



THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

Research Commons

<http://researchcommons.waikato.ac.nz/>

Research Commons at the University of Waikato

Copyright Statement:

The digital copy of this thesis is protected by the Copyright Act 1994 (New Zealand).

The thesis may be consulted by you, provided you comply with the provisions of the Act and the following conditions of use:

- Any use you make of these documents or images must be for research or private study purposes only, and you may not make them available to any other person.
- Authors control the copyright of their thesis. You will recognise the author's right to be identified as the author of the thesis, and due acknowledgement will be made to the author where appropriate.
- You will obtain the author's permission before publishing any material from the thesis.

An Investigation of Self-Learning and Self-Protection for Adaptive Digital Twins

A thesis

submitted in fulfilment

of the requirements for the Degree

of

Master's of Science (Research)

at

The University of Waikato

by

Chris Anderson



THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

2021

Contents

1	Introduction	12
1.1	Background	14
1.1.1	Self-Adaptive Systems	16
1.1.2	Control Theory	16
1.1.3	State-Space Models	17
1.1.4	Anomaly Detection	17
1.2	Aims	18
1.3	Structure	19
2	Related Work	20
2.1	Digital Twins	20
2.1.1	Digital Twins in Critical Infrastructure	21
2.1.2	Digital Twins in Critical Infrastructure for Self-Protection	23
2.1.3	Further Definition	24
2.2	Machine Learning	25
2.2.1	Neural Networks	25
2.3	Self-Adaptive Control Systems	26
2.4	Systematic Literature Review	27
2.4.1	Methodology	27

2.4.1.1	Manual Additions	29
2.4.2	Continuous User Training	30
2.4.3	Literature Crossover	31
2.4.3.1	Honeypots, Decoys, and Deception	31
2.4.4	Key Findings	34
2.5	Research Gaps	35
2.6	Conclusion	36
3	Research Method	37
3.1	Self-Learning and Self-Protection	38
3.2	Hypotheses	45
3.3	Summary	46
4	Requirements and Experimental Design	47
4.1	Simulation	47
4.1.1	Heater	48
4.1.2	PID Controller	50
4.1.3	Forecast Oracles	51
4.1.4	Error Metrics	53
4.1.5	Degradation	54
4.2	Tested Techniques	56
4.2.1	Base-Case	56
4.2.2	Same-Trend	57
4.2.3	Observer Kalman and Eigensystem Realisation	58
4.2.4	Dynamic Mode Decomposition with Control	58

4.2.5	Sparse Identification of Nonlinear Dynamics with Control	59
4.2.6	Recurrent Neural Network	59
4.3	Conclusion	60
5	Self-Learning Evaluation and Analysis	61
5.1	Performance over Simulation	62
5.1.1	Retraining	64
5.1.2	Model Stability	70
5.2	Performance over Varying Oracle Depths	72
5.3	Conclusion	75
6	Self-Protection Evaluation and Algorithms	77
6.1	Signal Processing	78
6.1.1	Nyquist Frequency	79
6.1.2	Transforms	79
6.2	Sampling and Command Frequency	79
6.3	Heater Failure Conditions	81
6.4	Temperature Prediction Handling	82
6.5	Input Frequency Handling	84
6.6	Conclusion	88
7	Engineering Design Discussion	89
7.1	Simulation	89
7.1.1	Simulation Design Limitations	91
7.2	Overall Architectures	92
7.2.1	Self-Learning	94

7.2.2	Self-Protection	96
7.3	Software Development Process	96
7.4	Conclusions	99
8	Conclusion	101
8.1	Research Hypotheses	102
8.1.1	Hypothesis 1	102
8.1.2	Hypothesis 2	103
8.1.3	Hypothesis 3	104
8.2	Threats to Validity	105
8.2.1	Simulator	105
8.2.2	Sampling	107
8.2.3	Data Types	108
8.3	Future Work	108
8.3.1	Pole Placement	108
8.3.2	Dynamic Retraining Thresholds	109
8.3.3	Compressed Sensing	109
8.3.4	Forecasting	110
8.3.5	Non-Human-In-The-Loop	110
8.3.6	Actual Energy Systems	110
8.3.7	Prediction Testing	111

List of Tables

2.1	Non-exhaustive list of ICS Honeypots	33
4.1	Raw Heater Variables	49
4.2	PID + Heater Variables	50
4.3	Techniques and Self-Learning coverage	60
5.1	Variable Weighting	63
5.2	Error for non-retraining models	63
5.3	Temperature Error Across All Retraining Strategies	64
5.4	Temperature Error Across All Retraining Strategies with minimum time before retrain	65
5.5	Total Change in DMDc A Matrix (Three Significant Figures) .	66
5.6	Total Change in DMDc B Matrix (Three Significant Figures) .	66
5.7	Temperature Error (Five Significant Figures)	73
5.8	Water Level Error (Five Significant Figures)	74
5.9	Power Error (Five Significant Figures)	75
7.1	Summary of Modules and Implementations	99

7.2	Summary of Example Implementations of Each Module	100
8.1	Complete Temperature Error Across All Retraining Strategies with minimum time before retrain	124

List of Figures

2.1	Lightwire Address tagged as ICS Honeypot in Shodan (n.d.) .	32
2.2	Step-7 port (102) flagged as a honeypot and Modbus shows two fake S7-200s in Shodan (n.d.)	32
2.3	Simplified Gap Analysis	36
3.1	Cloud-Machine Interface	38
3.2	Proposed Self-Learning Architecture	39
3.3	Three Pole Plot for DMDc model	41
3.4	Implemented Self-Learning Diagram	42
3.5	Self-Learning Framework Class Diagram	44
4.1	Modelled Water Heater	48
4.2	Heater Diagram with PID	51
4.3	Evolution of the heater over time	52
4.4	Simulated Heater Graphs	53
4.5	Error over the first oracle's window (n=2048)	54
4.6	Numeric Explosion with DMDc when Heater degrades to 95%	55

4.7	Overshoot with Same-Trend technique	57
5.1	Signed Error with Absolute A Matrix Difference over time	67
5.2	Absolute change in A matrix values between retraining	68
5.3	Principal Component Analysis of A matrix change over time	69
5.4	Principal Component Analysis of A matrix over time	70
5.5	DMDc First Retrain Bode Plot — Magnitude	71
5.6	Error mean and deviation over the primary oracle (25 th retrain)	72
6.1	Example of an aliased signal	80
6.2	Self-Protection by modifying PID target	83
6.3	Self-Protection by modifying raw heater output	84
6.4	Phase diagram for tank temperature	85
6.5	Phase diagram for tank temperature with interference	86
6.6	Heater Level of $\hat{\mathbf{B}}$ in frequency-domain showing oscillations (left) and filtered frequencies (right)	87
7.1	Updated Architecture as per Figure 3.2	93
7.2	Self-Learning Architecture	95
7.3	Self-Protection Architecture	97
7.4	Software Development Process Diagram	98
8.1	B Matrix Change	125

8.2	Principal Component Analysis of B matrix change over time	126
8.3	Absolute change in A matrix values between retraining	127
8.4	Absolute change in A matrix values between retraining	127
8.5	Principal Component Analysis of A matrix change over time	128
8.6	Principal Component Analysis of B matrix change over time	129
8.7	Integral of Absolute Error for DMDc Retrains with the Signed strategies	130
8.8	Principal Component Analysis of B matrix over time	131
8.9	Heater error only for Octuple length run	131
8.10	DMDc First Retrain Bode Plot — Phase	132

Abstract

Adaptive Digital Twins are applicable to a number of fields, including the cybersecurity of industrial control systems. This thesis prototypes a Self-Learning adaptive digital twin and posits an architecture for the creation of digital twins based on the learnings gained from the prototype. The prototype shows the efficacy of control theoretical approaches for adaptive digital twins for both modelling and protecting a system, and the architecture posits a generalised method for developing adaptive digital twins.

Acknowledgements

I have been lucky to be surrounded by a group of excellent people whose passion for their own interests is inspirational.

I would like to extend my gratitude to both of my supervisors — Dr Panos Patros and Dr Tim Walmsley — for their support and candid advice during the thesis, to the members of the ORCA and Ahuora Labs for their friendship and for keeping me sane, to my friends in other faculties for their advice and opinions on my research, and to my family and girlfriend for their understanding, their accommodation, and for their support.

Furthermore, I'd like to thank the University of Waikato and Project Ahuora for funding this research. Project Ahuora is a seven-year collaborative project funded by the New Zealand Ministry of Business, Innovation & Employment (2020) under the Strategic Science Investment Fund. As part of the Advanced Energy Technology platform fund, Project Ahuora is one of four currently funded programmes and is a collaboration between the University of Auckland, the University of Waikato, and Massey University.

Chapter 1

Introduction

Globally, 196 governments have signed the Paris Agreement (United Nations Framework Convention on Climate Change, 2015) to limit climate change. In 2019, Aotearoa New Zealand (NZ) passed the Climate Change Response Amendment Act into law (Parliamentary Counsel Office of New Zealand, 2019). This legislation mandates net-zero carbon emissions by 2050 and meets the requirements set out under the Paris Agreement. Achieving this goal requires changes in every part of society. Industrial uses that generate large quantities of greenhouse gases often have longer lifespans than consumer uses. For example, NZ's Ministry of Business, Innovation & Employment (2019a) find that the average boiler system has an economic lifespan of 15 to 20 years, though are often used for 20 to 40 years (Ministry of Business, Innovation & Employment, 2021). Long-lived systems, such as these, take longer to switch for low-emission versions and make retrofit solutions a more immediately viable option. A problem with retrofit solutions is their security, especially when current plants were not designed with internet connectivity in mind.

New Zealand's energy sector, as at 2019, contributed close to 40% of the country's total emissions, with 28% of this portion originating from process heat — the use of energy to create hot fluids or gases produced in an Industrial

Control System (ICS) setting, often in the form of steam — (Ministry of Business, Innovation & Employment, 2019a). In total, process heat contributed nine per cent of the country’s total emissions and consumed just over a third of the total energy demand (Ministry of Business, Innovation & Employment, 2019b, 2021). With the majority of these emissions in the process heat sector stemming from boiler systems (Ministry of Business, Innovation & Employment, 2019a), a focus on them creates the biggest effect on emissions. This reduction is at the heart of the motivation of this thesis.

As stated above, many sites use steam to operate, and this steam is raised by evaporating deionised water above its boiling point (Sarco, 2018). This process’ failure states can lead to significant economic damage or the serious harm or loss of human life. Self-Protecting digital twins can foresee these failures and take actions to warn or prevent them.

Digital twins — virtual models of a physical system based on real-time sensor data — that can adapt to changes in the physical twin via Self-Learning allow for more functional twins that can take protective actions and forecast hypotheticals, such as potentially malicious changes to an industrial plant that can result in catastrophic failures. Furthermore, an adaptive digital twin promises to create a high-fidelity simulation of the system that can vet changes to the plant, generate low-noise event reports, and in extreme cases, take preventative measures. To date, few algorithms for adaptive digital twins have been proposed due in no small part to the complexity and uncertainty in their requirements. To forward these ends, this project proposes an architecture and engineering method for creating Self-Learning and Self-Protecting Digital Twins, thereby making them adaptive. Such an adaptive digital twin fulfills Weyns (2018)

Moving toward the Internet of Things (IoT) — the use of many internet-connected sensors and actuators — in industrial control provides excellent operational visibility, and when combined with Cloud computing, offers operational optimisation. Cloud computing would give plants access to exceedingly large amounts of computation resources at low cost; however, allowing ICSs access to external networks is fraught with risk (Anton, Hafner, & Schotten, 2019). There are well-founded reasons for such an inherently risky proposition to not be adopted today, but the future adoption of such technology is pivotal for mitigating the effects of climate change. Handling this risk by developing a framework that guarantees quality in the presence of uncertainty is a motivating goal of this thesis.

To reiterate, securing the future of energy systems is an open problem. In furtherance of that goal, adaptive digital twins promise to provide a system that vets inputs, generates actionable reports, and takes measures to prevent damage to the plant. This thesis implements a digital twin of an energy system and uses the learnings of said implementation to contribute to the engineering design of future digital twins.

1.1 Background

Reducing the impact of the process heat sector on the environment requires radical changes to global energy technology. Part of this shift in operations is the move to “Industry 4.0” — the leveraging of interconnected systems to increase efficiency. A core challenge in connecting Industrial Control Systems (ICSs) to external networks is the lack of security these systems were designed around.

ICSs have long shelf-lives and rebuilding these systems is not often a viable solution to improve security. Additionally, even with modern systems, the criticality of industrial control systems, both economically and from a safety perspective, means that security needs to be watertight. For ICSs used in critical infrastructure, the security requirement is yet more important (Gazula, 2017; The Guardian, 2021). Unfortunately, critical infrastructure like the electric grid stands to benefit a great deal from a radical shift in functionality. Wang and Lu (2013) discuss the opportunities and challenges in Smart Grids. While not the exact topic of this thesis, Smart Grids highlight a complex energy system that promises great advances in efficiency if the security concerns can be assuaged.

Digital Twins promise to do exactly that. This thesis explores a framework for Self-Learning in Digital Twins to maintain a close pairing with the physical twin under uncertainty.

A Self-Learning system under the definition put forward in this thesis falls under the third, sixth and seventh waves of adaption as described by Weyns (2018), and advances goals beyond the waves insofar as handling unanticipated change. These waves describe the evolution and interrelation of problems in the field of self-adaptive systems. An unanticipated change in the underlying system could have many causes, but regardless of cause, the Self-Learning system must continue to model the underlying system faithfully to be capable of protecting it.

The long life of industrial facilities makes mandatory a solution that can be applied retrofit, and from machine learning to expert knowledge, Artificial Intelligence undoubtedly plays a role in this solution. Rebuilding these plants anew is a long term objective and not something that is economically viable in the present.

1.1.1 Self-Adaptive Systems

Self-Adaptive Systems are systems that modify themselves to adapt to changes in their environment Weyns (2021a). In computing, software often assumes that its environment does not change, and when it does, that a human can adapt and evolve the software instead. Self-Adaptive Systems as applied to software allow software-intensive systems to adapt to expected changes and variations in their environment. As a primitive example, a server hosting many applications may choose to devote more processing time to the application under the most load. In more complex cases, these systems can dynamically adjust their configurations to meet changing demands.

To achieve these forms of adaptation, many Self-Adaptive Systems use models of the underlying system. These models can be created using a range of methods, including using domain-expert knowledge or via system identification. Of fundamental benefit to this thesis is the ability to guarantee these models against undesirable outcomes. Van Zijl (2020) use formal methods in their thesis to check models of autoscaling in Cloud infrastructure. The goal of that thesis is to guarantee a model satisfies a contractually agreed level of service to the end-user of the infrastructure. In the same vein, energy systems have failure states that need to be guaranteed to a variety of stakeholders, and this goal can be achieved using the mathematical verification of models.

1.1.2 Control Theory

Weyns (2018) describes the use of control theory to achieve Self-Adaptivity as the sixth wave of Self-Adaptation. Control theoretical approaches often make use of mathematical models of underlying systems to create feedback loops that drive the systems toward some goal. These provide benefits over other solutions, such as guarantees under uncertainty and mathematical stability checking. For safety-critical systems, these features are paramount.

1.1.3 State-Space Models

A state-space model is a mathematical representation of a system. While other types of model would suffice for control, this type of model is useful for engineering self-adaptive systems as they provide a well-understood basis from which to build upward. For the purposes of this thesis, state-space models exist in the following discrete-time form:

$$x(t + 1) = \mathbf{A}x(t) + \mathbf{B}u(t) \quad (1.1)$$

Where x is the state vector representing the system at time t , u is a vector of inputs at time t , \mathbf{A} is a matrix that describes how the state x changes over time, and \mathbf{B} is a second matrix that describes how inputs affect said changes in x . Time increments, t , advance in whole units where each unit is the reciprocal of the sampling frequency.

$$|EigenValues(\mathbf{A})| < 1 \quad (1.2)$$

For the verification of stability in this case, the magnitude in the real and imaginary plane of the eigenvalues of \mathbf{A} must be below 1, as shown in Equation 1.2.

In later chapters, this thesis uses the continuous-time form $\hat{x} = \mathbf{A}x + \mathbf{B}u$ as a substitute as it allows for easy distinction between x , the current state, and \hat{x} , the future state.

1.1.4 Anomaly Detection

Anomaly detection is the detection of rare or out-of-distribution data. Commonly applied to fraud and intrusion detection (Chandola, Banerjee, & Kumar, 2009), anomaly detection takes many forms where outliers are typically problematic. This assertion is not always true, however, and some applications

of anomaly detection are used for change detection, such as event detection. Though digital twin technology has potential in furthering low-noise event reporting, this thesis focuses on the drift of a system toward a new normal.

Unanticipated changes are changes that were not explicitly defined and whose impacts can be unpredictable. While control theory can provide guarantees of the model in use, it does not guarantee that the model matches the real world. Handling these changes is subtly different from the common use of anomaly detection. Some of these changes are not anomalous and are instead an incorrect assumption in a model. Other anomalies are not harmful and the system can continue regardless. Changes to the real-world counterpart of these models do occur in practice in cases such as when a heat exchanger fouls or when boiler systems wear (Wade, 1995). These degradations are expected, but the exact impacts on the system are hard to predict.

1.2 Aims

This thesis aims to develop an adaptive digital twin for an energy system and to forward the engineering methods and architectures of digital twins in the process heat sector. In furtherance of the first aim, this thesis architects and prototypes a Self-Learning and Self-Protecting digital twin using a simulated energy system. The use of a simulated physical twin allows for rapid prototyping and the testing and solicitation of further requirements. In terms of iteration speed, stimulated energy systems hold a few key advantages over real-world energy systems. Including, the time investment to set up a testing environment, the speed of development, and the rate of data collection. The learnings of the first aim are then used to rearchitect the design of this type of digital twin and propose an engineering method for their creation.

To reach these aims, this thesis proposes two research questions. These questions reflect the respective aim as laid out above.

RQ1. How to develop adaptive digital twins that incorporate Self-Learning and Self-Protection in the process heat sector?

RQ2. How do the learnings of development expand the theory of Self-Adaptive Systems for the engineering of adaptive digital twins?

1.3 Structure

This thesis is laid out as follows. Chapter 2 covers the related work including the systematic literature around Digital Twins. Chapter 3 presents the method this thesis uses and the questions it answers. Additionally, that chapter defines Self-Learning in the context of this thesis. Chapter 4 describes the techniques and experiment design in addition to the requirements of the work. Chapter 5 discusses the results of the work as they pertain to Self-Learning. Chapter 6 discusses the results of the Self-Protection methods built atop the Self-Learning system. Chapter 7 discusses an updated design based on the lessons learned during Chapters 5 and 6. Finally, Chapter 8 concludes the thesis, discusses its limitations, and proposes avenues of future work.

Chapter 2

Related Work

This chapter covers the related work that surrounds the area of Self-Protecting Digital Twins. This thesis proposes a definition of Self-Learning systems that enables Self-Protection in a wide range of applications.

The upcoming section covers the systematic literature review that started from an ad-hoc literature search into Digital Twins. Before that, the first section covers Digital Twins and the working definition of Project Ahuora. Following digital twins, the Self-Adaptive Control Systems section explains the benefits of using control theory to create digital twins. Finally, the last chapter summarises the chapter and leads to the Requirements and Design chapter.

2.1 Digital Twins

The definition used for Digital Twins until further discussion in Section 2.1.3 originates from the joint work of the National Aeronautics and Space Administration (NASA) and the United States Air Force (USAF), in which Glaessgen and Stargel (n.d.) lay out the groundwork for a new manufacturing, certification, and fleet management paradigm.

2.1.1 Digital Twins in Critical Infrastructure

While Digital Twins are still a fairly new and upcoming area of research, they do see use in critical infrastructure. (Bécue et al., 2018) show twins in use as tools for both automated testing and personnel training. From the aforementioned paper, the following areas were considered operational challenges in security:

- Threat Prevention
- Detection
- Investigation
- Response
- Intellectual Property Protection

They also mention that security “must be understood as an enabler” in Industry 4.0 and highlight the necessity that security is included at a design level.

Similar to Self-Protecting honeypots, digital twinning in critical infrastructure has a relatively small pool of authors. This pool can be divided into two subsets: cyber-ranges and intrusion detection.

Eckhart and Ekelhart (2018a) uses digital twins and state replication to detect intrusions in industrial control systems. Unfortunately, this approach introduces latency and required a reduction in PLC scan time. In industrial control, real-time systems are a hard requirement, so adding latency to these devices is an issue, though not an insurmountable one. Many areas of critical infrastructure operate on time frames where these lowered scan times are acceptable; though, the impact this approach has on patching and updating plants due to requiring the model to update is a potential issue for industry. This paper also highlighted that it can notice an attack after the fact. For critical ICSs, this level of security may not be high enough. It did not deal with remedy-

ing the situation that arises from the event. Detecting a catastrophic event post-calamity is of limited real-world use.

Eckhart and Ekelhart (2018b) also relied on detection after the fact. It also appeared to rely on the virtual environment being hit, not physical hardware. Eckhart, Ekelhart, and Weippl (2019) expanded on this framework and example, bringing it into the realm of visualisation. This framework used by Eckhart and Ekelhart (2018b) does have available code¹.

Gehrmann and Gunnarsson (2020) proposed a framework for twinning an arbitrary device. The solution relied on Cloud connections in their diagrams and appeared to use a bidirectional synchronisation pattern. This solution also detected vulnerabilities after the fact. In many cases, this would be too late to sufficiently safeguard a legacy site. While this paper makes significant inroads in terms of performance and overhead, it requires that the PLC synchronise state with the twin at various points to guard against rogue stimuli targeting the physical twin. Provided these synchronisation points are sufficiently spaced, it may be possible to toggle the PLC state to avoid detection.

An area of commonality of these solutions is the requirement to change the PLC logic to enable state replication. These changes have potential side effects, both in terms of execution time and stability, that may hinder their adoption.

