ \frac{\partial}{\partial z} (L_f^2 h(z)) \\ \vdots \\ \frac{\partial}{\partial z} (L_f^{n-1} h(z)) \end{bmatrix} \tag{5.13}$$

Often, this is evaluated at some fixed value of the input δ_u^*. For time-varying inputs, the extended Lie derivative can be used [31], but is unnecessary for the system used in this work. A sufficient condition for local observability is: The system is locally observable around z_0 if $\text{rank}(\mathcal{O}(z_0, \delta_u^*)) = n$ and is locally observable if this is true for every $z_0 \in \mathbb{R}^n$ [29].

5.2 Solving for the Input

In order to make the system observable, the defender needs to insert a non-zero input into the system. It is important to note that this input will overwrite the controller for its duration. For ideal systems without noise, there are likely many small magnitude inputs that could result in accurate state estimates. However, for real systems with noise, small or poorly designed inputs will result in inaccurate estimates. This section discusses methods for determining an optimal input.

The primary objective in determining the input is to minimize the state estimation error. This can be achieved by optimizing a quantitative measure of observability rather than just the rank test used above. One such measure is the condition number with respect to inversion of the observability matrix [32, 33]. The condition number is a common metric for determining if a matrix is full rank. Here, we propose also using its numeric value in an optimization problem. Mathematically, it is defined for a generic matrix A as

$$\kappa(A) = \frac{\overline{\sigma}(A)}{\underline{\sigma}(A)} \tag{5.14}$$

where $\kappa(\cdot)$ is the condition number, and $\overline{\sigma}$ and $\underline{\sigma}$ are the largest and smallest singular values of A, respectively. The condition number bounds the error of a matrix inverse in the presence of a small perturbation in the matrix [34]. For our problem, that small perturbation is noise, so the condition number bounds the effects of noise. This makes it a good measure of estimation error.

The observability matrix is a matrix function of both the state and the input, so its condition number $\kappa(\mathcal{O}(z, \delta_u))$ is also a function of these two variables. The state is unknown a priori, so the input design should not use specific knowledge of it. Instead, it can be designed assuming the worst case of the state, resulting in an optimization problem to maximize the condition number over the set of states. Then, the input can be designed to minimize this upper bound on the condition number, resulting in a minmax optimization problem. The solution to this problem is a single optimal input value that can be inserted as a step input.

To fully define this input, the parameters that need to be solved for are the input vector and duration. The input vector is solved for using the minmax optimization problem discussed above

$$\arg\min_{\delta_u}\max_{z} \quad \kappa(\mathcal{O}(z,\delta_u)) \tag{5.15}$$

The duration is selected as a function of the state estimation error and the transient magnitude. The state estimation error statistics are determined numerically using repeated simulations with different noise realizations. These simulations also provide the transient magnitude. The final selection is a subjective weighting of these two parameters and is used later to make a decision rule about the occurrence of an attack.

5.3 Estimating the State

We estimate the state using unscented Kalman filters (UKFs), which use the system model along with sequences of input and measurement data to estimate the state for nonlinear stochastic systems. The challenge is that the equilibrium state has null dynamics $\dot{\bar{x}} = 0$, meaning the equilibrium state estimate will not change as the UKF gathers more measurement data. To account for this, we propose two approaches: we can give the equilibrium state random-walk dynamics, or we can use maximum likelihood estimation (MLE). We discuss both in this section and present results on both in Section 5.6, but ultimately focus on using MLE because it has performance advantages.

5.3.1 Unscented Kalman Filter

Our system can be rewritten as a discrete-time stochastic system with additive noise

$$\begin{aligned} z_{k+1} &= f(z_k, \delta_{u,k}) + w_k \\ y_k &= h(z_k) + v_k \end{aligned} \tag{5.16}$$

where w_k is process noise and v_k is measurement noise. Both noise sources are assumed to be Gaussian white processes described by $E[w_k] = 0$, $E[v_k] = 0$, $E[w_k w_l^T] = Q\delta_{kl}$, and $E[v_k v_l^T] = R\delta_{kl}$.

Using this system model, the goal of the UKF is to estimate a statistical distribution for the unknown state. For additive Gaussian process and measurement noise, the UKF calculates Gaussian state estimates described by $\mathcal{N}(\hat{z}_k, \hat{P}_k)$, where $\hat{z}_k$ and $\hat{P}_k$ are the mean and covariance of the state distribution, respectively.

This section follows [35, 36]. The UKF algorithm can be broken into prediction and correction steps:

Prediction The prediction step predicts the statistics of the state at the next time step. This step uses an approximation called the sigma-point transformation, which propagates the previous estimated distribution through the nonlinear system dynamics. The transformation uses a deterministic set of points, called sigma points, that captures the mean and covariance of the previous distribution. The transformation then propagates those points through the nonlinear system dynamics and uses the propagated points to approximate the predicted statistics.

The sigma points are defined by a set of vectors and weights. As a function of the previous distribution, the sigma vectors are defined for $i = 1, \ldots, n$ as

$$\mathcal{X}_{0,k-1} = \hat{z}_{k-1} \tag{5.17}$$

$$\mathcal{X}_{i,k-1} = \hat{z}_{k-1} + \left(\sqrt{(n+\lambda)\hat{P}_{k-1}}\right)_i \tag{5.18}$$

$$\mathcal{X}_{i+n,k-1} = \hat{z}_{k-1} - \left(\sqrt{(n+\lambda)\hat{P}_{k-1}}\right)_i \tag{5.19}$$

where n is the length of the state, $\lambda = \mu^2(n+\nu) - n$ is a scaling parameter, μ determines the spread of the sigma points, ν is a secondary scaling parameter usually equal to 0, and $\left(\sqrt{(n+\lambda)\hat{P}_{k-1}}\right)_i$ is the i^{th} row of the matrix square root. The weights are defined for $i = 1, \ldots, 2n$ as

$$W_0^m = \frac{\lambda}{n+\lambda} \tag{5.20}$$

$$W_0^c = \frac{\lambda}{n+\lambda} + (1 - \mu^2 + \beta) \tag{5.21}$$

$$W_i^m = W_i^c = \frac{1}{2(n+\lambda)} \tag{5.22}$$

where the m and c superscripts are used for calculating the transformed mean and covariance, respectively, and $\beta = 2$ incorporates prior knowledge of the known Gaussian distribution.

The sigma points and system model can then be used to calculate the predicted distributions. The predicted mean vectors for the state and output distributions are

$$\mathcal{X}_{i,k}^{-} = f(\mathcal{X}_{i,k-1}, \delta_{u,k-1}) \tag{5.23}$$

$$\hat{z}_k^{-} = \sum W_i^m \mathcal{X}_{i,k}^{-} \tag{5.24}$$

$$\mathcal{Y}_{i,k}^{-} = h(\mathcal{X}_{i,k}^{-}) \tag{5.25}$$

$$\hat{y}_k^{-} = \sum W_i^m \mathcal{Y}_{i,k}^{-} \tag{5.26}$$

where $\hat{z}_k^{-}$ is the predicted state estimate, $\hat{y}_k^{-}$ is the predicted output estimate, and summations are $\sum = \sum_{i=0}^{2n}$. The predicted covariance matrices are

$$\hat{P}_k^{-} = \sum W_i^c (\mathcal{X}_{i,k}^{-} - \hat{z}_k^{-})(\mathcal{X}_{i,k}^{-} - \hat{z}_k^{-})^T + Q \tag{5.27}$$

$$\hat{P}_{y_k y_k}^{-} = \sum W_i^c (\mathcal{Y}_{i,k}^{-} - \hat{y}_k^{-})(\mathcal{Y}_{i,k}^{-} - \hat{y}_k^{-})^T + R \tag{5.28}$$

$$\hat{P}_{z_k y_k}^{-} = \sum W_i^c (\mathcal{X}_{i,k}^{-} - \hat{z}_k^{-})(\mathcal{Y}_{i,k}^{-} - \hat{y}_k^{-})^T \tag{5.29}$$

where $\hat{P}_k^{-}$ is the predicted state-covariance estimate, $\hat{P}_{y_k y_k}^{-}$ is the predicted output-covariance estimate, and $\hat{P}_{z_k y_k}^{-}$ is the predicted cross-covariance estimate.

Correction The correction step corrects the predicted statistics based on the received measurement. The corrected state distribution is

$$\hat{z}_k = \hat{z}_k^{-} + K_k(y_k - \hat{y}_k^{-}) \tag{5.30}$$

$$\hat{P}_k = \hat{P}_k^{-} - K_k \hat{P}_{y_k y_k}^{-} K_k^T \tag{5.31}$$

$$K_k = \hat{P}_{z_k y_k}^{-} (\hat{P}_{y_k y_k}^{-})^{-1} \tag{5.32}$$

where $\hat{z}_k$ is the corrected state estimate, $\hat{P}_k$ is the corrected state-covariance estimate, and K_k is the Kalman gain and represents a weighting factor to calculate the corrected estimates.

5.3.2 Random-Walk Dynamics

To account for the null dynamics of the equilibrium state, a first approach is to give them random-walk dynamics. This can be expressed as

$$\overline{x}_{k+1} = \overline{x}_k + w_k \tag{5.33}$$

where w_k is the process noise. The noise term allows for the state to change at a rate that's related to the noise standard deviation. Or said another way, the larger the noise standard deviation, the faster the estimate can change, but also the more sensitive to other measurement and process noise it becomes.

This was the first approach that was tried in order to estimate the state, but it had some limitations. These limitations are discussed with the results in Section 5.6.

5.3.3 Maximum Likelihood Estimation

The second approach to account for the null dynamics of the equilibrium state is to use an initial state estimate $\hat{z}_0$ and keep the null dynamics. This creates a problem because during a zero-dynamics attack the initial state is unknown. A poor choice of the initial condition could result in a comparably poor estimate of the state. In order to address this, we use maximum likelihood estimation, which selects the initial condition that maximizes the total likelihood of the set of observations.

The UKF provides the likelihood of an individual measurement given past measurements through its predicted state distribution. The likelihood given both past measurements and an initial condition $p(y_k|Y_{k-1}, \hat{z}_0)$, with $Y_k = \{y_0, \ldots, y_k\}$, is normally distributed as $\mathcal{N}(\hat{y}_k^-, \hat{P}_{y_k y_k}^-|\hat{z}_0)$. Using these individual likelihood functions, the cumulative likelihood of the series of measurements Y_T is

$$p(Y_T|\hat{z}_0) = \prod_{k=0}^{T} p(y_k|Y_{k-1}, \hat{z}_0) \tag{5.34}$$

For numerical reasons, it is common to use the natural logarithm of the likelihood function. This creates a sum of log-likelihood values instead of a product, but still has the same maximum location. The resulting optimization problem is

$$\hat{z}_0^* = \arg\max_{\hat{z}_0} \quad \sum_{k=0}^{T} \log p(y_k | Y_{k-1}, \hat{z}_0) \tag{5.35}$$

where $\hat{z}_0^*$ is the optimal estimate of the initial augmented state.

Finally, we relate this augmented state in the variable space z back to the true system state in the variable space x. As a reminder, $z = \begin{bmatrix} \delta_x^T & \overline{x}^T \end{bmatrix}^T$ and $x = \overline{x} + \delta_x$. Therefore, we can calculate our optimal state estimate $\hat{x}_0^* = \overline{x}_0^* + \delta_{x,0}^*$, which is used for detecting and diagnosing attacks. Moving forward, this is simply called $\hat{x}_0$.

5.4 Detecting an Attack

After implementing the input and running the maximum likelihood estimation procedure, the final step is to make a decision about the presence of an attack. The challenge is that there is little prior knowledge about the attack that can be used. To counter this, we use a decision rule that is solely based on the state estimation statistics under nominal conditions. This type of decision making process is often referred to as one-class classification.

The decision is made based on the state estimation error statistics, where the state estimation error is defined as the difference between the actual and estimated initial states $\tilde{x} = \hat{x}_0 - x_0$. The general idea is to determine a mapping between the error and operating status $\mathcal{D} : \tilde{x} \to \{\text{normal},\ \text{attack}\}$. Then for the region $\mathcal{R}$ that maps to the normal status, the decision function is

$$\mathcal{D}(\tilde{x}) = \begin{cases} \text{normal}, & \text{if } \tilde{x} \in \mathcal{R} \\ \text{attack}, & \text{otherwise} \end{cases} \tag{5.36}$$

See Figure 27 for a generic nonlinear example. In our one-class classification problem, this boundary is determined purely by data from nominal conditions.

In this work, the state estimation error is approximately a multivariate Gaussian random variable. As such, we can create an analytical decision rule based on this distribution. It

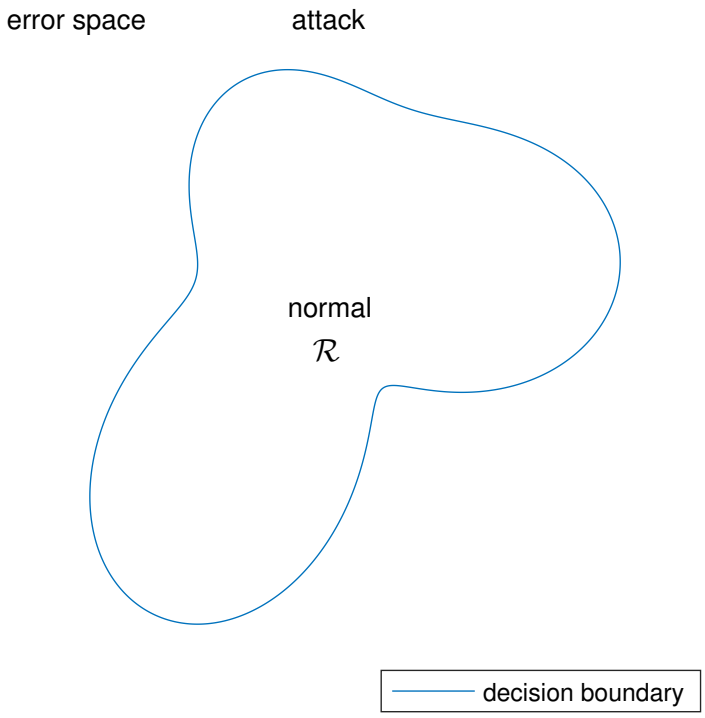


Figure 27: Example of a decision boundary in two-dimensional space. Any value that falls into the normal region is declared normal, and any value that falls outside the normal region is declared an attack.

makes sense to use a set of points for the boundary that are equally likely to occur; for the Gaussian distribution, this results in a hyper-ellipsoid contour. If the error did not follow a well-known distribution, various machine learning techniques could be used for a similar purpose.

There are an infinite number of hyper-ellipsoid contours, and the final one can be selected using the probability of seeing a false positive $P(\mathcal{D} = \text{attack} \mid \text{true} = \text{normal})$. Under normal conditions, we expect to see false positives at a rate equal to the total probability of values outside the normal region $\mathcal{R}$. Or mathematically, under normal conditions, $P(\tilde{x} \in \mathcal{R}) = 1-\alpha$, and $P(\tilde{x} \notin \mathcal{R}) = \alpha$. As such, we can calculate the corresponding boundary by selecting a value of α; see Figure 28 for a generic Gaussian example. The actual value of α is discussed in Section 5.6.

The rest of this section focuses on how to calculate the hyper-ellipsoid decision boundary as a function of α. The idea is to transform the multivariate random variable into a scalar random variable and the hyper-ellipsoid boundary region into a scalar decision threshold. Then, the scalar value can easily be compared to the threshold to make a decision. Additional background can be found in [37].

First, we need to estimate the unknown multivariate Gaussian statistics of the state estimation error. Note that this is different from the distribution calculated by the UKF. This is the distribution of estimating the initial condition as a function of the entire sequence of measurements. The mean is zero, but the covariance matrix is unknown. This is estimated using m simulations of the error, and the estimated covariance $\hat{\Sigma}$ is calculated as

$$\hat{\Sigma} = \frac{1}{m}\mathcal{E}^T\mathcal{E} \tag{5.37}$$

where $\mathcal{E} = \begin{bmatrix} \tilde{x}_1 & \tilde{x}_2 & \dots & \tilde{x}_m \end{bmatrix}^T$ is the $m \times n$ data matrix.

As mentioned, we now convert this multivariate random variable into a scalar random variable to more easily calculate the decision boundary. The weighted norm of a multivariate Gaussian random variable with estimated covariance is a T-squared random variable $T^2_{n,m}$

$$\tilde{x}^T\hat{\Sigma}^{-1}\tilde{x} \sim T^2_{n,m} \tag{5.38}$$

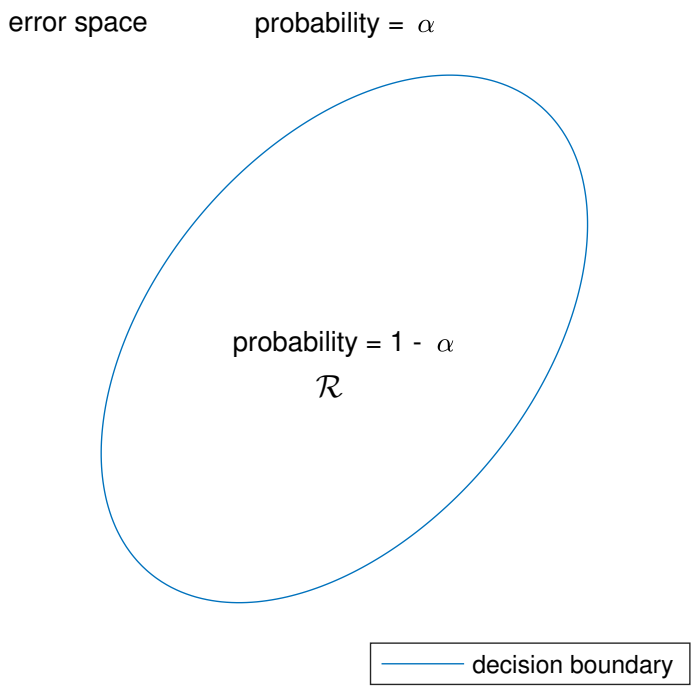


Figure 28: Example of a Gaussian decision boundary in two-dimensional space. Under nominal conditions, the probabilities of correctly declaring normal and falsely declaring attack are $1 - \alpha$ and α, respectively.

where the T^2 subscripts are distribution parameters. Using the T-squared distribution, the region $\mathcal{R}$ is defined as the set of all points such that the T-squared random variable is less than or equal to some threshold

$$\mathcal{R} = \{\tilde{x} : \tilde{x}^T \Sigma^{-1} \tilde{x} \leq T^*\} \tag{5.39}$$

where T^* is defined such that $p(T^2_{n,m} \leq T^*) = 1 - \alpha$. In addition, the boundary is the set above limited to those points that are strictly equal rather than less than or equal.

One challenge with calculating the above decision boundary is that the T-squared distribution is not often included in software libraries, so it is easier to convert the T-squared random variable into an F random variable. These two random variables are related through

$$T^2_{n,m} = \frac{mn}{m-n+1} F_{n,m-n+1} \tag{5.40}$$

and an F^* can be defined similarly such that $p(F_{n,m-n+1} \leq F^*) = 1 - \alpha$. This value can be calculated using a cumulative distribution function for the F distribution, which is included in many popular software packages. In addition, it is often tabulated for common values of α.

This decision process can be summarized through the following steps:

- **Step 1:** Calculate F^* as the value such that $p(F_{n,m-n+1} \leq F^*) = 1 - \alpha$.
- **Step 2:** Calculate $T^* = \frac{mn}{m-n+1} F^*$.
- **Step 3:** Determine if $\tilde{x} \in \mathcal{R}$ using (5.39).
- **Step 4:** Make a decision according to (5.36).

5.5 Additional Assumptions for the Pressurizer Model

As previously mentioned, the pressurizer dynamics are derived assuming the system is saturated. The primary advantage to this model, often called an equilibrium model, is that it is significantly simpler than a non-equilibrium model that allows for subcooled liquid or superheated steam. The primary disadvantage is that the model will prove inaccurate if the system strays from saturated conditions.

Table 4: Condition numbers for several input possibilities.

u_1(kW)	u_2(kg s^{-1})	Max κ	Min κ
-100	-5	1.30×10^5	4.88×10^4
0	-5	1.44×10^5	5.21×10^4
-100	0	1.54×10^{13}	8.92×10^5

In order to ensure model accuracy, we constrain our inputs to those that maintain the saturated system. The primary cause of non-equilibrium conditions is a large influx of colder water into the pressurizer from the surge line. As mentioned, there is a spray bypass line that continuously pushes mass into the pressurizer from a spray accumulator. To compensate for this flow in, there must be an equal flow out of the primary loop into the accumulator. Therefore, for there to be any influx of colder water into the pressurizer, the flow rate entering the primary loop plus any expansion upward from changing primary loop temperatures must be greater than the spray bypass flow rate. As a result, we can reduce the possibility of seeing non-equilibrium conditions by constraining the flow rate to either enter the primary loop at a rate below the spray bypass flow rate or leave the primary loop. For this work, we use the range $[-5\,\mathrm{kg\,s^{-1}}, 0.4\,\mathrm{kg\,s^{-1}}]$. We also constrained the heater input to $\pm 100\,\mathrm{kW}$ from its nominal value.

5.6 Results

For the pressurizer system model, we calculated the optimal input, the state estimation error statistics, and the detection rate using the decision rule.

The optimal input magnitude is the value that minimizes the maximum condition number over the state space. As might be expected, larger magnitude inputs generally result in lower condition numbers, making them more optimal. The input space for our problem is

asymmetrical, resulting in an optimal input of $u = \begin{bmatrix} -100\,\text{kW} & -5\,\text{kg}\,\text{s}^{-1} \end{bmatrix}^T$ with a maximum condition number of 1.30×10^5.

The input above is optimal with respect to the condition number, but it is also worth considering the transient produced. Looking at some other inputs, the maximum condition number is much more sensitive to the surge flow than to the heater; see Table 4. When the heater is removed from the input such that $u = \begin{bmatrix} 0\,\text{kW} & -5\,\text{kg}\,\text{s}^{-1} \end{bmatrix}^T$, the maximum condition number only increases by 10 %, but also reduces the pressure transient. By contrast, when the surge flow is removed from the input such that $u = \begin{bmatrix} -100\,\text{kW} & 0\,\text{kg}\,\text{s}^{-1} \end{bmatrix}^T$, the maximum condition number increases by eight orders of magnitude. As a result, we use $u = \begin{bmatrix} 0\,\text{kW} & -5\,\text{kg}\,\text{s}^{-1} \end{bmatrix}^T$ because it should achieve similar results to the optimal input, while reducing the transient magnitude.

In addition to the maximum condition number, it is worthwhile to look at the minimum condition number for the three inputs mentioned above; these are also shown in Table 4. For the two inputs that include the surge flow, the minimum and maximum condition numbers are less than one order of magnitude apart, and for the input that only includes the heater, the two condition numbers are over seven orders of magnitude apart. This suggests that if the heater-only input is used, it could distinguish some states but possibly not others, and emphasizes the importance of looking for the maximum value over the state space.

Before getting into the statistics of the state estimation error, it is worth looking at the results without any noise. This represents the nominal case and provides a look at whether our approximation is reasonable. We provide one example with the attacker targeting the level; see Figure 29. In this figure, the left two plots show the measured, true, and estimated parameters, and the right two plots show the estimation error. From these plots, the nominal error is extremely small, less than 0.005 MPa and 0.1 % for the pressure and level, respectively. This accuracy is typical of other attack simulations as well. This suggests the approximation is reasonable for this system.

Here, we also compare the two approaches to accounting for the null dynamics of the equilibrium state. As a reminder, the two approaches are assigning random-walk dynamics to the equilibrium state, or using maximum likelihood estimation to estimate the initial state; see Figure 30 for the results from a simulation. This example shows two advantages to the

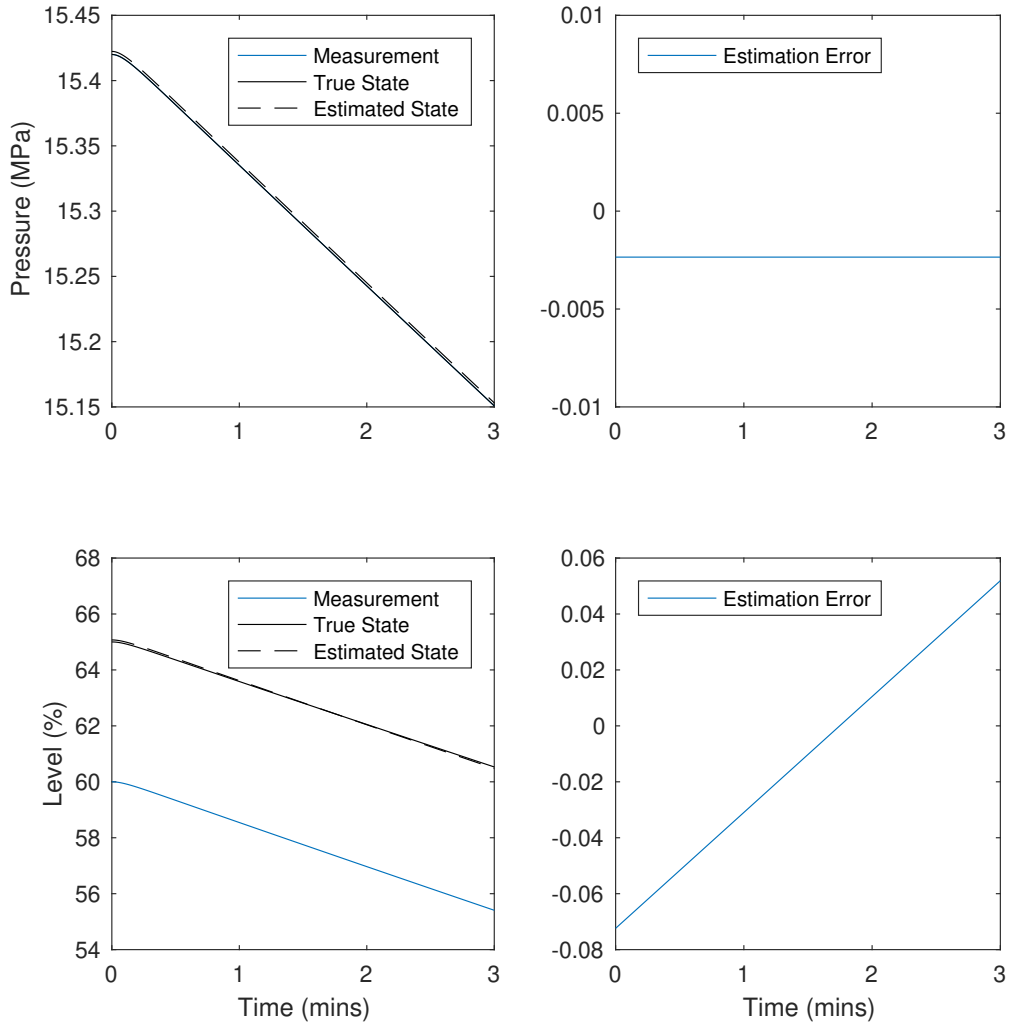


Figure 29: Results of the state estimation process on an attack targeting the level. This simulation does not include noise. The top plots refer to the pressure, and the bottom plots refer to the level. The right plots show the estimation error, both of which are small.