Some industries, such as aviation, take an altogether different approach, focusing heavily on personnel training. Airbus (n.d.) and Boeing (n.d.) both provide such cyber-range solutions to partners.

Additionally, both Airbus (ASCon Systems, 2017) and Boeing (The Boeing Company, 2019) announced the use of Digital Twins; however, they utilise them for manufacturing efficiency gains and to improve the design and safety of future aircraft.

¹<https://github.com/sbaresearch/cps-twinning>

At this time, there is little indication that the industry uses digital twinning defensively in the real world, and even if that case does see use, it is not widespread.

2.1.2 Digital Twins in Critical Infrastructure for Self-Protection

Self-Protection in Critical Infrastructure is a newer area of research and is one that is key to realising the goals of Industry 4.0. Combined with Digital Twins, Self-Protection allows production facilities to be placed on the internet with a high degree of security.

Unanticipated changes that seek to cause damage are, for all intents and purposes, a cyber attack. Current literature focuses on machine learning approaches to address this concern; however, other avenues are under-explored. Danny Weyns raises the issue of Self-Protection as an open challenge in Section 11.2.2.2: “Dealing with Unanticipated Change” in his book under the section titled “Open Challenges” (Weyns, 2021a).

Self-Adaptive systems have a wider reach than solely Industrial Control System security, and the creation of twinned, runtime models has applications ranging from Smart City planning to the decarbonisation of production facilities. These runtime models become even more potent when used to perform “What If?” studies, as is illustrated by Schluse, Atorf, and Rossmann (2017) in their paper that discusses using Digital Twins in conjunction with a simulator to perform these studies. Of note is that the aforementioned paper does not use a runtime synthesised model, and thus would struggle to handle unanticipated changes to the modelled system.

Schroeder et al. (2021) propose a method for the creation of digital twins based on modelling languages. Their approach describes an architecture where a

collection of smaller components that each fulfil a goal. These components cover many of the common features proposed in the literature but do not give rise to Self-Protection. Since the models are built on a static model, these twins are not adaptive and are thus unable to Self-Protect when the physical system changes.

2.1.3 Further Definition

Prior to this section, Digital Twins were defined based on NASA's 2010 definition. While this is a useful and apt description for the overall concept, it pays to differentiate levels of Digital Twin functionality.

This thesis defines Digital Twins as having three tiers depending on stages of functionality:

1. Digital Models
2. Digital Shadows
3. Digital Managers

It is this definition that this thesis will use herein.

Briefly, Digital Models are a standalone model built of a real-world system, Digital Shadows are Digital Models that uses sensor data from their real-world counterparts, and Digital Managers are Digital Shadows that feed data back to the physical twin in the form of control. This definition is very similar to the definition proposed in Kritzingler, Karner, Traar, Henjes, and Sihm (2018), except that this definition refers to Managers over Twins to avoid confusion.

As a first step, this thesis proposes a method for the automatic creation of Digital Shadows rather than Digital Twins under the above definition.

2.2 Machine Learning

Machine learning is the use of algorithms that learn to model a system using example data. These algorithms span a wide range of subcategories from decision tree learners to neural networks. A common feature of all machine learning algorithms is the creation of a model that can be used for prediction, regardless of complexity. There is an argument to be made that system identification of control-theoretical models falls under this definition. Accepting this categorisation, these models differentiate themselves by providing guarantees. As mentioned in Gunning (2017), machine learning is a broad field, and some methods do support forms of verification; however, there is a negative correlation between accuracy and explainability. This gap is no more evident than in the case of deep neural networks.

2.2.1 Neural Networks

Neural Networks (NNs) are a subcategory of machine learning that use layers of updatable biases and weights to approximate a function. While neural networks, especially Deep Learning (LeCun, Bengio, & Hinton, 2015), have shown exemplary performance in a range of fields in recent years, they are not without drawbacks.

Neural networks require considerable amounts of training data to generate an accurate model. For many applications, this demand for data is satisfiable, but for some applications, such as those later in this thesis, it is not. There are data augmentation techniques for improving the size of datasets, but few apply in this case.

Another downside of neural networks is the lack of transparency or verifiability they bring. With verifiable models, both in machine learning more generally and in other disciplines, all possible states are known. With a neural network

approach, the doubt that the model may suddenly behave unexpectedly cannot be removed. Even providing guarantees within safe ranges is prohibitively difficult, especially in the face of malicious modification, such as the one-pixel attack by Su, Vargas, and Sakurai (2019). This doubt presents an issue for the adoption of such techniques in critical infrastructure and is thus an area of active research.

2.3 Self-Adaptive Control Systems

The use of control theoretical models for the creation of digital twins is a novel step toward the synthesis of a self-adaptive system based on the live, real-world system.

There are other approaches to modelling systems, such as the variety of methods used in Burroughs (2021), Chew, Kumar, Patros, and Malik (2020), Patros, Kent, and Dawson (2017), Podolskiy, Patrou, Patros, Gerndt, and Kent (2020); however, these approaches usually require an expert to define them before runtime. For the application of automatically synthesising a system model based on real-world sensor data, prior created models are not a feasible approach.

There are machine learning approaches that can also learn from live data. Stream learning is a good example of such an approach, and stream anomaly detection techniques, like (Tan, Ting, & Liu, 2011), (Dawson, Patros, & Kent, 2021), and (Podolskiy, Mayo, Koay, Gerndt, & Patros, 2019), would work well in a security context. Machine learning approaches, while often highly accurate, suffer from a lack of explainability (Gunning, 2017).

This is in direct contrast to control theoretical models. Frequency response and pole locations of control models show how the model will respond to stimuli and how the model behaves over time.

Work by Angelopoulos, Papadopoulos, Silva Souza, and Mylopoulos (2016) proposes a similar Control-based Requirements-oriented Adaptation (CobRA) method to adapt to changes of an underlying system. CobRA uses a learning component to correct their model of the system based on measurements from the environment. This learning component is a Kalman filter that updates to correct the system's understanding based on how the real system tracked.

2.4 Systematic Literature Review

This section outlines the literature review of security techniques for Critical Infrastructure (CI). The below section will cover the techniques, the methodology, and the results of the review.

2.4.1 Methodology

A systematic literature review was conducted in this research. Before the discussion of the review, it is pertinent to cover some of the decisions that were made and influenced the findings of the review.

Firstly, the automatic portion of the literature review was conducted on Scopus due to both Scopus' API making mining relatively easy and to the number of results collected.

Secondly, the automatic search did not include other papers by the same author or papers citing the core papers when expanding its search radius. To combat this limitation, more recent papers were prioritised.

Thirdly, the algorithm implemented to search the papers assigned a weighting based on age and citation count in an attempt to order papers that were both new and impactful higher than those that were either older, prominent papers or recent, uncited papers.

The following is the scoring equation:

Result: Sets 1.5x multiplier if Title contains “Twin”

if *Title contains “Twin”* **then**

| Twin = 1.5

else

| Twin = 1.0

end

Algorithm 1: Part 1 of Scoring for Automated Literature Review

$$Score = (100 - (CurrentYear - PublicationYear)) \times Citations \times Occurrences \times Twin \quad (2.1)$$

The goal of the two equations above was to sort publications in an order that favours recent papers that occur many times as parents or children of the initial pool of papers or are, in general, commonly cited by other papers. It further prioritised papers whose titles contain the Twin to push Digital Twin papers higher.

For the systematic portion of the literature review, this thesis selected 13 papers using the following keyword searches:

- digital twin AND cyber security
- digital twin AND cyber-security
- digital twin AND cybersecurity

This thesis selected these papers based on the impact, citations, abstract, introduction, and relevance to this thesis’ research goals as stated in Chapter 2.4. This preliminary search left a group of recent papers from which to expand the search.

From there, the automated portion of the literature review collected the abstracts and metrics of related papers and ordered them for review.

After two expansions, the list of 539 papers was filtered by abstract.

During this process, an expansion to the search terms was necessary to capture the research intentions of this thesis. Including the added terms, the full list of terms was:

- digital twin AND cyber security
- digital twin AND cyber-security
- digital twin AND cybersecurity
- digital twin AND critical infrastructure
- digital twin AND critical-infrastructure
- digital twin AND process heat

This repeat yielded an additional 137 papers, bringing the total to 676 papers. A manual search over the abstracts brought this count down to 308.

In total, fewer than 100 were not related to motivation. This total of 676 is in line with findings by Jones, Snider, Nassehi, Yon, and Hicks (2020) that showed only very recent widespread interest in digital twins, so it stands to reason that there is not an overly large corpus of work.

2.4.1.1 Manual Additions

Because automatic searches may potentially miss high-value resources, the review included some additional papers on the merits of interest or academic relevance. This is especially true when dealing with governmental resources from comparatively small nations, such as New Zealand, as they are unlikely to see widespread citation outwith their country of origin.

Examples of such additions include Lockheed-Martin’s Cyber-Kill Chain paper (Hutchins, Cloppert, & Amin, 2011) and a paper evaluating the efficacy of reinforcement learning in intrusion response (Iannucci, Barba, Cardellini, & Banicescu, 2019).

A second, smaller literature pass over the following terms added additional context:

- Self-adaptive AND digital twins
- Self-adaptive AND digital twins AND cyber
- Self-adaptive AND honeypot

2.4.2 Continuous User Training

The Government Communications Security Bureau (GCSB) suggest in Section 7.1.7 — Detecting Information Security Incidents — of the New Zealand Information Security Manual (GSCB, 2020) that all agencies should “implement and maintain tools and procedures covering the detection of potential information security incidents...”. Of the items suggested to be included in these tools and procedures, User Awareness and Training appears to be an area of little academic research.

Karampidis, Panagiotakis, Vasilakis, Markakis, and Papadourakis (2019) found that 70% of enterprises they surveyed did not have scheduled awareness briefings. They also found that 75% do not have a traffic analysis tool, and 70% did not have tested backups. Regardless of the reason for this trend, it shows that a considerable number of enterprises could strengthen their security posture. The authors postulate that Information Technology (IT) operators in the ICS space “do not consider [the likelihood of cyber-physical attacks] seriously”.

Čeleda, Vykopal, Švábenský, and Slavíček (2020) proposed a training testbed for cybersecurity students. Citing industry issues with an insufficient quantity

of trained security personnel in the ICS space, this paper focused entirely on a course to build skills in a workforce of security experts rather than to reinforce the security posture of the workforce on the ground.

It is worthy of note that, unlike in other areas, Denial of Service (DOS) attacks in ICSs are a critical-level vulnerability. If a controller is brought down, it has a very real possibility of disrupting the entire site (Microsoft Corporation, n.d.; Reuters, n.d.). Such attacks against PLCs can be performed with open-source penetration testing tools, such as Metasploit (Wallace & Atkison, 2013).

While improved testing and training capability is vital for the security of industrial control systems, it's not the only avenue that can be explored.

2.4.3 Literature Crossover

In this second, ancillary literature pass, this thesis focused on a combination of four themes:

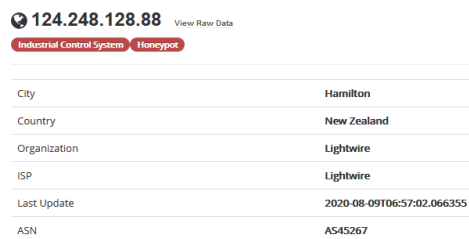
- Honeypots and Decoys
- Digital-Twins
- Self-Protection
- Critical Infrastructure

2.4.3.1 Honeypots, Decoys, and Deception

Self-Protecting Honeypots The first gap identified in this literature review is that very little work exists in the space of Self-Protecting honeypots. There is little existing research in this area, and what research there is written by a small pool of authors, such as Pauna (2012) and Pauna, Bica, Pop, and Castiglione (2019).

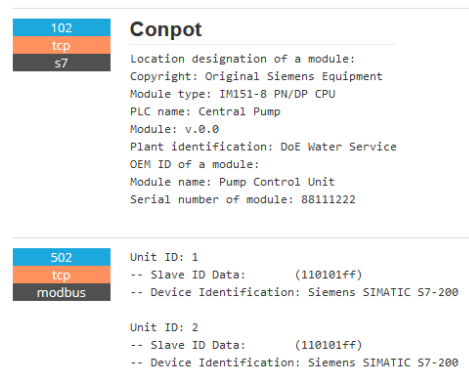
One of the primary benefits brought by a Self-Protecting honeypot is the ability to gather real-world data about how Self-Protecting fair in the wild. Additionally, Self-Protecting honeypots should be able to consume extra resources from the adversarial side.

Digitally-Twinned Honeypots Honeypots based on digital twinning is a recent research area by extension of digital twins being, themselves, a new field (Jones et al., 2020).



124.248.128.88 <small>View Raw Data</small>	
Industrial Control System Honeypot	
City	Hamilton
Country	New Zealand
Organization	Lightwire
ISP	Lightwire
Last Update	2020-08-09T06:57:02.066355
ASN	AS45267

Figure 2.1: Lightwire Address tagged as ICS Honeypot in Shodan (n.d.)



102 tcp s7	Conpot Location designation of a module: Copyright: Original Siemens Equipment Module type: IM151-8 PN/DP CPU PLC name: Central Pump Module: v.0.0 Plant identification: DoE Water Service OEM ID of a module: Module name: Pump Control Unit Serial number of module: 88111222
502 tcp modbus	Unit ID: 1 -- Slave ID Data: (110101ff) -- Device Identification: Siemens SIMATIC S7-200 Unit ID: 2 -- Slave ID Data: (110101ff) -- Device Identification: Siemens SIMATIC S7-200

Figure 2.2: Step-7 port (102) flagged as a honeypot and Modbus shows two fake S7-200s in Shodan (n.d.)

Digital Twins provide the ability to create accurate, high-interaction honeypots. The benefit of such a system is the increased difficulty an adversary would face detecting the honeypot, which alters the cost-benefit tradeoff of attacking the infrastructure. Zamiri-Gourabi, Qalaei, and Azad (2019) described the ease in which some honeypots may be fingerprinted and demonstrates the need for reduced fingerprint-ability in deployed honeypots. This point is reiterated by a cursory glance — Figures 2.1 and 2.2 — on the web indexing

platform Shodan (n.d.). Even when restricted to just New Zealand, it is relatively easy to spot a Conpot (MushMush Foundation, n.d.) instance operating on Lightwire (n.d.)’s network.

Honeypots in Critical Infrastructure Honeypots in Critical Infrastructure are not an uncommon occurrence in the literature (K. Wang, Du, Maharjan, & Sun, 2017), and neither are deception campaigns. There are a few reasons that deception makes sense in an industrial control setting.

Firstly, many systems in operation in production today are unable to use state-of-the-art security methods. Due to the long life of infrastructure, many plants were not built with interconnection, Cloud, or Internet of Things (IoT) in mind. These plants, built on protocols like DNP3 and Modbus, lack any form of security (Amoah, Camtepe, & Foo, 2016; Hayes & El-Khatib, 2013).

Secondly, honeypots can provide detailed threat intelligence about attacks in the wild. This information can be used to strengthen areas that see high-frequency or high-severity threats occur (Bahşi & Maennel, 2015). Deutsche Telekom (n.d.-a) host a dashboard showing open-source intelligence (OSINT) created from the data of both their own and community honeypots.

Honeypot	Interactivity	Protocols/Ports	OSINT
Conpot	Med	13/13	HPFeeds
GasPot (Trendmicro, 2015b)	Low	1/1	None
Artillery (BinaryDefense, n.d.)	Low	1/16	None
Cowrie (Oosterhof et al., n.d.)	Med-High	1/1	None
Dionaea (DinoTools et al., n.d.)	High	14/14	HPFeeds
T-Pot (Deutsche Telekom, n.d.-b)	High	39/39	HPFeeds

Table 2.1: Non-exhaustive list of ICS Honeypots

As an aside in Table 2.1, Conpots maintainers merged Gaspot into Conpot in 2015 (Trendmicro, 2015a) and instances were detected via fingerprinting in (Zamiri-Gourabi et al., 2019). Additionally, HPFeeds is an open-source intelligence feed that is supported by a wide range of tools.

From the academic side, many papers, such as Bernieri, Conti, and Pascucci (2019)² or Pauna, Iacob, and Bica (2018), do not provide code with which to reproduce results. Additionally, some papers, like Pauna et al. (2018), evaluate on the open internet, which, while a real-world example, lacks reproducibility.

2.4.4 Key Findings

This section lays out the key findings of the literature review. In the literature, there is a significant focus given to the areas of Intrusion Detection and Anomaly Detection (Feng, Li, & Chana, 2017; Lai, Liu, Song, Wang, & Gao, 2016; Zhou et al., 2015). The literature also appears to have an overarching focus on the defensive side of security. This focus is justified in the findings of Urias, Van Leeuwen, and Richardson (2012) and Green et al. (2017).

Without investment in the development of standardised testbeds and “cyber ranges”, such as Gao, Peng, Jia, Dai, and Wang (2013), Bitton et al. (2018) and Bécue et al. (2018), penetration testing is hampered significantly by either the cost or risk when testing on in-house deployments or production sites, respectively. Not all testbeds are equal, however, and Gao et al. (2013) lacks actionable results. Both Bitton et al. (2018) and Bécue et al. (2018) are stronger, but focus on how cost and the potential impact of digital twins impact testbeds, respectively.

²Additionally, this paper arguably falls within the realm of Digital Twins

Davis and Magrath (2013) survey cyber ranges and testbeds and categorise them into various areas, such as testbed technology and use case. A key finding in this paper was that many current testbeds utilise either simulation or emulation with an overall focus on training. Another fundamental point raised was the implementation and monetary cost associated with large scale deployments of these testbeds.

Holm, Karresand, Vidström, and Westring (2015) also surveyed testbeds intending to answer four research questions. In their survey of 30 Industrial Control System (ICS) testbeds, 16 had vulnerability analysis as their key objective. Education and the evaluation of defence mechanisms ranked joint second with nine testbeds a piece. Another finding from this paper is that only 11 testbeds in this study attempt to verify fidelity with only four being based on standards.

2.5 Research Gaps

The following items were identified as research gaps in the literature review:

- Self-Protecting Digital Twins
- Testing Methodology for ICS Systems
- Framework for Fidelity Analysis of ICS testbeds (Holm et al., 2015)
- Automated Security Testing using twinned CPSs
- Hardware-directed decoys or twins
- A scalable, digitally twinned cyber-range

Additionally, Asghar, Hu, and Zeadally (2019) state that current ICS security solutions carry a high cost, both in terms of implementation and maintenance cost. For this reason, this thesis begins by discussing an approach to enabling internet connectivity of industrial control facilities aided by Self-Protecting Digital Twins.

B	F	J	K	N	S	V
	Twins	State replication	IDS	Visualisation	Framework	Self-Protection
Deriving a cost-effective digital twin of an ICS to facilitate security evaluation						
A specification-based state replication approach for digital twins						
CyberFactory#1 - Securing the industry 4.0 with cyber-ranges and digital twins						
Towards security-aware virtual environments for digital twins						
Characterising the Digital Twin: A systematic literature review						
A digital twin based industrial automation and control system security architecture						
Enhancing Cyber Situational Awareness for Cyber-Physical Systems through Digital Twins						

Figure 2.3: Simplified Gap Analysis

A Digital Twin capable of Self-Protection and Self-Healing can tackle some of the key areas of future research raised by the aforementioned paper, such as reduced implementation time and increased adaptability under uncertainty.

2.6 Conclusion

In summary, there are many avenues of research still to be fully explored in the area of Digital Twins. This thesis focuses on their uses in the Self-Protection of dynamic energy systems.

One of the challenges of Digital Twins for Self-Protection is maintaining an accurate model of the physical twin when said system is subject to change. Weyns (2021b) envisions the use of learning techniques, such as Bayesian networks or Machine Learning, to keep runtime control models up-to-date in response to uncertainty. In response, this thesis describes a Self-Learning framework that aims to dynamically maintain an accurate, within defined tolerances, model required for such Self-Protection.

Chapter 3

Research Method

From the gaps in the literature highlighted in Chapter 2, this thesis started by conceptualising a Digital Twin-enabled Cloud-Machine Interface (CMI). This interface creates a human-in-the-loop system that leverages the Digital Twin paradigm and lessen the attack surface. A barrier to using Cloud computing to optimise plant efficiency is the possible attack surface that connecting a plant to the internet exposes. By running “What If?” tests on the digital twin and passing the updated state to a human operator, the CMI aims to reduce the risk of compromise for twinned infrastructure.

Figure 3.1 depicts this proposed concept of a CMI that uses digital twins as a method of validating the legitimacy of data received from the Cloud. Using control theoretical approaches to digital twins provides guarantees and explainability, both of which are necessary for Industrial Control.

In Figure 3.1, the connection that the CMI uses to interface with the Human-Machine Interface (HMI) has an electric disconnect to allow the HMI to isolate the network at any moment. The CMI interfaces with a read-only network splice to retrieve live system data, with the HMI to send vetted suggestions, and with the Cloud to retrieve suggestions. The digital twins perform “What If?” tests on optimisations from the Cloud to ensure safe operation.