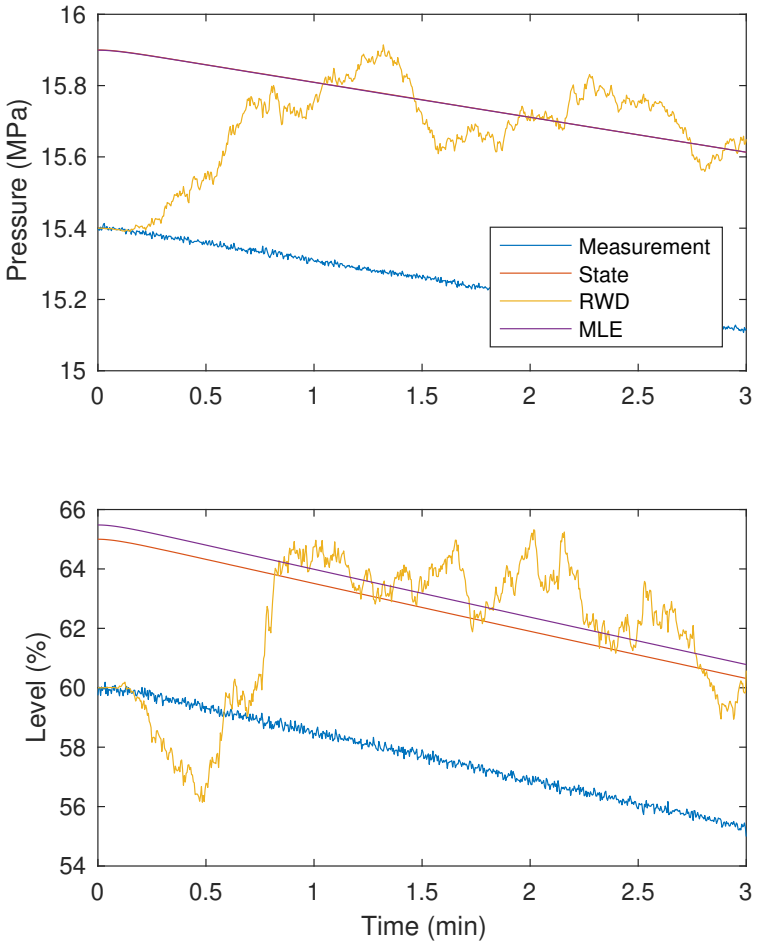


Figure 30: Plots of the two approaches to accounting for the null dynamics of the equilibrium state. The first, labeled RWD, is for the random-walk dynamics, and the second, labeled MLE, is for the maximum likelihood estimation. Both methods are able to estimate the state, but the MLE has advantages.

maximum likelihood approach: (i) it is generally more accurate, and (ii) it is not a function of stopping time. For this second advantage, note that the state estimate using the random-walk dynamics changes dramatically as a function of the time the algorithm is stopped, whereas this is not the case for the maximum likelihood approach. These advantages are why maximum likelihood estimation is used as the primary estimation approach.

The surge-flow input is used to estimate the statistics of the state estimation error for varying input durations. This is done by running 100 simulations at nominal conditions and calculating sample standard deviations; see Figure 31 for the results as a function of the duration. In this section, the error is normalized such that 100 % is equal to the control system alarm values of 100 psi and 10 % for pressure and level, respectively. The final selection, which is a subjective compromise between estimation error and transient magnitude, is 120 s.

With these statistics calculated, a suitable α is required to finalize the decision rule. For this application, we select a more conservative α that results in fewer false positives because as the true state gets further from the nominal condition, it becomes easier to detect the attack. Therefore, the value α predominantly determines whether we can differentiate nominal conditions from the early stages of an attack. From our previous work detailing zero-dynamics attacks targeting the pressurizer [17], we expect an attacker could reach the alarm values in approximately four hours. If we then implement the detection mechanism every two hours, we would have at least two attempts to detect the attack before reaching the alarm values. Using a rate of once per two hours, we can correlate α to the expected number of false alarms per year.

Finally to select α and look at overall detection performance, we test the decision rule and detection routine frequency using simulations with varying values of α. If an attacker can reach the alarm values in four hours, this equals a maximum normalized attack rate of $25\,\%\,\mathrm{h}^{-1}$. In order to simulate the detection routine and assuming it is implemented every two hours, the routine is first implemented at a random initial attack state between 0 % to 50 %. If the decision rule detects the attack, then the simulation stops. Otherwise, the detection routine is implemented at the equivalent of two hours later by adding 50 % to the original initial condition. This process is continued until the attack is successfully detected,

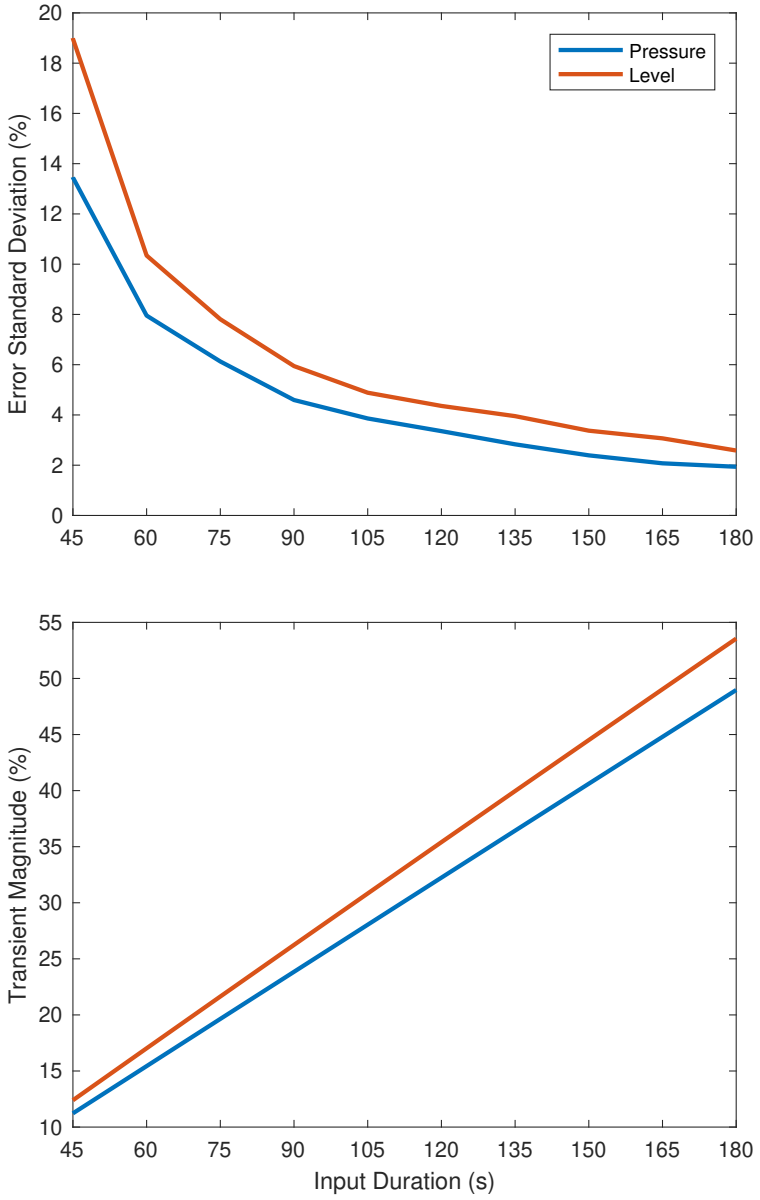


Figure 31: Plots of the standard deviation of the estimation error and the transient magnitude versus the input duration. As the input duration increases, the error decreases, but the transient magnitude increases.

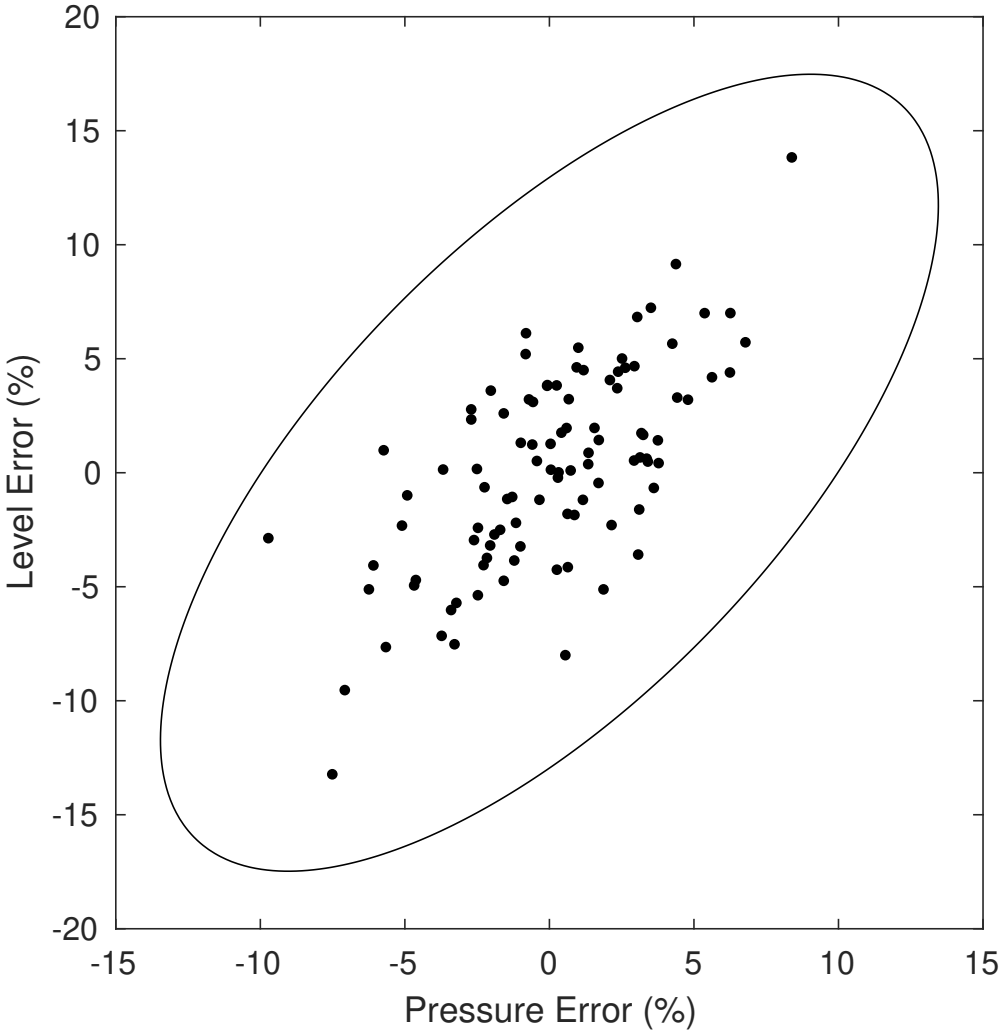


Figure 32: Plot of the decision region and the data from the simulations for the physics-based model. The black dots are each estimates for a different noise simulation, and the boundary is the decision region. Note these are in units of percentage of alarm values.

and is done for multiple different α values. Over 100 simulations, our detection algorithm caught 74 % on the first attempt and the remaining 26 % on the second attempt with an α of one false alarm per month, and 70 % on the first attempt and the remaining 30 % on the second attempt with an α anywhere from one false alarm per year to one per century. This means we detected 100 % of the attacks before the attacker reached the alert state, regardless of α. As such, we use a value of once per century. Our true positive detection rate was 100 % and our theoretical false positive rate is once per 100 years. Given how rare we expect to see false positives, it would be hard to verify using simulations, so we rely on the theoretical number.

5.7 Validating Using Simulator Data

The previous sections of this chapter use the physics-based model for both generating data and detecting attacks. Here, we validate the detection approach using simulator data and the data-driven model developed in Chapter 3. In this section, we: (1) discuss a change to the estimation algorithm that results in better estimates, (2) present an analysis that correlates output estimation accuracy with state estimation accuracy, and (3) show the results of the validation.

5.7.1 Estimating the State

For the theoretical work above, we both simulate the dynamics and estimate the state using the same physics-based model. This results in a detection algorithm that has perfect model knowledge, meaning that the UKF provides nearly optimal results. However, when validating the approach using simulator data, our model is not perfect. This led to changes in how we estimate the state for this validation process.

The UKF can only provide accurate estimates if the noise assumptions are reasonable approximations. For our data-driven model, the Gaussian process noise assumption is not a

good approximation, and the UKF did not provide accurate estimates. Instead, we directly implemented maximum likelihood estimation to estimate the unknown initial state.

The estimates can be calculated in a similar manner to the system identification techniques. Similar to the UKF, these are calculated in the z space. The estimated measurements are calculated as

$$\begin{aligned} \hat{z}_{k+1} &= A\hat{z}_k + Bu_k + E\hat{\zeta}_k \\ \hat{y}_k &= C\hat{z}_k \end{aligned} \tag{5.41}$$

where, as a reminder, ζ_k is a vector of monomials made from the components of z_k and u_k, and the output estimation error $\tilde{y}_k = \hat{y}_k(z_0) - y_k$ is a function of the unknown z_0.

Similar to the system identification case, the maximum likelihood problem is proportional to the negative of the weighted least-squares error using the inverse of the covariance matrix as the weight. As such, we get the following optimization problem

$$\hat{z}_0^* = \arg\min_{z_0} \quad V(z_0) \tag{5.42}$$

where $V(z_0) = \sum_{k=0}^{T} \tilde{y}_k^T R^{-1} \tilde{y}_k$ is the objective function, and z_0^* is the optimal initial condition. This problem is solved using gradient descent methods, where the gradients are solved numerically.

Similar to before, we convert from the augmented state z back to the state x. As a reminder, $z = \begin{bmatrix} \delta_x^T & \overline{x}^T \end{bmatrix}^T$ and $x = \overline{x} + \delta_x$. Therefore, we can calculate our optimal state estimate $\hat{x}_0^* = \overline{x}_0^* + \delta_{x,0}^*$, which is used for detecting and diagnosing attacks. Moving forward, this is simply called $\hat{x}_0$.

5.7.2 Necessary Output Accuracy

The changes in the previous section account for the mismatch in our assumptions about the noise statistics. In addition, there is also a mismatch between the simulator dynamics and our model dynamics. This is an expected result as all models are approximations of real system dynamics. Here, we discuss how to quantify output accuracy and, more importantly, how to correlate it with achieving our objectives.

Our research objectives are to be able to detect attacks and estimate the state. This means the parameter that we really care about is the accuracy of the state estimate. As

a result, we want to be able to correlate the accuracy of the output with the accuracy of the state estimate. This will help determine whether our model is accurate enough and also provides an important sensitivity study that can determine how realistic it is for plants to implement these methods.

For this section, we focus on nominal accuracy of the output. This means that we care about how accurate just the model is and ignore the measurement noise. Later, measurement noise will be incorporated to create decision rules, similar to the physics-based work described earlier in the chapter.

The two measures that we want to correlate are output accuracy and state estimation accuracy. The output accuracy measures the distance between the true measurements and estimated measurements over the duration of the transient. To account for the time-series data, the weighted root-mean-square (RMS) error between these two signals is used

$$\eta_O = \sqrt{\frac{1}{T}\sum_{k=0}^{T} \tilde{y}_k^T W \tilde{y}_k} \tag{5.43}$$

where η_O is the output accuracy, $\tilde{y}_k = \hat{y}_k - y_k$, $\tilde{y}_0$ is assumed zero, and $W^{-\frac{1}{2}}$ is a diagonal matrix containing the alarm values of 100 psi and 10 %, respectively, that accounts for the different variable scales. The half power is used for the weigh matrix because the matrix is squared inside the sum, i.e. $(W^{\frac{1}{2}}\tilde{y}_k)^T(W^{\frac{1}{2}}\tilde{y}_k) = \tilde{y}_k^T W \tilde{y}_k$

The state estimation accuracy measures the distance between the true state and estimated state. This could be some average value similar to the above, but that is not necessary for this application. Instead, just the difference at the initial condition is used. Similar to the output accuracy, the state estimation accuracy uses a weighted norm, resulting in

$$\eta_E = \sqrt{\tilde{x}_0^T C^T W C \tilde{x}_0} \tag{5.44}$$

where η_E is the state estimation accuracy and $\tilde{x}_0 = \hat{x}_0 - x_0$.

With these terms defined, we can now correlate output accuracy with state estimation accuracy. Our goal is to norm bound these terms. Or mathematically, this can be written find $f(\cdot)$ such that $\eta_E \leq f(\eta_O)$. This provides a quantitative measure that correlates these two parameters. We will solve this through optimization techniques by running simulations

where we impose output error on the measurements, called y^{new}, and then use the incorrect sequence for our state estimation algorithm. This is comparable to our model incorrectly predicting the sequence of outputs, and can be measured using our output accuracy defined above.

In order to simulate these scenarios, we need a standard method of approximating an output accuracy with a certain magnitude. This is done by adding a term that is proportional to the measurement value, but subtracting off the nonzero initial condition. This can be written as

$$y_k^{\text{new}} = y_k + A(y_k - y_0) \tag{5.45}$$

where A is a diagonal matrix that results in some known output accuracy. See Figure 33 for an example of a measurement dataset with simulated error.

This new dataset can then be used to estimate the state. This is done using the maximum likelihood estimation described above, resulting in an optimal estimate of the state for that given output error scenario.

The full problem is solved as a series of optimization problems. For some fixed scalar output accuracy c, there is a set of A matrices as defined above that result in that output accuracy. The optimization problem is to maximize the state estimation accuracy over the set of A matrices. Or mathematically,

$$\max_{\eta_O = c} \quad \eta_E \tag{5.46}$$

This problem is solved for varying values of c to calculate the correlation curve.

This optimization is solved using particle swarm optimization. This is a heuristic-based optimization method that makes few assumptions about the objective function and has been shown to have improved performance in global optimization problems over some of the standard gradient-descent algorithms.

It is worth noting that this approach can only ever be an approximation on the upper bound of the accuracy. First, there are an infinite number of ways that the output error can be simulated, and we only implement one way. Second, the data-driven model may not be uniformly accurate. In other words, it may be more accurate at some states than others,

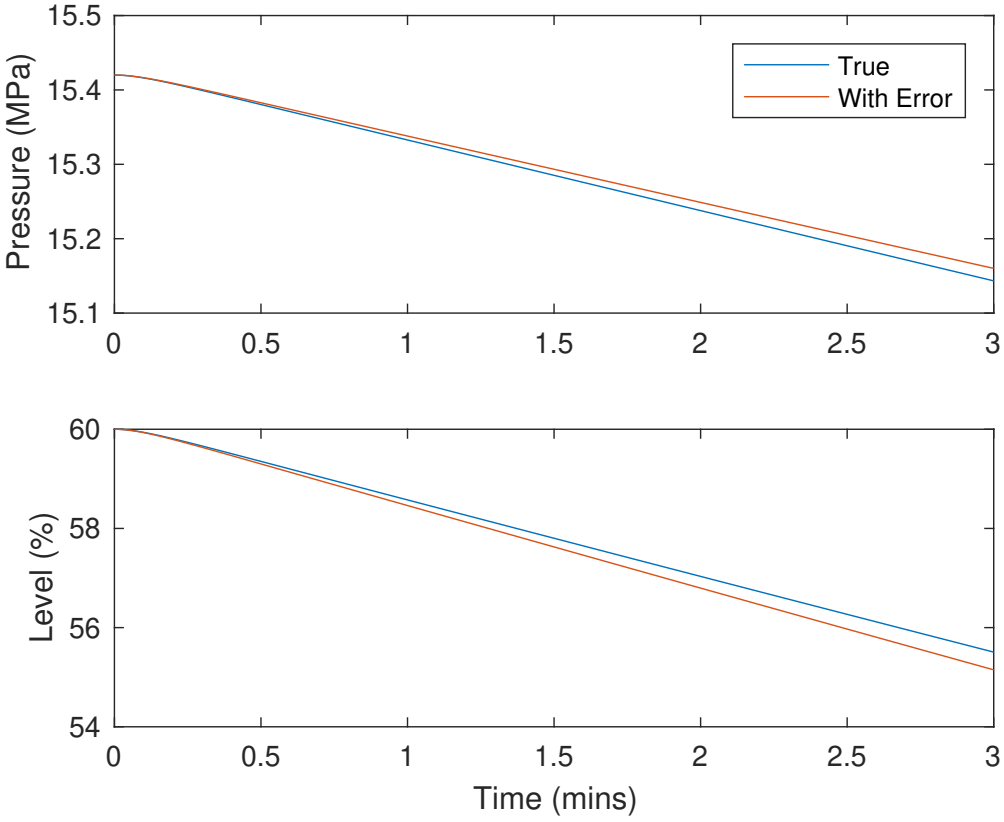


Figure 33: Plot of the simulated output error. The blue lines are the true signals, and the red lines show how output error is introduced for the analysis.

which could cause larger estimation accuracy values under certain conditions. However, this approach still represents a reasonable attempt to estimate this upper bound.

5.7.3 Results

Using the data-driven model, we calculate the accuracy of the state estimation techniques on the training, testing, and validation datasets. These are compared with the estimate of the bounded output accuracy calculated using the approach described above. Finally, the state error statistics are used to create a decision rule and calculate a detection rate on the validation datasets.

Before discussing the results, we first discuss in more detail the datasets contained in the training, testing, and validation sets. The primary difference between the different datasets is the starting initial conditions; see Figure 34 for a plot of the different initial pressure and level values. In addition, this plot shows the largest bounding window around the training set. Within this training window, we would expect our model to perform fairly well. Outside of this training window, it is hard to know how well the model will generalize. We have included three points in the validation set that are outside of the training window to see how well our model is able to detect attacks in this region of the state space. These points are discussed in more detail in the remainder of this section.

The first result we discuss is the correlation between output accuracy and state estimation accuracy. From our optimization analysis, this correlation is essentially linear and equal to $\eta_E = 78\eta_O$. It is later referred to as the cutoff line.

With this cutoff line calculated, we can look at the accuracy of the state-estimation techniques using nominal data without noise. Ideally, these values should all be near or below the estimated cutoff line. We calculate the state estimation accuracy for each dataset within the training, testing, and validation sets. Starting with just the datasets that fall within the training window, Figure 35 shows the state estimation accuracy against the output accuracy for the three types of datasets. Then, the points outside the training window are included in Figure 36, which as expected have much larger output errors. In these plots, there are

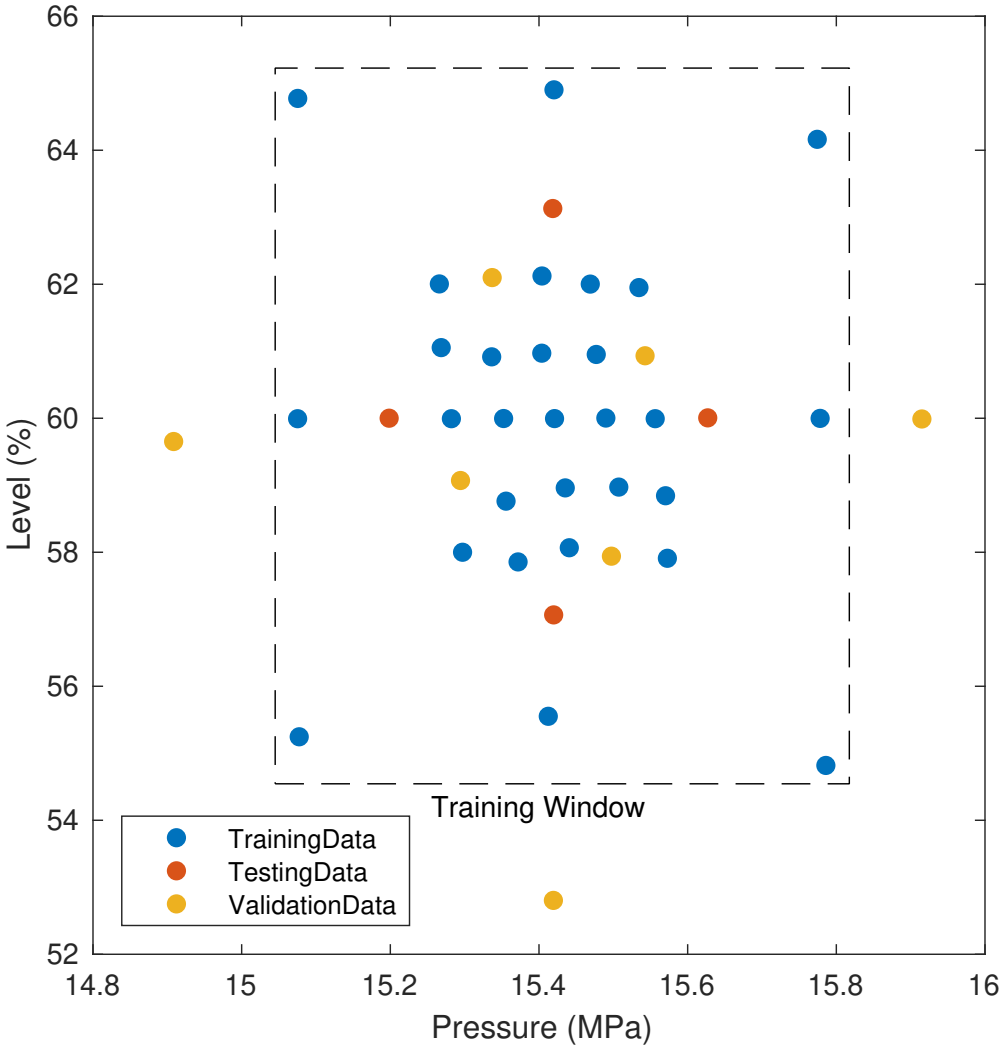


Figure 34: Plot of the initial conditions for the training, testing, and validation datasets. In addition, the dashed box shows the range of data included in the training set, and there are a few points outside of it.