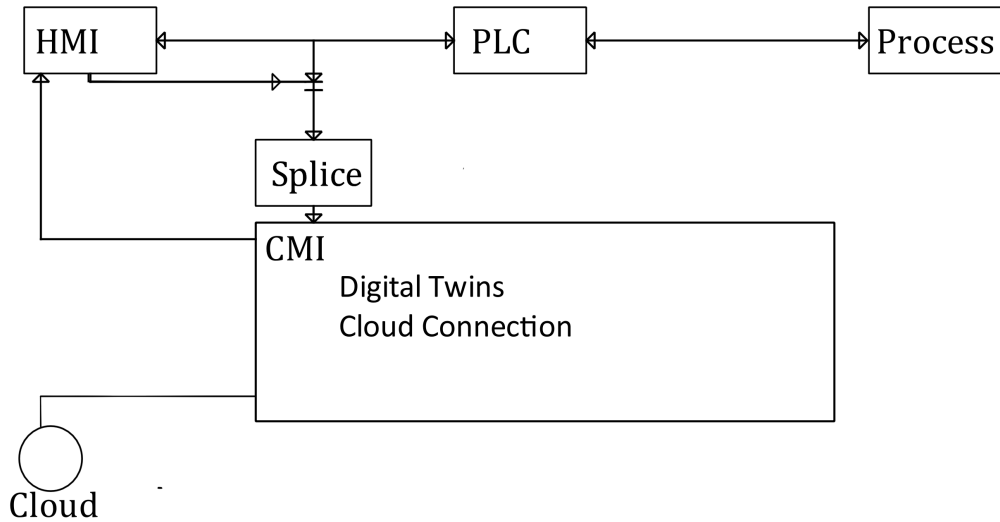


Figure 3.1: Cloud-Machine Interface

In this interface, no Cloud optimisation suggestion is deployed directly. The digital twin vets the suggestions before presenting them to the human operator for the final decision.

This chapter outlines the research questions that this thesis tackles, the definitions used in achieving those ends, and how the experiments and systems function. To the best of the knowledge of this thesis, the Self-Learning architecture presented herein is a novel contribution.

3.1 Self-Learning and Self-Protection

The definition of Self-Learning in the context of Self-Adaptive Systems is only vaguely defined. Machine Learning aptly describes the above cases, and in fact, many use techniques well regarded to fall under that umbrella, such as neural networks and genetic algorithms. Self-Learning, in the context of a system that adapts to its environment, should be able to adapt to said environment. The above mentioned approaches are trained once at creation and do not

adapt; instead, Self-Learning is used to denote a system that learns without external aid. In common machine learning parlance, this class of approaches are referred to as unsupervised learning (Ghahramani, 2004).

Instead, this thesis posits a framework for creating Self-Learning systems that learn from and adapt to their environments. A framework for a generic system such as this Self-Learning system should also be generic. To those ends, this thesis argues for the definition of Self-Learning as a system of the following components: the monitoring component, the modelling component, the verification component, the imagining component, the falsification component, and the reconsidering component. These components enable for the substitution of implements of each module while retaining the overall Self-Learning nature.

This thesis' solution involves fully re-identifying the model to incorporate fundamental changes that were unanticipated at design time. This is in contrast to CoBRA that, as mentioned in Section 2.3, builds a Kalman filter to transform and adapt to the data it reads from the system.

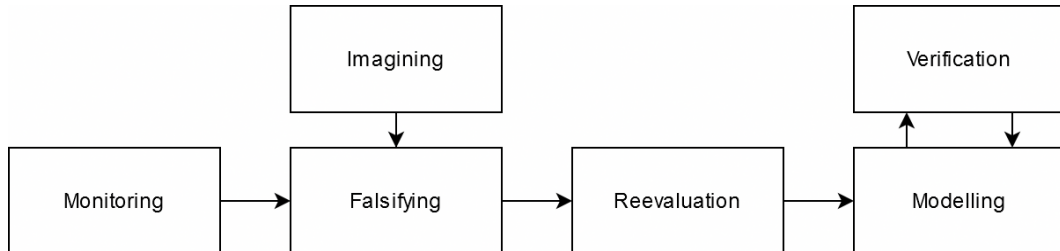


Figure 3.2: Proposed Self-Learning Architecture

These additional terms are defined below and their interrelation is depicted in Figure 3.2. The implementations of these terms depend on the underlying techniques that constitute the overall system.

The **Monitoring** Module provides the system the ability to monitor its surroundings. In this thesis' case, it is the ability of a digital twin to read live data from its physical twin.

The second module — the **Modelling** Module — provides the system the ability to generate a model of the physical twin based solely on inputs and state of that twin. This property draws an analogy to the ability to formulate beliefs based on observations. There are a plethora of ways to attain this property. Many of the aforementioned papers implement unique modelling techniques, and in fact, almost all techniques under the umbrella of Machine Learning should fit this bill.

As part of this modelling, the system needs to be capable of verification, provided by the **Verification** Module. The ability of a system to verify itself is necessary to ensure that the model is stable and correct. For control theoretical systems, there exist defined verification techniques, such as frequency response plots and pole plots. Some Machine Learning algorithms, such as decision tree learners, can be audited, but even these lack guarantees on stability. An example of pole plots showing stability is depicted in Figure 3.3. The three poles in this figure have a distance of slightly less than one, which means the system will slowly converge.

Determining model correctness is, at least partially, domain-specific. For example, in the area of process-heat, a model that describes a change in internal energy without energy entering or exiting the system violates the first law of thermodynamics. For some domains, the aforementioned sanity check may not be feasible. Another example is whether the model is logically consistent.

Much like a human using their knowledge of a subject to estimate “What If?” scenarios, the **Imagining** Module relates to the capability of a system to imagine a future state based on forecasted inputs. The imagining module also covers such a system’s ability to produce said forecasted inputs through a variety of substitutive means. In this thesis’ case, the forecasting is handled by one of two oracles as covered later in Section 4.1.3. These oracles allow the system to see future states and ask limited “What If?” questions.

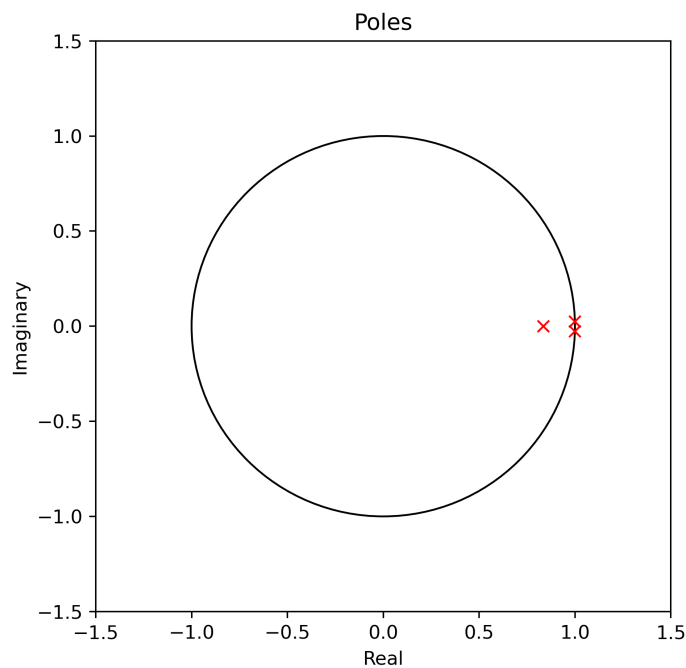


Figure 3.3: Three Pole Plot for DMDC model

The **Falsification** Module, alternatively the **Validation** module, allows the system to determine how far from reality an imagined value is after reality reaches that point. It is the ability to realise that the beliefs, which are the predictions in this case, one holds are incorrect. If where the system imagined itself to be is sufficiently different from where it ended up, the system needs to reevaluate its beliefs. As this definition pertains to digital twins, it refers to the ability of the digital twin to compare its imagined values to the physical twin’s true values. Continuing from the prior example on the imagining module, the system then gets an answer to the “What If?” question.

In the below figure, solid lines represent the control flow and dotted lines represent data flow.

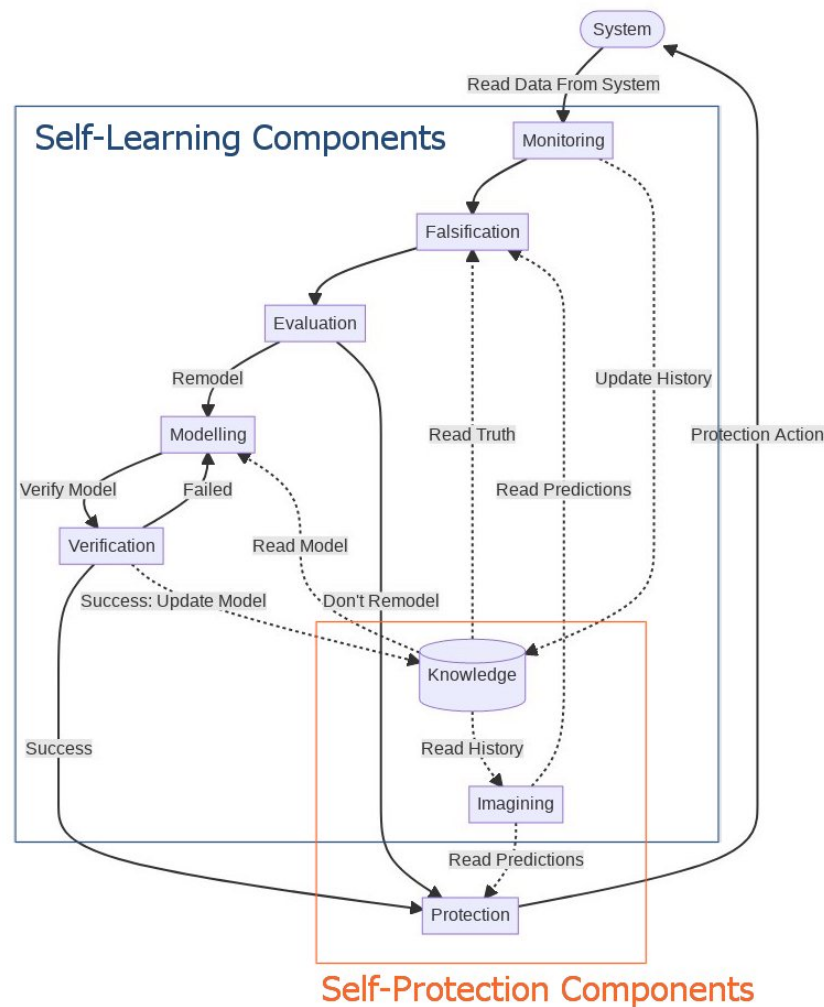


Figure 3.4: Implemented Self-Learning Diagram

Finally, the **Reevaluation** Module provides a system's ability to update its model based on how the system progressed in reality. This state only occurs if the system was previously falsified. As depicted in Figure 3.2, a model that does not support a partial update may instead recreate itself. The reevaluation module may choose not to update a model in cases such as the new model failing non-functional requirements.

As shown in Figure 3.4, part of the duty of the Monitoring component is to read sensor data from the system — the simulator — and update the stored history. The Monitoring Module then passes control flow to the Falsification Module.

The Falsification Module first requests predictions from the Imagining Module, then it compares the predictions to ground truth from the knowledge store. The falsification data is then passed to the Reevaluation Module.

The Reevaluation Module checks if the Model was falsified and whether the model should be retrained. Some other considerations in this module are restricting the module from retraining the model too often, especially if retraining has a high cost. When training a new model, the control is passed on to the Modelling Module; otherwise, control is returned to the Falsification Module. In the event that control is returned to the Falsification Module, it then calls on the Self-Protection Module.

If the control flows to the Modelling Module, said module handles fully or partially updating a copy of the model, depending on what the Reevaluation Module decided was necessary. Once the copy of the model of the system is updated, the module passes it to the Verification Module for verification.

The Verification Module verifies the model, and if it passes verification, commits it to the knowledge store. Upon a verification failure, the control flow returns to the Modelling Module to resolve potential issues. In this thesis, verification and feedback were human-in-the-loop designs that required manual intervention. The experiments in Chapter 5 and Chapter 6 do not wait for the human-in-the-loop to act and simply continue regardless, though the latter chapter does discuss automatically reacting.

Lastly, the Self-Protection Module handles preventing the real system from venturing into these states. In this thesis, this module engages to prevent the system from exceeding 100°C. Section 6.3 discusses the states in further detail.

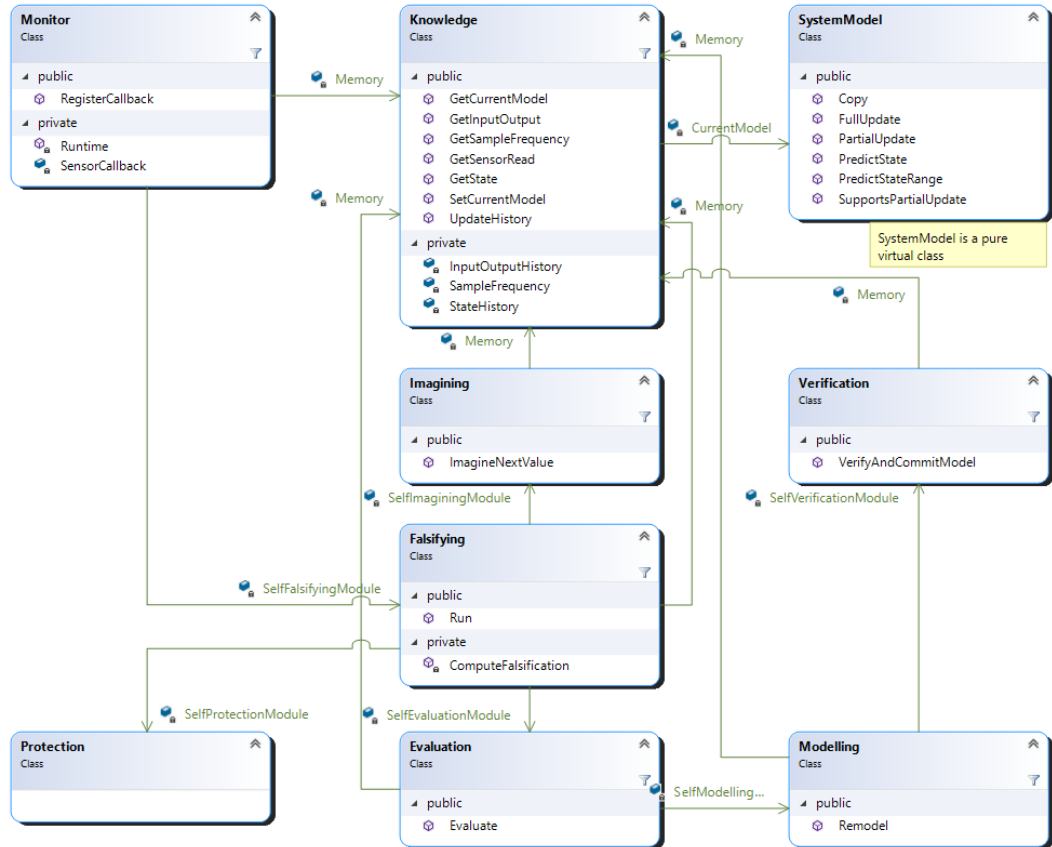


Figure 3.5: Self-Learning Framework Class Diagram

Figure 3.5 shows an example class diagram for Self-Learning. Where applicable, the classes shown in the figure do not include the “Self-” prefix in their class names. Of note in this diagram is the SystemModel class. SystemModel is an interface that any implementation of a model must implement to be compatible with the Modelling Module.

These classes are minimal: they do not try to cover every possible end-use. Instead, they provide a starting point from which to build additional functionality, such as model adjustment. The diagram depicts Self-Protection without member variables or methods as it lies outwith the scope of Self-Learning.

3.2 Hypotheses

Before moving on to cover the research questions in further sections, there are a few terms that need defining in the context of digital twins. The first term this thesis defines is Self-Learning as it is a precursor to both Self-Protection and Self-Healing in digital twins.

Other research uses the term Self-Learning in many different ways. In Al-Habaibeh and Gindy (2001), the authors use automated sensor and signal processing to create a self-learning system to detect degradation and failure of machining equipment. The application of self-learning in that paper is in determining which sensors are most representative of a fault. The use case of that paper is similar to the case laid out later in this chapter; however, the key difference is that this thesis' approach focuses on enabling a digital twin that can handle unanticipated changes of the physical twin.

Zhao et al. (2021) use Self-Learning to refer to a system that learns which parts of its training data hold the most relevance. The training is conducted on real-world data, as per a Digital Shadow, but it is only run during a training phase and is not a continuous process.

Indeed, the use of Self-Learning to refer to an algorithm that learns without expert knowledge is fairly common. Both of the above examples appear to fit this classification, and so do Oliveira-Neto, Han, and Jeong (2013), Jang (1992), and Wei et al. (2021). Peng Wen, Zhang Dianhua, and Gong Dianyao (2012) uses the term to represent an online, proportional control strategy guided by mass-flow equations.

As Self-Learning is a precursory challenge to Self-Protection, this thesis needs to ascertain a fit-for-purpose Self-Learning architecture before moving on to Self-Protection. As such, this thesis plans to answer the following research questions:

- H1. Is the proposed software architecture for Self-Learning Systems fit-for-purpose?
- H2. Are the proposed Self-Learning algorithms fit-for-purpose as applied to Dynamical Energy Systems?
- H3. Assuming adequate Self-Learning, are the proposed Self-Protection algorithms fit-for-purpose?

3.3 Summary

In summary, this thesis focuses on fit-for-purpose Self-Learning methodologies and algorithms. A fundamental part of the aforementioned definition of Self-Learning is being able to update the model in use. As Filieri et al. (2017) mentions in their Section 5 subheading “Automatic synthesis and update of controllers”, the ability to dynamically create and update models at run time is an open challenge, and this thesis investigates methodologies that aim to achieve exactly that.

Chapter 4

Requirements and Experimental Design

This section covers the requirements and design of the simulator, the oracles, and the experiments. For the reproduction of results, where applicable and when not otherwise stated, all randomness is pseudorandom and the seed shall be zero (0).

4.1 Simulation

To test both the first and second research question, this thesis uses a water heating simulation written in Python 3. Without a real-world boiler to model nor a built simulator for these experiments, this thesis builds a water heater rather than a boiler due to the equations used being faster to implement. Overall, this choice should not alter the validity of the findings as the proposed framework is not implementation-specific.

The simulation makes use of a deterministic random number generator (RNG), meaning that runs with a given seed value always progress identically, which allows for direct comparisons between Self-Learning and Self-Protection strate-

gies. Additionally, the simulator runs at a fixed 100-hertz internal rate, and the test harness samples the system at 0.1 hertz.

Although the simulation is not overly true to real-world industrial boilers — boiling is considered a failure state by the simulation — it does allow testing hypotheses, which is what is important in this context.

The design of the heater is detailed in the following sections.

4.1.1 Heater

The first component of the simulation is the heater itself. The heater consists of a water tank, a heating element, an inflow pipe, and an outflow pipe.

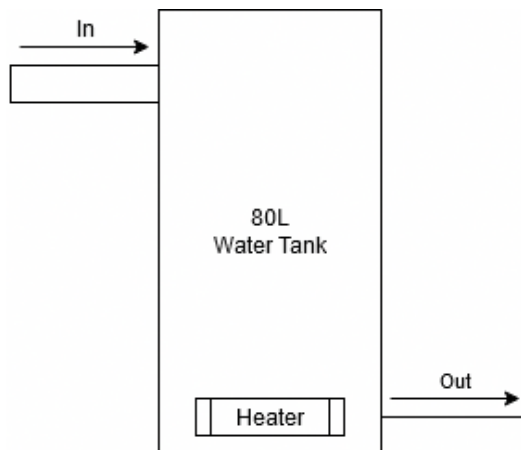


Figure 4.1: Modelled Water Heater

In all tests, the tank is configured as having an upper limit of 75 litres and a minimum of five litres. The heating element is relatively large for the amount of water at 20,000 watts. The size of the heating element gives the heater fast dynamics and creates a need for a good twinning solution. Many boilers in the energy sector have slower dynamics than this heater, so provided a Digital Twin can be synthesised for the heater, it should apply to slow systems.

While the Digital Twin of this heater should generalise to slower systems, some pertinent assumptions are listed. First, the water in the heater mixes perfectly.

While not true on short time scales, water does mix over longer time scales, and these long scale trends will amortise. Second, the Twin has complete visibility to both the internal state and the inputs of the heater, which is not always true in real-world applications. As discussed in Section 4.1.2, this implementation does change which input and state values are exposed, but that does not completely mitigate this limitation.

On the note, Table 4.1 depicts the variables of the heater.

Inputs and Outputs	Internal State
Water Inflow Rate	Water Volume
Inflow Temperature	Water Temperature
Water Outflow Rate	Heater Maximum Power
–	Heater Output Level

Table 4.1: Raw Heater Variables

The simulation uses the same seeded RNG to influence the inflow and outflow rates. The logic for water inflow is simplistic: if and only if the tank is not full, water can flow. Water outflow is equally simplistic with the only difference being that it checks that the tank is not empty. Despite the simplicity, the simulation does

The boiler updates its temperature using the following equations:

$$AvgHeat = \frac{T_w * V_w + T_i * V_i}{V_w + V_i} \quad (4.1)$$

where T_w and V_w are the temperature and volume of water in the tank, and T_i and V_i are the temperature and volume of the input water.

$$HeatGain = \frac{H_p * H_c}{4200 * V_w} \quad (4.2)$$

where H_p is the heaters maximum power, H_c is the heater output level, and V_w is the water volume in the tank.

In the below algorithm, *HeatLoss* is a small amount of lost temperature.

Result: Temperature and Volume are updated

Volume -= OutFlow * 0.5

Temperature = AvgHeat + HeatGain - HeatLoss

Volume -= OutFlow * 0.5

Volume += InFlow

Algorithm 2: On Tick Boiler Update

In Algorithm 2, the Volume is modified by *OutFlow* both before and after the temperature update. The reason for this is that in-flowing water, *InFlow*, is already computed analytically, and so should be handled after the update, where out-flowing water is not, and thus splitting it into two helps avoid biasing the update due to sample timing.

4.1.2 PID Controller

A proportional-integral-derivative (PID) controller handles the heater output level. This controller is tuned by hand and is far from perfect, adding further challenge and motivation to the Self-Protection use case. Figure 4.2 depicts how this PID controller is implemented in respect to the water heater.