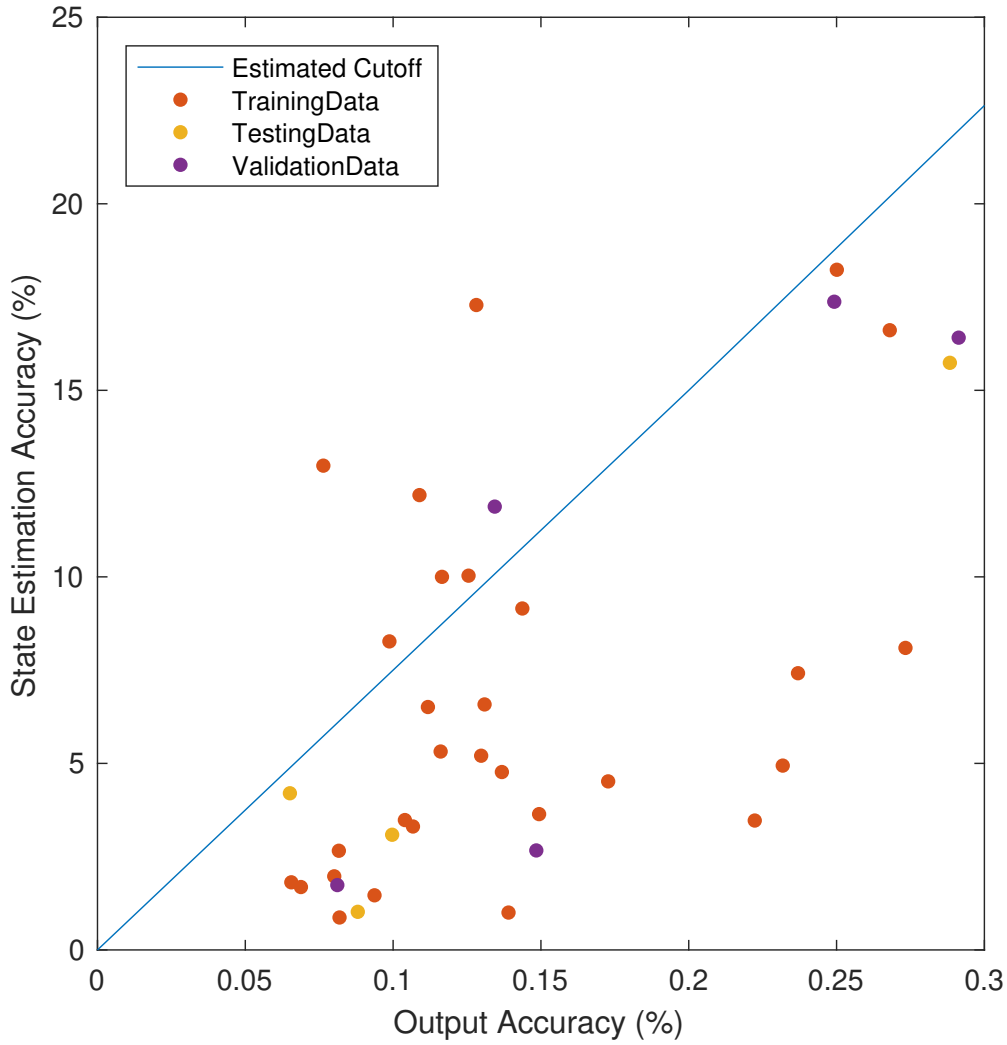


Figure 35: Plot of output accuracy versus state estimation accuracy for the training, testing, and validation sets. The blue line represents the estimated cutoff, and we expect points to lay at or below the blue line. This curve is missing two data points that are outside our training window, and are used to see how well the model generalizes far from data.

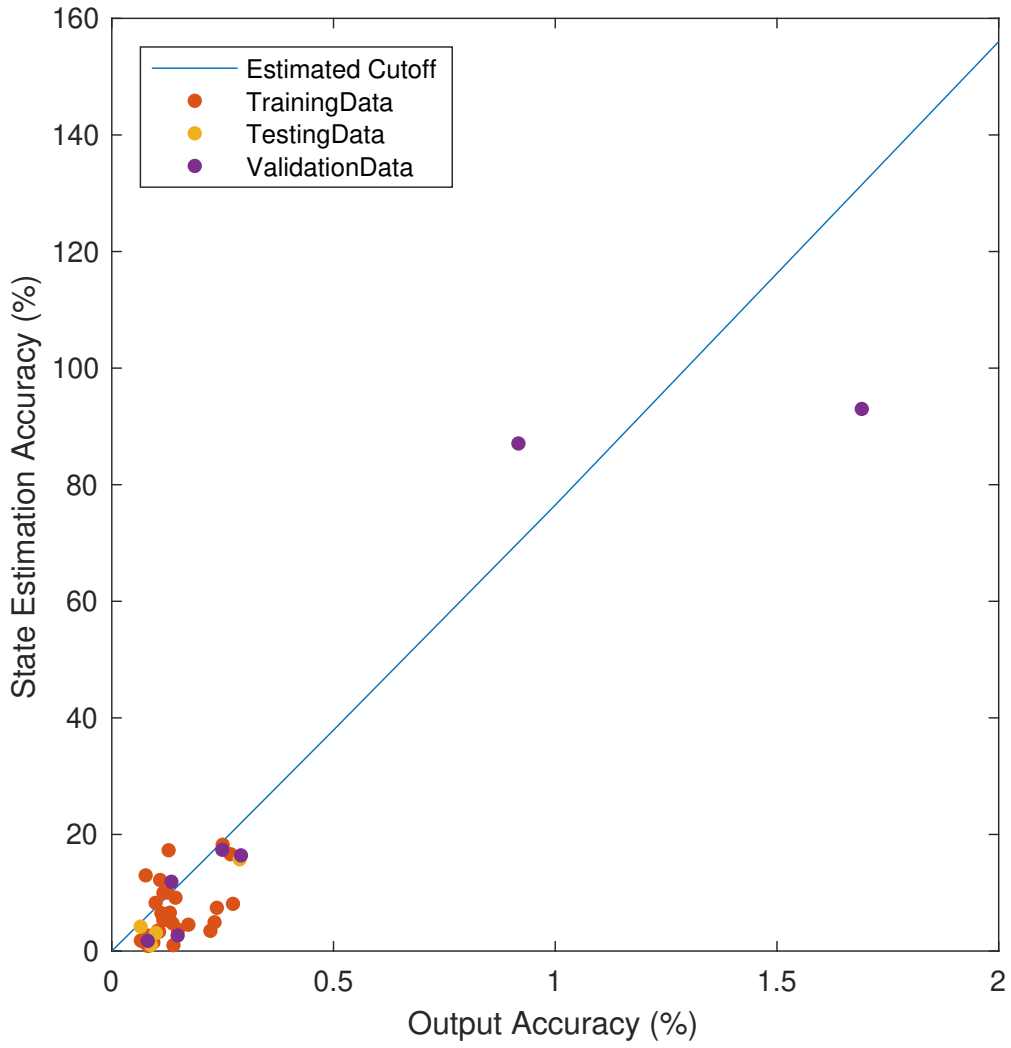


Figure 36: Plot of output accuracy versus state estimation accuracy for the training, testing, and validation sets. The blue line represents the estimated cutoff, and we expect points to lay at or below the blue line. This curve includes two data points that are outside our training window, and are used to see how well the model generalizes far from data. Even at these higher errors, the data does not stray far from our estimated cutoff values.

a total of 40 datasets. And of these, 33 are below the norm bounds calculated using the approach in the previous section, which suggests our cutoff line is a decent estimate.

It is also worth looking more closely at these state estimation accuracy results to better understand where the large state estimation errors are coming from. To do this, we look at the pressure and level errors individually; see Figures 37 and 38 for similar plots of the state estimation accuracy against the output accuracy, but with pressure and level broken out separately. Figures 37, which only includes those datasets within the training window, shows that the error for the pressure is bounded near 5 %, while the error for the level is bounded near 20 %. Even including the datasets outside the training window, the pressure is bounded near 20 %, while the level is bounded closer to 100 %. This means our methods work much better on the pressure than the level. We believe that understanding why this is and what system properties result in easier or harder estimation is a good candidate for future work and will be discussed more in Chapter 6.

Next, we look at the state error statistics at nominal operating conditions including measurement noise. This enables us to develop a decision rule similar to the one used for the physics-based model. Using the same approach and the same value of α, this results in the decision region found in Figure 39, along with the error data from the simulations. For this set of data, all error values are within the region, so would not result in any false positives.

Finally, we test the decision rule and detection routine frequency using simulations. We do not have as much control over the initial state as in the physics-based efforts. Here, we take the four validation sets inside the training window as our first attempt and the three validation sets outside the training window as our second attempt. And, these simulations include measurement noise. Over 100 simulations each for the set inside the training window and outside the training window, our detection algorithm caught 100 % of the attacks. This means our approach is successfully validated using simulator data. Similar to with the physics-based case, however, attacks may not be detected in the earliest stages; the validation sets were all detected because they are sufficiently far from nominal conditions.

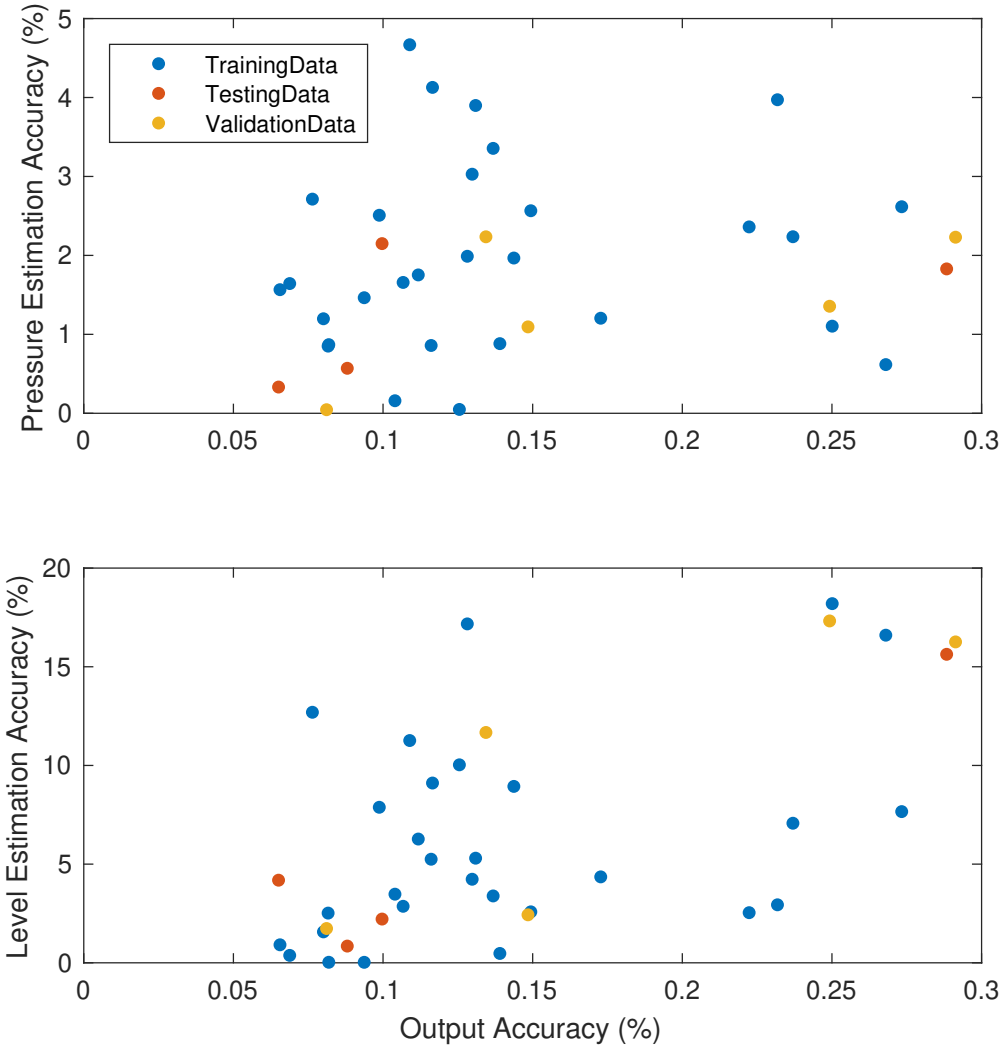


Figure 37: Plot of output accuracy versus state estimation accuracy for the training, testing, and validation sets. Compared to the previous plot, this breaks out pressure and level separately, instead of using a norm. This curve is missing two data points that are outside our training window, and are used to see how well the model generalizes far from data.

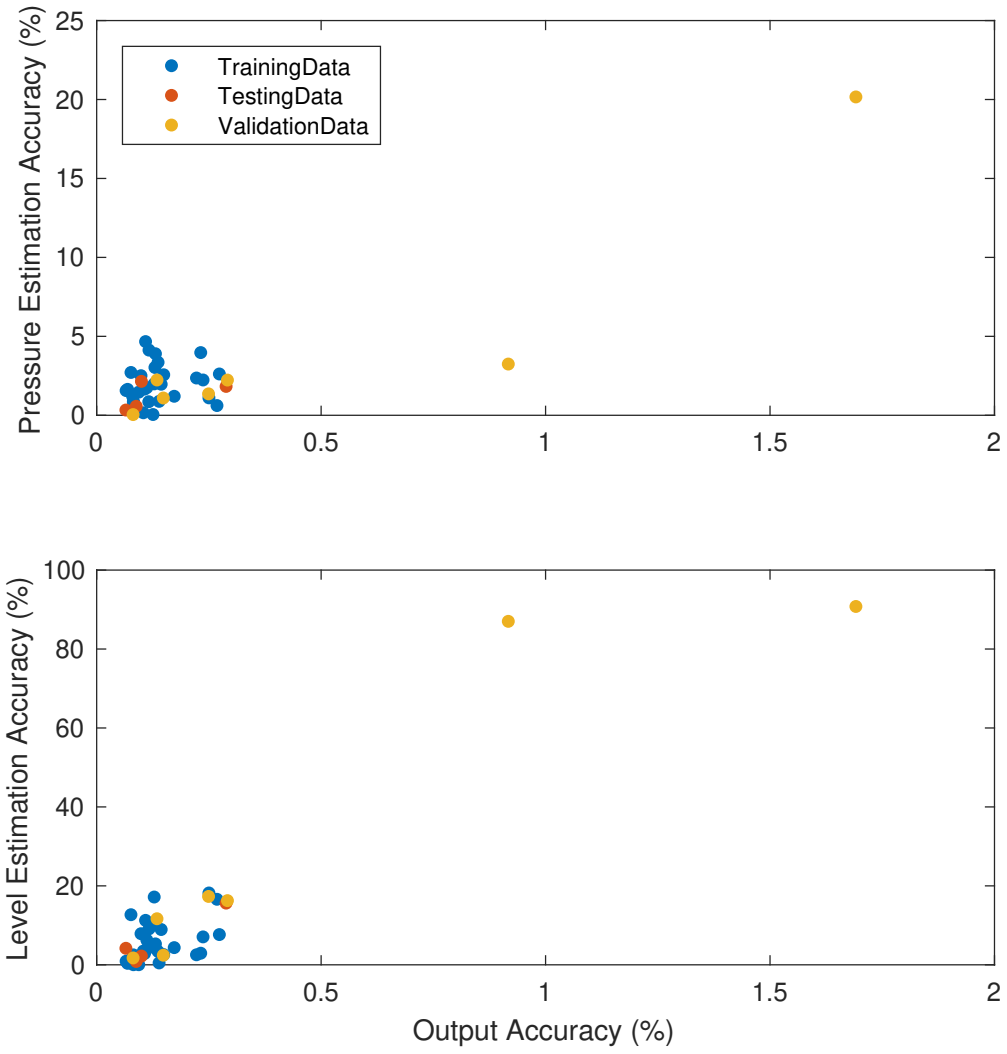


Figure 38: Plot of output accuracy versus state estimation accuracy for the training, testing, and validation sets. Compared to the previous plot, this breaks out pressure and level separately, instead of using a norm. This curve includes two data points that are outside our training window, and are used to see how well the model generalizes far from data.

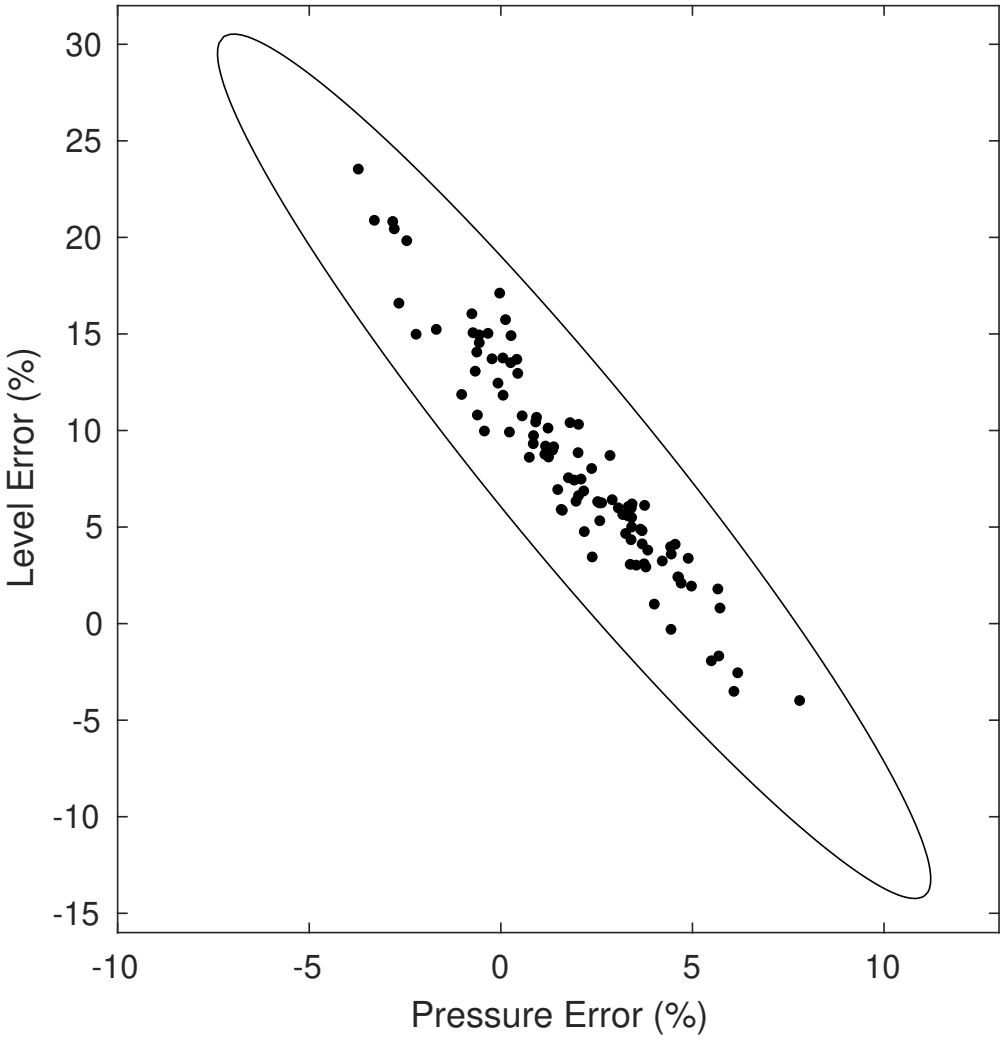


Figure 39: Plot of the decision region and the data from the simulations for the data-driven mode. The black dots are each estimates for a different noise simulation, and the boundary is the decision region. Note these are in units of percentage of alarm values.

5.8 Detecting Zero-Dynamics Attacks Offline

One potential drawback to this approach is that the system must be perturbed from nominal operating conditions to detect attacks. In general, this is something that plants try to minimize. So a natural follow-up question is: can this approach be done without affecting the plant? For example, could it be done purely computationally using a parallel offline mode?

This type of detection scheme is sometimes called a digital twin. These are generally digital models that capture desired behavior or dynamics of some physical object. They use historical data to try and estimate parameters. Here, a hypothetical digital twin would purely use the input and output data to determine whether the plant is under attack. For reference, state estimators can be used as digital twins.

The challenge with any offline detection mode is that zero-dynamics attacks do not leave evidence of the attack. In fact, that is precisely what makes them so dangerous. As such, historical data would not provide any insight into whether the plant is in the middle of a zero-dynamics attack. What this means is that in order to detect these zero-dynamics attacks, the system must be perturbed. And, this cannot be done in any offline manner.

5.9 Detecting Other Stealthy Attack Strategies

In addition to detecting zero-dynamics attacks, the perturbation proposed here can also be used to detect an additional stealthy attack strategy: the replay attack. These attacks are not the focus of this work, but still represent a dangerous attack strategy that should be considered during a cyber security risk assessment.

In Chapter 2, we discuss the replay attack, and how they can be detected by inserting a signal, unknown to the attacker, into the system to create expected transients. To detect replay attacks, previous researchers have proposed inserting zero-mean random control inputs. But, those zero-mean inputs would not detect the zero-dynamics attacks that are the focus of this work. By contrast, our perturbation would create an expected transient, so can be used to detect both zero-dynamics attacks and replay attacks.

5.10 Economic and Safety Impacts of the Perturbation

The proposed perturbation reduces both the pressurizer level and pressurizer pressure. And, the pressure reduction cascades to the rest of the primary side, reducing pressure throughout. It is worth discussing the economic and safety impacts of these changes, as well as implementation of the perturbation.

Economic Impact Regarding plant economics, we discuss two possible financial implications of using the proposed detection mechanism: (i) the change in pressure could result in a change in power output, and (ii) there could be additional operating costs.

Without going into all the technical details, a change in coolant pressure will change its density and could result in a change in power output. However, pressurized water reactors use liquid water, which is often approximated as incompressible because there is little change in density over pressure changes. This means any density changes will be small compared to other parameter changes, like temperature. As such, we expect there to be some very small change in power. It is worth investigating this further in future work, but our expectation is that it will be very small given the small change in pressure and the nearly incompressible nature of liquid water.

The second point on additional operating costs depends heavily on the plant in question and the level of automation. If there is little automation, the perturbation would need to be implemented manually, which could generate a noticeable cost increase. However if there is significant automation or if the perturbation could be designed into the control system from the vendor, the detection method would run automatically without requiring operator intervention. Therefore, further analysis on the costs associated with additional labor would need to be done on the specific plant.

Safety Impact From a safety point of view, the perturbation was designed so that the transient magnitude was significantly smaller than any alarm values. As such, we do not expect the transient to pose a safety threat when not under attack.

One exception to this scenario is if the plant truly is under attack. In this scenario, our transient could move the plant further into unsafe operating conditions, which could impact

safety. However, we note that in this case, the attack would eventually reach these unsafe conditions anyway, or could reach worse conditions. Therefore, if our method successfully detected the attack, it would still improve safety because the operators would know the plant was under attack and could respond accordingly.

5.11 Chapter Summary

As a reminder, our research objectives relating to this chapter are to detect attacks by monitoring physical measurements and provide diagnostic information to enhance plant response. In this chapter, we accomplish the following:

- the exact and approximate problems are setup, and a test for whether systems are detectable is proposed;
- the approach for solving for the optimal input is presented using optimization methods;
- the state-estimation techniques are discussed for how to estimate the unknown state in an optimal manner;
- the results of the approach implemented on the pressurizer are discussed;
- an analysis on how accurate the system model needs to be is presented; and
- the results are validated using the data-driven model and data from the commercial simulator.

This work completes the objectives because we are able to estimate the state providing both diagnostic information and, combined with the decision rules developed, determinations of whether the system is under attack. We then implement them on a critical subsystem of pressurized water reactors to show that they can successfully be used to detect the attacks outlined in the previous chapter.

This concludes the analysis from the defender's perspective on how to defend against zero-dynamics attacks targeting nonlinear systems.

6.0 Conclusions and Future Work

The goal of this research is to improve cyber security for NPPs by addressing the following questions: How might an attacker attack NPPs? And how can automated defenses defend against those attacks? These concepts are expanded through the following research objectives:

1. characterize stealthy cyber vulnerabilities targeting nuclear power plants;
2. detect attacks by monitoring physical measurements; and
3. provide diagnostic information to enhance plant response.

All of these objectives have been met through new theoretical tools and their implementation on the pressurizer subsystem.

Our first objective is met through the work described in Chapter 4. In that chapter, we extend previous results on zero-dynamics attacks to systems with nonlinear dynamics. To accomplish this, we transform a control-affine nonlinear system into a system under attack. We then discuss an iterative algorithm to calculate nonlinear zero-dynamics attacks based on this attack model. Finally, we demonstrate the techniques on the pressurizer subsystem by analyzing all combinations of attackable signals and characterizing the resulting zero-dynamics attacks by stability and damage time. Ultimately, the advantage of using these nonlinear methods is that linearization is not required, allowing us to be more exact about the zero dynamics of the system.

Our second and third objectives are met through the work described in Chapter 5. In that chapter, we develop an approach to detect zero-dynamics attacks targeting systems with nonlinear dynamics. We first approximate the problem to constrain the number of unknowns and create a more tractable problem. This approximate problem still requires a nonzero control input, and we show how the nonlinear observability matrix can be used to optimize this control input for detecting attacks. Using the physics-based model, we use unscented Kalman filters and maximum likelihood estimation to estimate the true state, enabling us to both detect the attack and provide important diagnostics. Finally, we approximate the

required output accuracy and use the data-driven model to validate the results using data from the commercial simulator.

6.1 Summary of Contributions

To reiterate from the beginning of the dissertation, our main contributions to the fields of nonlinear controls and nuclear cyber security are that we:

1. extend zero-dynamics attacks to nonlinear systems through the development of an attacked system model and an algorithm;
2. develop metrics for zero-dynamics attacks to compare different attack strategies;
3. characterize zero-dynamics attacks targeting the pressurizer subsystem of pressurized water reactors;
4. develop tools to determine whether zero-dynamics attacks targeting a specific nonlinear system are detectable;
5. develop detection methods for detectable systems under zero-dynamics attacks;
6. quantify required model accuracy to be able to accurately detect attacks; and
7. implement the detection methods on the pressurizer subsystem using both a theoretical first-principles model and a data-driven model.