Inputs and Outputs	Internal State
Water Inflow Rate	Water Volume
Inflow Temperature	Water Temperature
Water Outflow Rate	Heater Output
Target Temperature	–

Table 4.2: PID + Heater Variables

The effect of the PID controller is that the boiler is that there is now an additional input variable — the PID’s target temperature. The internal state also shrinks with *HeaterMaximumPower* and *HeaterOutputLevel* becoming a single value representing the current operating power in watts.

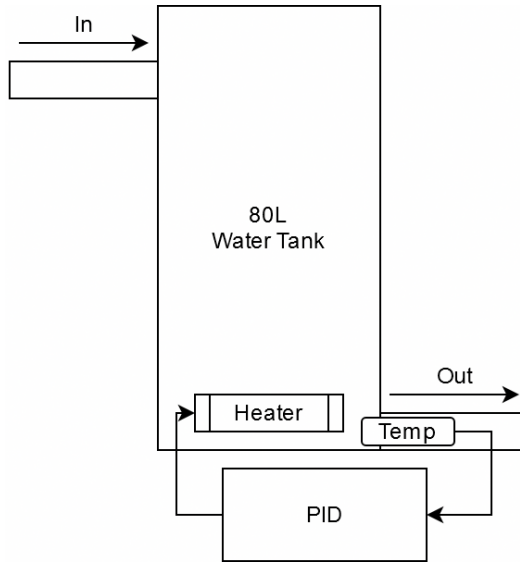


Figure 4.2: Heater Diagram with PID

The values in Table 4.2 are used by all twinning techniques in the coming chapters. The change to monitored variables should better reflect real-world scenarios, even if these are still fairly best-case examples.

4.1.3 Forecast Oracles

The implementation of the simulator uses two forecast oracles. The first of these oracles, as depicted in the lower graph of Figure 4.4 by the first of two coloured bars, is a perfect representation of the future. As seen in the upper graph, the simulator is running ahead of the twin, which provides the first oracle window with a perfectly accurate set of future inputs. In this window, the only variable is the model in use as the forecasted values are perfect.

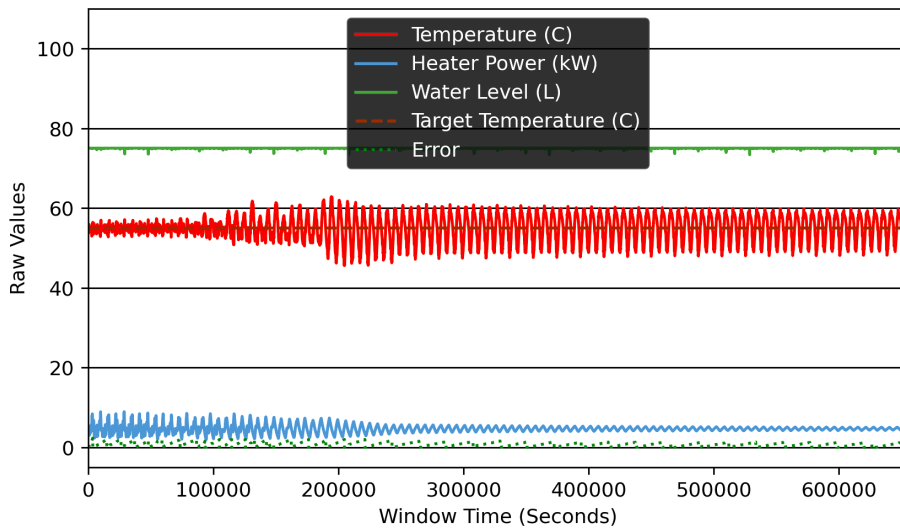


Figure 4.3: Evolution of the heater over time

In the second oracle, the last value of the first oracle is replicated over the entire window. This oracle emulates a poorly performing forecasting algorithm. The benefit of using a constant value over the window is that it shows the temporal dynamics of the model in use.

From the perspective of the model, the oracles both represent future values. The oracles only provide the input and output data, and the model must propagate the state correctly to predict future trends. This thesis only runs the simulator in the future to provide ground-truth values.

Figure 4.4 uses a Dynamic Mode Decomposition with Control (DMDc) model that shows good temporal dynamics and resistance to poor forecasting. As a note, this thesis does not cover the use of forecasting algorithms, but they do form an integral part of real-world implementation.

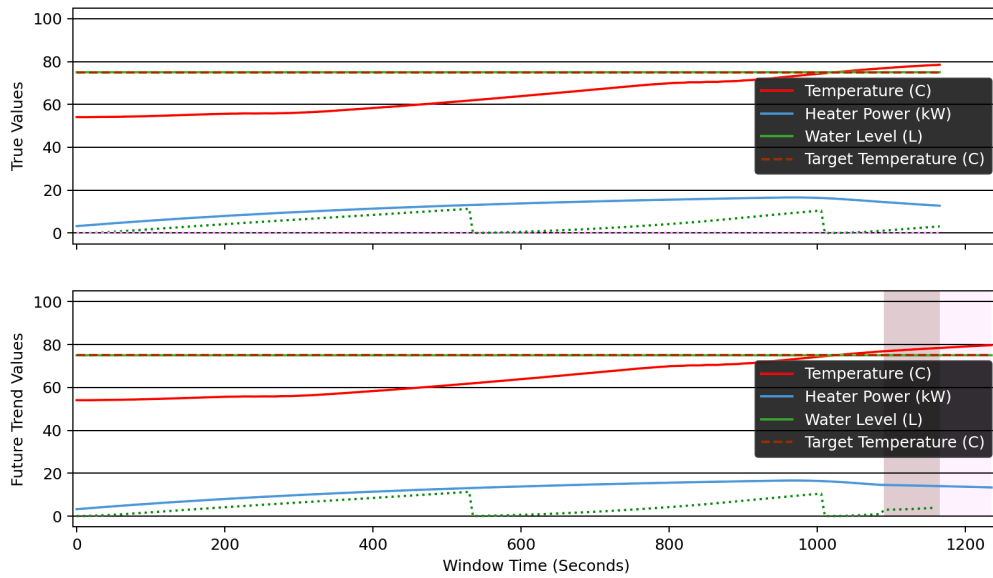


Figure 4.4: Simulated Heater Graphs

4.1.4 Error Metrics

A secondary benefit of the simulator running in the future is that the error can be computed at various points over the first oracle’s window. In the results section, all results pertaining to model error use the error over a single time step.

As shown in Figure 4.5, the error over oracles of various lengths remains relatively constant. This figure measures the error of each step rather than the accumulated error. The tighter the 95th percentile confidence interval — depicted in blue — and the flatter the line, the better the model is handling state propagation. In this case, DMDc was the model in use.

This graph is the aggregate of a sliding window of results rather than stochastic predictions against a fixed ground truth. For this reason, these graphs are representative of the initial 10,240 seconds of the simulation. More graphs showing error over the oracle window appear in Chapter 6.

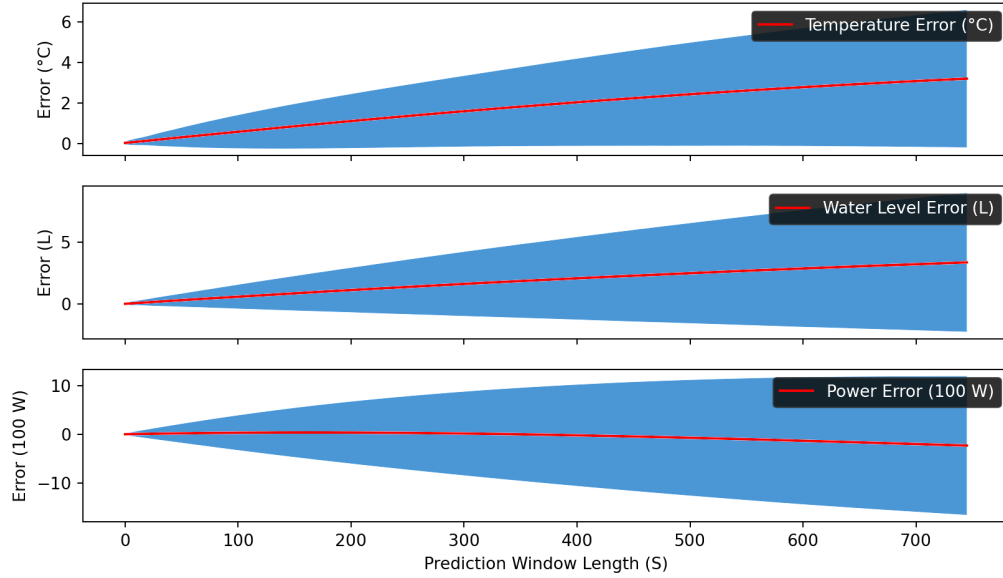


Figure 4.5: Error over the first oracle's window ($n=2048$)

4.1.5 Degradation

As an operative question in this thesis is to adapt to unanticipated changes in the underlying system, the heater degrades over time. It does so in several ways:

Firstly, the heater's maximum power degrades over time. When this degradation starts and how long it takes to reach maximum degradation are both randomly selected from a Gaussian distribution added to a constant.

Secondly, the interpolation rate of the heater in response to the PID controller slows, leading to longer dynamics over time. Both the time to start and the onset time are selected the same way as above. This lag time exists to simulate a heating element that cannot immediately change in power level, such as gas-fired boilers.

The simulation allows a maximum of 60% degradation. When the power degradation causes the heater to be incapable of reaching the target, the DMDc model becomes highly inaccurate. This inaccuracy does cause a retrain that solves the issue, but it affects the graph scale. This particular case is interesting as it does show the retraining working. In the tests conducted in this thesis, the heater reaches full degradation at approximately 248,550 seconds. The instability occurs considerably after said point, as shown in Figure 4.6, and produces warnings relating to numeric stability of the matrices.

With the graph scale ranging so drastically, it becomes hard to pick out small-scale inaccuracies in the model. Considering that the degradation of the heater is still a large, unanticipated change, this thesis limits degradation to 60% — an 8,000-watt maximum output power. This is sufficient to demonstrate the Self-Learning properties of the system.

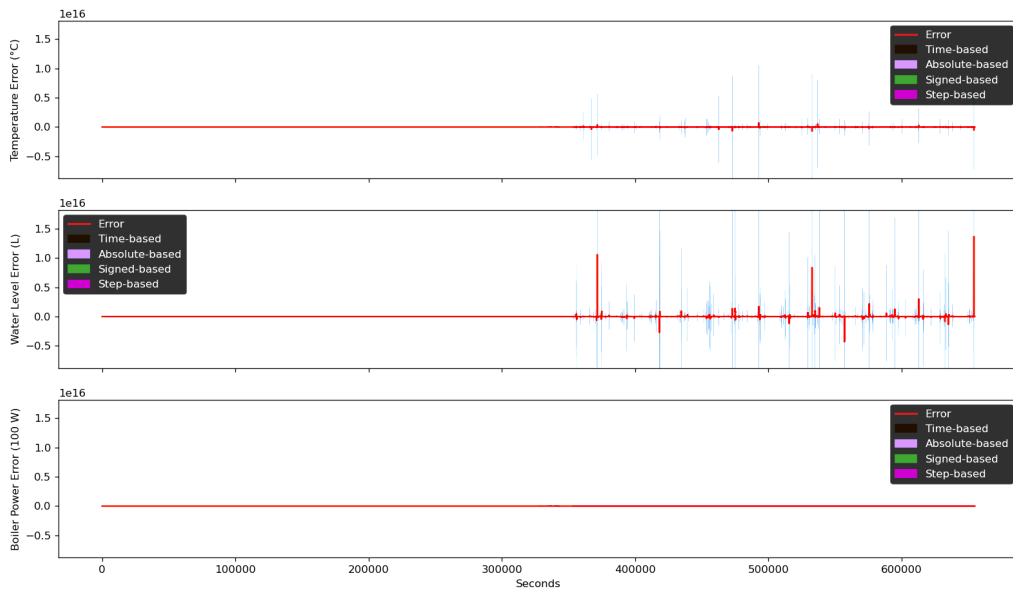


Figure 4.6: Numeric Explosion with DMDc when Heater degrades to 95%

4.2 Tested Techniques

This thesis evaluates several techniques against the heater defined in Section 4.1. Each technique is discussed in detail below. Some techniques are not tested and will be discussed in the limitations section.

Before moving to definitions, this thesis defines an axis of conformity for each of the six attributes. Monitorability, Modellability, Verifiability, Falsifiability are binary. Either the system supports it, or it does not. For both Imaginability and Reevaluability, any given system can have either full, partial, or no conformity. Imaginability can have partial conformity in the form of a system that can only predict the future using state data without factoring in input and output data. Reevaluability can also have partial conformity insofar as partially conforming systems do not support partial updates to models and instead require recreating the model.

The reason that Imaginability has partial conformity is that a model that does not use input and output data cannot perform “What If?” experiments where the question revolves around changes to inputs or outputs.

Table 4.3 depicts the techniques as they sit in terms of conformity.

4.2.1 Base-Case

The Base-Case technique assumes that the state at time t will be identical to $t - 1$. This technique does not make any use of input or output state information and is therefore only partially capable of Imagining.

As this technique does not create a model, it is incapable of Modelling, Verification, and Reevaluation.

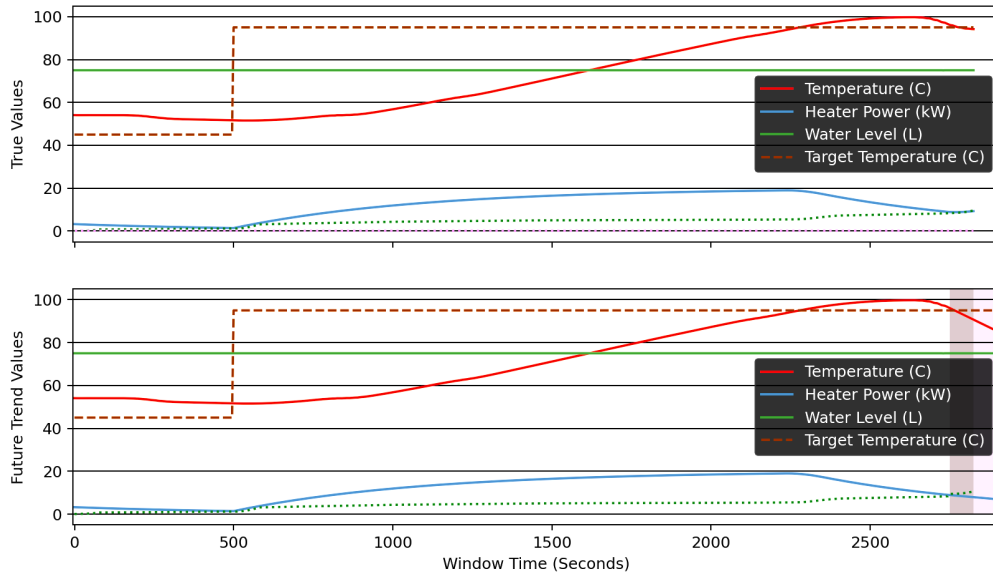


Figure 4.7: Overshoot with Same-Trend technique

4.2.2 Same-Trend

The Same-Trend technique assumes that the state at time t will be as follows:

$$t = (t - 1) + ((t - 1) - (t - 2)) \quad (4.3)$$

This technique offers surprisingly good accuracy due to the long response times and high linearity of the system; however, it does suffer from overshooting in long-range predictions, as shown below.

In Figure 4.7, the physical twin has plateaued from close to 100°C down to the target temperature of 95°C, while the digital twin predicts it would have dropped to around 90°C. At the end of the long-range forecast, this technique predicts close to 85°C. If the physical twin were allowed to continue, it would drop slightly below 95°C before trending back towards it. The Same-Trend technique causes this behaviour by being unaware of the PID driving the physical twin.

4.2.3 Observer Kalman and Eigensystem Realisation

The Observer Kalman Identification (OKID) (Juang, Phan, Horta, & Longman, 1993) and Eigensystem Realisation Algorithm (ERA) (Juang & Pappa, 1985) technique — known herein as OKID + ERA — uses the two namesake algorithms to create a state-space model of the system. This thesis implements this technique by creating many models and ranking them with a heuristic function to determine the best. The process of creating multiple models generates twenty models, of which only around ten are valid, and the heuristic function is a simple count of the number of stable poles.

OKID + ERA fully conforms to Monitorability, Verifiability, and Imaginability as it creates a model. It does not support partial model updates, however, and therefore partially conforms to Reevaluability.

Of issue with OKID + ERA is the inability to model the PID controller involved in the simulation. Where the two techniques described below are both capable of modelling an underlying controller as evidenced by the “with Control” suffix, OKID + ERA is not, and that leads to significant error.

4.2.4 Dynamic Mode Decomposition with Control

The Dynamic Mode Decomposition with Control (DMDc) (Proctor, Brunton, & Kutz, 2016) algorithm creates a state-space model that encapsulates underlying controllers, such as the PID controller in the simulation. As DMDc creates a state-space model, it covers the same conformity as OKID + ERA.

Demo, Tezzele, and Rozza (2018) wrote the DMDc implementation used in this thesis.

4.2.5 Sparse Identification of Nonlinear Dynamics with Control

The Sparse Identification of Nonlinear Dynamics with Control (SINDYc) (Brunton, Proctor, & Kutz, 2016) does not build a state-space model and therefore does not naively meet the Verifiability requirement. Given SINDYc’s construction, verification is possible.

De Silva et al. (2020) wrote the SINDYc implementation used in this thesis.

4.2.6 Recurrent Neural Network

Using Tensorflow (Martín Abadi et al., 2015), this thesis implements a recurrent neural network based on two, 1024-cell LSTM (Hochreiter & Schmidhuber, 1997) layers with dropout followed by two dense layers. The first dense layer has 256 neurons, and the second is the output layer that produces the predicted state.

Networks, in general, support Monitorability, Imaginability, and full Reevaluability. That is to say that they can model a system, imagine future states based on inputs and outputs, and both partially and fully recreate the aforementioned model.

Where networks fall short is in Verifiability. As shown in Gunning (2017), explainability in Artificial Intelligence is negatively correlated with predictive accuracy. While networks perform well, they lack explainability, and without explainability, it is exceeding challenging to mathematically prove stability and correctness.

As an aside, this thesis ran experiments using Tensorflow 2.2.0, which does not support weight updates after serialisation; however, Tensorflow 2.4.0 and greater do support it. This limitation does not prevent proving the concept but does preclude running tests in this configuration as tests occur after serialisation.

4.3 Conclusion

In conclusion, this thesis proposes a redefinition of Self-Learning that includes six core attributes. To the best of this thesis’ knowledge, no current definition of such a system that can adapt to unanticipated changes exists. A more in-depth discussion of stability measures is featured in the results chapters for supported models.

As depicted in Table 4.3, few current techniques have the requirements of a Self-Learning system. It should be noted that this table omits Monitorability and Falsifiability as the test harness implements these requirements.

Technique	Modelling	Verification	Imagining	Evaluation
Base-Case	–	–	Partial	–
Same-Trend	–	–	Partial	–
OKID + ERA	Yes	Yes	Full	Partial
DMDc	Yes	Yes	Full	Partial
SINDy	Yes	Yes ¹	Full	Partial
Recurrent Network	Yes	–	Full	Full

Table 4.3: Techniques and Self-Learning coverage

For state-space models, such as OKID + ERA and DMDc, there exist supplementary techniques to allow full conformity in Reevaluability. These techniques are discussed in Chapter 8.3 — Future Work.

Chapter 5

Self-Learning Evaluation and Analysis

This chapter discusses the efficacy of proposed Self-Learning algorithms and the retraining of system models. The inputs and outputs of the system — shown in Table 4.2 — along with the creation of an ancillary data, such as matrices, are described in the prior chapter.

For completeness, all test iterations used an Intel® Core™ i7-8700 and an NVIDIA® GeForce® GTX 1050 Ti. In terms of software, all tests used Python 3.8.

Do note that this thesis only performs experiments once, unless otherwise stated, as the simulator is deterministic and produces the same results every time. This determinism also applies in terms of different hardware: better hardware will only complete the experiments faster.

5.1 Performance over Simulation

In this section, this thesis tests retraining as used in Self-Learning. To achieve retraining, this thesis uses four falsification strategies: time-based, absolute error-based, signed error-based, and step error-based.

The time-based strategy uses the passage of time since the last successful retraining to determine if the model has become stale. Although this strategy could be dynamic, this thesis enforces a retraining every six simulated hours.

The absolute error-based strategy accumulates the magnitude of the system's error. This value always increases in the presence of error until the retraining threshold is reached. This thesis sets this threshold at 200 as it performed well in testing.

In contrast with the above strategy, the signed error-based strategy accumulates the error while respecting its direction, leading to some errors cancelling out. Because error can cancel itself out, this strategy has a comparatively low threshold of 100 when compared with the absolute error strategy. The threshold for this approach should be below the absolute error threshold otherwise it will never be reached; however, the exact value chosen by this thesis has no special relation to absolute error and a half of that metrics threshold was merely convenient and performant.

Finally, the step-based error strategy only considers if the last prediction was sufficiently far from the actual values. This strategy exists to quickly bail from wildly inaccurate models. This thesis sets the threshold for this strategy at two as it only triggers when the weighted error for a single step is large.

When evaluating the error, a multiplicative weighting — depicted in Table 5.1 — is applied to the delta between the true and predicted values before the Self-Falsifying implementation decides on falsification. Weighting the values emphasises the primary predictive attribute used in Self-Protection. Errors in Heater Output Power are not as valuable for falsification as the range of values is two orders of magnitude larger.

Variable	Multiplier
Temperature	1.1
Water Level	1.0
Power	0.01

Table 5.1: Variable Weighting

As the Base-Case and Same-Trend do not feature retraining, those results will not be in Table 5.3 below and are instead listed in Table 5.2.

Error Type	Base	Follow
Temperature	3320.3	1126.7
Water Level	399.33	581.47
Power	7366.1	175.33

Table 5.2: Error for non-retraining models

The graphs below use the total accumulated error. Even though the simulator resets the tracked error, the accumulated error remains the same.