6.2 Implementation on Other Systems

In this dissertation, the pressurizer system is used to demonstrate the approaches on characterizing and detecting attacks. But, it is important to note that the same techniques can be applied to many other systems, both in nuclear power plants and in other industries, without major modifications. The reason for this is that the work uses the very general state-space representation, which enables systems from many fields to be described using a common form.

In order to implement our work on other systems, the following steps should be carried out:

1. Derive system dynamics using state-space representation
2. Transform dynamics into model under attack (Section 4.1)
3. Implement algorithm for calculating zero-dynamics attacks (Section 4.2)
4. Analyze stability of characterized attacks (Section 4.4)
5. Transform dynamics into augmented model (Section 5.1)
6. Ensure augmented model is observable (Section 5.1)
7. Calculate optimal defender control input (Section 5.2)
8. Estimate the state under attack conditions (Section 5.3)
9. Develop a decision rule using one-class classification (Section 5.4)

By following these steps, this work can be extended to safety-critical systems across many industries to help safeguard them.

6.3 Limitations

Before concluding this work, it is important to note any limitations that may prevent its use or require additional research. We discuss two limitations in this section.

First, for characterizing zero-dynamics attacks, many of the attack strategies require the attacker to have significant access to the system inputs and outputs. This access makes the attacks particularly dangerous because they result in zero measurable response. However, it also means the attacks may be harder to implement than other strategies that could have some similar properties. If there are other strategies that are almost as dangerous but easier to implement, they should be evaluated and compared to the attacks discussed here based on a cyber security risk assessment.

Second, when applied to the pressurizer system, the detection algorithm for zero-dynamics attacks is sensitive to output accuracy. This means that accurately estimated outputs can result in accurate state estimates, enabling effective attack detection. But it also means that

inaccurate models could result in unreliable estimates. Further analysis is required to better understand this relationship and to extend the ideas to general systems.

If these limitations can be overcome, it would further advance the field of nuclear cyber security. Possible directions for future work that address these limitations are discussed in the next section.

6.4 Future Work

Future work could focus on four primary goals, where the latter three aim to address the limitations discussed above:

1. characterizing zero-dynamics attacks for other subsystems or for larger models of full systems;
2. investigating attack strategies that have small outputs, but are not exactly zero;
3. quantifying whether a nonlinear system is a good candidate for the detection methods; and
4. determining how design choices affect the relationship between model and estimation accuracy.

Starting with characterizing zero-dynamics attacks for other systems, this work can be applied to full nuclear power plant models to look for vulnerabilities outside of just the pressurizer. This would give a more full picture of potential vulnerabilities, rather than just those from a single subsystem. In addition, this would enable collaborations with other fields like game theory. These collaborations could optimize defenses based on the potential damage and attacker effort of these attacks.

The next proposed area for future work is investigating attack strategies that have small outputs, but are not exactly zero. This is motivated by the attacks characterized in this research that are stable as a purely zero-dynamics attack, but can become unstable, and thus cause damage, if some measurable output is allowed. And, these attacks require less attacker access to implement, making them theoretically easier to implement. This direction

for future work is a result of the strict definition of zero dynamics that requires zero output. If this requirement is relaxed, other attacks may be discovered that are nearly as stealthy, but require less attacker access.

Moving to the detection side of this research, another direction for future work is quantifying the feasibility of implementing the detection methods on general nonlinear systems. This can focus on understanding how hard it is to estimate different states within a system or comparing entire systems to determine the difficulty of estimating all the states. The result could aid in understanding what types of systems these methods would work best with.

A final direction for future work, that expands upon the previous item, is determining how design choices affect the relationship between model and estimation accuracy. For example looking at the pressurizer, how do dimensions and operating conditions affect this relationship? And are there other designs for controlling and monitoring pressure that could result in easier estimation under attack? The answers to these questions could further enable designers to design-in security using the system dynamics.

Appendix A Description of Appendices

During the first half of my graduate program, we worked on several fault detection applications focused on detecting loss-of-coolant accidents (LOCAs) in Pressurized Water Reactors. These accidents occur when the working fluid leaks out of the system, and are dangerous because the systems are pressurized to ensure the water remains a liquid; a leak could cause the liquid to flash to steam and prevent critical cooling of the reactor core.

In these Appendices, we present three applications of detecting LOCAs using three different approaches. These works are presented here because they are related to the final dissertation work, but do not directly solve the problems the dissertation addresses. In addition, these works can be found published in [38, 39, 17].

Appendix B Using Model-Based Fault Detection to Differentiate Transients and Loss of Coolant Accidents

B.1 Introduction

Detecting loss of coolant accidents (LOCAs) is a safety-critical task for nuclear power plant (NPP) operators. In order to manually detect LOCAs, operators must, in addition to their other responsibilities, constantly monitor multiple process parameters, which makes it difficult to detect LOCAs early. Earlier detection could avoid or reduce the large transients, like a reactor trip, that stress systems, further overload operators, and might ultimately lead to core damage [40]. Furthermore, operators might not be able to diagnose specific characteristics of the LOCA, such as the magnitude of the leak, which could influence the best response to the accident. In order to alleviate operator workload and improve detection capabilities, online monitoring tools would automatically alert operators to the onset of LOCAs and provide additional diagnostics about the accident, including the leak magnitude.

Previous research for detecting the onset of LOCAs used artificial neural networks (ANNs) [41]. Based on measurements at each time step, an ANN detected several faults, including the onset of LOCAs. This method looked at LOCAs initiated while the plant was at steady-state, but they could be initiated at anytime. It is necessary to investigate the performance of the methods during transient operation as well.

Other research estimated leak magnitudes for LOCAs using support vector regression (SVR) [42]. The SVR method accurately estimated the leak magnitudes in all leak scenarios tested. However, the estimation process did not begin until the reactor had already tripped and could only provide a single estimate, which would not work for a time-varying leak magnitude.

These efforts implemented data-driven methods for LOCA detection, meaning they used large quantities of dynamic training data to develop the detection tools. These types of methods provide a powerful framework when training data exists. However, these methods may not work when training data does not exist, or when scenarios occur that fall outside the training envelope.

In this paper, we detected a LOCA using a model-based approach that does not depend on large quantities of training data and works in all scenarios for which the models are appropriate. Specifically, we detected the onset and estimated the leak magnitude of a LOCA that was initiated during a transient and with a time-varying magnitude. This was done using multiple-model adaptive estimation (MMAE), which uses multiple mathematical models to estimate the most likely operating conditions [43]. In this work, one model represented plant dynamics during normal operating conditions, and the remaining models represented the dynamics during LOCA conditions with different leak magnitudes.

The models were derived using first principles and system identification. First principles were used to derive a general form for the model equations and system identification was used with process data to estimate the parameter values. Using this procedure, the same approach could be easily implemented for plants with different sizes or configurations. In addition, while this procedure required process data, it did not require process data from a LOCA, which differentiates it from the data-driven methods described above.

Using the same measured inputs to the system, the models produced different estimates of a particular sensor measurement, and the estimates were compared using Bayesian hypothesis testing. This statistical testing method calculated a real-time probability for each of the models, enabling the method to identify the most likely operating condition. This operating condition could provide both detection of the onset and an estimate of the magnitude of the leak.

This method has previously been used to detect a LOCA [44]. That work detected a constant-magnitude LOCA initiated at steady-state without sensor noise. This work improves upon that previous research.

This paper used process data from a pressurized water reactor simulator. This simulator was developed by GSE Systems and models a 970 MWe, three-loop, generic pressurized water reactor. The simulator has a model accuracy that is compliant with ANS-3.5, which is the American Nuclear Society's standard for "Nuclear Power Plant Simulators for Use in Operator Training and Examination" [21]. The simulator is used to train supervisors, operators, and engineers around the country.

This chapter is structured as follows. Section 2 describes the process data used for this research. Section 3 derives the general form for the model equations and explains the system identification methods used. Section 4 describes how the models are used in the multiple-model adaptive estimation technique. Section 5 provides the results of the research. Finally, Section 6 concludes the chapter.

B.2 Process Data

Process data was collected for normal and faulted operating conditions. The data from normal operating conditions was used for system identification, and the data from faulted operating conditions was used to test the fault detection techniques. This data included the measurable temperatures, pressures, mass-flow rates, pump currents, reactor power, and liquid levels throughout the primary and secondary sides. All data was collected at a sample rate of 2 Hz.

Data for normal operating conditions was collected by varying the electricity generated from 100 to 75% and then back to 100%. These power changes were done at a rate of 0.5% (5 MWe) per minute. This scenario was selected because it represents start-up and shutdown conditions for a plant, so comparable process data could be collected for a real plant.

Data for faulted operating conditions was collected by running simulations of small-break LOCAs with the same-time varying magnitude but two different conditions. Both LOCA magnitudes increased linearly from 2 to 4% over 60 s, where 100% is equivalent to a full break in a 4.5-inch-diameter pipe. The first LOCA was initiated at full power to show how the method works during steady-state operating conditions, and the second LOCA was initiated during a transient while the plant was increasing power at a rate of 0.5% per minute.

To make the data more representative of real plant data, additive Gaussian white noise was added to each of the relevant sensor measurements. Because neither noise variance estimates nor real process data are readily available, noise variance values for each sensor type were estimated using datasheets for commercial components. These components are manufactured by Emerson as an indicator of typical instrument accuracy. Using the listed ac-

Table 5: Listed accuracies of some commercial sensors for different sensor types.

Sensor Type	Accuracy (% of maximum value)
Temperature	0.5
Pressure	0.15
Flow	0.25
Level	0.2

curacies from the datasheets, shown in Table 5, we equated the accuracies to 95% confidence intervals for the Gaussian curve (equal to 2σ)

$$\text{accuracy} = \frac{2\sigma}{U_{\max}} \tag{B.1}$$

$$\text{var} = \sigma^2 = \frac{(\text{accuracy} \times U_{\max})^2}{4} \tag{B.2}$$

where $U_{\max}$ is the maximum sensor value. This provided a systematic estimate for the noise magnitudes.

Adding noise to the process data made it more representative of real plant data. For use with the online monitoring tools, that data was then filtered to remove some of the noise. The noisy signals were filtered using first-order low-pass filters. The time constants were chosen for each signal individually and the filtered values were used for the remainder of the research.

B.3 Pressurizer Model

The pressurizer model was derived using conservation of volume and mass within the primary loop. To simplify the problem, two assumptions were made. First, the piping does not expand considerably from temperature changes, so volumes remain constant. Second, the coolant inventory in the primary loops is always a liquid. This second assumption could

be violated late in an accident scenario; however, the goal of this research is to detect the fault well before reaching that point.

The volume in the primary loop, consisting of the hot and cold-leg water in the reactor coolant system, water in the pressurizer, and vapor in the pressurizer, is

$$V_{PL} = V_L + V_V + V_{HL} + V_{CL} \tag{B.3}$$

and its rate of change is

$$\dot{V}_L = -\dot{V}_V \tag{B.4}$$

where V is the volume, the PL subscript refers to the primary loop, the V subscript refers to the vapor part of the pressurizer, the L subscript refers to the liquid part of the pressurizer, and the HL and CL subscripts refer to the hot and cold-leg sections of the reactor coolant system, respectively.

The mass in the primary loop, consisting of the masses of the volumes listed above, is

$$m_{PL} = \rho_L V_L + \rho_V V_V + \rho_{HL} V_{HL} + \rho_{CL} V_{CL} \tag{B.5}$$

and its rate of change, equal to the net mass-flow rate entering and existing the system, is

$$\begin{aligned} \dot{m}_{PL} &= \dot{\rho}_L V_L + \rho_L \dot{V}_L + \dot{\rho}_V V_V + \rho_V \dot{V}_V + \dot{\rho}_{HL} V_{HL} + \dot{\rho}_{CL} V_{CL} \\ &= \dot{m}_{\text{in}} - \dot{m}_{\text{out}} - \dot{m}_{\text{leak}} \end{aligned} \tag{B.6}$$

where ρ is density, $\dot{m}_{\text{in}}$ is the charging mass flow entering the primary loop, $\dot{m}_{\text{out}}$ is the letdown mass flow leaving the primary loop, and $\dot{m}_{\text{leak}}$ is any leaking mass flow leaving the primary loop.

Combining Equations B.4 and B.6 and solving for $\dot{V}_L$ gives

$$\dot{V}_L = \frac{1}{\rho_V - \rho_L}(V_L \dot{\rho}_L + V_V \dot{\rho}_V + V_{HL} \dot{\rho}_{HL} + V_{CL} \dot{\rho}_{CL} + \dot{m}_{\text{out}} - \dot{m}_{\text{in}} + \dot{m}_{\text{leak}}) \tag{B.7}$$

This can be further simplified using the definitions for the volumes as

$$V_L = AL_L + V_{LC} \tag{B.8}$$

$$V_V = AL_V + V_{VC} \tag{B.9}$$

$$L_V = 100 - L_L \tag{B.10}$$

where V_{LC} and V_{VC} are constant liquid and vapor volumes that fall below and above the sensor boundaries respectively. Combining all of this and converting the constants into unknown parameters gives

$$\dot{L}_L = \frac{1}{\rho_V - \rho_L}\Big((\dot{\rho}_L - \dot{\rho}_V)L_L + c_1\dot{\rho}_L + c_2\dot{\rho}_V + c_3\dot{\rho}_{HL} + c_4\dot{\rho}_{CL} + c_5(\dot{m}_{\text{out}} - \dot{m}_{\text{in}} + \dot{m}_{\text{leak}})\Big) \tag{B.11}$$

Finally, this equation can be rewritten by grouping like terms together

$$\dot{L}_L = -\frac{\dot{\rho}_L - \dot{\rho}_V}{\rho_L - \rho_V}L_L - \frac{c_5}{\rho_L - \rho_V}(\dot{m}_{\text{eff}} + \dot{m}_{\text{net}} + \dot{m}_{\text{leak}}) \tag{B.12}$$

where

$$\dot{m}_{\text{eff}} = \frac{c_1\dot{\rho}_L + c_2\dot{\rho}_V + c_3\dot{\rho}_{HL} + c_4\dot{\rho}_{CL}}{c_5} \tag{B.13}$$

$$\dot{m}_{\text{net}} = \dot{m}_{\text{out}} - \dot{m}_{\text{in}} \tag{B.14}$$

The effective mass-flow rate represents the changes in mass due to the changing density of the coolant.

B.3.1 Model Structure

A small-break LOCA affects the coolant volume in the primary loop, which can be determined from the pressurizer liquid level. The level is measured through a level sensor that indicates the percentage of the total level that is liquid. Its value is maintained within a desired range by controlling the letdown and charging flows in the CVCS. And because the pressurizer has the highest elevation in the primary loop, the level indicates total coolant volume in the primary loop.

In this work, system models were required and were described using discrete-time state-space representations. For a linear system, these are written as

$$x_{k+1} = A_k x_k + B_k u_k + G_k w_k \tag{B.15}$$

$$z_k = C_k x_k + v_k, \tag{B.16}$$

where $x_k \in \mathbb{R}^n$ is the system state vector, $u_k \in \mathbb{R}^p$ is the input vector, $z_k \in \mathbb{R}^r$ is the measurement vector, A_k is the state matrix, B_k is the input matrix, G_k is the noise input matrix, C_k is the output matrix, w_k is process noise, and v_k is measurement noise. Both noise sources are assumed to be Gaussian white processes, described by $E[w_k] = 0$, $E[v_k] = 0$, $E[w_k w_l^T] = Q\delta_{kl}$, and $E[v_k v_l^T] = R\delta_{kl}$.

When creating the model, factors that affect the pressurizer level are temperatures and pressures throughout the primary loop, temperatures and pressures inside the pressurizer, and charging and letdown mass flow rates. For a derivation of the model, see the appendix. This model was parameterized with unknown constants that were determined using system identification methods, described in the next section. The final model is

$$\dot{L}_L = -\frac{\dot{\rho}_L - \dot{\rho}_V}{\rho_L - \rho_V} L_L - \frac{c_5}{\rho_L - \rho_V}(\dot{m}_{\text{eff}} + \dot{m}_{\text{net}} + \dot{m}_{\text{leak}}) \tag{B.17}$$

where

$$\dot{m}_{\text{eff}} = \frac{c_1\dot{\rho}_L + c_2\dot{\rho}_V + c_3\dot{\rho}_{HL} + c_4\dot{\rho}_{CL}}{c_5} \tag{B.18}$$

$$\dot{m}_{\text{net}} = \dot{m}_{\text{out}} - \dot{m}_{\text{in}} \tag{B.19}$$

and L_L is the liquid level, the V subscript refers to the vapor part of the pressurizer, the L subscript refers to the liquid part of the pressurizer, the HL and CL subscripts refer to the hot and cold-leg sections of the reactor coolant system, respectively, $\dot{m}_{\text{in}}$ is the charging mass flow entering the primary loop, $\dot{m}_{\text{out}}$ is the letdown mass flow leaving the primary loop, and $\dot{m}_{\text{leak}}$ is any leaking mass flow leaving the primary loop.

This model was then transformed into the standard format for a time-varying continuous-time state-space representation

$$\dot{x} = A(t)x + B(t)u^{(i)} \tag{B.20}$$

$$z = x \tag{B.21}$$

where the state and input are, respectively,

$$x = L_L \tag{B.22}$$

$$u^{(i)} = \dot{m}_{\text{eff}} + \dot{m}_{\text{net}} + \dot{m}_{\text{leak}}^{(i)} \tag{B.23}$$

and the matrices are defined as

$$A(t) = -\frac{\dot{\rho}_L - \dot{\rho}_V}{\rho_L - \rho_V} \tag{B.24}$$

$$B(t) = -\frac{c_5}{\rho_L - \rho_V} \tag{B.25}$$

Note that $A(t)$ and $B(t)$ are time-varying continuous-time matrices. The value that changes between the different models is the leak mass-flow rate $\dot{m}_{\text{leak}}^{(i)}$, where i indexes the different models. The normal-operating-condition model has a zero leak mass-flow rate, and the other operating conditions vary over the range of possible values.

These techniques were implemented on digital control systems, so a discrete-time state-space representation was used. Because the time constant of the system was long compared to the sample period ($h = 0.5\,\text{s}$), the time-varying system matrices were assumed constant over the duration of the sample period. As such, the discrete-time matrices were calculated by

$$A_k = e^{hA(t)} = e^{hA(kh)} \tag{B.26}$$

$$B_k = \left(\int_{\tau=0}^{h} e^{hA(kh)} d\tau \right) B(kh) \tag{B.27}$$

where e^{hA} is the matrix exponential.

B.3.2 System Identification

The goal of system identification is to identify an unknown model based on data [45]. Starting with the model form for the pressurizer liquid level, data was used to determine the unknown parameters. This data is described in Section B.2.

This system identification problem was formulated as a least-squares error problem

$$\min_{v} \sum_{k=1}^{n} e_k^2 \tag{B.28}$$

where $v = [c_1, \ldots, c_5]^T$ is the unknown parameter vector, n is the number of data points used in the system identification, and $e_k = z_k - \hat{z}_k$ is the error between the measurement and the estimated measurement. The estimated measurements were calculated using

$$\hat{z}_k = A_{k-1}\hat{z}_{k-1} + B_{k-1}u_{k-1} \tag{B.29}$$

with assumed known initial condition $\hat{z}_0 = z_0$, calculable A_k, and $B_k u_k$ equal to a linear function of the unknown parameters. This error function needed to be transformed in order to use the standard solution. Moving one step forward in time gave

$$\hat{z}_1 = A_0 z_0 + B_0 u_0 \tag{B.30}$$

$$= \overline{z}_1 + \phi_1 v \tag{B.31}$$

where $\overline{z}_k$ and $\phi_k v$ are the known and unknown portions, respectively. Next, this was generalized to form the recursive equation

$$\hat{z}_k = A_{k-1}\overline{z}_{k-1} + A_{k-1}\phi_{k-1}v + B_{k-1}u_{k-1} \tag{B.32}$$

$$= \overline{z}_k + \phi_k v \tag{B.33}$$

where

$$\overline{z}_k = A_{k-1}\overline{z}_{k-1} \tag{B.34}$$

$$\phi_k v = A_{k-1}\phi_{k-1}v + B_{k-1}u_{k-1} \tag{B.35}$$

with $\overline{z}_0 = z_0$ and $\phi_0 = 0$.

Table 6: Least-squares estimates of the unknown parameters.

c_1	c_2	c_3	c_4	c_5
-6.94×10^1	-3.50×10^1	1.99×10^2	4.30×10^2	7.73×10^{-3}

This is now the standard least-squares problem with $e_k = (z_k - \overline{z}_k) - \phi_k v$. This can be solved by writing it in matrix form

$$Y = \Phi v \tag{B.36}$$

where $Y = [y_0, \ldots, y_n]^T$, $y_k = z_k - \overline{z}_k$, and $\Phi = [\phi_0, \ldots, \phi_n]^T$. The least-squares solution to the problem is the pseudo-inverse

$$\hat{v} = (\Phi^T \Phi)^{-1} \Phi^T Y \tag{B.37}$$

where $\hat{v}$ is the least-squares estimate of the unknown parameter vector.

Using the normal operating data, we implemented the system identification techniques to calculate least-squares estimates of the unknown parameters; see Table 6. Note that the units are arbitrary based on dimensionless units. The fully defined model was then simulated and compared with the raw data. The results are shown in Figure 40. This figure shows that the model is a good approximation of the GPWR simulator data.

B.4 Multiple-Model Adaptive Estimation

B.4.1 Kalman Filters

The models were implemented using Kalman filters, which are model-based filters used to filter noisy signals and for model-based fault detection. For linear systems with Gaussian noise and ideal system models, the Kalman filter provides optimal linear estimates that

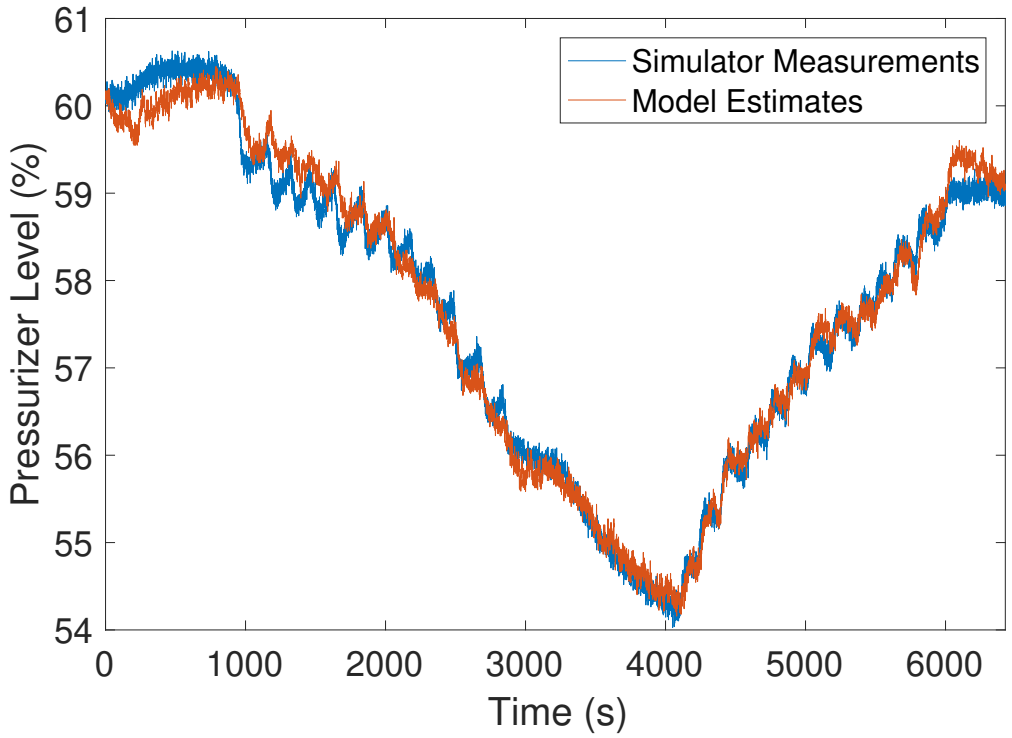


Figure 40: Simulator data and system identification estimates from the normal operating conditions.

minimize the mean-squared error [46]. In the MMAE approach, N Kalman filters run in parallel, where each filter represents a different operating condition.

The Kalman filter can be broken into prediction and correction steps. In the prediction step, the previous state and state-covariance estimates are used to predict the next values

$$\hat{x}^{(i)}_{k|k-1} = A_k \hat{x}^{(i)}_{k-1|k-1} + B_k u^{(i)}_k \tag{B.38}$$

$$\hat{z}^{(i)}_{k|k-1} = C_k \hat{x}^{(i)}_{k|k-1} \tag{B.39}$$

$$P^{(i)}_{k|k-1} = A_k P^{(i)}_{k-1|k-1} {A_k}^T + G_k Q {G_k}^T, \tag{B.40}$$

where $\hat{x}^{(i)}_{k|k-1}$ is the predicted state estimate, $\hat{z}^{(i)}_{k|k-1}$ is the predicted measurement, $P^{(i)}_{k|k-1}$ is the predicted state-covariance estimate, and the i superscript refers to the i^{th} filter in the set of filters and matches the operating conditions above. This predicted estimate is the same formula as the estimate used for system identification.

In the corrector step, updated measurements are received and weighted to correct the predictions

$$\hat{x}^{(i)}_{k|k} = \hat{x}^{(i)}_{k|k-1} + K^{(i)}_k \tilde{z}^{(i)}_k \tag{B.41}$$

$$P^{(i)}_{k|k} = (I - K^{(i)}_k C_k) P^{(i)}_{k|k-1} \tag{B.42}$$

$$K^{(i)}_k = P^{(i)}_{k|k-1} {C_k}^T \Sigma^{(i)-1}_k \tag{B.43}$$

$$\tilde{z}^{(i)}_k = z_k - \hat{z}^{(i)}_{k|k-1} \tag{B.44}$$

$$\Sigma^{(i)}_k = C_k P^{(i)}_{k|k-1} {C_k}^T + R, \tag{B.45}$$

where $\hat{x}^{(i)}_{k|k}$ is the corrected state estimate, $P^{(i)}_{k|k}$ is the corrected state-covariance estimate, $K^{(i)}_k$ is the Kalman gain and represents a weighting factor to calculate the corrected estimates, $\tilde{z}^{(i)}_k$ is the innovation and is the difference between the measurements and the predicted measurements, and $\Sigma^{(i)}_k$ is the innovation-covariance estimate.