Do note that Table 5.4 is not complete and similar values between Table 5.3 removed for brevity. A complete version of this table is available in the Appendix Table 8.1. For all removed OKID values, they shared an error of $2.45e9$ rather than $2.37e9$.

Strategy	DMDc	OKID	SINDYc
No Retraining	5622.5	1.36e9	3320.3
Step	5622.5	2.37e9	3320.3
Signed	2890.8	2.34e9	3320.3
Sign., Step	2552.5	2.37e9	3320.3
Absolute	2692.9	2.37e9	3320.3
Abs., Step	2624.1	2.37e9	3320.3
Abs., Sign.	2675.9	2.37e9	3320.3
Abs., Sign., Step	2629.5	2.37e9	3320.3
Time	2804.9	2.92e9	3320.3
Time, Step	2668.8	2.37e9	3320.3
Time, Sign.	2693.3	2.34e9	3320.3
Time, Sign., Step	2604.3	2.37e9	3320.3
Time, Abs.	2760.6	2.37e9	3320.3
Time, Abs., Step	2585.7	2.37e9	3320.3
Time, Abs., Sign.	2698.6	2.37e9	3320.3
All	2573.0	2.37e9	3320.3

Table 5.3: Temperature Error Across All Retraining Strategies

5.1.1 Retraining

Note that the recurrent neural network results are omitted from many tests due to poor performance after retraining. In part, this performance is likely due to insufficient training data being present during retraining. When retraining, the neural network has access to the last 1300 steps of history, and it segments this dataset further into a 9:1 split for training and testing data. Without more data, the recurrent neural network is unable to completely retrain and partial updates do not function as mentioned in Section 4.2.6.

Strategy	DMDc	OKID
Step	5622.5	2.45e9
Signed	2545.3	2.45e9
Sign., Step	2877.8	2.45e9
Absolute	2699.0	2.45e9
Abs., Step	2629.4	2.45e9
Abs., Sign.	2683.4	2.45e9
Abs., Sign., Step	2636.1	2.45e9
Time, Sign.	2742.7	2.45e9
Time, Sign., Step	2610.1	2.45e9
Time, Abs.	2766.7	2.45e9
Time, Abs., Step	2591.1	2.45e9
Time, Abs., Sign.	2706.1	2.45e9
All	2610.1	2.45e9

Table 5.4: Temperature Error Across All Retraining Strategies with minimum time before retrain

The rest of this subsection covers how this thesis handled retraining for DMDc. To measure retraining stability, this thesis uses a mix of visual metrics, such as the below Principal Component Analysis (PCA) graphs, and comparisons of retraining strategies. Tables 5.5 and 5.6 demonstrate the amount of a change seen by the DMDc matrices during retraining. For both tables, the rows represent input data and the columns represent the impact on the state variables as a result of $\hat{x} = \mathbf{A}x + \mathbf{B}u$ where \hat{x} is the new state, \mathbf{A} is the discrete-time state matrix and \mathbf{B} is the discrete-time transition matrix.

Given that the degradation of the system applies mostly to the heater power, the largest changes being those that modified the power state lines up. The two changing components of the underlying system are the heater power and interpolation rate degrading. Both of these components directly affect the dynamics of the internal power variable.

—	Temperature	Water Level	Power
Temperature	0.00154	0.00138	1.04
Water Level	1.10	0.899	19.7
Power	0.0000181	0.0000101	0.00934

Table 5.5: Total Change in DMDc A Matrix (Three Significant Figures)

—	Temperature	Water Level	Power
In Flow Rate	0.0690	0.155	2.54
In Flow Temperature	0.00148	0.00197	0.271
Target Temperature	1.50	1.23	25.1
Out Flow Rate	0.120	0.0416	17.8

Table 5.6: Total Change in DMDc B Matrix (Three Significant Figures)

For example, the widest difference for DMDc in Table 5.4 is from the No Retraining entry at 5622.5 to the Signed strategy entry at 2545.3. This result shows that retraining, in this instance, can reduce error by 54.7% in the best-recorded case. On average, the reduction falls to 48.9%, including both Step-based only cases where the improvement was 0.0%. Excluding those cases gives a reduction of 52.3%.

The Step-based strategy is not useless, however, and aids the Signed strategy in Table 5.3. Interestingly, this does not apply to Table 5.4. The helping is further illustrated in Figure 5.1 where the Step strategy causes some of the largest changes in A matrix values. In this figure, the size of the bars — both those at the bottom and those on the line — represents the magnitude of the change in the A matrix.

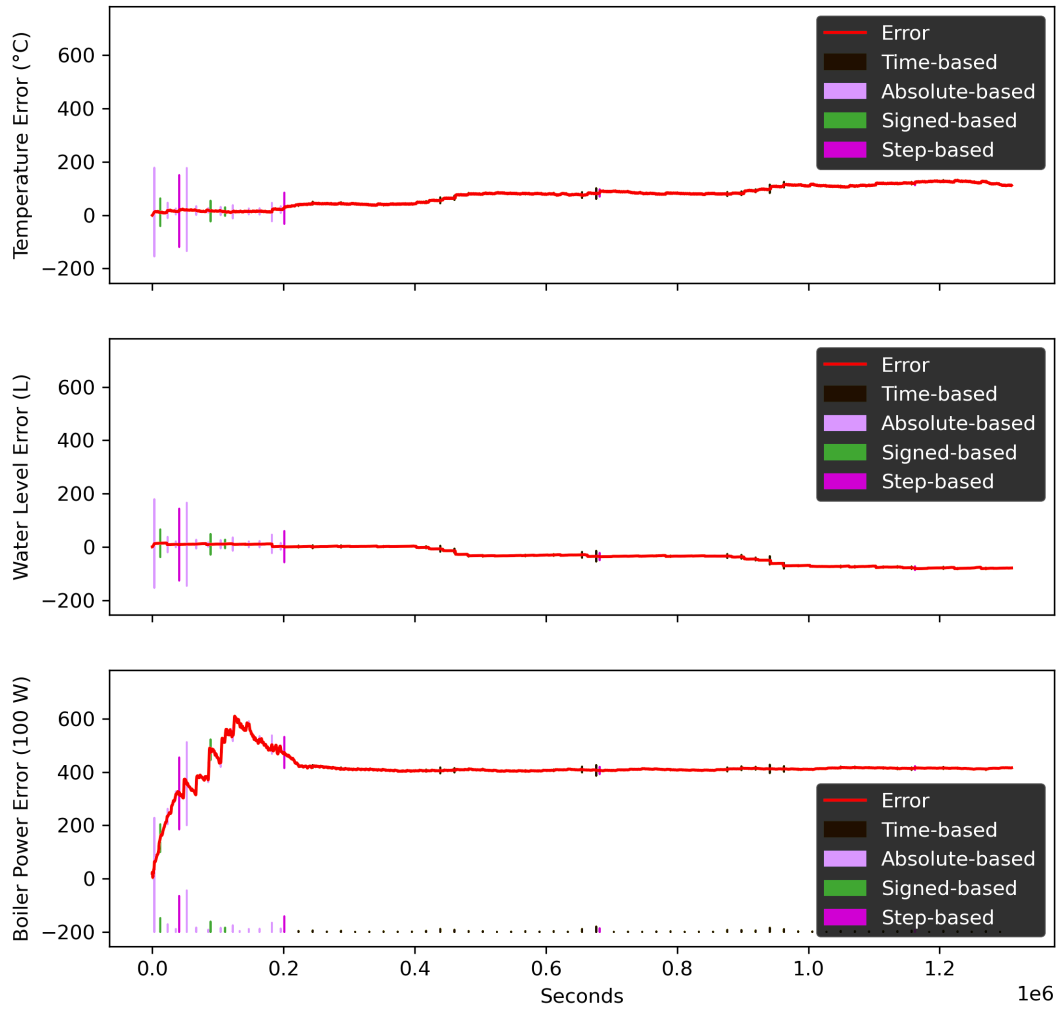


Figure 5.1: Signed Error with Absolute A Matrix Difference over time

To visualise how the A and B matrices for DMDc changed, this thesis performs PCA on said matrices and demonstrates the clustering that occurs during the end of the portion of the test runs in Figure 5.3. This area lines up with a trough shown in Figure 5.2. The equivalent absolute change graph and the visualisation for the B matrix is shown in Appendix Figures 8.1 and 8.2.

In all of the 3D figures, the scale to the left of the vanishing point ranges from 0 to 120.

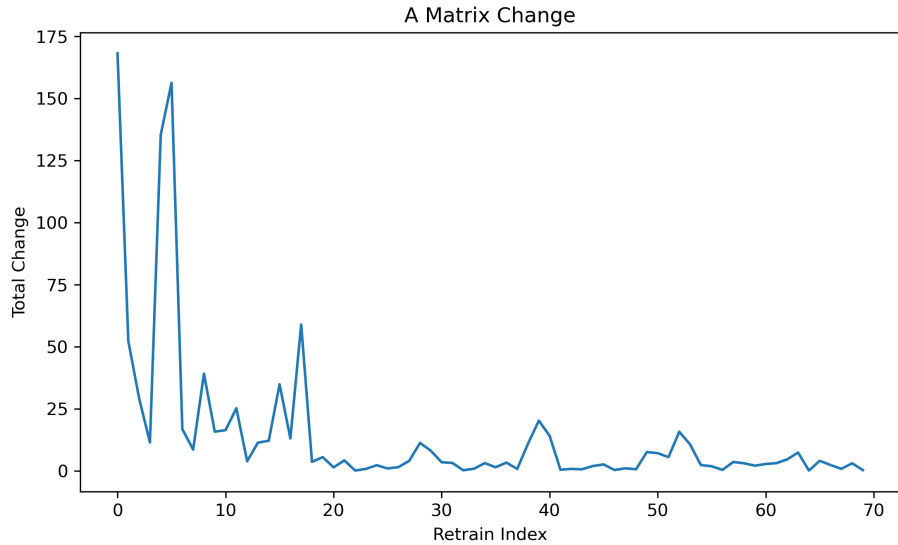


Figure 5.2: Absolute change in A matrix values between retraining

The index of each point is represented by a coloured dot that ranges from green to red where green is the lower indices and red, the higher. In both cases, the PCA visualisations show a clustering late on in the data. While the final few points of both cases show a trend away from the cluster, note that the graph scale on that axis is $1e-16$. Additionally, Appendix Figure 8.5 shows the result of a longer run. This run encompassed eight times the duration and shows a similar result. The graph is not identical because the longer run enforced a minimum time between retraining while the initial run did not.

This thesis also performed tests using the best performing retraining strategy, the Signed strategy. These images have been omitted because they show is largely identical to the more frequently retrained runs. The difference being that the Signed strategy only meaningfully retrained at the beginning of the run when the underlying system was changing and not after the system reached steady-state. This trend is shown in Appendix Figure 8.7.

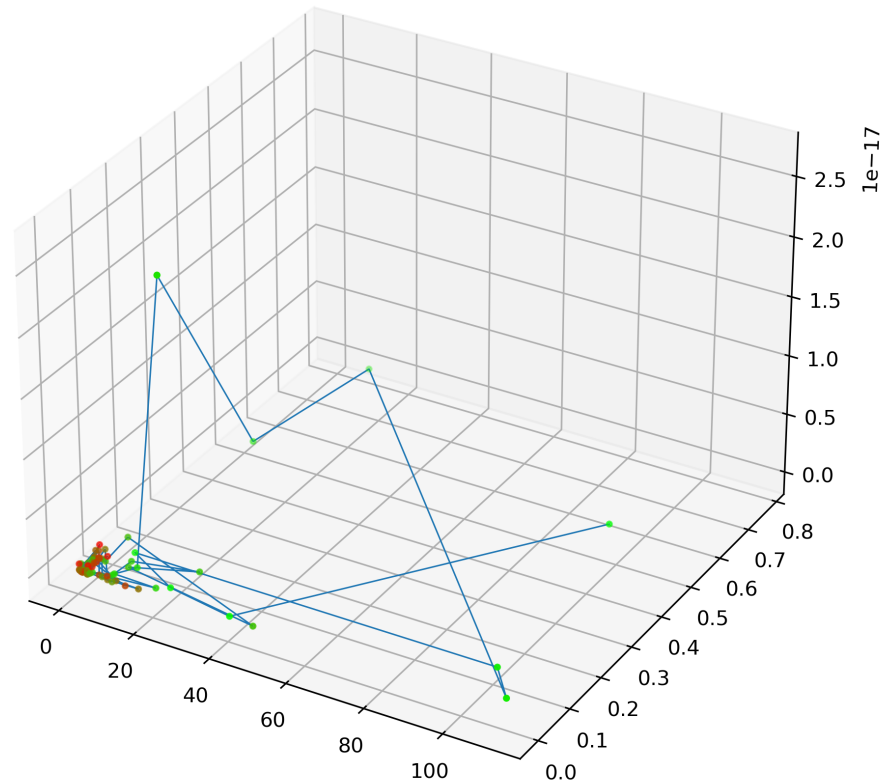


Figure 5.3: Principal Component Analysis of A matrix change over time

The aforementioned PCA visualisations show the PCA of the difference between each matrix after each retraining. Figure 8.8 shows the PCA of the A matrices themselves over an octuple length run. In this visualisation, the model converges on a small region after only a few iterations after changing fairly drastically over the beginning iterations, which lines up with the retraining magnitudes in Appendix Figure 8.9.

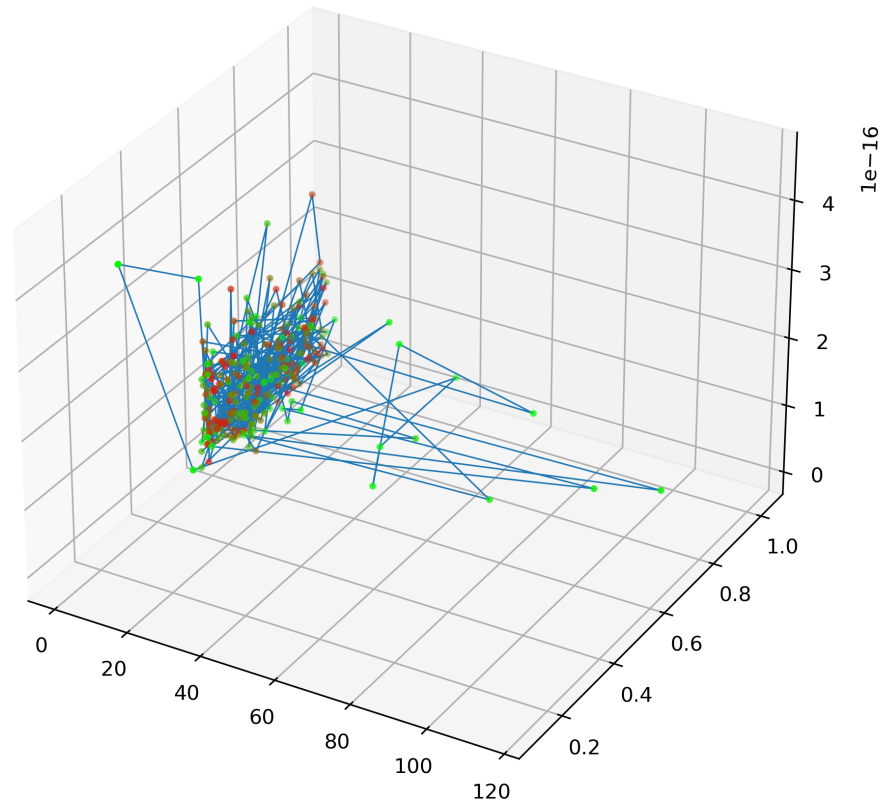


Figure 5.4: Principal Component Analysis of A matrix over time

5.1.2 Model Stability

In the framework outlined in the previous chapter, Self-Verification checks the model for correctness and stability. In this thesis, the verification step involves a human in the loop and does not check for correctness. Additionally, verification was only implemented for the DMDc model.

The human-in-the-loop can validate the model stability by checking pole placement and comparing the bode plots to the dominant frequencies in the input and output data. Figure 3.3 shows the pole placement graph for DMDc with a 0.1 Hz sampling rate. Figure 5.5 shows the magnitude half of the bode plot and the phase plot can be found in Appendix Figure 8.10. Both figures follow the same layout as Table 5.6.

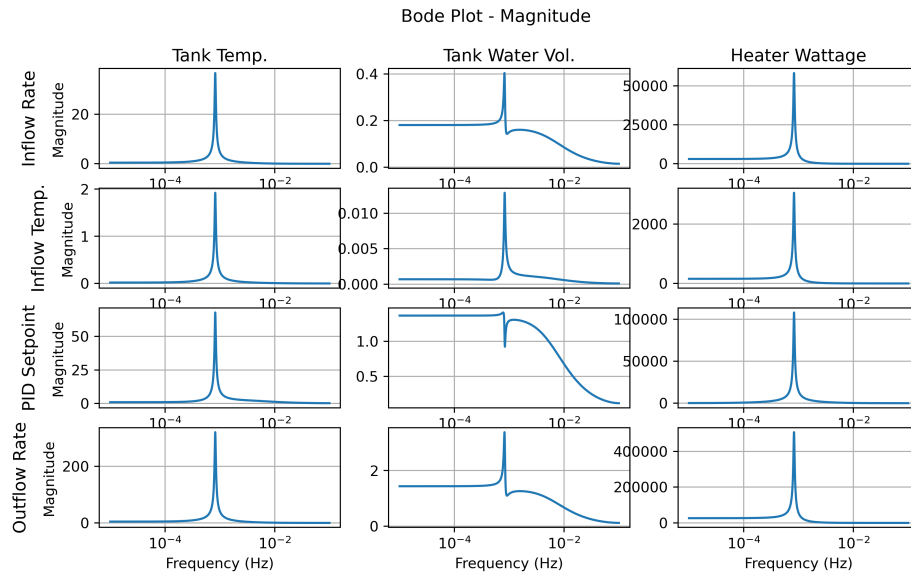


Figure 5.5: DMDc First Retrain Bode Plot — Magnitude

In Figure 5.5, there is a peak in magnitude of frequency response around 0.0001 Hz. In no cases do higher, non-aliased frequencies cause spurious spikes in response, and given that the dominant frequencies in the simulator reside at or around 0.00015 Hz, the model should be resistant to higher-frequency manipulation. The implications of this resilience and these results are discussed in the next chapter — Chapter 6.

As already mentioned in Chapter 3, another model stability check is pole location, and as depicted in Figure 3.3, which is of the same DMDc system as this discussion uses, the poles lie within a unit circle and are thus stable and converging.

5.2 Performance over Varying Oracle Depths

A discussion on the fitness-for-purpose of Self-Learning algorithms would be incomplete without also covering the ability to imagine future states. As such, this thesis touches on this in this section. To do so, this section presents three tabular summaries of the errors over a perfect oracle using multiple modelling techniques. For each table, the name of the best solution is bold, as are the individual best error mean and variance metrics. Unlike in Section 5.1, the following experiments used oracle depths greater than one, where a depth of one is, in effect, not an oracle.

The OKID and ERA model is omitted from these results as it generated a mean error of $7.4592e60$. As mentioned in Section 4.2.3, this result is not unexpected. The inability to model the controller causes significant drift from the ground truth, and this drift is accumulated over the entire oracle.

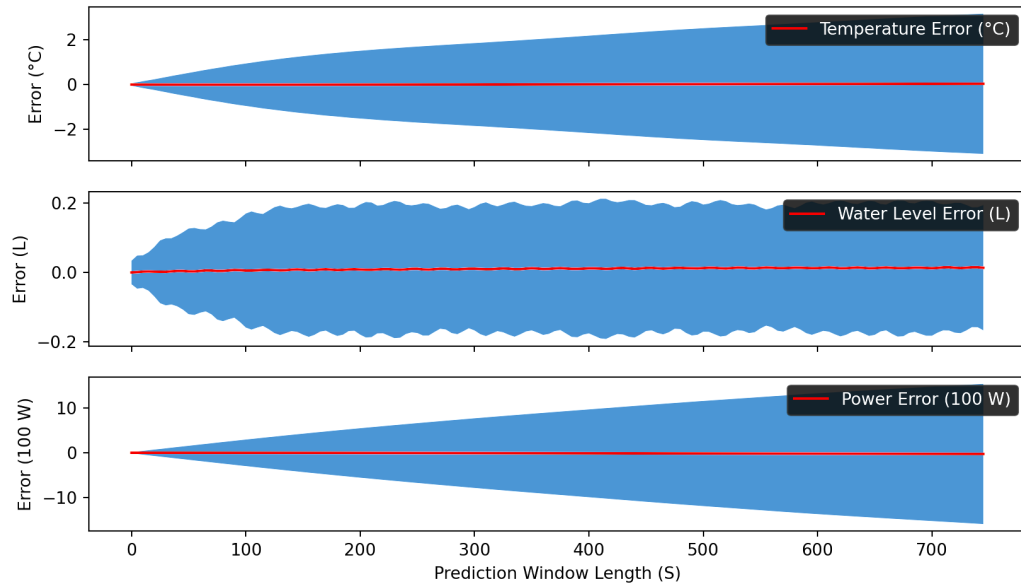


Figure 5.6: Error mean and deviation over the primary oracle (25th retrain)

These metrics represent area under the means and deviations — as shown in Figure 5.6 or earlier in Figure 4.5 — respectively. The data in this figure are shown as “DMDc 25” in the below tables. In Tables 5.7, 5.8, and 5.9, Integral Error represents the total area under the mean line, and Integral Error Variance represents the total area between the mean and two standard deviations. For integral values, lower is better.

Type	Integ. Error	Integ. Var.
Base	0.39207	426.64
SINDYc	0.39207	426.64
Same-Trend	5.7667	601.98
DMDc 0	271.81	295.12
DMDc 1	53.228	897.46
DMDc 2	34.106	994.42
DMDc 10	16.645	334.39
DMDc 25	1.9841	296.28
Recurrent	365.65	963.35

Table 5.7: Temperature Error (Five Significant Figures)

As in Figure 4.5, each technique listed below used 2048 samples for every one of the 150 steps into oracle. Each technique is tested with identical input and ground-truth data.

When considering the best solution of each table, this thesis adds the error variance to the mean twice and uses the lowest result. This result represents the total area contained within the 95th percentiles in the above-mentioned figures.

Note that the SINDYc and the base case have identical results. This is not a mistake: the test runs in question are repeatable and continue to yield identical results. Instead, it is due to the implementation of SINDYc seemingly modelling the same dynamics as the base case. The results here form a trend that continues in later results.

Type	Integ. Error	Integ. Var.
Base	0.11911	21.296
SINDYc	0.11911	21.296
Same-Trend	17.248	358.10
DMDc 0	280.37	454.11
DMDc 1	0.76662	17.031
DMDc 2	0.24930	17.914
DMDc 10	1.6035	33.936
DMDc 25	1.4677	26.190
Recurrent	50.112	17.035

Table 5.8: Water Level Error (Five Significant Figures)

The reason for the uneven progression of retrained DMDc state-space models is that each represents a different period of the simulation state. These periods are well depicted in Figure 5.1. The majority of the DMDc values depicted in the three tables mentioned above are from the initial period where the state-space model changes significantly. Model 25 occurs shortly into the stable region and is the third model that was triggered by the time-based strategy. This stable region is also visible as the cluster of Figure 5.3.

For Self-Protection, the most important state variable is the temperature, as this variable has a defined failure state. As shown in Table 5.7, DMDc model 25 performs the best in temperature error despite having neither the lowest integral error nor the lowest integral error variance.

Type	Integ. Error	Integ. Var.
Base	2.1151	1096.4
SINDYc	2.1151	1096.4
Same-Trend	10.069	1423.3
DMDc 0	101.10	1355.3
DMDc 1	366.24	9975.0
DMDc 2	335.97	10099
DMDc 10	352.22	4242.0
DMDc 25	17.711	1327.4
Recurrent	3590.5	1822.4

Table 5.9: Power Error (Five Significant Figures)

Note in these results the comparatively high performance of the base case and SINDYc. In all three variables, these two techniques performed the best in integral error and well in integral error variance, being close to the best performer in that metric as well. This result is likely due to the dynamics being slow enough that changes between samples are minor.

5.3 Conclusion

In conclusion, the results in this chapter present a compelling argument for the fitness-for-purpose of the framework laid out in Chapter 3 and the algorithms used in this chapter as applied to dynamic energy systems.

In testing the Falsification and Evaluation modules, this chapter has shown the performance benefits of retraining control models with a range of thresholds. Without retraining, the prebuilt DMDc model would have been unable to model the changing dynamics of the system. Without multiple strategies, the retraining may have been too sparse or too frequent to provide benefit, as evidenced by the difference between the rows of Table 5.3.

The ability to check model stability is a boon of using control theory, and as such, this chapter covered a rudimentarily implemented of Verification module. Despite the human-in-the-loop nature of this module, the system could be automated.

Finally, this chapter explored the ability of the tested digital shadow to Self-Imagine. This sufficiently accurate imagination allows the system to predict future states and enables a form of Self-Protection that is discussed in the next chapter.

Chapter 6

Self-Protection Evaluation and Algorithms

This chapter covers Self-Protection and input frequency handling. This thesis implements both of these features as human-in-the-loop interactions; therefore, discussions of automated actions are reserved for Section 8.3.5 under Future Work. The automated actions that do occur in the below sections are rudimentary but serve as an example from which to build atop in the future.

The first experiment covers the case where an input causes the system to enter a failure state. In 1982, a Siberian gas pipeline exploded as a result of a cyber-attack that caused over-pressurisation. This attack involved resetting pump speeds to maintain flow rates greater than the controller intended, leading to a failure of the physical pipeline (Gazula, 2017). While there are benefits in connecting plants to the internet, it is cases such as these that show why detecting and preventing dangerous trends in a system is required.

The second experiment covers the need to detect trends that degrade a system without directly leading to failure. Motivated by the same cause as the first experiment, the second analyses the frequencies of input and output data and how that data influences the system. The motivation for this experiment is

the infamous case of Stuxnet. In 2010, the Iranian nuclear reactor at Natanz suffered damage to its centrifuges caused by excessive oscillations in speed. The cause of these oscillations was a computer worm named Stuxnet (Gazula, 2017). While the centrifuges were not directly put into failure state as per the Siberian pipeline case, the oscillations in speed did nevertheless cause damage.

Because the digital shadow of the simulator is capable of Self-Learning, disturbances arising from the system are minimised and the impacts of both of these experiments are isolated from underlying changes in the system.

In previous tests and as mentioned in Section 4.1.3, the simulator ran ahead of the real predictions to provide true input and output data as if the forecasting were perfect. A problem arises, however, in the event that control actions need to be performed since the simulator has already processed beyond the time step for which the control arrived. To handle this issue, while retaining the accurate input and output data, the simulator recomputes future states every time a control action is taken. In essence, the simulator becomes able to time travel. When a control signal arrives, the simulator reverts time to the appropriate time step, applies the action, and returns back to the future from whence it came.

The impact being that for experiments in this section, an external component can modify the system. In practice, the time travel invalidates the future data and the simulator builds a new history that encompasses any changes resulting from the external influence.

6.1 Signal Processing

Before discussing the Self-Protection techniques, this thesis introduces the concept of Nyquist Frequency and mentions two transforms used by the second Self-Protection technique.

6.1.1 Nyquist Frequency

In signal processing, the Nyquist frequency is the frequency above which signal aliasing occurs in an input signal. It is defined as one half the frequency of the sampling rate. Provided the highest frequency in a given input signal is below the Nyquist frequency, it is free from aliasing. An example of this aliasing is covered in Section 6.2. There are some cases, discussed in Chapter 8, where a signal can be reconstructed with fewer samples.

6.1.2 Transforms

The two transforms used in this thesis are the Chirp Z-Transform (CZT) and the Fast-Fourier Transform (FFT). Note that both of these methods are discrete-time transforms as the systems in this thesis are in discrete-time.

The Chirp Z-Transform takes a time-domain signal and expresses it in the frequency domain. It is a more general form of the aforementioned FFT that makes a very similar transform. The CZT can be used to compute pole locations and the region of convergence of a transfer function (Shilling, 1972); however, this thesis did not complete the computation of the latter for the input data, so the FFT alone would have been sufficient.

6.2 Sampling and Command Frequency

The sampling rate for the simulation is 0.2 hertz. Whilst the sampling rate is customisable, this chapter uses one sample every five simulated seconds. Due to an oversight in designing the simulator, the command rate is currently fixed to the sampling rate. The import of this limitation makes it worth mentioning ahead of the two Self-Protection methods as it affects them both.

For the first method, the limitation means that any actions taken, such as moving the target temperature, are processed once per five seconds. Largely, this limitation does not adversely interfere with the protection technique.

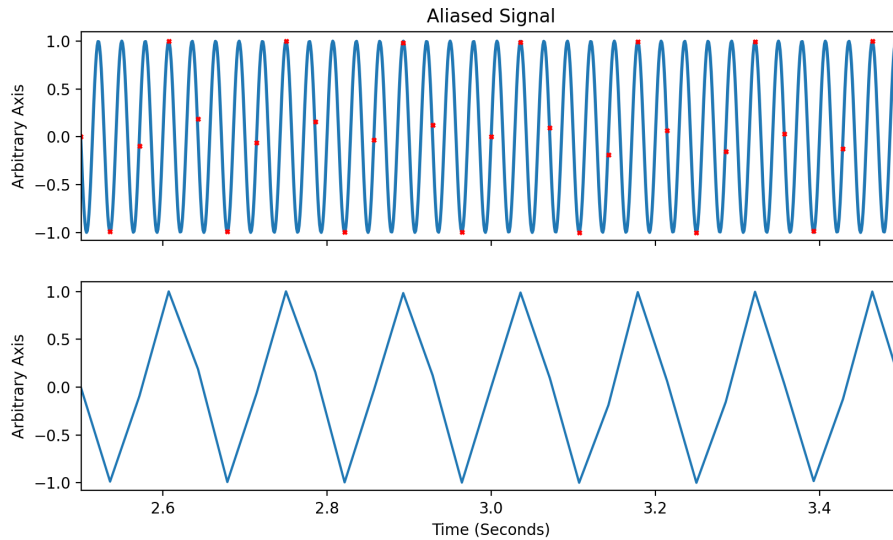


Figure 6.1: Example of an aliased signal

However, the second method is impacted more. The inability to issue commands faster than the sample rate limits the ability to create oscillations above the aliasing limit of the sampling rate with meaning. It is possible to feed a higher frequency signal into the system, but as the aliasing in issuing commands occurs at the exact time as in sampling the system, it is somewhat meaningless as a signal above the Nyquist frequency is aliased by definition, and as such, no longer controls the system as intended.

Figure 6.1 shows the impacts of an aliased signal. The upper diagram shows the raw signal with red crosses showing the sampling points at one-fifth the frequency. The upper signal is oscillating at five times the Nyquist frequency of the sampling rate. The lower diagram shows the reconstructed waveform, and it is this signal that the simulator can read. The issue with the command frequency being limited to the bottom diagram's sampling frequency is that it is impossible to issue commands that achieve the upper diagrams oscillations.

For this system where the sampling rate is 0.2 hertz, the Nyquist frequency is 0.1 hertz.

6.3 Heater Failure Conditions

In the heater system, there are three internal variables with temperature being the only variable that can result in a failure that Self-Protection can affect.

For water level, the heater implements “hardware” high- and low-level cutoffs. If the water reaches the high-level alarm, the inlet valve is instantly closed. If the water reaches the low-level alarm, the outlet valve is instantly closed. In both cases, the valves return to normal operation when the cutoffs are no longer in effect.

For heater power, it is possible that the boiler sufficiently degrades — as is mentioned in Section 4.1.5 — or outright fails. Since this variable is the main control axis, if it is uncontrollable, then the entire system is uncontrollable: a heater without sufficient energy entering will be unable to reach the target temperature. The only possible action to take in such a scenario is to warn a human operator.

In the case of temperature, allowing the water to reach 100°C would be considered a failure mode. The simulator has three ways of modifying the temperature by way of modifying the heater. The first is changing the PID target temperature; the second, the heater level; and the third, the raw heater output.

In the experiments below, without intervention, the water temperature would reach a failure at around 3200 seconds due to the PID controller being slightly modified from the one used in Chapter 5. This chapter doubles the PID’s integral weighting as otherwise the temperature grazes, but does not exceed, 100°C.

6.4 Temperature Prediction Handling

This section discusses the only automated portion of Self-Protection tested in this thesis. The Temperature Prediction method works by predicting the future using both oracles mentioned in Section 4.1.3. Both of these oracles predict 15 steps for a total of 30 steps or 150 seconds. As mentioned in that section, the first oracle has perfect information and the second assumes no changes to the inputs or outputs in the system. Both oracles are used in full when accumulating a variable referred to herein as the limiter. This variable is used by each method to limit the heater and prevent it from exceeding the maximum safe temperature.

When the end of the second oracle exceeds the safe maximum temperature, the simulation increases the limiter by 32. Additionally, for every predicted step where the temperature exceeds the safe maximum, the simulator further increments the limiter. When no prediction exceeds the safe maximum, the variable decays by 10% every step. This decaying is visible in Figure 6.2, where the target temperature interpolates back to the set temperature. This smoothed response is present in all methods but is most evident here. The amount the limiter is increased by is arbitrary, and the response it causes is further modified by the methods. For example, the PID method scales the limiter by two when reducing the target temperature, but the heater level method uses the same limiter variable to reduce the heater's output in hundred-watt increments.

The first two methods have an almost functionally identical outcome: both cause the heater to reduce output while respecting relevant delays. Setting the PID target to zero will quickly produce a zero heater level, but the reaction will be delayed by the PID in addition to the delay in the heating element acting normally and could lead to a reaction that is too slow to prevent failure. Although the simulated version of this method — shown in Figure 6.2 — reacts

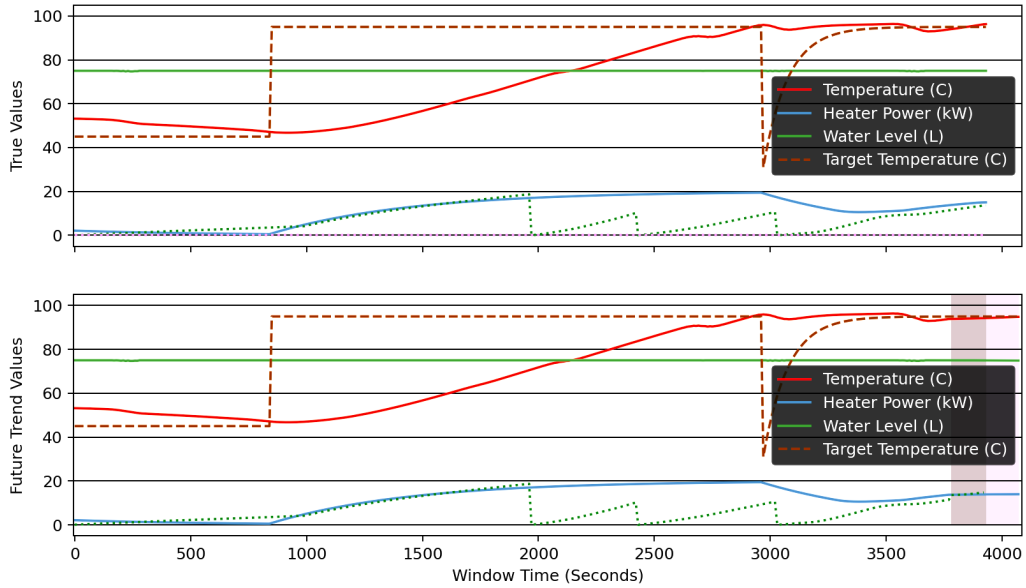


Figure 6.2: Self-Protection by modifying PID target

fast enough. Directly modifying the heater level is an effective way of reducing the heater power, and it bypasses the PID delay while remaining true to the simulation.

Lastly, modifying the heater output directly is only useful for one situation: modelling an emergency stop. Cutting off the heater immediately has no delay period but also no controllability. When allowed more granularity than a binary on-off, this method — as depicted in Figure 6.3 — behaves much like the aforementioned heater level method minus the delay.

In the case of the PID, the temperature levels off at the target temperature of 95°C , whereas with both the heater level and the raw heater output, the levelling off occurs above the target temperature. The former method achieves the target temperature better in this case as the Self-Protection component is modifying, by proxy of briefly moving the target, the PID controllers state. The other two methods modify the underlying system and leave the PID controller unmodified. It will eventually converge on the target temperature, but the modified integral weighting makes this process take some time.

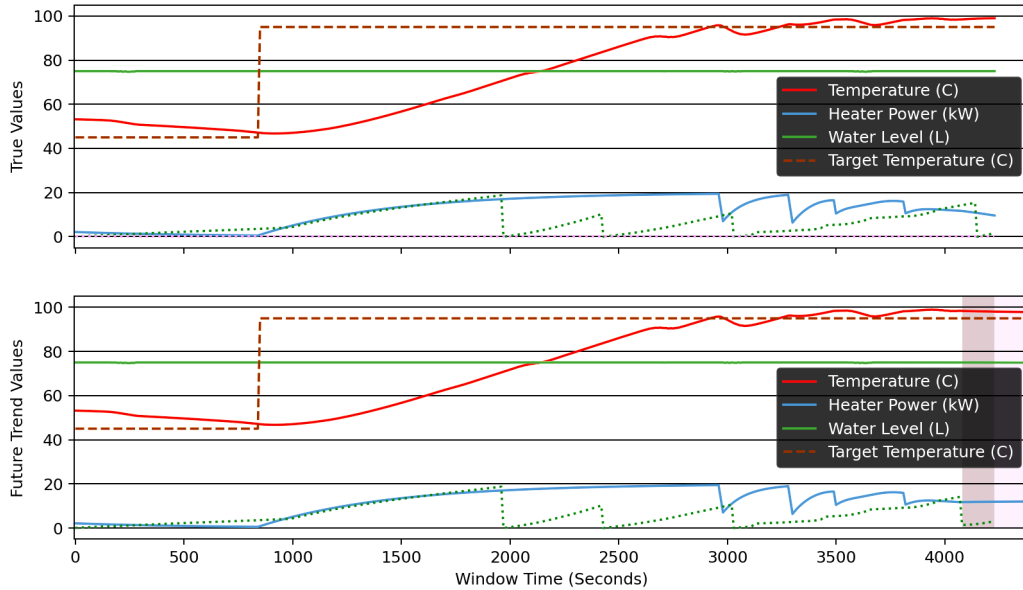


Figure 6.3: Self-Protection by modifying raw heater output

It should be noted that the Self-Protection shown in this section is a proof-of-concept to validate the proposed design. The results are not meant to prescribe how Self-Protection should be implemented, merely what it may look like. As mentioned above, the methods are basic: they use what is in effect an untuned integral controller, but they do still prevent the system from exceeding safe limits. Additionally, due to the sampling frequency, it is not possible to only measure the system in flight. One measurement could be sufficiently far away from failure to satisfy a check, only for the system to encounter failure before the next reading occurs.

6.5 Input Frequency Handling

The second form of Self-Protection implemented is the input frequency handling experiment. This type of protection is aimed at oscillating changes to the system's target temperature. By quickly moving the set target, the system may itself oscillate in a way that would cause damage, and although this

example system is not affected in terms of degradation by repeated changes, others are. In this experiment, the oscillation moves the target temperature between 25 and 125 degrees at 0.05 hertz.

This experiment generates a graph of magnitude and phase for the input and output data. Because retraining may modify the \mathbf{B} matrix, the graph data is the product of the \mathbf{B} matrix and inputs, as given in Equation 6.1 as an array of $\hat{\mathbf{B}}$ vectors, where \mathbf{B} is the \mathbf{B} matrix, u is each input-output vector, and $\hat{\mathbf{B}}$ is each resulting vector.

$$\hat{B} = B \cdot u \quad (6.1)$$

The array of $\hat{\mathbf{B}}$ vectors then undergoes a Chirp Z-transform (Rabiner, Schafer, & Rader, 1969) into the frequency-domain. The magnitude and the phase of the frequency-domain data create Figures 6.4 and 6.5. The magnitude graphs have been omitted for brevity as the disturbances as easier to spot in the phase plot.

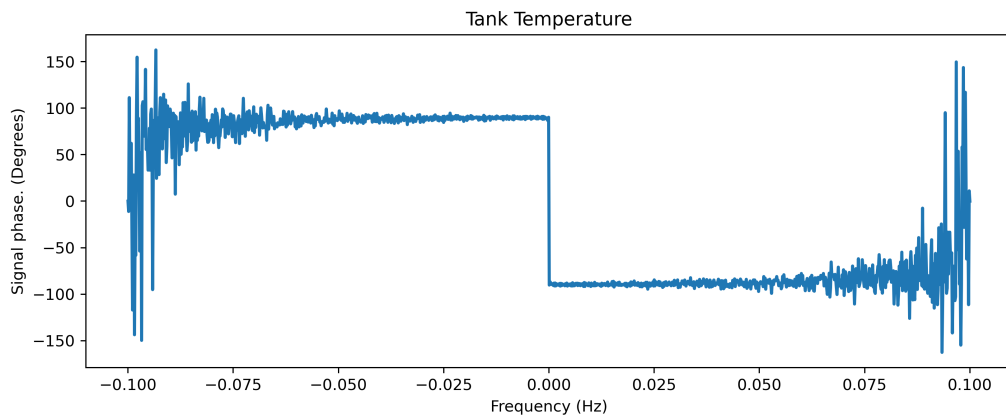


Figure 6.4: Phase diagram for tank temperature

The primary differences between Figure 6.4 and Figure 6.5 is the expected spike around 0.03 hertz to around 0.05 hertz. These are the phase graphs, so there isn't much to analyse here aside from where the oscillation frequencies sit. This thesis detects these frequencies using a human-in-the-loop design, though automating the process is possible. Analysing a known-good signal for a ground-truth would provide enough context to determine anomalous frequencies, especially if only frequencies that cause instability in the model need considering.

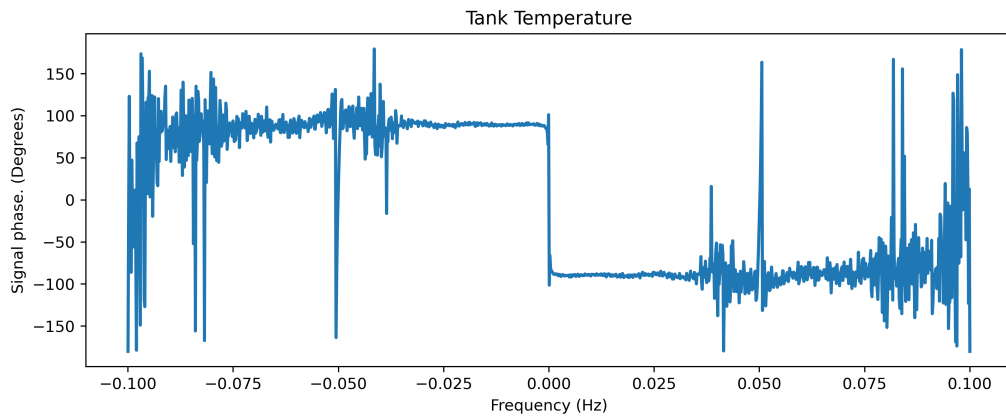


Figure 6.5: Phase diagram for tank temperature with interference

Detection is but one half of protection a system, however. The protection component works by removing frequencies within a range. In this case, that range is 0.035 to 0.055 hertz. To remove the range, the simulation converts the input data from the heater into the frequency domain and sets the frequencies within the range to zero before transforming back to the time domain. To remain consistent with the z-transform performed to detect the frequency, the removal function transforms the data using Equation 6.1 before removing the frequencies. The data cannot be immediately returned as it is still in $\hat{\mathbf{B}}$ form; however, the original data format can be restored by reapplying Equation 6.1 with the inverse of \mathbf{B} , \mathbf{B}^{-1} .