In addition to estimating the states, the Kalman filter also predicts a statistical model for the innovations sequence. For an ideal system model, the statistical model of the innovations is Gaussian white noise with covariance equal to the innovation-covariance estimate defined above [47]. This means that the model that matches the current operating conditions will

produce an innovations sequence that is normally distributed with zero-mean and covariance approximated by the innovation-covariance.

Using the same normal operating data as for system identification, the parameters R and Q were calculated. These parameters determine both the measurement-feedback gain and the theoretical innovation covariance. The value for R, the measurement-noise covariance, was calculated as the sample covariance estimate of the measurement during steady-state operation. Then, the value of Q, the process-noise covariance, was calculated by running the normal operating data through Kalman filters with varying Q values until the sample innovation covariance matched the theoretical innovation covariance. The final values were: $R = 5.4 \times 10^{-3}$ and $Q = 1.4 \times 10^{-3}$, where both values are unitless because they are associated with the normalized level.

B.4.2 Bayesian Hypothesis Testing

Bayesian hypothesis testing is a method of assigning probabilities to the set of statistical models based on measured data. It starts with a prior probability for each model, which is the probability that the model is the best fit before incorporating data. It then calculates the likelihood of seeing the sampled data given each model. Finally, it calculates a posterior probability for each model, which is the probability that the model is the best fit for the new data. It does this by weighting the prior probabilities using the likelihoods.

The goal is to implement Bayesian hypothesis testing as a recursive update equation. In other words, the equation should calculate new probabilities at each time step based on the likelihoods of the current data and the probabilities at the previous time step. The equation considers N statistical models, called hypotheses, denoted by $\{H^{(1)}, \ldots, H^{(N)}\}$, and is defined as [48]

$$P(H^{(i)}|Z_k) = \frac{1}{c} P(z_k|Z_{k-1}, H^{(i)}) P(H^{(i)}|Z_{k-1}) \tag{B.46}$$

where z_k is a single measurement, $Z_k = \{z_0, \ldots, z_k\}$, $P(H^{(i)}|Z_k)$ is a posterior probability, $P(z_k|Z_{k-1}, H^{(i)})$ is a likelihood function, $P(H^{(i)}|Z_{k-1})$ is a prior probability, and c is a normalizing constant to ensure the probabilities sum to unity.

In order to implement the above equations, likelihood values for each hypothesis need to be calculated. Using the Gaussian model for the innovation, likelihood values can be calculated using the Gaussian density function

$$P(z_k|Z_{k-1}, H^{(i)}) = |2\pi\Sigma_k^{(i)}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\tilde{z}_k^{(i)T}\Sigma_k^{(i)-1}\tilde{z}_k^{(i)}\right). \tag{B.47}$$

The result of Bayesian hypothesis testing is a set of posterior probabilities for the different operating conditions. These posterior probabilities, calculated in Equation B.46, were then used for detecting the onset and estimating the leak magnitude of the LOCAs by calculating the expected value of the leak rates

$$E\left\{\dot{m}_{\text{leak}}\right\} = \sum_{i=1}^{N} \dot{m}_{\text{leak}}^{(i)} P(H^{(i)}|Z_k). \tag{B.48}$$

This entire procedure is shown in Algorithm 2.

A few notes on Algorithm 3. First, it was assumed that the plant was initially operating under normal conditions, $H^{(1)}$, so $P(H^{(1)}|Z_0)$ was significantly higher than the other probabilities. This was selected to reduce false alarms in the beginning that occurred from noise. This did not affect the algorithm's ability to detect the onset of the LOCA. Second, the threshold mentioned was a free design parameter defined as the leak size that distinguishes a LOCA from a false alarm. This value was selected as the smallest leak magnitude associated with any of the filters.

B.5 Results

The first LOCA tested was the baseline scenario to demonstrate how the methods both detected the onset of the accident and estimated the leak magnitude. For this scenario, the pressurizer level is shown for the times until the reactor tripped and until the LOCA was detected; see Figure 41. The time to trip, equal to 176 s, corresponded to a 28% drop in the level, and the time to detect, equal to 12.5 s, corresponded to just a 1% drop in the level.

The methods also estimated the time-varying magnitude of the leak, and the estimates and actual leak magnitudes are shown in Figure 42. From this plot, the estimates in the

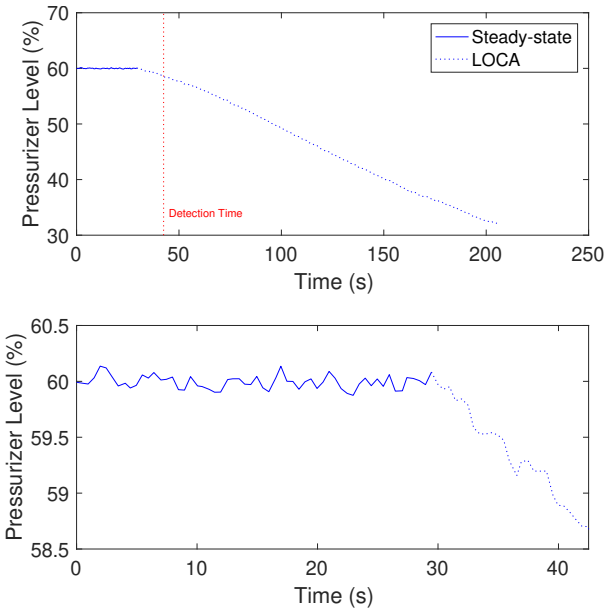


Figure 41: Level measurements from the simulator for the steady-state case. The top plot shows the entire accident scenario, and the bottom plot shows up until the accident was detected.

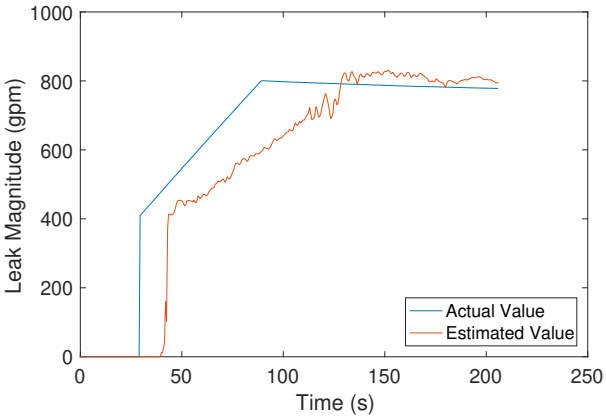


Figure 42: Comparison between the actual leak magnitudes and the estimated leak magnitudes for the steady-state case.

Algorithm 2 Method to detect LOCAs using MMAE.

```
assign P(H^(1)|Z_0) = 0.99
for each H^(i) in {H^(2), ..., H^(N)} do
    assign P(H^(i)|Z_0) = (1−0.99)/(N−1)
end for
assign status = Normal
for each time step k do
    for each H^(i) in {H^(1), ..., H^(N)} do
        calculate P(H^(i)|Z_k)
    end for
    calculate E{ṁ_leak}
    if E{ṁ_leak} ≥ threshold then
        status = LOCA
        magnitude = E{ṁ_leak}
    else
        continue
    end if
end for
```

middle of the scenario react more slowly than the leak magnitude increases. This results from trying to balance accuracy with insensitivity to noise. The estimates in the beginning and end are much closer to the true values, and show a clear increasing trend as the estimates try to converge to the true values.

Previous research has detected LOCAs initiated when the plant started at steady-state. However, none of these works have shown their techniques implemented during plant transients. The second LOCA tested was initiated while the plant power was increasing, causing normal changes in the pressurizer level. For this scenario, the pressurizer level is again shown for the time to trip and time to detect; see Figure 43. Towards the beginning of the transient, the level drops by nearly 1%, and the methods accurately ignore this as normal operating conditions. When the LOCA is initiated, the methods quickly detect the leak. The time to

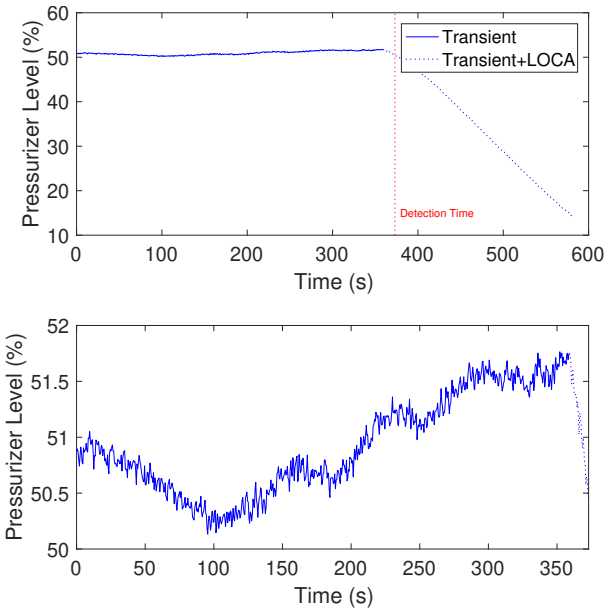


Figure 43: Level measurements from the simulator for the transient case. The top plot shows the entire accident scenario, and the bottom plot shows up until the accident was detected.

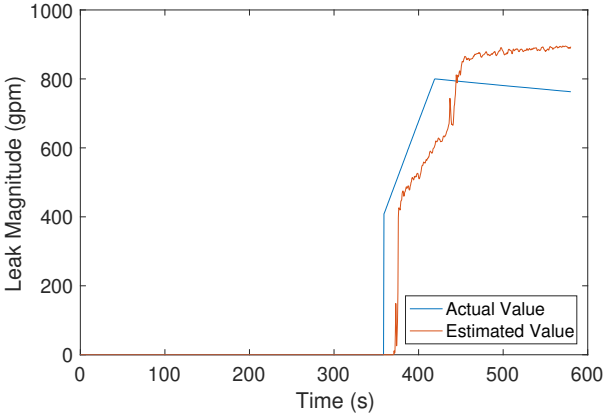


Figure 44: Comparison between the actual leak magnitudes and the estimated leak magnitudes for the transient case.

trip, equal to 220 s, corresponded to a 38% drop in the level, and the time to detect, equal to 13.5 s, corresponded to just a 1% drop in the level.

For this scenario, the estimates for the time-varying magnitude were also calculated. These values are compared to the actual magnitudes; see Figure 44. As noted above, the estimates remain small during the transient, and only increase when the LOCA is initiated. The estimates here show a similar trend to the previous scenario, although the final estimation error is larger compared with the steady-state case above.

B.6 Conclusion

In this chapter, we detected loss of coolant accidents in a GPWR simulator using a model-based approach. This approach used system models of the pressurizer under various operating conditions to identify the most likely operating conditions. The parameter values of the models were tuned to the specific plant using system identification and normal operating data. The models were implemented using Kalman filters for each operating condition, which used feedback from the measurements to estimate the pressurizer level. Finally, the outputs from the different models were used with Bayesian hypothesis testing to detect the loss of coolant accidents and estimate the leak magnitudes.

These methods were able to quickly and accurately detect the LOCAs and differentiate them from an operating transient. For the steady-state case, the LOCA was detected in 7% of the time it took for the reactor to trip, and for the transient case, the LOCA was detected in 6% of the time it took for the reactor to trip. In addition, the transient was correctly identified as normal operating conditions, demonstrating the methods have some robustness. These quick detection times and correct identification of a transient means that the methods could be highly beneficial for plant operators. The detection times would give them warning before the plant had even tripped that the fault occurred, and the identification of transients suggests the methods could be robust to false alarms. Based on these benefits, the methods would provide a powerful online monitoring tool to help operators.

One of the major advantages to this model-based approach for detecting LOCAs is that it does not require process data from a LOCA. This is significant because very little, if any, real process data from LOCAs exists. However, it is not difficult to capture operating data from start-up and shutdown procedures, which is all that is necessary for this methodology. This simplified data set makes these methods particularly powerful for detecting LOCAs compared to other methods.

Appendix C Using Kernel Density Estimation to Detect Loss of Coolant Accidents in a Pressurized Water Reactor

C.1 Introduction

Detecting loss of coolant accidents (LOCAs) is a safety-critical task for nuclear power plant (NPP) operators. Quick and accurate detection enables them to rapidly take the necessary actions to safely shut down the plant. In order to manually detect LOCAs, operators must, in addition to their other responsibilities, constantly monitor multiple process parameters, which makes it difficult to detect LOCAs early. If LOCAs could be detected earlier, it may be possible to avoid or reduce the large transients, like a reactor trip, that stress systems, further overload operators, and may ultimately lead to core damage [40]. Furthermore, operators may be unable to diagnose specific characteristics of the LOCA, such as the location of the leak. In order to alleviate operator workload and improve detection capabilities, online monitoring tools would automatically alert operators to the onset of LOCAs and provide additional diagnostics about the accident, including the leak location.

Previous research for detecting the onset of LOCAs used artificial neural networks (ANNs) [41]. Based on measurements at each time step, an ANN detected several faults, including the onset of several LOCAs. This method looked at each time step individually, meaning it did not take advantage of accumulated information over time. In addition, the method detected each of these faults using a common set of measurements; however, the generality of the measurement set made it less sensitive to LOCAs. As a result of these limitations, the LOCAs were detected well after the plant had tripped.

Other research identified leak locations for LOCAs using both ANNs and support vector machines (SVMs) in a one-loop plant [41, 42]. Both methods were differentiating between the hot and cold legs of the plant. While these methods were successful in identifying the leak locations, the plant configuration used is not typical for NPPs, which generally have multiple primary loops.

The research mentioned above, specific to LOCAs, used data-driven techniques, meaning they used large quantities of dynamic training data to develop the detection tools.

Fault detection has also been developed using physics-based techniques. For example, state-estimation techniques have used measurements and a system model to predict the behavior of a system [49, 50, 51]. The predictions and measurements were then compared using statistical testing to detect faults. Another technique has detected faults by analyzing qualitative trends in conservation of mass, momentum, and energy [52, 53]. Fault decisions were made using qualitative reasoning. The challenge with estimation and physics-based techniques is that they can be difficult to implement in certain cases [54].

In this paper, we detected LOCAs and identified the leak locations for a three-loop pressurized water reactor. This was done using data-driven kernel density estimation techniques, which estimated nonparametric probability density functions. These density functions provided likelihood values of the current measurements for the different fault status. This information was combined over time to take advantage of temporal data and improve detection speed.

We combined the likelihood values using two decision rules: Bayesian hypothesis testing (BHT) and maximum likelihood estimation (MLE). Using these two decision rules, we detected both the onset and location of the leaks. BHT used the likelihood values at each time step to calculate real-time probabilities for both normal and faulted operating conditions. These probabilities were then used to make decisions about whether a LOCA has occurred. MLE combined data starting from LOCA detection to provide the location that is most likely. Two different methods were used because, when identifying the leak locations, the goal was to select optimal locations, rather than to detect them quickly.

The statistical methods used in this paper have a broad range of applications. A few of those applications are given below. Kernel density estimation has been used for fault detection in radar systems [55], dam safety [56], and flood modeling [57]. Bayesian hypothesis testing has been used for detecting satellite faults [58], structural health monitoring of aerospace components [59], and military target tracking applications [60]. These statistical methods have a long history of use for fault detection and safety analysis purposes.

This paper used process data from a pressurized water reactor simulator. This simulator was developed by GSE Systems and models a 970 MWe, three-loop, generic pressurized water reactor. The simulator uses the RETACT thermal hydraulics package, which can

Table 7: Variable states used to generate multiple LOCA scenarios.

Location (6 values)	Loops A, B, C; hot leg and cold leg (Figure 45)
Magnitude (20 values)	1, 2,. . . , 20 %
Initial Condition (10 values)	90, 91,. . . , 100 %

model non-homogeneous and non-equilibrium conditions. It has a model accuracy that is compliant with ANS-3.5, which is the American Nuclear Society's standard for "Nuclear Power Plant Simulators for Use in Operator Training and Examination" [21]. The simulator is used to train supervisors, operators, and engineers around the country.

This chapter is structured as follows. Section 2 describes the process data used for this research. Section 3 derives the two sets of variables used for the data-driven techniques. Section 4 explains kernel density estimation, Bayesian hypothesis testing, and maximum likelihood estimation. Section 5 provides the results of the research. Finally, Section 6 concludes the chapter.

C.2 Process Data

To capture the correlations between different variables, process data was collected for normal and faulted operating conditions. This data included the measurable temperatures, pressures, mass-flow rates, pump currents, reactor power, and liquid levels throughout the primary and secondary sides. All data was collected at a sample rate of 2 Hz.

Data for normal operating conditions was collected by varying the electricity generated and the control-rod positions. The electricity generated was varied from 100 to 80% and then back to 100%. These power changes were done at a rate of 0.5% (5 MWe) per minute. Then, a single bank of control rods was adjusted by ±3 positions. Both scenarios were tested because they captured different time constants for the processes.

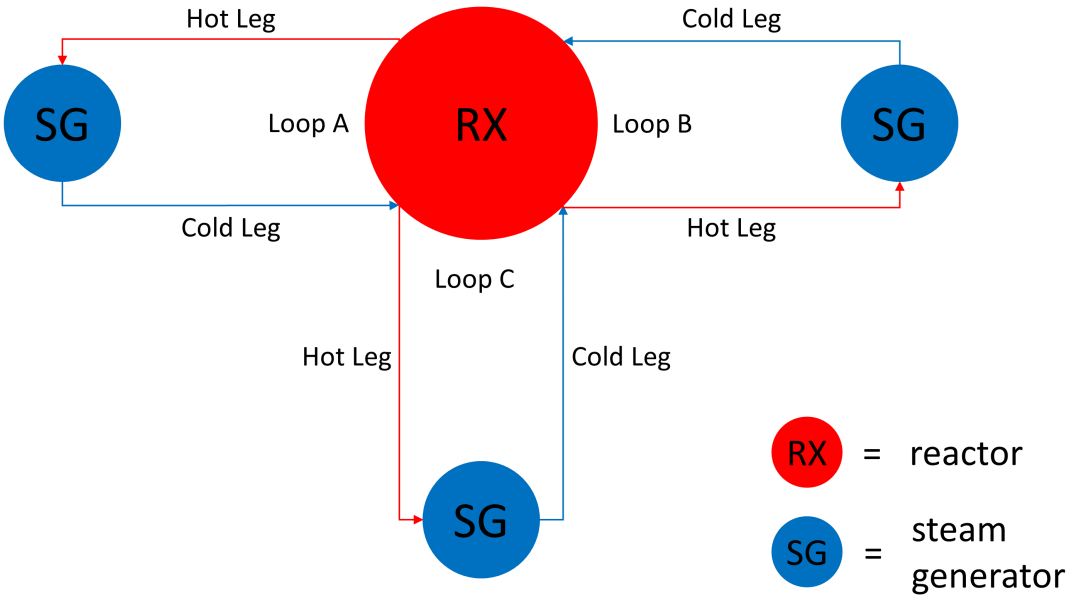


Figure 45: Schematic of the different leak locations considered in the three-loop pressurized water reactor.

Table 8: Listed accuracies of some commercial sensors for different sensor types.

Sensor Type	Accuracy (% of maximum value)
Temperature	0.5
Pressure	0.15
Flow	0.25
Level	0.2

Data for faulted operating conditions was collected by running simulations of small-break LOCAs and varying the LOCA location, LOCA magnitude, and the plant initial conditions. From the simulator, the magnitude was specified by a percentage, where 100% is equivalent to a full break in a 4.5-inch-diameter pipe. The initial condition was specified by varying the output electrical power to a percentage of full power. The different states of these three variables are shown in Table 7, and the possible locations are shown in Figure 45. Based on these states, data was collected as a full factorial design. This means every combination of the variable states was used, so the complete data set included 1200 LOCA scenarios. Each data set began with 30 seconds of normal operation and ended when the reactor tripped.

The collected data was split into training and validation sets. The training set was used to train the kernel density estimates, and the validation set was used to test the densities using Bayesian hypothesis testing and maximum likelihood estimation.

One method of selecting the training and validation sets is to use orthogonal arrays [61]. Orthogonal arrays are fractional factorial designs that optimally span the entire space of the design parameters. In this case, the design parameters were the LOCA location, the LOCA magnitude, and the plant initial conditions. The minimum orthogonal array cuts the full factorial design in half, creating equally sized training and validation sets.

To make the data more representative of real plant data, additive Gaussian white noise was added to each of the relevant sensor measurements. Because neither noise variance estimates nor real process data are readily available, noise variance values for each sensor

type were estimated using datasheets for commercial components. These components are manufactured by Emerson as an indicator of typical instrument accuracy. Using the listed accuracies from the datasheets, shown in Table 8, we equated the accuracies to 95% confidence intervals for the Gaussian curve (equal to 2σ)

$$\text{accuracy} = \frac{2\sigma}{U_{\max}} \tag{C.1}$$

$$\text{var} = \sigma^2 = \frac{(\text{accuracy} \times U_{\max})^2}{4} \tag{C.2}$$

where $U_{\max}$ is the maximum sensor value. This provided a systematic estimate for the noise magnitudes.

Adding noise to the process data made it more representative of real plant data. For use with the online monitoring tools, that data was then filtered to remove some of the noise. The noisy signals were filtered using first-order low-pass filters. The time constants were chosen for each signal individually and the filtered values were used for the remainder of the research.

C.3 Selecting Variable Sets

NPPs have a large number of measurable process variables, but they should not all be used for fault detection. First, the high variable-dimensionality results in solutions that are considerably more computationally expensive as the number of variables increases. Second, additional variables may overfit the data, making the methods less robust to new cases. Therefore, the total number of variables used should be reduced to the smallest set that can accurately differentiate the possible conditions.

C.3.1 Variable Set for Detecting Onset

The first variable set needed to distinguish between normal and LOCA operating conditions. A LOCA reduces the mass in the primary side, so one reasonable variable set correlates the rates of change of the masses in the primary-side subsystems. These subsystems include

the liquid coolant in the primary loops, the mixed liquid-vapor mass in the pressurizer, and the charging and letdown mass-flow rates.

For each of these subsystems, the rates of change of the masses were calculated based on bulk properties. These rates were based on two assumptions. First, the piping does not expand considerably from temperature changes, so volumes remain constant. Second, the coolant inventory in the primary loops is always a liquid. This second assumption could be violated late in an accident scenario; however, the goal of this research is to detect the fault well before reaching that point.

For the primary loop, the total mass is

$$m_P = \rho_P V_P \tag{C.3}$$

and its rate of change is

$$\begin{aligned} \dot{m}_P &= \dot{\rho}_P V_P + \rho_P \dot{V}_P \\ &= \dot{\rho}_P V_P \end{aligned} \tag{C.4}$$

where m is mass, ρ is density, V is volume, and the P subscript refers to the primary loops. Because V_P is constant, the first variable representing the primary loop mass-rate-of-change is reduced to $x_1 = \dot{\rho}_P$.

The pressurizer mass, which includes both liquid and vapor phases, is

$$m_{PZR} = \rho_V V_V + \rho_L V_L \tag{C.5}$$

and its rate of change is

$$\dot{m}_{PZR} = \dot{\rho}_V V_V + \rho_V \dot{V}_V + \dot{\rho}_L V_L + \rho_L \dot{V}_L \tag{C.6}$$

where the PZR subscript refers to the pressurizer, the V subscript refers to the vapor phase, and the L subscript refers to the liquid phase. The pressurizer system is a cylinder with cross-sectional area A, height H, and measurable liquid level L_L. The height can be normalized such that $H = 100$ and $L_V = 100 - L_L$. Using this normalization, Equation C.6 can be simplified to

$$\dot{m}_{PZR} = A\left(\dot{L}_L(\rho_L - \rho_V) + L_L(\dot{\rho}_L - \dot{\rho}_V) + 100\dot{\rho}_V\right) \tag{C.7}$$

Because A is constant, the second variable representing the pressurizer mass-rate-of-change is reduced to $x_2 = \dot{L}_L(\rho_L - \rho_V) + L_L(\dot{\rho}_L - \dot{\rho}_V) + 100\dot{\rho}_V$.

The final variable is derived from the charging and letdown mass-flow rates. These values are the mass-rates-of-change between the primary loop and the chemical and volume control system and can be used directly. Therefore, the third variable is the net mass-flow rate, $x_3 = \dot{m}_{\text{charging}} - \dot{m}_{\text{letdown}}$.

These three variables make up the final variable set for detecting whether the plant is in a normal or faulted operating condition. The set is denoted by $X_1 = \{x_1, x_2, x_3\}$.

C.3.2 Variable Set for Identifying Location

The second variable set needed to distinguish between the six leak locations. For this problem, mass terms could not be used because mass flow is only measured at one location per primary loop. Instead, data correlations were used to narrow down the relevant variables from a short list to the final variable set.

The short list of variables was identified visually based on plotting the measurable signals for LOCAs in each of the locations. The list included fourteen variables: reactor power, (3×) hot-leg temperatures, (3×) cold-leg temperatures, (3×) primary-loop mass-flow rates, (3×) reactor-coolant-pump amperages, and reactor-coolant-system pressure. Each of the variables with (3×) includes one measurement per primary loop.

The variable set was narrowed by analyzing the correlations between each pair of variables. Correlation is calculated as

$$r_{ij} = \frac{\sum_{k=1}^{n}(x_i^k - \bar{x}_i)(x_j^k - \bar{x}_j)}{\sqrt{\sum_{k=1}^{n}(x_i^k - \bar{x}_i)^2}\sqrt{\sum_{k=1}^{n}(x_j^k - \bar{x}_j)^2}} \tag{C.8}$$

where r_{ij} is an individual correlation coefficient between variables i and j, k indexes time, n is the number of time steps, and $\hat{x}_i$ is the mean of variable x_i. A correlation value close to ± 1 indicates that the variables are highly correlated. This means that the two variables provide nearly identical information, so one of them can be removed without much loss of information.