Figure 6.6 shows the effects of this transform. Both sides of the graph in the frequency domain are symmetrical, so the positive frequencies are omitted. As mentioned in Section 5.1.2, the dominant frequencies of this heater reside at around 0.00015 hertz, which shows as a comparatively tall spike centred around zero. In the left portion of the figure is another, smaller peak at 0.05 hertz that disappears in the right portion. The presence of the oscillating frequency also causes minor distortions that are most noticeable between the two peaks.

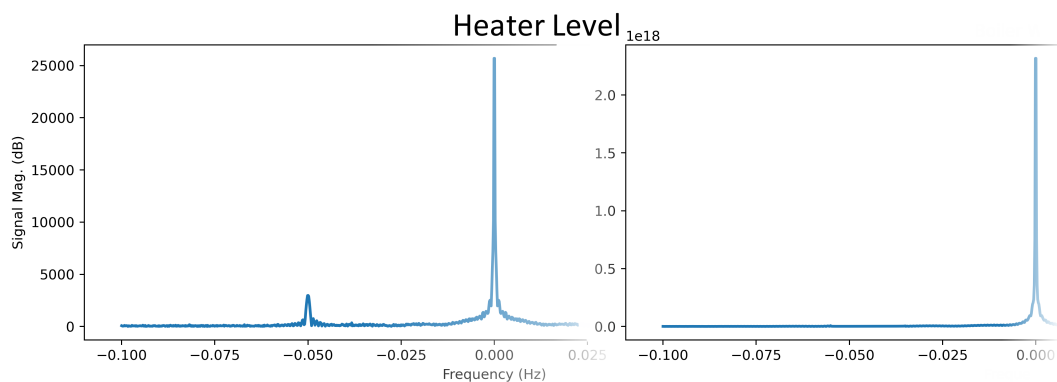


Figure 6.6: Heater Level of $\hat{\mathbf{B}}$ in frequency-domain showing oscillations (left) and filtered frequencies (right)

This section does not modify the input and output signals generated for use in the heater directly, so no accuracy results can be provided as this method is not applying anything to the underlying system. In an ideal world, it would be, but for now, that is left as future work and is discussed in Section 8.3.

6.6 Conclusion

The Self-Protection methods outlined in this chapter are somewhat basic but provide a starting point for further research. Firstly, the prediction Self-Protection method demonstrates a proof-of-concept fail-safe for use with Digital Twins. Even in a contrived and primitive form, the system handled failure cases and kept the system under control. The oracles used in Self-Protection can predict further into the future than the 150 seconds. The accuracy of an increased oracle size was shown earlier in Figure 5.6, and it relevant due to the non-linear accuracy across the window. For example with DMDC, if the window length increased from 150 to 300, a linear increase would cause the 95th percentiles to double. In the aforementioned figure, this scaling does not occur, and the 95th percentiles fall short of doubling.

Secondly, the frequency filtering Self-Protection method discusses the use of input filtering for improving the resilience of the system. As mentioned above, the inability to apply filtering to the underlying system limits any conclusions that can be drawn about the efficacy of this method for that purpose, though the filtering does clean up the signal as expect and so should be applicable.

Interestingly, the oscillations introduced in Section 6.5 have an adverse impact on the functionality of Section 6.4. The oscillations cause the prediction to momentarily exceed 100°C, which spuriously triggers the Self-Protection mechanisms. In actuality, the system continues with only minor fluctuations in the heater level. Because the filtering technique is applied before the predictions are computed, the removal of these frequencies stabilises the predictions and removes the spurious activations.

Chapter 7

Engineering Design Discussion

This chapter discusses some of the shortcomings of the design of both the simulator and the overall Self-Learning architecture and proposes further engineering design methodologies going forward. These updated architectural designs are based on the limitations of and the lessons learned from the created system in this thesis. Note that this chapter only discusses architectural limitations that are relevant to the future development of digital twins. Further discussion of more general, out-of-scope limitations is saved for the Section 8.2.

7.1 Simulation

The use of a simulator in this thesis serves as an efficient and highly effective prototyping tool. For Self-Learning systems, rapid prototyping is a valuable tool as building a Self-Learning system is deeply involved. Doubly so when factoring in the complexity in configuring a real-world steam boiler for this task. Depending on the exact requirements of each physical twin, certain methods may be unsatisfactory. A robust simulator allows the methods and overall implementation of the Self-Learning system to be tested before attempting to twin a real-world system.

The benefit here is that twinning a real-world system is far from trivial. Not only does testing on a real energy system, whether a boiler or an automotive engine, burn fossil fuels, it can also result in the downtime of the system. Each iteration of the Self-Learning system may require additional operation and downtime, something that is unlikely to be granted in a production environment.

Furthermore, for Self-Protecting systems, the state the physical twin must be in to cause a self-protection event is inherently risky, and the uncertainty of the system or the implementation makes this proposition doubly so. To deal with this risk, the physical twin would need isolation and rigorous risk assessment before each test, in addition to code correctness validation.

These challenges slow the creation of self-learning digital twins to the extreme. By contrast, simulators enable much faster iteration and the ability to test in a safe environment. They do add an additional threat to their validity insofar as a twin of a simulator can only be as accurate as the simulator, but nonetheless, they allow that testing and validation be carried out in a safe environment.

From a design standpoint, a simulator used in the creation of digital twins benefits from the ability to “time travel”. In Chapter 6, this thesis mentioned that the simulator was extended to enable the simulator to both generate future data for the first forecast oracle and to act in the present, and by acting in the present, to change said future data. In Chapter 6, this extension involved the simulator being able to restore past states and recompute the future. While the implementation used in this thesis was not efficient, the algorithm itself is as the Self-Protection system seldom acts on every sample. For the majority of samples, the system observes and only incurs any computation penalty when it must act. This design further enhances the performance of the simulator.

7.1.1 Simulation Design Limitations

The simulator, as implemented in this thesis, has several limitations due to the nature of its ad-hoc construction. None of these limitations affect the results of the thesis and serve instead to make further development harder. This section addresses these limitations

The simulator was built around the core concept that it would be modelled by a digital shadow. Largely, the design functioned in that regard; however, limitations become more apparent when adding Self-Learning and Self-Protection features.

The simulator referred to in the prior chapters is actually two subsystems that operate together: the engine system that manages the registered components and handles advancing time, and the heater simulation module that registers with the engine and handles heating virtual water. A core premise of this design was allowing multiple components to pass data between each other, and although this functionality is not used in the thesis, it is used in the improved design.

Firstly, the simulation module should be split apart as, as it stands, it is a monolithic design that encompasses too many responsibilities. Chiefly, it handles not just the dynamics but also the generation of disturbances and input data. This design became a considerable obstacle for Self-Protection in conjunction with the inability to save the simulator state, as discussed later.

Self-Protection must be able to influence input and output data to function. As the simulation module sits very deep in the overall architecture and acts immediately on the generated inputs, it was not feasible to modify it without redesigning the system.

A better approach is to create a second module — the disturbance module — that computes input and output data with an API sufficient to modify how input and output data look before the simulation module acts on that data.

Secondly, the system should gracefully save its state and should be able to predict future states without impacting the current state. The simulation module does not support either, but Python’s pickle utility can work around this limitation. Because pickle serialises the entire object, however, it makes influencing input and output data impossible as it would be immediately overridden by the object being deserialised during a rollback.

The final change to the design would be explicit support for modification. Some supporting functions support modifying the system by calling user-defined functions. These functions allow for rapid prototyping beyond that of using an inherited model of simulator alone and extending this functionality to the entire system doubles down on this flexibility.

The limitations aside, however, the simulator enabled this thesis to conduct the work of the previous two chapters. Connecting to and actuating a physical energy system is a complex endeavour, and one best left for future work where resources are less constrained.

7.2 Overall Architectures

The lessons of the past two chapters identified a series of changes to the architectures of Self-Learning and Self-Protection. Additionally, there are further changes to how the requirements of the system are elicited. Before moving on to discuss the changes in the architectures, this section covers the requirement changes.

The use of simulation in this thesis enabled the rapid prototyping to elicit further requirements of Self-Learning and Self-Protection. Due to the time investment required to get a physical energy system operational for these experiments, and the inevitable scheduling issues that occur with a shared resource, the use of a simulator allowed for an implementation that was iteratively tested during development. Without a simulator, the difficulty of this iterative testing would slow to an extreme. Additionally, the simulator can host future optimisation experiments without impacting the running energy system. Note that these experiments refer to testing other configurations of adaptive digital twins and not process optimisations that would ideally be tested on the digital twin itself.

Furthermore, the simulator allows for exploratory dives into the requirements of the system. Initial requirements for Self-Learning were considered when this thesis architected and implemented the simulator; however, iteration is a vital step in developing software designs, and the inability to test quickly would have hindered that. During the prototyping and testing of the simulator with both Self-Learning and Self-Protecting, this thesis proposes additional requirements that are addressed in the updated architectures.

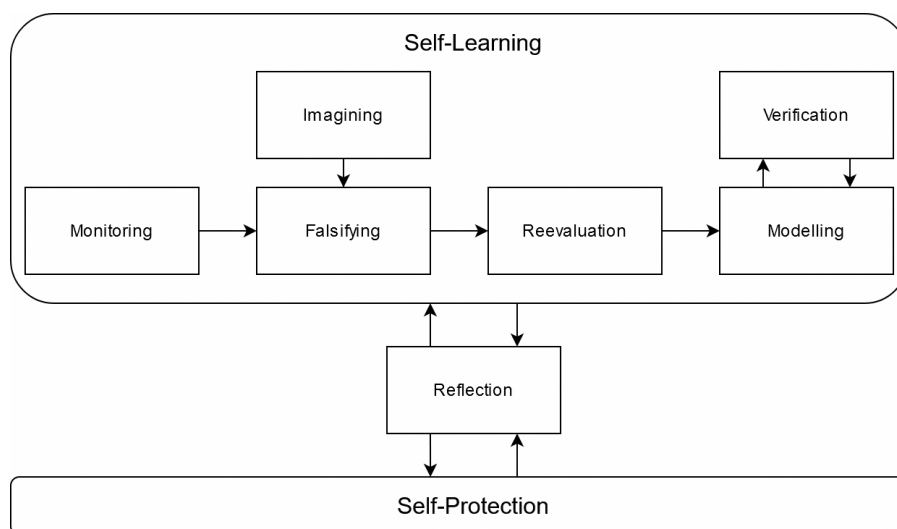


Figure 7.1: Updated Architecture as per Figure 3.2

Figure 7.1 shows an updated overall self-adaptive system architecture for Self-Learning and Self-Protecting adaptive digital twins. The **Reflection** module between Self-Learning and Self-Protection is a meta-control module that enables further adaption at a Self-Learning and Self-Protection configuration level. For example, the reflection module could switch the implementation of the modelling module if the overall system is not performing well.

From here on in, this section leverages the lessons learned in the prior chapters and proposes updated, fit-for-purpose Self-Learning and Self-Protection architectures.

7.2.1 Self-Learning

The Self-Learning framework remains largely the same, but for clarity, Self-Protection is removed from the diagram. Self-Protection is its own bespoke subsystem, and its presence detracts from the presentation of the core idea: Self-Learning.

In the amended Figure 7.2, the Monitoring module is a bespoke subsystem that reads from the physical twin. This decoupling helps facilitate the solution for the limitation raised in Section 6.2. By making the Self-Learning system — for which the Monitoring module is the entry point — a separate system, the locked sampling rate issue disappears by design. A second and potentially more impactful benefit is that this decoupled system mirrors the real-world design goals of Digital Twin-based Cloud Machine Interface.

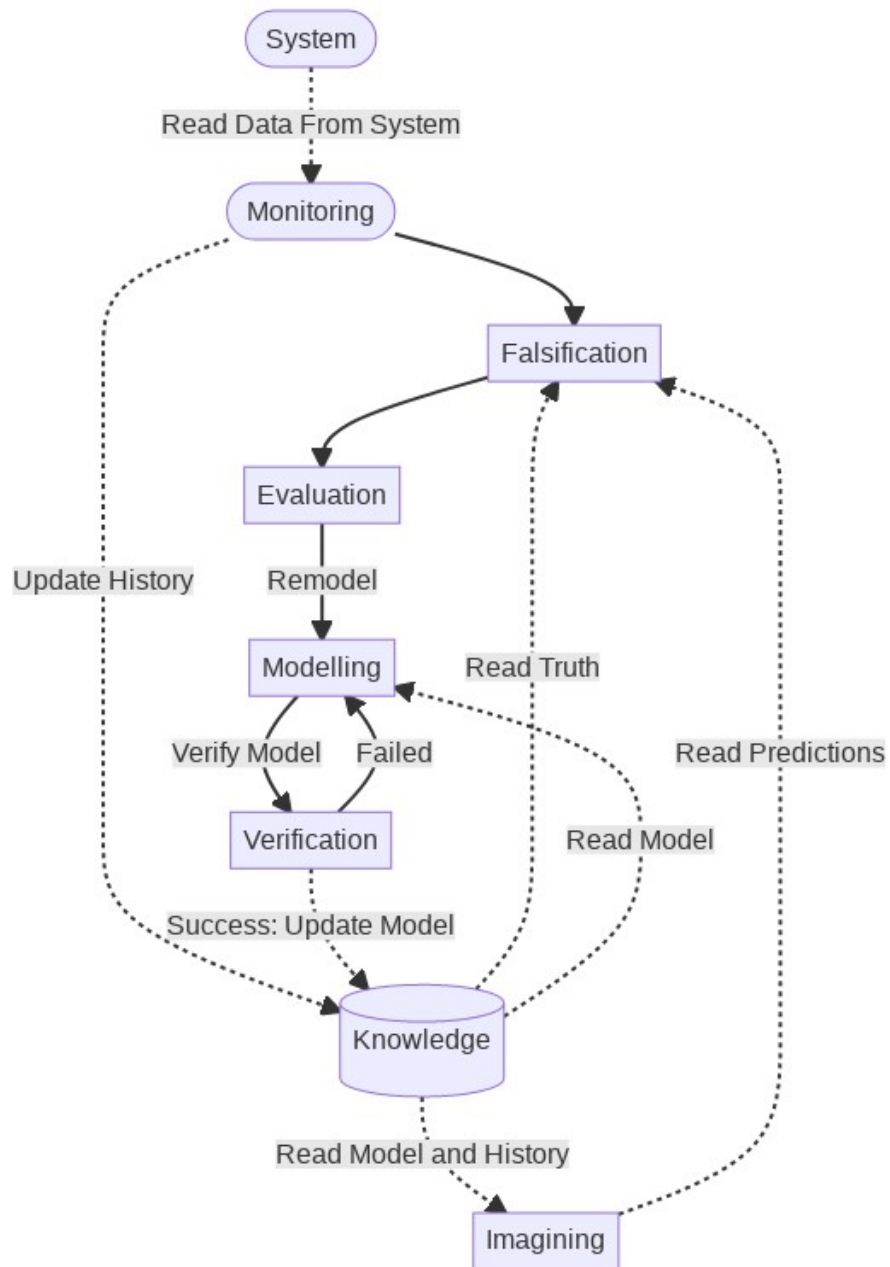


Figure 7.2: Self-Learning Architecture

In contrast to the implementation in Chapter 5 where the physical system drove the Self-Learning modules in synchrony. The design in this section is not synchronous with the physical twin, which mirrors what a real-world implementation would necessarily be.

7.2.2 Self-Protection

The architecture for Self-Protection is not discussed in detail in Chapter 3. So instead of refining the Self-Protection design, this section proposes a general Self-Protection architecture in Figure 7.3. This architecture builds on CoBRA, as mentioned in Section 2.3, with regards to the filter stack on the physical twin's target and sensor data. This stack can contain any arbitrary number of filters, including none. As in CoBRA, this stack could include a Kalman filter, or as in Section 6.5, it could include a frequency filter.

Additionally, Self-Protection in the form shown in Section 6.4 is split into two components, including a second stack of modifiers. As with the previous stack, these modifiers are executed sequentially by the Protection module, which then updates input filters or engages protection actions in physical twin.

In difference from the implementation in Chapter 6, the Protection module is now a bespoke entity that sits decoupled from the physical twin and the Self-Learning system, though it does rely on the model from the Self-Learning system.

7.3 Software Development Process

This thesis posits the following software engineering process for the development of adaptive digital twins. Firstly, the data that can reasonably be collected from the physical twin should be identified including how frequently this data is available, how reliable it is, and what form it takes. The monitoring implementation requires this information if it is to be successful.

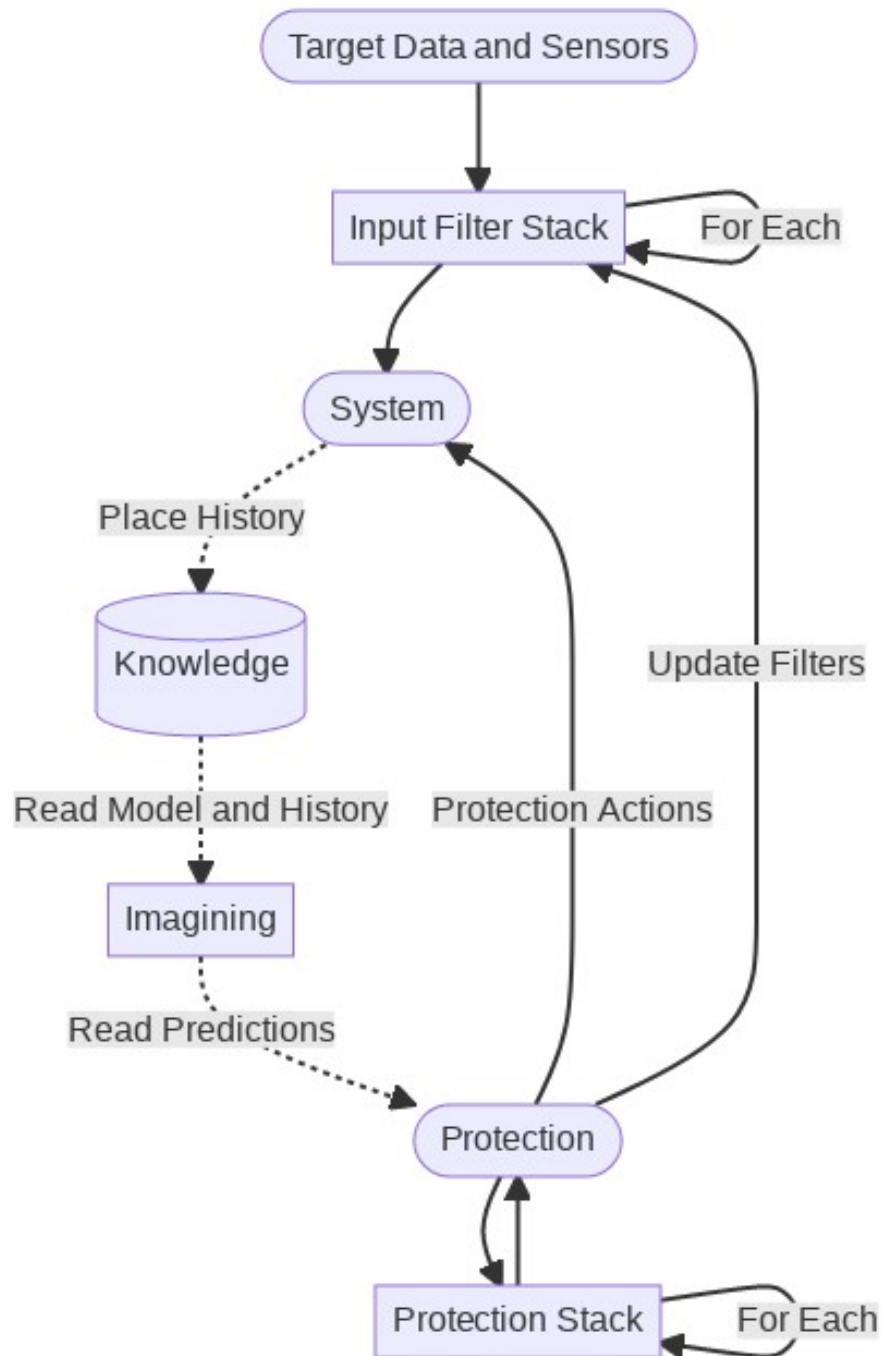


Figure 7.3: Self-Protection Architecture

Secondly, the simulation of the system should be constructed that mirrors the physical twin. At the same time, effort into preparing the physical system for twinning can begin in parallel. This step enables the software development process to begin in earnest. This thesis assumes that both preparation and iteration time for the physical system are slower than that of the simulation.

The software process can iterate in the design and implementation of the modules put forth in Figures 7.2 and 7.3 based on the simulation. This process acknowledges that the simulation is unlikely to be perfect or to capture the complexity of the physical system in full, but it does allow for the bulk of the software development process to occur where without it, this process would not. Essentially, this process enables the rapid prototyping and timely development of adaptive digital twins.