Using the training data set, the correlations were calculated for each pair. This can best be shown in matrix form

$$R = \begin{bmatrix} r_{11} & \cdots & r_{1d} \\ \vdots & \ddots & \vdots \\ r_{d1} & \cdots & r_{dd} \end{bmatrix} \tag{C.9}$$

where R is a symmetric matrix showing the correlations between each pair. Positive and negative correlations are treated the same, so the absolute value of R is shown

$$R = \begin{bmatrix} 1.00 & 0.91 & 0.90 & 0.91 & 0.26 & 0.24 & 0.26 & 0.33 & 0.25 & 0.34 & 0.53 & 0.51 & 0.56 & 0.47 \\ 0.91 & 1.00 & 0.99 & 1.00 & 0.13 & 0.15 & 0.13 & 0.31 & 0.27 & 0.32 & 0.45 & 0.39 & 0.45 & 0.60 \\ 0.90 & 0.99 & 1.00 & 0.99 & 0.14 & 0.18 & 0.14 & 0.30 & 0.31 & 0.30 & 0.43 & 0.34 & 0.42 & 0.57 \\ 0.91 & 1.00 & 0.99 & 1.00 & 0.13 & 0.15 & 0.13 & 0.31 & 0.27 & 0.32 & 0.45 & 0.39 & 0.45 & 0.60 \\ 0.26 & 0.13 & 0.14 & 0.13 & 1.00 & 0.98 & 1.00 & 0.28 & 0.21 & 0.29 & 0.25 & 0.30 & 0.33 & 0.15 \\ 0.24 & 0.15 & 0.18 & 0.15 & 0.98 & 1.00 & 0.98 & 0.29 & 0.12 & 0.29 & 0.26 & 0.37 & 0.34 & 0.12 \\ 0.26 & 0.13 & 0.14 & 0.13 & 1.00 & 0.98 & 1.00 & 0.29 & 0.21 & 0.28 & 0.25 & 0.30 & 0.34 & 0.15 \\ 0.33 & 0.31 & 0.30 & 0.31 & 0.28 & 0.29 & 0.29 & 1.00 & 0.68 & 0.81 & 0.21 & 0.39 & 0.40 & 0.68 \\ 0.25 & 0.27 & 0.31 & 0.27 & 0.21 & 0.12 & 0.21 & 0.68 & 1.00 & 0.68 & 0.19 & 0.00 & 0.24 & 0.49 \\ 0.34 & 0.32 & 0.30 & 0.32 & 0.29 & 0.29 & 0.28 & 0.81 & 0.68 & 1.00 & 0.32 & 0.39 & 0.27 & 0.68 \\ 0.53 & 0.45 & 0.43 & 0.45 & 0.25 & 0.26 & 0.25 & 0.21 & 0.19 & 0.32 & 1.00 & 0.54 & 0.50 & 0.47 \\ 0.51 & 0.39 & 0.34 & 0.39 & 0.30 & 0.37 & 0.30 & 0.39 & 0.00 & 0.39 & 0.54 & 1.00 & 0.55 & 0.55 \\ 0.56 & 0.45 & 0.42 & 0.45 & 0.33 & 0.34 & 0.34 & 0.40 & 0.24 & 0.27 & 0.50 & 0.55 & 1.00 & 0.54 \\ 0.47 & 0.60 & 0.57 & 0.60 & 0.15 & 0.12 & 0.15 & 0.68 & 0.49 & 0.68 & 0.47 & 0.55 & 0.54 & 1.00 \end{bmatrix}$$

The order in the matrix matches the order described in the beginning of this section. From this matrix, reactor power, (2×) hot-leg temperatures, and (2×) cold-leg temperatures were removed, leaving nine variables.

One limitation of kernel-density methods is that they can require large amounts of memory to store likelihood values for a d-dimensional space. And, the memory required increases exponentially with each variable. We estimated that at most seven variables could be used before running into memory problems. Therefore, we needed to remove two more variables to determine the final variable set.

The two variables removed were the remaining hot and cold-leg temperatures. These two variables were selected because they were correlated with the other hot and cold-leg temperatures, respectively, which suggested that temperature variations were independent of the leak location. In contrast, the (3×) primary-loop mass-flow rates and (3×) reactor-coolant-pump amperages were less correlated between the loops, suggesting they provided more leak-location dependent information.

After removing these two groups of variables, the second variable set, X_2, was reduced to seven variables: the (3×) primary-loop mass-flow rates, (3×) reactor-coolant-pump amperages, and reactor-coolant-system pressure.

This variable set can be explained by the mass-flow physics. When a leak occurs, there is a localized pressure-change in a part of the plant. This will cause a change in the relationship between the mass-flow rates and the reactor-coolant-pump amperages that is a function of the total system pressure. This variable set captures those changes to be able to identify the correct location.

C.4 Methods

C.4.1 Kernel Density Estimation

Kernel density techniques estimate nonparametric probability density functions over a set of given variables. These nonparametric density functions are desirable when a parametric density is unable to describe the data set. When used for differentiating between different operating conditions, they need to be conditional probability density functions, $P(X_i|\text{Condition})$, over a specific variable set, X_i, and for a given condition. Detecting the onset and identifying the location together required eight different density functions

$$P(X_1|\text{Normal}), P(X_1|\text{LOCA})$$
$$P(X_2|L_1), \ldots, P(X_2|L_6)$$

where L_i is the i-th location.

A kernel density estimator uses training data to estimate the density value at each given point in the variable space. The result is a d-dimensional grid of likelihood values calculated by [62]

$$\hat{P}_{H_i}(X) = \frac{1}{n}\sum_{k=1}^{n} K_{H_i}(X - X_i^k) \tag{C.10}$$

$$K_{H_i}(X) = |H_i|^{-\frac{1}{2}} K(H_i^{-\frac{1}{2}} X) \tag{C.11}$$

where $\hat{P}_{H_i}(X)$ is the density estimate at X for variable set X_i, n is the number of time steps, $K(\cdot)$ is a kernel smoothing function, and H_i is the bandwidth matrix. The kernel smoothing function is a non-negative function that integrates to one and helps smooth the density estimate. The bandwidth matrix defines the amount of smoothing.

The kernel smoothing function and the bandwidth matrix need to be determined. This research used the multivariate Gaussian probability density function as the kernel function. The bandwidth matrix was calculated using Silverman's rule of thumb [62]

$$H_i = \begin{bmatrix} h_1 & & \\ & \ddots & \\ & & h_d \end{bmatrix} \tag{C.12}$$

$$h_j = \sigma_j \left(\frac{4}{(d+2)n} \right)^{\frac{1}{d+4}} \tag{C.13}$$

where σ_j is the standard deviation of x_j, and d is the dimensionality of X. This method of calculating the bandwidth matrix correlates it to the dimensionality of the problem and the standard deviations of each individual variable. Kernel density estimation was implemented using Matlab's *mvksdensity* function [63].

For each density estimate, the above equations resulted in a discretized, d-dimensional matrix of likelihood values over the span of the variable space. In order to calculate the values between this discretized matrix, linear interpolation was used.

C.4.2 Bayesian Hypothesis Testing

To detect the onset of a LOCA, the density functions over variable set X_1 are used with BHT. This method assigns probabilities to a set of statistical models based on measured data. It starts with a prior probability for each model, which is the probability that the model is the best fit before incorporating data. It then calculates the likelihood of seeing the sampled data given each model. Finally, it calculates a posterior probability for each model, which is the probability that the model is the best fit for the new data. It does this by weighting the prior probabilities using the likelihoods.

Algorithm 3 Method to detect LOCAs using BHT.

assign $P(\text{Normal}|X_1^{0:0}) = P(\text{LOCA}|X_1^{0:0}) = 0.5$

assign status = Normal

for each time step k **do**

  calculate $P(\text{Normal}|X_1^{0:k})$, $P(\text{LOCA}|X_1^{0:k})$

  if $P(\text{LOCA}|X_1^{0:k}) >$ threshold **then**

    status = LOCA

  else

    continue

  end if

end for

BHT can be formulated as a recursive update equation, meaning that it calculates new posterior probabilities at each time step based on the likelihood values of the current data and the probabilities of the previous time step. In this research, the equations consider the two operating conditions and are defined as [48]

$$\begin{aligned} P(\text{Normal}|X_1^{0:k}) &= \frac{1}{c} P(X_1^k|\text{Normal}) P(\text{Normal}|X_1^{0:k-1}) \\ P(\text{LOCA}|X_1^{0:k}) &= \frac{1}{c} P(X_1^k|\text{LOCA}) P(\text{LOCA}|X_1^{0:k-1}) \end{aligned} \tag{C.14}$$

where X_1^k is variable set 1 at time k, $X_1^{0:k} = \{X_1^0, \ldots, X_1^k\}$, $P(\text{Condition}|X_1^{0:k})$ is a posterior probability, $P(X_1^k|\text{Condition})$ is the kernel density estimate calculated in the previous section, $P(\text{Condition}|X_1^{0:k-1})$ is a prior probability, and c is a normalizing constant to ensure the probabilities sum to unity. These equation also show more explicitly that the probabilities are functions of the variable sets over multiple time steps.

The result of BHT is a set of posterior probabilities for the different operating conditions. These posterior probabilities, calculated in Equation C.14, were then used for detecting a LOCA. This procedure is shown in Algorithm 3.

The threshold mentioned in Algorithm 3 is a free design parameter defined as the probability that distinguishes normal from faulted operating conditions. When the posterior

probability for the LOCA operating condition exceeded the threshold, a LOCA was declared. This meant that higher values of the threshold can reduce the possibility of false alarms, but can also increase the detection delay during an actual LOCA. Its value was determined experimentally, as shown in Section C.5.

C.4.3 Maximum Likelihood Estimation

To identify the leak location, the density functions over variable set X_2 are used with MLE. This method selects parameter values based on maximizing the total likelihood of a set of observations. Here, the parameter values were the different leak locations.

To calculate the MLE, the total likelihood for each condition needed to be determined. The total likelihood of a series of measurements is equal to the product of each individual likelihood value. For location L_i, this is defined as

$$\prod_{k=n_1}^{n_2} P(X_2^k|L_i) \tag{C.15}$$

where n_1 and n_2 are the times when the fault was detected and the reactor tripped, respectively.

In implementation, it is common to use the natural logarithm of the likelihood function. This creates a sum of log-likelihood values instead of a product, but still has the same maximum location. Based on this, the most likely location, L_i^*, was found using

$$L_i^* = \arg\max_{L_i} \sum_{k=n_1}^{n_2} \log P(X_2^k|L_i) \tag{C.16}$$

C.5 Results

The methods were implemented on each of the LOCA scenarios from the validation set. The metrics used to analyze performance were the number of false alarms, the detection delay, and the number of correctly identified locations.

The threshold used with BHT in Section C.4.2 was selected to balance the number of false alarms with the detection delay. These two metrics are shown in Figure 46 for different

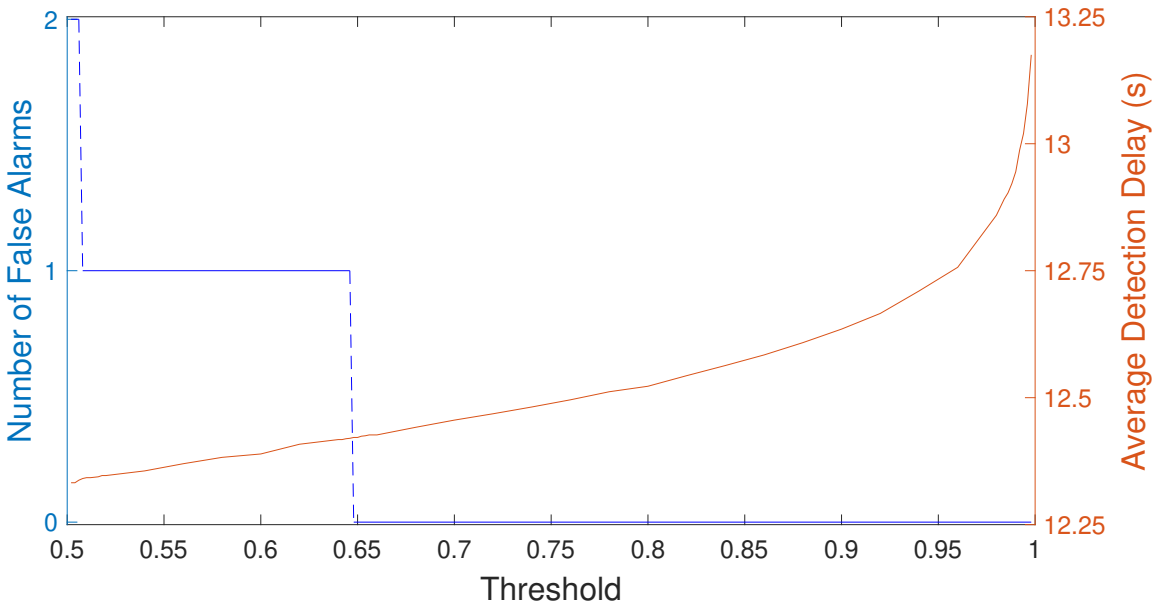


Figure 46: A plot of the number of false alarms and average detection delay as a function of the decision threshold used with BHT.

values of the threshold. From the figure, the number of false alarms drops to zero when the threshold value passes 0.65. Therefore, a threshold value of 0.65 was selected to give the shortest detection delay that still reduces the number of false alarms to zero.

The scenarios were grouped by leak magnitude, and the average, minimum, and maximum delay values were calculated. These results were also compared with the time it took for the reactor to trip; see Figure 47. This plot shows that the detection methods detect the fault well before the plant trips for all magnitudes examined. The average detection delay was one-seventh the time to trip, giving operators significant warning before the plant would trip. Overall, the methods provided reliable detection in a short period of time.

It is also worth comparing these results to those found by Bartlett and Uhrig [41]. Their research detected multiple faults, including two LOCAs in the hot and cold legs of a one-loop plant. These accidents tripped the reactor in 4.0 and 4.5 seconds, respectively, and the detection delays were 47.5 and 37.5 seconds, respectively. Based on the times to trip the reactor, their LOCAs were larger in magnitude than any tested in this research. But, the detection delays for our largest LOCAs were approximately one-tenth the delays for their LOCAs. By using a variable set specific to loss of coolant accidents and taking advantage of temporal data, we were able to detect the loss of coolant accidents more quickly.

Finally, the results of the location detection problem were assessed based on the selected threshold. The results are again broken out by magnitude; see Table 9. For very small magnitude LOCAs, the methods did a poor job of identifying the location. This is because there is insufficient change in the loops for the methods to differentiate the fault from noise. However, once the leak magnitude exceeds 2%, the location was nearly always detected.

The maximum log-likelihood values that were used to make the decisions about the locations are shown in Figure 48. These values give a measure of the relative evidence collected for each location decision. This figure shows that the leaks with 1 and 2% magnitudes, which are the smallest leak sizes looked at in this study, had significantly lower log-likelihood values than for the larger magnitudes. This occurred because the changes in signals for the smaller leaks were so small that the detected locations were more a function of noise than a detectable change in the physics.

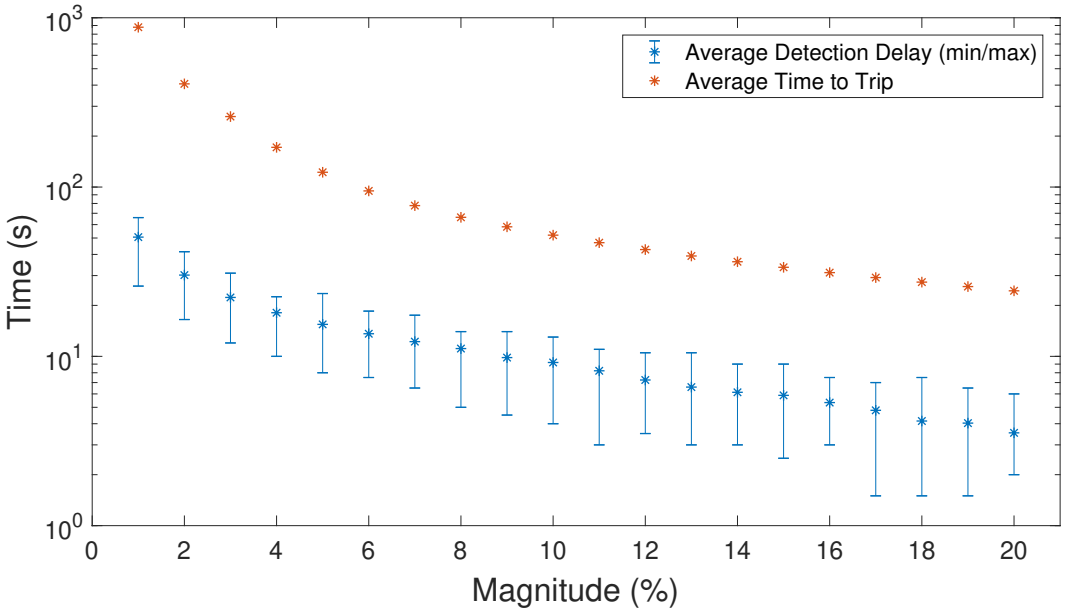


Figure 47: A plot of the average, minimum, and maximum detection delays compared to times for the reactor to trip as a function of fault magnitude.

Table 9: Percentage of correctly identified leak locations as a function of the leak magnitude.

Magnitude (%)	Total correct location (%)
1	3.3
2	46.7
3-20	99.8

To address this problem, we have included a cutoff value for the log-likelihood, also shown in Figure 48. If the log-likelihood is below this cutoff, the algorithm will determine that there is insufficient evidence to detect the location. This will reduce the number of incorrectly identified locations.

C.6 Conclusion

To ensure safe operation, NPP operators are required to constantly monitor a large number of process variables. In addition, they need to be ready to rapidly diagnose and respond to accidents. Having accident alerts and diagnostic information automatically provided can help them focus on the response. This paper presented online monitoring tools for detection of LOCAs and identification of the leak locations.

To accomplish these tasks, data-driven methods were developed using process variables. First, the set of measurable variables in a NPP was reduced to two sets capable of detecting leaks and identifying their locations. Then, these variable sets were used to train kernel-density estimates that provided likelihood information at each time step. Finally, these likelihood values were used with Bayesian hypothesis testing and maximum likelihood estimation to detect the LOCAs and identify the leak locations.

Looking at the results of this research, the LOCAs were detected for all scenarios tested with an average detection delay of one-seventh the time for the reactor to trip. In addition,

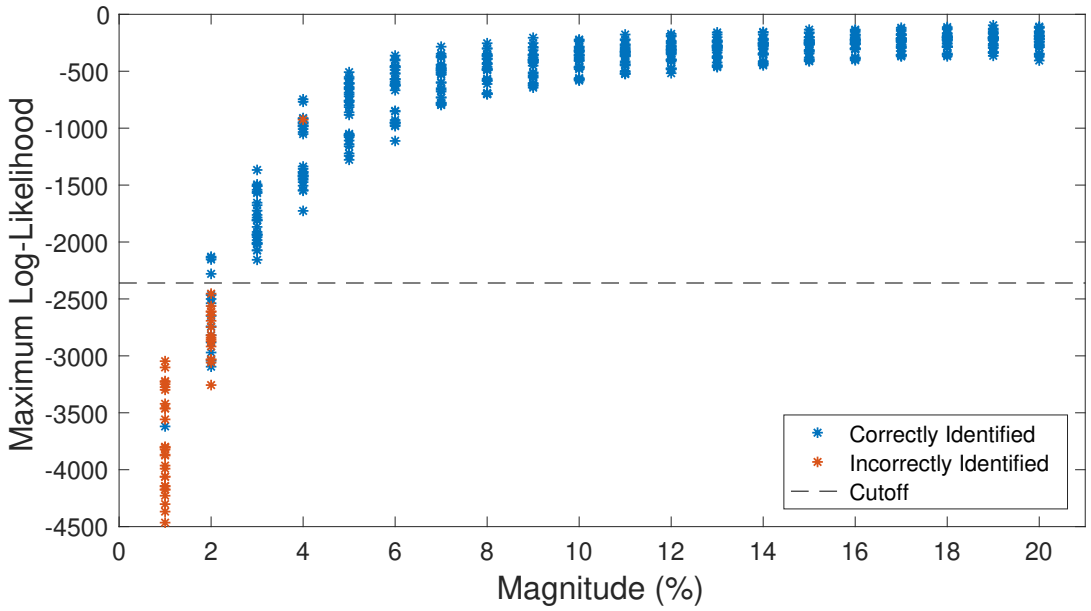


Figure 48: A plot of the maximum log-likelihood values for each scenario as a function of the leak magnitude.

the leak locations were correctly identified for 92.3% of the scenarios tested, with higher success rates for larger leaks.

These detections times should be compared against some criteria. If the monitoring tool detected a LOCA, it would trip the plant. Compared with waiting until the control system tripped the plant, tripping the plant early would reduce the plant transient and could prevent the plant from reaching unsafe conditions before the safety systems started working. Therefore, the primary criteria for detection time is faster than the control system would trip the plant, which these techniques accomplish by a factor of seven.

It is also important to consider the detection accuracy. These methods were able to accurately detect the LOCAs, reducing the burden on operators. The importance of this can be seen by examining the Three Mile Island accident, where operators essentially misdiagnosed a LOCA. During that accident, the leak rate was estimated at $1270\,\mathrm{kg\,min^{-1}}$ [64], which corresponds to a magnitude of roughly 2% in this paper. At that leak rate, these methods could have detected the LOCA in around 30 seconds. Automated techniques would have removed the human element and reduced the severity of that accident.

Appendix D Detecting Loss-of-Coolant Accidents Without Accident-Specific Data

D.1 Introduction

Detecting loss-of-coolant accidents (LOCAs) is a safety-critical task for nuclear power plant operators. For medium to large LOCAs, operators may be able to do this manually without considerable effort. However, it is more challenging for very small LOCAs that result in minimal changes to process measurements. Without a proper fault diagnosis, operators would have difficulty implementing appropriate responses to ensure safe plant shutdown. This is exactly what happened during the Three Mile Island accident, when an undiagnosed small LOCA resulted in a partial reactor meltdown. In order to aid operators, this paper develops an automated fault detection tool to detect these very small LOCAs in pressurized water reactors.

Often, fault detection techniques are separated into data-driven and physics-based categories, where the goal of both is to differentiate between normal and faulted operating conditions. Data-driven methods rely on information contained in large quantities of historical data, whereas physics-based methods use first-principles to develop system models. This paper will focus on data-driven techniques for fault detection.

There have been several previous works that used data-driven methods to detect LOCAs and other leaks. The problem has been solved using kernel density estimation [39], artificial neural networks [41, 65, 66, 67, 68], and support vector machines [13]. Each of these data-driven methods used simulator-generated data from both nominal and faulted operating conditions to achieve their goals. The challenge with the above methods is that real operational LOCA data is difficult to obtain. Thus, the various detectors cannot be trained using operational data. In order to implement them, the detectors would almost certainly have to be trained using plant-specific simulator data. This would remove one of the primary advantages of data-driven methods, which is that they estimate and make decisions based on operational data.

As an alternative, this work develops data-driven methods that could be implemented using only real operating data. It does this using a physics-inspired approach that equates the physical effects of a LOCA to changes in known variables. This means that the methods can be trained using nominal operating data, but are still able to detect LOCAs. In this work, we exclusively use simulated data; however, we expect the nominal data that we use to be consistent with real data that is available to plants, such that it could be implemented using real data.

Our approach combines data-driven modeling and control theoretic estimation techniques. The model captures the physics of relevant plant systems to be able to predict dynamic system behavior. It is created using artificial neural network regression and is trained using only nominal operating data. The estimation technique uses the data-driven model to estimate the system behavior in the presence of uncertainty. This enables it to provide real-time LOCA detection and to estimate the unknown leak magnitude. The estimation is implemented using particle filtering techniques, which are nonlinear estimators. The methods are verified using LOCA data.

This chapter is structured as follows. Section 2 describes the system modeling, including the model input and output variables, the process data, and the model structure and training. Section 3 describes the state-estimation process and how it is used to detect LOCAs. Section 4 describes the results of the methods applied to detecting LOCA simulations. Finally, Section 5 concludes the chapter.

D.2 System Modeling

In order to detect LOCAs, we developed a data-driven predictive model that captured the physics of the relevant systems. This model required three things: input and output variables, process data, and the regression model structure. Once the input and output variables were determined, process data was collected and used to train the system model based on the selected regression model structure.

D.2.1 Model Input and Output Variables

The model variables can be split into outputs and inputs, where outputs are the variables being modeled to detect LOCAs, and inputs are the variables used to predict the time-varying behavior of the outputs. Additional details on physics-based modeling of the pressurizer can be found in [22].

The output variables were selected based on what physical processes should show the largest response to LOCAs. These variables were determined to be the pressure and liquid level in the pressurizer system, denoted p and L, respectively. They were chosen because LOCAs result in loss of coolant inventory, which will reduce the mass in the primary loop. The pressure and liquid level are good proxies for the mass because the pressurizer has the highest elevation in the primary loop; therefore, changes in the primary loop mass will first appear in the pressurizer.

The input variables were selected as the variables that affect pressurizer mass and energy; see Figure 49 for a schematic of the pressurizer. These variables are

$$\begin{aligned}
\dot{m}_{SL} &= \text{surge-line mass-flow-rate} \\
T_{SL} &= \text{surge-line temperature} \\
\dot{m}_{SP} &= \text{spray mass-flow-rate} \\
T_{SP} &= \text{spray temperature} \\
\dot{Q}_H &= \text{heater power}
\end{aligned}$$

These variables also capture the saturated two-phase condition in the pressurizer, which includes both liquid and vapor phases.

The problem with the above list of inputs is that the surge-line mass-flow-rate is not directly measured. Rather, it is a combination of (i) the net mass-flow-rate to and from the chemical and volume control system (CVCS), and (ii) the change in density of the primary loop from changing temperature. The net mass-flow-rate from the CVCS is measured as the difference between the charging and letdown flows, denoted $\dot{m}_{CV}$. The net mass-flow rate from the change in primary loop density can be estimated using other properties.

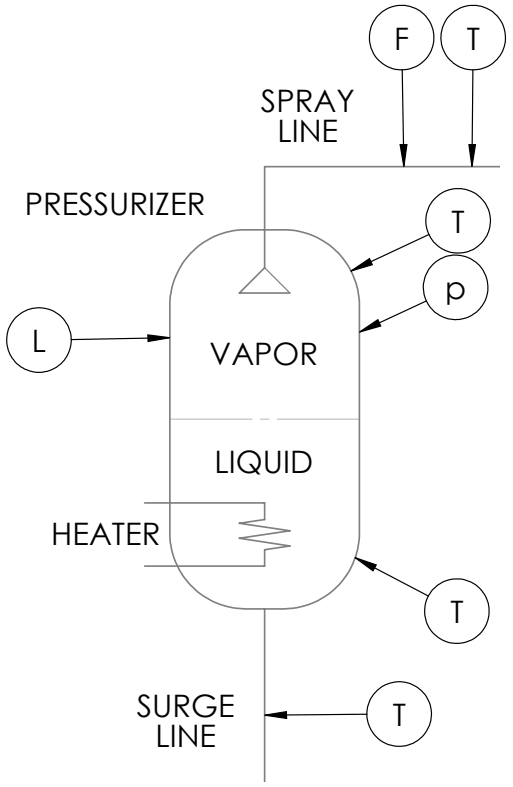


Figure 49: Schematic of the inputs to the pressurizer system. These inputs are all the variables that affect the pressurizer mass and energy, so are used in the model.

The change in primary loop density can be written analytically using the chain rule

$$\dot{\rho} = \frac{\partial \rho}{\partial p}\dot{p} + \frac{\partial \rho}{\partial T}\dot{T} \tag{D.1}$$

where ρ is density, p is pressure, T is temperature, and the dot notation signifies $\frac{d}{dt}$. This is subcooled liquid, which is often approximated as an incompressible fluid. This means the change in density is approximately proportional to $\dot{T}$, with additional coefficients determined from the data. This was done for the average hot-leg and average cold-leg temperatures separately, denoted $\dot{T}_H$ and $\dot{T}_C$, because the coefficients change significantly over that amount of temperature variation.

There is one additional input that is beneficial under certain operating conditions. Generally, the pressurizer is a saturated system, so the temperatures of both the vapor and liquid portions are related to the pressure. However, during large in-surges, the liquid temperature can fall below the saturated temperature due to the large influx of colder liquid entering the pressurizer. Therefore, the pressurizer liquid temperature, denoted T_L, has also been included as an input.

The final model variables are a set of inputs used to predict outputs. The inputs are $\dot{m}_{CV}$, $\dot{T}_H$, $\dot{T}_C$, T_{SU}, $\dot{m}_{SP}$, T_{SP}, $\dot{Q}_H$, and T_L. The outputs are p and L.

As mentioned in the introduction, this selection of variables enabled us to detect LOCAs without using LOCA data. A LOCA has a comparable effect on mass and energy to a net mass-flow out of the primary side. This means that LOCA-similar data can be captured when there is a net mass-flow out through the CVCS system. Then during implementation of the model, the net mass-flow-rate can be calculated as the sum of the rate from the CVCS and the rate from a LOCA. This can be used to predict the unknown LOCA mass-flow-rate.

D.2.2 Process Data

Simulated process data containing the relevant inputs and outputs was collected for nominal conditions and LOCA conditions. The nominal data was used for training models and other parameters, and the LOCA data was only used for final verification testing. Here, we discuss how the nominal data was collected.

All data was generated using a commercial nuclear power plant simulator developed by GSE Systems. The plant simulated is a 970 MWe pressurized water reactor with three primary loops and a data collection rate of 2 Hz. The simulator uses the RETACT thermal hydraulics package, which can model non-homogeneous and non-equilibrium conditions. In addition, it has a model accuracy that is compliant with ANS-3.5, which is the American Nuclear Society's standard for "Nuclear Power Plant Simulators for Use in Operator Training and Examination" [21].

The nominal data needed to contain unfaulted operating transients to be able to capture system dynamics. This was achieved by varying the reference signals to the pressure, level, and rod control systems. The pressure control system uses the heaters and spray to control the pressure, the level control system uses the CVCS mass-flow-rate to control the level, and the rod control system uses control-rod position to control the primary loop temperature. This process was selected in order to artificially capture data in a shorter period of time that might normally occur over an extended period of time.

The varying reference signals were selected using a random number generator in order to collect multiple datasets. In total, twenty datasets were collected, each containing ten hours of simulated data; see Figure 50 for an example of one dataset. The datasets were split into three groups: (i) eighteen datasets were used to train the neural network model, (ii) one dataset was used to estimate particle filter parameters, and (iii) one dataset was used to estimate appropriate decision thresholds. This was done to reduce biases that can occur from using the same data for multiple training purposes.

To make the data more representative of real plant data, additive Gaussian white noise was added to the input and output values. The standard deviations of these noise distributions were selected as 0.1 % of the full sensor range, as defined by the simulator. These values are shown in Table 10.

D.2.3 System Model

The goal is to develop a model that captures the physics of the pressurizer without requiring thermodynamic models. We do this by training an artificial neural network (ANN)

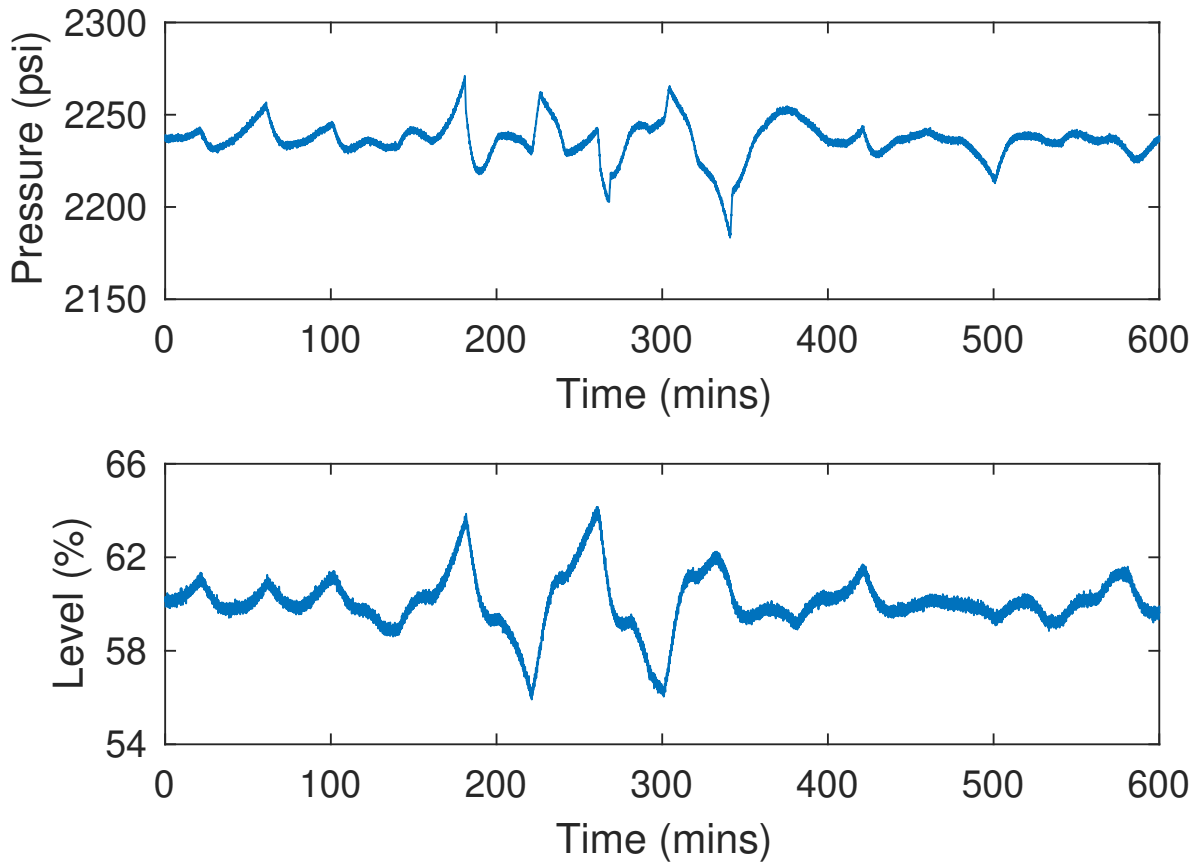


Figure 50: An example of the pressure and level measurements for one dataset, including Gaussian white noise. The fluctuations result from two phenomena: (i) the signals diverge from their nominal values due to the altered reference signals being sent to the controllers, and (ii) they return to their nominal values once the nominal reference signals are resumed to the controllers.

Table 10: List of the noise standard deviations, σ, for the input and output sensors.

Sensors	σ
p	$0.7\,\text{psi}$
L	$0.1\,\%$
$\dot{m}_{SP}$	$0.7\,\text{gpm}$
$\dot{m}_{CV}$	$0.2\,\text{gpm}$
T_H, T_C	$0.12\,^\circ\text{F}$
T_{SU}, T_{SP}, T_L	$0.6\,^\circ\text{F}$
$\dot{Q}_H$	$0.3\,\text{kW}$

to model the nonlinear state update equation. Here, we discuss the model and the training process.

Model Structure We described the pressurizer dynamics using a nonlinear state-space model

$$\begin{aligned} x_{k+1} &= f(x_k, u_k) + w_k \\ y_k &= h(x_k) + v_k \end{aligned} \tag{D.2}$$

where $x \in \mathbb{R}^n$ is the state vector, $u \in \mathbb{R}^p$ is the input vector, $y \in \mathbb{R}^r$ is the output vector, w is process noise, v is measurement noise, and k indexes time. Advantages to this system description are that it (i) is inherently auto-regressive, and (ii) can be directly implemented in the particle filtering algorithms described in the next section. For additional details on nonlinear state-space systems, refer to [27]. In a state-space model, the states are variables that completely describe the state of the system at a point in time. To simplify the model, the states were chosen as the pressure and level. This choice of state vector meant that the output function was equal to the identity function, $h(x_k) = x_k$. The inputs were described in Section D.2.1. This left the state function, $f(x_k, u_k)$, to be calculated using the data.

In order to calculate the state function, we needed direct access to state data; however, this data was corrupted by measurement noise. This was resolved by pre-filtering the data

and approximating the states by the filtered output data. The input data was also pre-filtered to improve model accuracy. The filtering techniques used were zero-phase low-pass filters, which filter the data forward and backward in order to remove any phase-delay. The filter time constants were chosen for each signal individually based on their frequency content. After filtering, the data was also down-sampled to 0.2 Hz because the signal-to-noise ratios were too large.

The state function was modeled using ANN regression, which is a data-driven nonlinear regression technique that maps inputs to outputs. An ANN contains an input layer, one or more hidden layers, and an output layer. In the standard feedforward network, data goes from the input layer, through each hidden layer sequentially, and then to the output layer. The input layer contains nodes for each input, the output layer contains nodes for each output, and the hidden layers contain a user-defined number of intermediate nodes, called neurons, which add degrees of freedom. For additional details on ANNs beyond what is discussed in this section, refer to [69, 70]. It has been shown that an ANN with a single hidden layer and sufficient neurons can uniformly approximate any continuous function [71]; therefore, an ANN with just a single hidden layer was used to capture the state function.

The output of each layer after the input layer depends on a set of weights and an activation function. These intermediary outputs are calculated as the weighted sum of that layer's inputs and then run through an activation function. For the hidden layer, the activation function is the hyperbolic tangent function, and for the output layer, the activation function is simply the linear output. The final result applied to our problem is a regression model that maps state and input vectors to state function

$$f(x_k, u_k) = W_o \sigma \left(W_i \begin{bmatrix} x_k \\ u_k \end{bmatrix} + b_i \right) + b_o \tag{D.3}$$

where $W_o \in \mathbb{R}^{n \times q}$ is the output weight matrix, $b_o \in \mathbb{R}^n$ is the output bias vector, $W_i \in \mathbb{R}^{q \times (n+p)}$ is the input weight matrix, $b_i \in \mathbb{R}^q$ is the input bias vector, q is the number of neurons, and σ is the activation function and is calculated element-wise. For training purposes, the weights and biases together are called the weights. The weights and number of neurons are calculated during the training process based on the collected process data.

Training For training, we divided the data into training and test sets. The split of training and test data was approximately 85% and 15%, respectively. The training set was used to find optimal weight values, and the test set was used to evaluate the performance of networks with different numbers of neurons. In addition, we used Bayesian regularization to prevent overfitting.

The training performance function measured both predictive accuracy and overfit, with the measure of overfit included as part of the regularization step. Predictive accuracy was measured as the sum of squares error, and overfit was measured as the sum of squares of the weights. The resulting optimization problem is

$$\min_{\mathbf{W}} \quad \alpha \sum_{k=0}^{N_T-1} (x_{k+1} - f(x_k, u_k))^2 + \beta \sum_{l=1}^{N_W} \mathbf{W}_l^2 \tag{D.4}$$

where $\mathbf{W}$ is a vector containing all the weights, N_T is the training set size, N_W is the total number of weights, and α and β are regularization parameters for the performance function. In its most basic form, the regularization parameters are user-defined; however, this adds additional parameters that the user needs to determine.

Bayesian regularization automatically calculates optimal regularization parameters using a Bayesian framework. A full discussion of this regularization technique is outside the scope of this paper; refer to [72, 73] for details. In short, Bayesian regularization calculates the regularization parameters that correspond to the maximum posterior probabilities given the data and given uniform priors on the unknown regularization parameters.

This optimization problem was solved using the Levenberg Marquardt algorithm [74], which is a steepest-descent optimization algorithm that is commonly used for training neural networks. Both the Levenberg-Marquardt algorithm and Bayesian regularization were implemented using Matlab's `trainbr` training function, which is part of their Deep Learning Toolbox [75].

In order to select the number of neurons, we trained networks with a range of neuron values and compared them based on their test set performance. That performance was measured using the mean absolute scaled error (MASE). This metric is a measure of accuracy

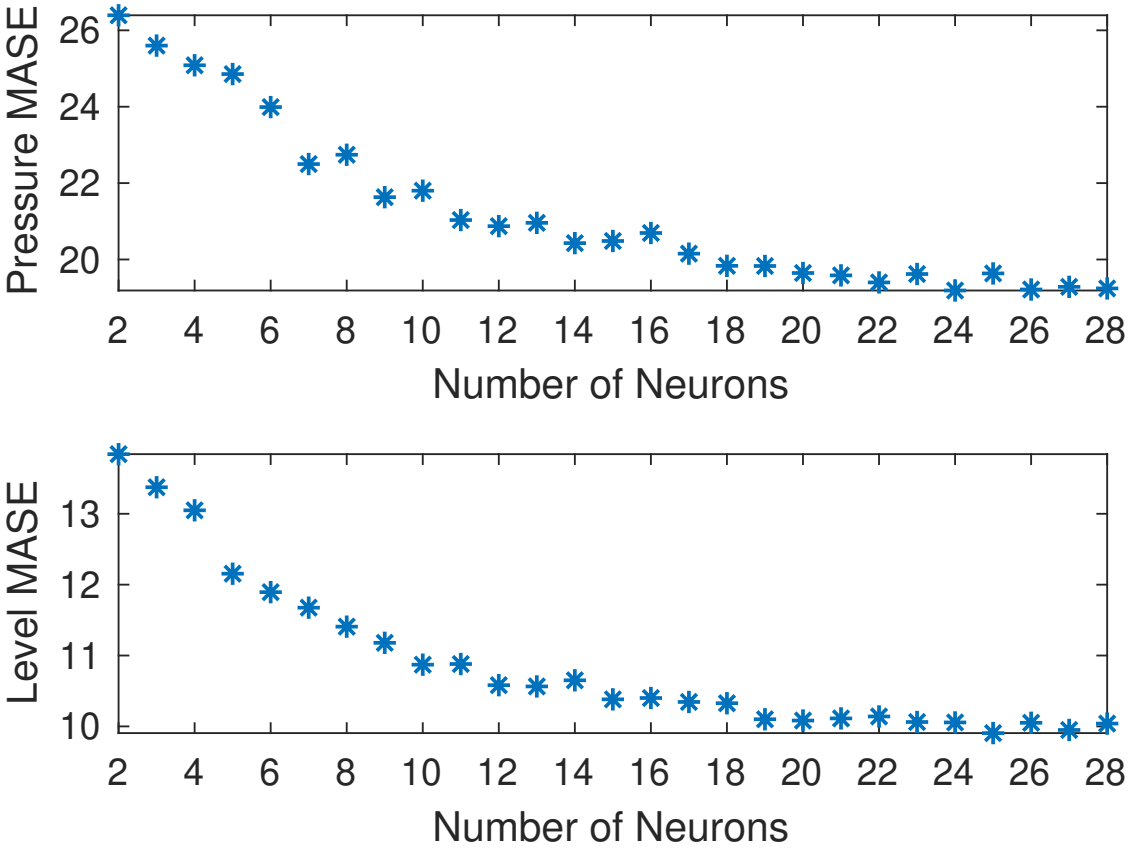


Figure 51: Plot of the test set performance as a function of the number of neurons. The performance converges as the regularization process prevents the ANN from overfitting the data.

used for time-series data that (i) is independent of the scale of the data, (ii) and is robust to zero-value measurements or constant measurements [76]. It is calculated element-wise as

$$\text{MASE} = \frac{\frac{1}{N}\sum_{k=0}^{N-1}|x_{k+1} - f(x_k, u_k)|}{\frac{1}{N}\sum_{k=0}^{N-1}|x_{k+1} - x_k|} \tag{D.5}$$

where the numerator is the mean absolute estimation error, the denominator is the mean absolute naive estimation error, the naive model is defined as $x_{k+1} = x_k$, and the $\frac{1}{N}$ terms are shown despite cancelling to emphasize they are mean values. The results for the range of neuron values are shown in Figure 51. This plot shows the successful regularization step as the test set performance levels off with increasing neurons. The final network chosen contained 28 neurons, with MASE values of 19.2 % and 10.0 % for the pressure and level, respectively. This model proved accurate enough to be able to detect LOCAs and estimate their magnitudes; therefore, no further improvement was needed.

The final result of the training process was a fully defined ANN regression model. This model approximated the unknown state function in the state-space model, and the full state-space model was then used for state estimation to detect LOCAs.

D.3 State Estimation

Using this state-space model, we detected LOCAs using state-estimation techniques. State estimation is the process of estimating the state based on a sequence of input and measurement data. These techniques require accurate system models to estimate the state, which is why the previous section focused on developing the model. This section discusses the state-estimation technique used to estimate the state and the leak model used to detect LOCAs.

D.3.1 Particle Filters

Particle filters are nonlinear state-estimation techniques that estimate the state using sequential Monte Carlo algorithms. This means they are implemented as a set of sample

states, called particles, that are propagated using the system model and sampled noise. These particles are then weighted using collected measurement data. The resulting particles and weights can be used to estimate a posterior distribution of the state at each point in time, $\hat{p}(x_k|Y_k)$, where $Y_k = \{y_0, \ldots, y_k\}$. Here, we provide a high-level overview of the algorithm; for additional details on the particle filtering algorithm used, refer to [77].

The first step in the algorithm is to propagate the set of particles using the system model and sampled noise. The system model was determined in Section D.2.3, but the noise term was not defined. We assumed it was Gaussian process noise, and its covariance matrix is discussed later in this section. This step can be expressed using our state-space model

$$\begin{aligned} \hat{x}_{k+1}^{(i)} &= f(\hat{x}_k^{(i)}, u_k) + w_k^{(i)} \\ \hat{y}_k^{(i)} &= h(\hat{x}_k^{(i)}) \end{aligned} \tag{D.6}$$

where $\hat{x}_k$ is the state estimate, $w_k^{(i)}$ is a sample from the noise distribution, i indexes the particles, and this propagation step is calculated for each of the N_P particles.

The second step is to weight the particles based on the measurement data. The weights are calculated using the probability density function of the known measurement noise distribution. These are unnormalized weights, which can then be normalized so that they sum to unity. This step can be summarized as

$$\begin{aligned} \phi_k^{(i)} &= p(y_k|\hat{x}_k^{(i)}) \\ &= \mathcal{N}(y_k : \hat{y}_k^{(i)}, R) \end{aligned} \tag{D.7}$$

$$\Phi_k^{(i)} = \frac{\phi_k^{(i)}}{\sum_{i=1}^{N_P} \phi_k^{(i)}} \tag{D.8}$$

where $\phi_k^{(i)}$ is the unnormalized weight, $\Phi_k^{(i)}$ is the normalized weight, and $\mathcal{N}(y_k : \hat{y}_k^{(i)}, R)$ is the likelihood of the sample vector y_k for the Gaussian distribution with mean vector $\hat{y}_k^{(i)}$ and meausurement covariance matrix R.

Algorithm 4 Particle filtering algorithm.

for each particle $i = 1, \ldots, N_P$ **do**

$\hat{x}_0^{(i)} \Leftarrow p(x_0)$

$\Phi_0^{(i)} = \frac{1}{N_P}$

end for

for each time step $k \geq 1$ **do**

for each particle $i = 1, \ldots, N_P$ **do**

$\hat{x}_k^{(i)} = f(\hat{x}_{k-1}^{(i)}, u_{k-1}) + w_{k-1}^{(i)}$

$\hat{y}_k^{(i)} = h(\hat{x}_k^{(i)})$

$\phi_k^{(i)} = p(y_k|\hat{x}_k^{(i)})$

end for

$\Phi_k^{(i)} = \frac{\phi_k^{(i)}}{\sum_{i=1}^{N_P} \phi_k^{(i)}}$

$\hat{p}(x_k|Y_k) \approx \sum_{i=1}^{N_P} \Phi_k^{(i)} \delta(x_k - \hat{x}_k^{(i)})$

$E[x_k|Y_k] = \sum_{i=1}^{N_P} \Phi_k^{(i)} \hat{x}_k^{(i)}$

for each particle $i = 1, \ldots, N_P$ **do**

$\hat{x}_k^{(i)} \Leftarrow \hat{P}(x_k|Y_k)$

$\Phi_k^{(i)} = \frac{1}{N_P}$

end for

end for

Using the combination of state values and weights, the set of particles can then be used to calculate the expected value of the state. This is the weighted sum of the individual particle state values

$$E[x_k|Y_k] = \sum_{i=1}^{N_P} \Phi_k^{(i)} \hat{x}_k^{(i)} \tag{D.9}$$

This expected value is often used for any decisions that involve the state values.

The final step is to resample the particles. Resampling duplicates particles of high weight and removes particles of low weight. Particles are resampled with a probability that is proportional to their weight. Without resampling, the majority of the weight would eventually be held by only a few particles, resulting in poor filter performance.

The particle filtering algorithm is shown in Algorithm 4. In this algorithm, $\Leftarrow$ means that the variable to the left of the arrow is a random sample from the distribution to the right.

Finally, we calculated the process noise covariance used in the algorithm. The value was chosen so that an empirical residual distribution would approximate its theoretical statistical distribution. The residual is calculated element-wise using [78]

$$\epsilon_k = \frac{1}{N_P} \sum_{i=1}^{N_P} F(y_k | \hat{x}_k^{(i)}) \tag{D.10}$$

where ϵ_k is the residual value, and F is the cumulative distribution function (CDF). The residual's theoretical distribution is the standard uniform distribution. To make the theoretical distribution easier to work with, we transformed the residual using the inverse standard Gaussian CDF; this transforms the uniform distribution into the standard Gaussian distribution. The problem becomes selecting a process noise covariance that makes the transformed residual approximate the standard Gaussian distribution.

This problem was solved using optimization techniques to minimize the error between the empirical residual distribution and the standard Gaussian distribution. The objective function was the Anderson-Darling test statistic [79], resulting in the optimization problem

$$\min_{Q} \quad N \int_{-\infty}^{\infty} \frac{(\hat{F}(\epsilon|Q) - F(\epsilon))^2}{F(\epsilon)(1 - F(\epsilon))} dF(\epsilon) \tag{D.11}$$

where $\hat{F}(\epsilon|Q)$ is the empirical CDF for a given process noise covariance Q, $F(\epsilon)$ is the theoretical CDF, and N is the number of samples.

The result is a fully defined particle filter that can be used for nonlinear state estimation. However with the current model, it can only estimate the state containing pressure and level.

D.3.2 Leak Model

In order to detect LOCAs, we augmented the state vector to include the leak magnitude and augmented the state-space model to include a leak model. With the model and this additional state, the particle filter could then estimate the leak magnitude in real-time. The challenge with determining the leak model is that without knowledge of the exact failure mechanism, it would be nearly impossible to come up with either a physics-based or data-driven model.

This challenge was overcome by designing a leak model that could handle arbitrary changes in leak magnitude. This meant it needed to account for both slow and abrupt changes in the leak magnitude. These two speeds were handled by different propagation mechanisms.

For handling slow changes or small jumps in the leak magnitude, a Gaussian random walk model was used [80]. This model can be expressed as

$$\mathcal{L}_{k+1} = \mathcal{L}_k + w_k \tag{D.12}$$

where $\mathcal{L}$ is the leak magnitude. The added noise term perturbs the estimate, allowing it to change over time; without it, the estimate would converge to a point estimate. The larger the noise standard deviation, the faster the estimate can change, but also the more sensitive to noise it becomes. This value was chosen experimentally as 0.1 gpm using nominal operating data.

For handling larger jumps in the leak magnitude, we monitored the particle filter for biased estimates. If the leak magnitude was far from the estimated value, the filter estimates would become biased. When this occurred, the particle filter was re-initialized, and the leak magnitude sample states were drawn from a uniform distribution. After a few time steps, the filter estimates converged back to a smaller set, enabling the re-initialization process to detect larger jumps in the magnitude.

It is important to note that this re-initialization process will occur during normal operation. This is a common challenge in any statistical testing and is often called a false alarm. However, this case does not represent a false alarm because no LOCA decision has been

reached. Therefore, it is better for this to happen even when there is no leak rather than to not happen fast enough when a leak occurs.

In order to determine if the estimates were biased, we needed to be able to measure bias. This was approximated using the auto-correlation of a sequence of residual values. Each sequence was two minutes, and the auto-correlation of lag one time-step was calculated element-wise as

$$\rho_k = \frac{1}{N_\rho} \frac{\sum_{k=1}^{N_\rho} (\epsilon_k - \mu)(\epsilon_{k-1} - \mu)}{\sigma^2} \tag{D.13}$$

where ρ is the auto-correlation coefficient, $\mu = 0$ is the theoretical mean of the residual, $\sigma^2 = 1$ is the theoretical variance of the residual, $N_\rho = 24$ is the number of time-steps in two minutes, and this calculation is done element-wise. In addition, we did the following: to ensure only non-negative values, the squared auto-correlation was used; and to calculate a single value, the maximum value of the residual vector was used. The final result was a single value at each time step that approximated the amount of bias in the past sequence of residual values.

This measure of bias was used to re-initialize the filter whenever the squared auto-correlation coefficient exceeded a threshold. The threshold was calculated experimentally as the upper limit of the values seen for nominal data. The threshold value was determined to be 0.8.

These two leak model propagation methods worked together to detect LOCAs. At each time step, the random walk model was implemented in the particle filtering algorithm. Before moving to the next time step, the auto-correlation coefficient was calculated and compared to the threshold. Whenever the value exceeded the threshold, the particle filter was re-initialized, and then returned to using the random walk model. These two methods are shown implemented in Section D.4.

The final result of this process was a state-estimation algorithm that used the data-driven ANN model and leak model to estimate the leak magnitude. This algorithm was then used for real-time LOCA detection.

D.4 Results

The methods were developed and trained using simulated nominal operating data. Here, we show the verification results from implementing the methods on simulated LOCA scenarios. In particular, we show how the two leak model propagation methods detect leaks of different magnitudes.

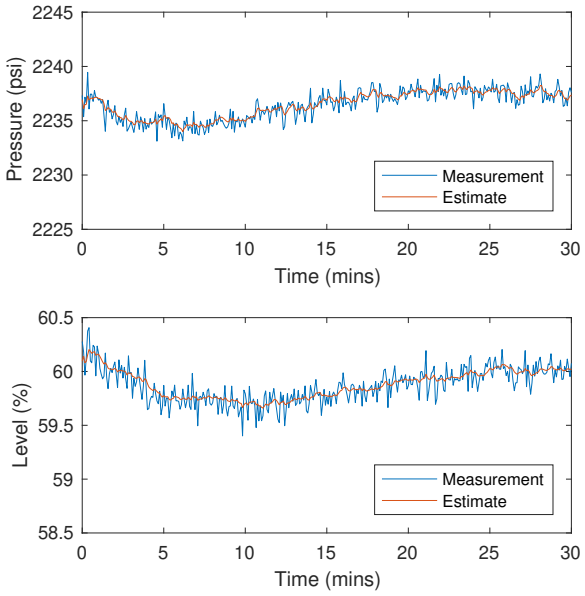
In order to discuss detection times, we needed to determine a threshold for detecting a leak using the particle filter. The detection time was defined as the time between the start of the LOCA scenario and the estimate exceeding this threshold. This was calculated experimentally as the upper limit of the values seen for nominal data. The threshold value was determined to be 3 gpm.

The LOCA data was collected by running simulations of small-break LOCAs of various sizes. In the simulator, the size is specified by a percentage, where 100 % is equivalent to a full break in a 4.5-in.-diameter pipe. The sizes simulated were 0.025 %, 0.05 %, and 0.1 %, which corresponded to 5.4 gpm, 10.8 gpm, and 21.6 gpm, respectively.

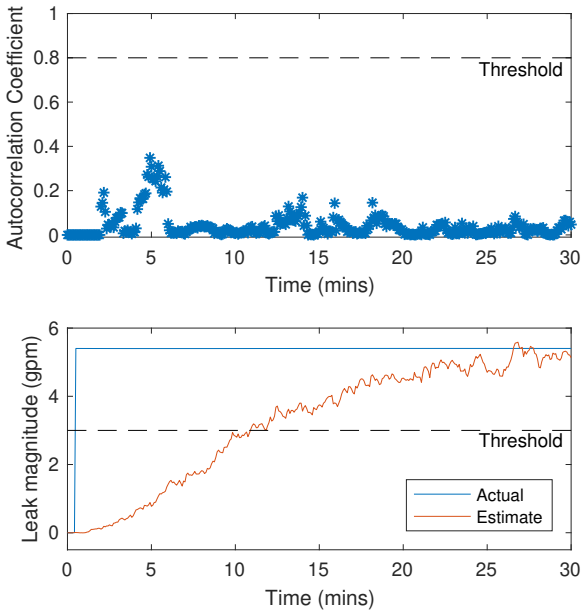
For each LOCA scenario, we provide two plots of the results. The first shows the measured outputs and the state estimates for the pressure and level. These plots are all shown with a consistent scale. The second plot shows the auto-correlation coefficient, the true leak magnitude, and the estimated magnitude. On these plots, vertical dashed lines show when the auto-correlation coefficient exceeded the threshold, causing the particle filter to be re-initialized.

The methods were first implemented on a LOCA simulation with a magnitude of 5.4 gpm; see Figure 52. For such a small LOCA, only the random walk propagation method was active because the auto-correlation coefficient never exceeded the threshold. This resulted in a slow increase in the estimated leak magnitude until it converged near the true value. Using the 3 gpm threshold, it took roughly 10 minutes until an operator would have been alerted. This corresponded to a 2.4 psi and 0.4 % drop in pressure and level, respectively, giving the operator plenty of time to react.

Next, the methods were implemented on a LOCA simulation with a magnitude of 10.8 gpm; see Figure 53. This LOCA was slightly larger than the previous scenario, which

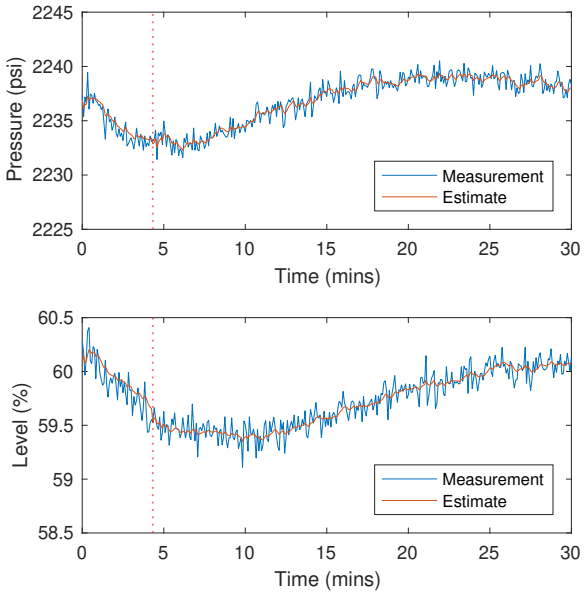


(a)

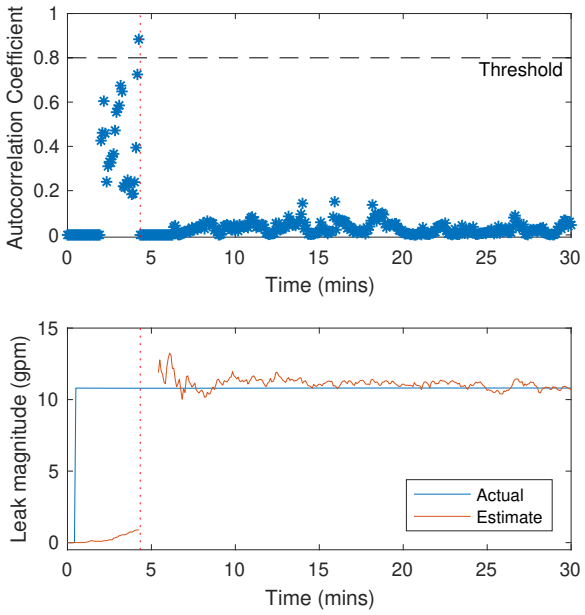


(b)

Figure 52: Results for the 5.4 gpm LOCA scenario. Figure 52a shows the pressure, level, and particle filter estimates. Figure 52b shows the auto-correlation coefficient and leak magnitude.



(a)



(b)

Figure 53: Results for the 10.8 gpm LOCA scenario. Figure 53a shows the pressure, level, and particle filter estimates. Figure 53b shows the auto-correlation coefficient and leak magnitude.

caused the auto-correlation coefficient to exceed the threshold just before 5 minutes. After the particle filter reinitialized and a short delay, the estimate converged near the true value. The converged value exceeded the threshold, meaning it took roughly 5 minutes until an operator would have been alerted. This corresponded to a 4.0 psi and 0.6 % drop in pressure and level, respectively. This simulation shows how the re-initialization process worked, and how it was able to more quickly detect the larger LOCA.

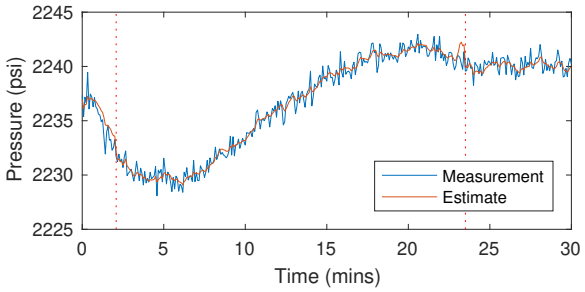
Finally, the methods were implemented on a LOCA simulation with a magnitude of 21.6 gpm; see Figure 54. This was the largest LOCA analyzed, and the auto-correlation coefficient quickly exceeded the threshold at 2 minutes. This meant it took roughly 3 minutes until an operator would have been alerted, which corresponded to a 6.6 psi and 0.7 % drop in pressure and level, respectively. These drops still would have provided an operator with plenty of time to decide how to proceed.

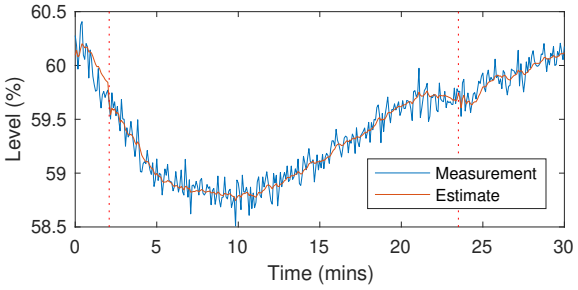
In this simulation, the filter also reinitialized around 24 minutes. This was a limitation in the collected dataset; the operating conditions reached a region in the data space that the data-driven model did not capture. However, the estimate returned after a few minutes.

The results from these LOCA simulations show that the data-driven approach was able to successfully detect these very small LOCAs; see Table 11 for a summary of the results. Two leak model propagation methods were used to detect LOCAs of varying sizes, helping improve detection time for larger LOCAs without increasing noise in the estimates. In addition, the methods provided accurate estimates of the leak magnitude, providing operators with useful diagnostic information.

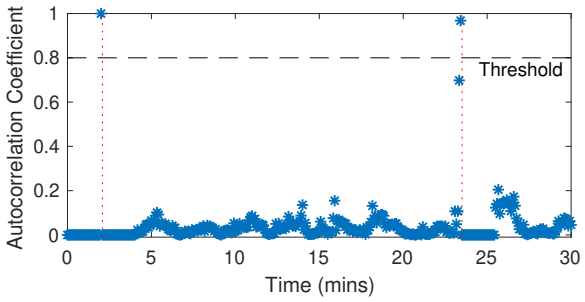
D.5 Conclusion

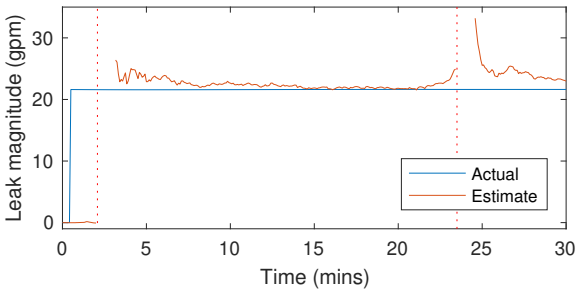
In this chapter, we developed a data-driven tool to detect very small LOCAs. This was accomplished by training an ANN regression model of the pressurizer dynamics and combining it with particle filters that detected LOCAs in real-time. All of this was done using only data captured from simulated nominal operating conditions, making it more implementable using real operational data than previous methods.





(a)





(b)

Figure 54: Results for the 21.6 gpm LOCA scenario. Figure 54a shows the pressure, level, and particle filter estimates. Figure 54b shows the auto-correlation coefficient and leak magnitude.

Table 11: Summary of the results of the methods implemented on LOCA scenarios. The pressure and level drops were calculated at the detection time.

Leak Magnitude (gpm)	Detection Time (min)	Pressure Drop (psi)	Level Drop (%)
5.4	10	2.4	0.4
10.8	5	4.0	0.6
21.6	3	6.6	0.7

The methods were tested using LOCA simulations to verify their effectiveness. The simulations had leak magnitudes of 5.4 gpm, 10.8 gpm, and 21.6 gpm, and were detected in 10 min, 5 min, and 3 min, respectively. Based on these results, the methods detected each case well before an operator would need to react and provided accurate estimates to aid in operator response.

The primary limitation to the methods developed here is that they rely on comparable data from the CVCS system. Specifically, they require transient data for the mass-flow-rate through the CVCS system, which limits the maximum LOCA size these methods could detect. In order to detect larger LOCAs, we would either need data from those LOCA scenarios, or would need to combine these methods with physics-based models that would be valid for a larger range of operating conditions. However, these larger LOCAs would be easier for operators to detect, making them less critical to be detected automatically.

As a final comment, we suggest that the use of physics-inspired variables is a powerful way of extending the use of data-driven methods. By considering variables in terms of their effects on mass and energy, variables can be more readily compared and used for prediction even when exact data does not exist. This approach can result in a balance between allowing the data to purely define the model and using expert knowledge of the physical processes.

Bibliography

[1] Y. Mo, R. Chabukswar, and B. Sinopoli. Detecting integrity attacks on scada systems. *IEEE Transactions on Control Systems Technology*, 22(4):1396–1407, July 2014.

[2] Lee T. Maccarone, Christopher J. D'Angelo, and Daniel G. Cole. Uncovering cyber-threats to nuclear system sensing and observability. *Nuclear Engineering and Design*, 331:204 – 210, 2018.

[3] A. Teixeira, I. Shames, H. Sandberg, and K. H. Johansson. Revealing stealthy attacks in control systems. In *2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1806–1813, Oct 2012.

[4] Mohammad Naghnaeian, Nabil H. Hirzallah, and Petros G. Voulgaris. Security via multirate control in cyber-physical systems. *Systems & Control Letters*, 124:12 – 18, 2019.

[5] B. Satchidanandan and P. R. Kumar. Dynamic watermarking: Active defense of networked cyber–physical systems. *Proceedings of the IEEE*, 105(2):219–240, 2017.

[6] Chongrong Fang, Yifei Qi, Peng Cheng, and Wei Xing Zheng. Optimal periodic watermarking schedule for replay attack detection in cyber–physical systems. *Automatica*, 112:108698, 2020.

[7] R. Langner. Stuxnet: Dissecting a cyberwarfare weapon. *IEEE Computer and Reliability Societies*, pages 49–51, May 2011.

[8] D. Albright, P. Brannan, and C. Walrond. Did stuxnet take out 1,000 centrifuges at the natanz enrichment plant? Technical report, Institute for Science and International Security, December 2010.

[9] Sigurd Skogestad and Ian Postlethwaite. *Multivariable Feedback Control: Analysis and Design*. John Wiley and Sons, Second edition, 2001.

[10] R. B. Amir, S. T. Gul, and A. Q. Khan. A comparative analysis of classical and one class svm classifiers for machine fault detection using vibration signals. In *2016 International Conference on Emerging Technologies (ICET)*, pages 1–6, 2016.

[11] B. Samanta and K.R. Al-Balushi. Artificial neural network based fault diagnostics of rolling element bearings using time-domain features. *Mechanical Systems and Signal Processing*, 17(2):317 – 328, 2003.

[12] S. H. Lee, Y. G. No, M. G. Na, K. Ahn, and S. Park. Diagnostics of loss of coolant accidents using svc and gmdh models. *IEEE Transactions on Nuclear Science*, 58(1):267–276, 2011.

[13] K. H. Yoo, Y. D. Koo, J. H. Back, and M. G. Na. Identification of loca and estimation of its break size by multiconnected support vector machines. *IEEE Transactions on Nuclear Science*, 64(10):2610–2617, 2017.

[14] F. Pasqualetti, F. Dorfler, and F. Bullo. Attack detection and identification in cyber-physical systems. *IEEE Transactions on Automatic Control*, 58(11):2715–2729, Nov 2013.

[15] H. Fawzi, P. Tabuada, and S. Diggavi. Secure estimation and control for cyber-physical systems under adversarial attacks. *IEEE Transactions on Automatic Control*, 59(6):1454–1467, June 2014.

[16] Mohammad Majidi, Alireza Erfanian, and Hamid Khaloozadeh. A new approach to estimate true position of unmanned aerial vehicles in an ins/gps integration system in gps spoofing attack conditions. *International Journal of Automation and Computing*, 15(6):747–760, Dec 2018.

[17] J. A. Farber and D. G. Cole. Detecting loss-of-coolant accidents without accident-specific data. *Progress in Nuclear Energy*, 128(103469), 2020.

[18] R.M. Singer, K.C. Gross, J.P. Herzog, R.W. King, and S. Wegerich. Model-based nuclear power plant monitoring and fault detection: Theoretical foundations. In *International Conference on Intelligent Systems Applications to Power Systems*, 6 1997.

[19] N. Zavaljevski and K. C. Gross. Sensor fault detection in nuclear power plants using multivariate state estimation technique and support vector machines. In *Third International Conference of the Yugoslav Nuclear Society*, 10 2000.

[20] A. Hoehn and P. Zhang. Detection of covert attacks and zero dynamics attacks in cyber-physical systems. In *2016 American Control Conference (ACC)*, pages 302–307, 2016.

[21] ANSI/ANS–3.5–2009: Nuclear power plant simulators for use in operator training and examination.

[22] Neil Todreas and Mujid Kazimi. *Nuclear Systems I: Thermal Hydraulic Fundamentals*. Taylor and Francis, 1990.

[23] Revised release on the IAPWS industrial formulation 1997 for the thermodynamic properties of water and steam. Technical report, International Association for the Properties of Water and Steam, 2007.

[24] Johan Paduart, Lieve Lauwers, Jan Swevers, Kris Smolders, Johan Schoukens, and Rik Pintelon. Identification of nonlinear systems using polynomial nonlinear state space models. *Automatica*, 46(4):647 – 656, 2010.

[25] Giuseppe Basile and Giovanni Marro. *Controlled and Conditioned Invariants in Linear System Theory*. Prentice Hall Professional Technical Reference, First edition, 1991.

[26] Alberto Isidori. *Nonlinear Control Systems*. Springer, Third edition, 1995.

[27] Hassan Khalil. *Nonlinear Systems*. Prentice Hall, Third edition, 2002.

[28] Gene Golub and Charles Van Loan. *Matrix Computations*. Johns Hopkins, Third edition, 1996.

[29] Alejandro F. Villaverde, Antonio Barreiro, and Antonis Papachristodoulou. Structural identifiability analysis via extended observability and decomposition. *IFAC-PapersOnLine*, 49(26):171 – 177, 2016.

[30] R. Hermann and A. Krener. Nonlinear controllability and observability. *IEEE Transactions on Automatic Control*, 22(5):728–740, 1977.

[31] Alejandro F. Villaverde, Antonio Barreiro, and Antonis Papachristodoulou. Observability and structural identifiability of nonlinear biological systems. *Complexity - Special Issue*, 2019, 2019.

[32] A. J. Krener and K. Ide. Measures of unobservability. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 6401–6406, 2009.

[33] B. T. Hinson, M. K. Binder, and K. A. Morgansen. Path planning to optimize observability in a planar uniform flow field. In *2013 American Control Conference*, pages 1392–1399, 2013.

[34] James Demmel. The componentwise distance to the nearest singular matrix. *SIAM Journal on Matrix Analysis and Applications*, 13(1):10–19, 1992.

[35] E. A. Wan and R. Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*, pages 153–158, 2000.

[36] Eric A. Wan and Rudolph van der Merwe. *The Unscented Kalman Filter*, chapter 7, pages 221–280. John Wiley & Sons, Ltd, 2002.

[37] Wolfgang Karl Hardle and Leopold Simar. *Applied Multivariate Statistical Analysis*. Springer, Fourth edition, 2015.

[38] J. A. Farber and D. G. Cole. Using model-based fault detection to differentiate transients and loss of coolant accidents. In *11th Nuclear Plant Instrumentation, Control and Human-Machine Interface Technologies Conference, Orlando, Florida*, 2018.

[39] Jacob A. Farber, Daniel G. Cole, Ahmad Y. Al Rashdan, and Vaibhav Yadav. Using kernel density estimation to detect loss-of-coolant accidents in a pressurized water reactor. *Nuclear Technology*, 205(8):1043–1052, 2019.

[40] Tennessee Valley Authority. Browns ferry nuclear plant (bfn)—multi-unit probabilistic risk-assessment (pra). Technical report, Apr 1995.

[41] Eric B. Bartlett and Robert E. Uhrig. Nuclear power plant status diagnostics using an artificial neural network. *Nuclear Technology*, 97(3):272–281, 1992.

[42] M. G. Na, W. S. Park, and D. H. Lim. Detection and diagnostics of loss of coolant accidents using support vector machines. *IEEE Transactions on Nuclear Science*, 55(1):628–636, Feb 2008.

[43] D. Magill. Optimal adaptive estimation of sampled stochastic processes. *IEEE Transactions on Automatic Control*, 10(4):434–439, Oct 1965.

[44] J. A. Farber and D. G. Cole. Using multiple-model adaptive estimation and system identification for fault detection in nuclear power plants. In *International Mechanical Engineering Congress and Expo, Pittsburgh, PA*, 2018.

[45] Karel J. Keesman. *System Identification*. Springer London, 2011.

[46] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[47] R. Mehra and J. Peschon. An innovations approach to fault detection and diagnosis in dynamic systems. *Automatica*, 7(5):637 – 640, 1971.

[48] Brian D. O. Anderson and John B. Moore. *Optimal Filtering*. Dover Publications, 2005.

[49] E. Chow and A. Willsky. Analytical redundancy and the design of robust failure detection systems. *IEEE Transactions on Automatic Control*, 29(7):603–614, Jul 1984.

[50] P.M. 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.

[51] Rolf Isermann. Process fault detection based on modeling and estimation methods: A survey. *Automatica*, 20(4):387 – 404, 1984.

[52] Young Soo Park and Richard Vilim. Implementation of new prodiag algorithm and simulation-based acceptance test. In *11th Nuclear Power Instrumentation, Control and Human-Machine Interface Technologies Conference*, pages 884 – 893, San Francisco, CA, June 11-15, 2017, 2017.

[53] J. Reifman and T.Y.C Wei. "PRODIAG" a process-indepedent transient diagnostic system - i: Theoretical concepts. *Nuclear Science and Engineering*, 131:329 – 347, 1999.

[54] Jianping Ma and Jin Jiang. Applications of fault detection and diagnosis methods in nuclear power plants: A review. *Progress in Nuclear Energy*, 53(3):255 – 266, 2011.

[55] M. J. Desforges, P. J. Jacob, and J. E. Cooper. Applications of probability density estimation to the detection of abnormal conditions in engineering. *Journal of Mechanical Engineering Science*, 212:687 – 703, 1998.

[56] Lin Cheng and Dongjian Zheng. Two online dam safety monitoring models based on the process of extracting environmental effect. *Advances in Engineering Software*, 57:48 – 56, 2013.

[57] Kyung-Duk Kim and Jun-Haeng Heo. Comparative study of flood quantiles estimation by nonparametric models. *Journal of Hydrology*, 260(1):176 – 193, 2002.

[58] Qianqian Zhang and Qingming Gui. A new bayesian raim for multiple faults detection and exclusion in gnss. *Journal of Navigation*, 68(3):465–479, 2014.

[59] You Ling and Sankaran Mahadevan. Integration of structural health monitoring and fatigue damage prognosis. *Mechanical Systems and Signal Processing*, 28:89 – 104, 2012.

[60] E. Mazor, A. Averbuch, Y. Bar-Shalom, and J. Dayan. Interacting multiple model methods in target tracking: a survey. *IEEE Transactions on Aerospace and Electronic Systems*, 34(1):103–123, Jan 1998.

[61] Nist/sematech e-handbook of statistical methods. `https://www.itl.nist.gov/div898/handbook/pri/section3/pri3342.htm`,. Accessed: 2018-08-08.

[62] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, 1986.

[63] Matlab statistics and machine learning toolbox release 2017b. The MathWorks, Inc., Natick, Massachusetts, United States.

[64] R. O. Wooton, R. S. Denning, and P. Cybulskis. Analysis of the three mile island accident and alternative sequences. Technical report, Battelle, Columbus Laboratories, Jan 1980.

[65] Jaemin Yang and Jonghyun Kim. An accident diagnosis algorithm using long short-term memory. *Nuclear Engineering and Technology*, 50(4):582 – 588, 2018. International Symposium on Future I&C for Nuclear Power Plants (ISOFIC2017).

[66] Seung Jun Lee and Poong Hyun Seong. A dynamic neural network based accident diagnosis advisory system for nuclear power plants. *Progress in Nuclear Energy*, 46(3):268 – 281, 2005. Computational Intelligence in Nuclear Applications: Lessons Learned and Recent Developments.

[67] Kun Mo, Seung Jun Lee, and Poong Hyun Seong. A dynamic neural network aggregation model for transient diagnosis in nuclear power plants. *Progress in Nuclear Energy*, 49(3):262 – 272, 2007.

[68] T.V. Santosh, Gopika Vinod, R.K. Saraf, A.K. Ghosh, and H.S. Kushwaha. Application of artificial neural networks to nuclear power plant transient diagnosis. *Reliability Engineering & System Safety*, 92(10):1468 – 1472, 2007.

[69] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[70] Sergios Theodoridis and Konstantinos Koutroumbas. *Pattern Recognition*. Academic Press, fourth edition, 2009.

[71] K. Hornik. Some new results on neural network approximation. *Neural Networks*, 6(8):1069 – 1072, 1993.

[72] David J. MacKay. Bayesian interpolation. *Neural Computation*, 4(3):415–447, May 1992.

[73] F. Dan Foresee and M. T. Hagan. Gauss-newton approximation to bayesian learning. In *Proceedings of International Conference on Neural Networks (ICNN'97)*, volume 3, pages 1930–1935 vol.3, June 1997.

[74] Donald W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

[75] Matlab deep learning toolbox release 2017b. The MathWorks, Inc., Natick, Massachusetts, United States.

[76] Rob J. Hyndman and Anne B. Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4):679 – 688, 2006.

[77] N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEE Proceedings F - Radar and Signal Processing*, 140(2):107–113, April 1993.

[78] James V. Candy. *Bayesian Signal Processing*. Wiley, second edition, 2016.

[79] M. A. Stephens. Edf statistics for goodness of fit and some comparisons. *Journal of the American Statistical Association*, 69(347):730–737, 1974.

[80] C. Andrieu, A. Doucet, S. S. Singh, and V. B. Tadic. Particle methods for change detection, system identification, and control. *Proceedings of the IEEE*, 92(3):423–438, March 2004.