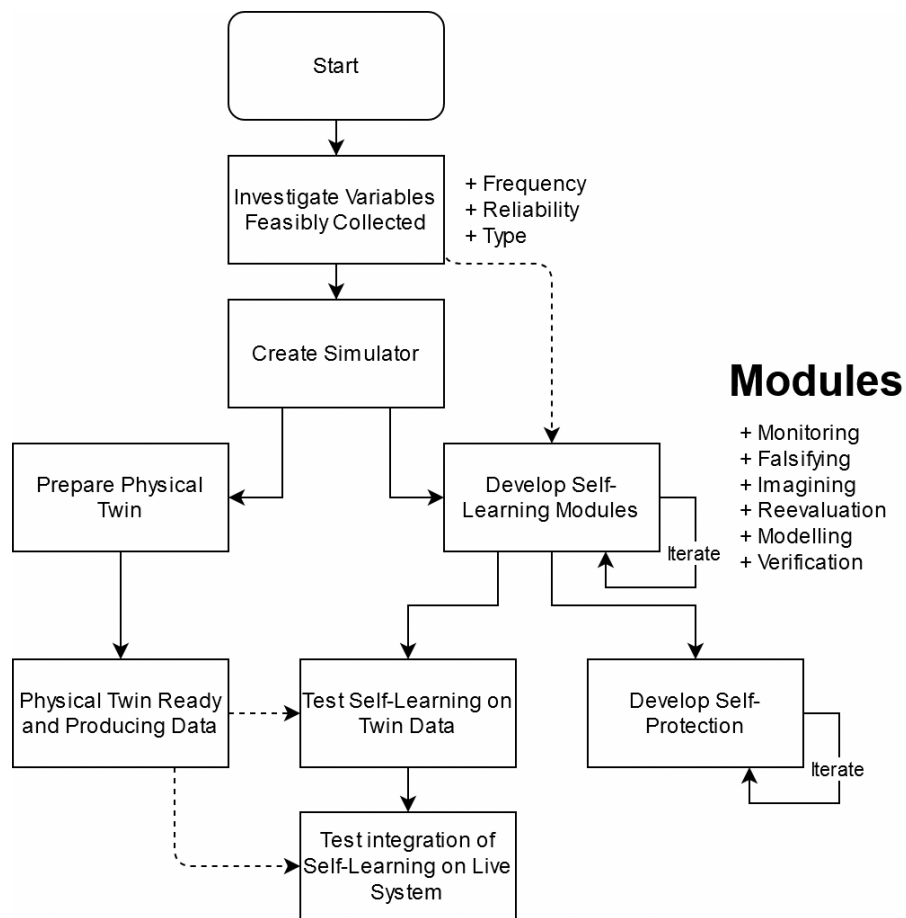


Figure 7.4: Software Development Process Diagram

When the physical is prepared to be twinned, offline data collection should be captured and introduced as a testing dataset for the in-development digital twin. From here, the digital twin's Self-Learning component can process real-world data as it would in deployment. The Self-Protection module does still require the simulator at this stage in development, however, as it cannot modify the future of the replayed data.

When the Self-Learning component is considered stable, it can begin to twin the physical system. As this component cannot modify the physical twin, beginning twinning with only that component active carries minimal risk.

In short, Figure 7.4 details the above process visually.

7.4 Conclusions

This chapter concludes the discussion of the design of the simulator and the Self-Learning and Self-Protection frameworks. In the case of the simulator, the benefits of using a rapid prototyping platform outweigh the potential negatives. In both Self-Learning and Self-Protection, the updated designs refine the concept and address limitations found during this thesis at a design level. Furthermore, as mentioned in Section 7.2, the use of simulations to prototype and rapidly ascertain and refine missed requirements prove to be a useful method for the solicitation of requirements for digital twins.

Module	Implementation
Monitoring	Via Simulator
Falsifying	Ad-Hoc
Imagining	Forecast Oracles
Reevaluation	Technique-dependant
Modelling	Technique-dependant
Verification	Manual

Table 7.1: Summary of Modules and Implementations

As shown in Table 7.1, many implementations were ad-hoc or manual. Only the reevaluation and modelling modules were tested in an interchangeable form. The monitoring and falsifying modules were not changed because they are closely matched to the system rather than the technique. For a different simulator or for a real system, these modules would need to change. The imagining and verification modules were not tested as both lie outwith the scope of this thesis and are left as future work.

Table 7.2 below shows three example implementations for four of the modules. The examples methods for implementing the monitoring and the reevaluation have been omitted as both depend on external variables. The monitoring module implementation depends on the system being twinned and the reevaluation module implementation depends on the modelling module implementation. These are examples as this thesis only investigated the efficacy of a variety of modelling implementations as shown in Section 4.2.

Falsifying	Imagining	Modelling	Verification
Threshold	Same-Trend	System Ident,	Pole Placement
Adaptive Threshold	State-Space	Domain Expert	Domain Expert
Machine Learning (ML)	ML	ML	Model Checking

Table 7.2: Summary of Example Implementations of Each Module

Chapter 8

Conclusion

Realising reduced greenhouse gas emissions from the energy-intensive process heat sector requires retrofit solutions. While replacing existing energy systems is not economically viable, improving operational efficiency provides an immediate benefit on the road to furthering renewable energy systems.

Energy systems, among other critical infrastructure, are prime targets for motivated and sophisticated cyberattacks. To leverage Cloud computing in the optimisation of energy system operations, security challenges must be addressed. The usage of Cloud computing, and the serious flexibility it provides, depends on building robust and safe systems.

In Chapter 3, this thesis conceptualised a Cloud-Machine Interface for connecting the Cloud to industrial plants. Digital Twins played a pivotal role in this interface as they provide a robust and provable model with which to test Cloud optimisation suggestions. To handle unanticipated changes or model inaccuracy, Digital Twins require Self-Learning properties, and it is these properties for which this thesis proposes and tests a definition.

8.1 Research Hypotheses

In Section 3.2, this thesis asked three hypotheses in furthering the definition of Self-Learning. These hypotheses are restated below.

- H1. Is the proposed software architecture for Self-Learning Systems fit-for-purpose?
- H2. Are the proposed Self-Learning algorithms fit-for-purpose as applied to Dynamical Energy Systems?
- H3. Assuming adequate Self-Learning, are the proposed Self-Protection algorithms fit-for-purpose?

8.1.1 Hypothesis 1

In furthering Research Question 2, Hypothesis 1 discusses the engineering design of adaptive digital twins. This hypothesis asks what a fit-for-purpose architecture for Self-Learning systems looks like. To answer this question, a definition of fitness-for-purpose in this context needs discussion. Through this lens, a system is fit-for-purpose if it can adapt to unanticipated changes in a system that it is modelling. The framework described, implemented and evaluated in answering Hypothesis 2 in Section 8.1.2 shows a system that fits this description; however, it lacks refinement.

An improved version of the engineering design is discussed in Chapter 7. There are two main points of import in that design: the use of simulated energy systems, and the compartmentalised adaptive digital twin framework. The use of simulated energy systems allows for the rapid prototyping of adaptive digital twin designs to close in the optimal configuration before applying it to a real system. This approach cuts down the iteration time, reduces risk, and vitally, allows preparations of the real energy system to continue in parallel with design considerations. Secondly, the use of a compartmentalised framework allows

for easily substituted components. For cases where safety criticality is not a high-priority factor, machine learning techniques could stand-in for the control theoretical techniques used in Chapters 5 and 6.

In summary, the answering of Hypothesis 1 contributes to Research Question 2 and the second aim of this thesis. The design discussion in Chapter 7 posits an approach to the development of adaptive digital twins that furthers the theory of self-adaptive systems as used in digital twins. This method is not intended to be an authoritative solution; instead, it is proposed as a single, possible avenue that can excel in some cases and falter in others. In the same vein, as there are myriad sorting algorithms where the best for a given use case can be chosen, this thesis contributes an algorithm for the development of digital twins.

8.1.2 Hypothesis 2

Fitness-for-purpose in the context of Hypothesis 2 is the ability to adapt to reasonably conceivable real-world changes in the physical energy system. For reference, this thesis' energy system was a heater system that degraded in terms of the maximum heater output and the speed of heater output change.

The digital shadow created in Chapter 5 followed the Self-Learning methodology laid out in Section 3.1. In testing, this model-based digital shadow was able to consistently model the physical system more accurately than the base case that assumed nothing changed. The test case that assumed the system continued to move in the derivative direction did outperform the modelled techniques in temperature error. For the avoidance of doubt, it also at least matched the modelled techniques in water level error and beat them in heater output power.

The reason for the “Same-Trend” technique performing as well as it did is discussed later in Section 8.2.1. To summarise, the experiments are conducted with a single step into the future, which benefits techniques that perform well in short term predictions, such as the “Same-Trend” technique. Longer-term prediction leans more heavily on the ability to model a system, and this leaning is highlighted in Section 5.2. Additionally, the “Same-Trend” approach falls short in Self-Protection as discussed in the next hypothesis.

The techniques used in Chapter 5 demonstrate the foundations of fit-for-purpose Self-Learning algorithms and methodologies as applied to dynamical energy systems and contribute to Research Question 1. As the physical system degrades and its dynamics change, the techniques in the aforementioned chapter adapt to the changes and continue to accurately model the system.

8.1.3 Hypothesis 3

Fit-for-purpose Self-Protection algorithms and methodologies are techniques that use Self-Learning to handle situations that could cause damage to the physical hardware. In Chapter 6, this thesis demonstrated two example methods of Self-Protection for energy systems.

The first technique, predicting the future and avoiding failure cases, uses the oracles to a greater extent. The limitation discussed at the end of Section 8.2.1 needs addressing before additional work on this technique can be evaluated in a more grounded manner. This limitation is addressed in Section 8.3.4, and a second piece of future work — Section 8.3.7 — rounds out the final addition required to gather comparable metrics. In this method’s current form, it has shown the promise and the ability of various modelling techniques in handling failure cases but lacks objective data to prove each techniques relative efficacy. Notably, Dynamic Mode Decomposition with Control (DMDc) performed better than the “Same-Trend” case mentioned in the discussion

of the above hypothesis as that technique’s accuracy varies wildly when the prediction is sufficiently far into the future.

The input frequency filtering technique in Section 6.5 generates frequency data over the history of the digital shadow and removes select frequencies. In the tested implementation in this thesis requires a human to select these frequencies; however, this implementation is sufficient for showing that removing frequencies from the input signal can be beneficial. There was a decrease in overall model accuracy with a filtered signal compared to the unmodified signal; however, the modified oscillating signal causes a drastic increase in error and causes the first Self-Protection to trigger fail-safes spuriously. It should be noted that the decrease in model accuracy with a filtered signal is likely caused by the oscillating signal still affecting the heater but no longer affecting the Modelling process, which would cause the Modelling process to, in essence, be acting in inaccurate data from the start.

In summary, the two proposed Self-Protection techniques show promise when based on the ability of a Digital Shadow to learn and adapt to unanticipated changes in the physical twin. Answering this hypothesis contributes to Research Question 1, and in combination with Hypothesis 2, answers it.

8.2 Threats to Validity

This thesis addresses the limitations in its discussions in this section. The limitations are grouped by the category they impact.

8.2.1 Simulator

One major limitation of the methods tested was the lack of testing with batched, non-neural-network approaches, such as decision trees, or stream

learning algorithms. Offering these techniques in addition to those provided in Section 4.2 renders a more complete answer to the efficacy of the Self-Learning framework.

Batch learning models like decision trees are explainable. Whilst the exact level of explainability depends, among other factors, on the size of the tree, it is possible to enumerate and check all possible outcomes. Do note that the models referred to here are not bagged or boosted as these techniques reduce their explainability (Gunning, 2017). Decision trees, such as the type proposed by Liang, Zhang, and Song (2010), also perform well in terms of retraining and updating speed.

Stream Learning has a concept called “Concept Drift” (Gama, Žliobaitė, Bifet, Pechenizkiy, & Bouchachia, 2014). Concept drift is a movement in the underlying system being modelled. The implementation of retraining in Section 5.1.1 bearing a similarity to concept drift is no accident: Concept drift satisfies the Self-Learning requirement of Falsifiability. In many ways, stream learners satisfy a large portion of the Self-Learning framework and, as such, are excellent testing candidates, excluding, of course, the difficulty of proving guarantees and stability.

Another limitation in the simulation portion of this thesis is the deviation from real-world process heat energy systems. In real-world boilers, water is heated past its boiling point to raise steam, and this difference changes a considerable amount in terms of the dynamics of the physical system. For one, the failure states of a boiler are different from those of a heater. Secondly, a boiler has non-linear dynamics compared to the heater’s linear dynamics. Although these differences exist, the toy example of a heater demonstrates the Self-Learning framework for linear systems at the very least.

A further limitation in this portion of the thesis is the length of the oracles generated in Section 5.1. In this section, all evaluations of error were carried out by comparing the simulation to the model as it stepped forward in time without using any oracle data. In essence, these comparisons occurred at an oracle depth of one, and while there is a good reason for doing so, it gives an advantage to some techniques over others. The reasons for using this oracle depth is to eliminate the effects of the oracle from the results and because in real-world situations, only the first prediction would be comparable to the physical system for retraining. The downside of this depth is that some techniques, such as DMDC, perform comparatively better with deeper oracles, at least in temperature and water level predictions.

Lastly, the simulator is responsible for generating the input and output data it uses. This lack of separation causes an issue that is discussed above in Section 8.1.3 whereby the simulator is not affected by the impacts of filtering select frequencies from the system. The way to solve this limitation is mentioned in Section 8.3.6.

8.2.2 Sampling

As mentioned in Section 6.2, the current implementation of the simulator locks the control action rate to the sampling rate and vice versa. As a result, it is not possible to test the detection of higher frequency inputs or the impacts of allowing state updates more frequently than those updates can issue control actions. Disconnecting these rates would provide additional data points for heterogeneously sampled systems, though it is likely only to affect the Frequency Self-Protection method. A more complete analysis of the benefits of this approach is discussed in the aforementioned section.

Secondly, the Frequency Self-Protection method discusses the use of input filtering for improving the resilience of the system. As mentioned in Section 6.5, the inability to apply filtering to the underlying system limits any conclusions that can be drawn about the efficacy of this method for that purpose, and though the filtering does clean up the signal, the system's stability remains to be more thoroughly tested.

8.2.3 Data Types

This thesis uses exclusively numeric data. It is worth noting as the finding within this thesis may not apply directly to categorical applications, such as in work by Chew et al. (2020).

8.3 Future Work

As with any research, this thesis leaves a variety of future avenues to research. Below is a non-exhaustive list of future works. Further Self-Protection is not discussed below as it is a far larger topic than the future work of this thesis.

8.3.1 Pole Placement

As explained in Section 4.2.3 and 4.2.4, OKID and ERA together and DMDC both create state-space models. Further, as discussed in Section 5.1.2, State-space models can be checked for stability in a few ways. The method of import in this context is checking the pole placement of the system. For the discrete-time systems used in this thesis, stable poles lie within a unit circle of origin on the real and imaginary axes, as shown in Figure 3.3. This technique is not limited only to detecting stability, however, and can be used to move the poles of a system into stability.

During the longer retraining runs, a subset of the DMDc models generated poles outwith a unit circle. These models were not used as they were rejected by the Verification module of the Self-Learning framework; however, in the flow in Figure 3.4 shows that Verification module can feedback to the Modelling module, and in this case, the model could be “cured” by moving the poles back into stability.

8.3.2 Dynamic Retraining Thresholds

In this thesis, the thresholds of the four strategies mentioned in Section 5.1 were static, but the thresholds do not need to be so. From better tuned manual or automatic thresholds to machine learning, there is no limit to the complexity nor the variety of methods that would satisfy the intent of the Reevaluation module. Future work in this vein could explore the use of control-based techniques, among a plethora of others, to dynamically move thresholds or replace the thresholds with an entirely different piece of logic.

8.3.3 Compressed Sensing

As mentioned in Section 8.2.2, the simulator locked the control action rate and sampling rate together. If the sampling rate is lower than the action rate, aliasing will occur and the samples will be unable to detect frequencies above the Nyquist frequency. In Section 6.1.1, this thesis mentioned that signals can be reconstructed with fewer samples in select cases. Provided the Fourier basis of the signal is sparse and samples are not evenly spaced, Compressed Sensing (Donoho, 2006) allows for the reconstruction of a signal higher than the apparent aliasing limit. The benefit of compressed sensing is that the exact timing of samples in the real world may vary enough to enable better signal recovery from the physical system, provided the clock is accurate enough.

8.3.4 Forecasting

As mentioned in Section 4.1.3, the simulator uses two forecasting oracles: one perfect and one simplistic. These two oracles represent the two ends of the spectrum of working forecasters. A natural extension to the work done in this thesis would be to utilise forecasting techniques that are more faithful to real-world use, such as machine learning.

8.3.5 Non-Human-In-The-Loop

The removal of the Human-In-The-Loop portion of both Self-Protection methods in Chapter 6 would create a self-contained and automated system. From the system frequency responses in Section 5.1.2 to the input frequencies in Section 6.5, much of the data required is already in the simulation. What remains to remove the human-in-the-loop is research into the best method to stably remove frequencies that can cause damage and ignore those that have no effect.

Secondly, to remove humans from the loop in the verification module, robust verification for the chosen model needs to be built. A single, comprehensive approach for all models is unlikely to be attainable as each system has different failure and success modes. For example, in the simulation, heating water above 100°C was a failure state, but in a boiler, that same state is a requirement for its functioning.

8.3.6 Actual Energy Systems

Addressing the real-world energy system limitation mentioned above in Section 8.2.1, a future step in the vein of this research is the use of a real energy system, possibly simulated but preferably data from a physical system.

Putting Self-Protection aside for a moment, using a simulator that replays real boiler data allows for the testing in Chapter 5 to be applied to real-world systems. However, testing Self-Protection is more complicated as it requires, at least to go beyond simply detecting problematic situations, the ability to influence a running system.

8.3.7 Prediction Testing

The prediction testing method used in this thesis does not generate metrics to evaluate the performance of each technique. Using the implementation of the first oracle used by this thesis allows the testing of prediction techniques without external influence from the forecasting method. To evaluate prediction techniques, the implementation requires considering the cases to test and the metrics to collect.

Bibliography

- Al-Habaibeh, A., & Gindy, N. (2001). Self-Learning Algorithm for Automated Design of Condition Monitoring Systems for Milling Operations. *The International Journal of Advanced Manufacturing Technology*, 18(6), 448–459. doi:10.1007/s001700170054
- Airbus. (n.d.). CyberRange. Retrieved August 13, 2020, from <https://airbus-cyber-security.com/products-and-services/prevent/cyberberrange/>
- Amoah, R., Camtepe, S., & Foo, E. (2016). Securing DNP3 broadcast communications in SCADA systems. *IEEE Transactions on Industrial Informatics*, 12(4), 1474–1485.
- Angelopoulos, K., Papadopoulos, A. V., Silva Souza, V. E., & Mylopoulos, J. (2016). Model Predictive Control for Software Systems with CobRA. In *Proceedings of the 11th international symposium on software engineering for adaptive and self-managing systems* (pp. 35–46). SEAMS '16. doi:10.1145/2897053.2897054
- Anton, S. D. D., Hafner, A., & Schotten, H. D. (2019). Devil in the detail: Attack scenarios in industrial applications. In *2019 IEEE Security and Privacy Workshops (SPW)* (pp. 169–174). IEEE.
- ASCon Systems. (2017). Digital Twin is about to rollout by Airbus. Retrieved August 13, 2020, from <https://ascon-systems.de/en/digital-twin-is-about-to-rollout-by-airbus/>

- Asghar, M. R., Hu, Q., & Zeadally, S. (2019). Cybersecurity in industrial control systems: Issues, technologies, and challenges. *Computer Networks*, *165*, 106946. doi:<https://doi.org/10.1016/j.comnet.2019.106946>
- Bahşi, H., & Maennel, O. M. (2015). A Conceptual Nationwide Cyber Situational Awareness Framework for Critical Infrastructures. In S. Buchegger & M. Dam (Eds.), *Secure it systems* (pp. 3–10). Cham: Springer International Publishing.
- Bécue, A., Fourastier, Y., Praça, I., Savarit, A., Baron, C., Gradussofs, B., ... Thomas, C. (2018). CyberFactory#1 — Securing the industry 4.0 with cyber-ranges and digital twins. In *2018 14th ieee international workshop on factory communication systems (wfcs)* (pp. 1–4).
- Bernieri, G., Conti, M., & Pascucci, F. (2019). MimePot: a Model-based Honey-pot for Industrial Control Networks. In *2019 ieee international conference on systems, man and cybernetics (smc)* (pp. 433–438).
- BinaryDefense. (n.d.). Artillery. Retrieved August 10, 2020, from <https://www.binarydefense.com/>
- Bitton, R., Gluck, T., Stan, O., Inokuchi, M., Ohta, Y., Yamada, Y., ... Shabtai, A. (2018). Deriving a Cost-Effective Digital Twin of an ICS to Facilitate Security Evaluation. In J. Lopez, J. Zhou, & M. Soriano (Eds.), *Computer security* (pp. 533–554). Cham: Springer International Publishing.
- Boeing. (n.d.). Cyber-Range-in-a-Box. Retrieved August 13, 2020, from <https://www.boeing.com/defense/cybersecurity-information-management/>
- Brunton, S. L., Proctor, J. L., & Kutz, J. N. (2016). Sparse identification of nonlinear dynamics with control (SINDYc). *IFAC-PapersOnLine*, *49*(18), 710–715.
- Burroughs, S. (2021). *Towards predictive runtime modelling of Kubernetes microservices* (Doctoral dissertation, Hamilton, New Zealand). Masters. Retrieved from <https://hdl.handle.net/10289/14091>
- Čeleda, P., Vykopal, J., Švábenský, V., & Slavíček, K. (2020). KYPO4INDUSTRY: A Testbed for Teaching Cybersecurity of Industrial Control Sys-

- tems. In *Proceedings of the 51st acm technical symposium on computer science education* (pp. 1026–1032). SIGCSE '20. doi:10.1145/3328778.3366908
- Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly Detection: A Survey. *ACM Comput. Surv.* 41(3). doi:10.1145/1541880.1541882
- Chew, C. J.-W., Kumar, V., Patros, P., & Malik, R. (2020). ESCAPADE: Encryption-Type-Ransomware: System Call Based Pattern Detection. In M. Kutylowski, J. Zhang, & C. Chen (Eds.), *Network and system security* (pp. 388–407). Cham: Springer International Publishing.
- Davis, J., & Magrath, S. (2013). A Survey of Cyber Ranges and Testbeds Executive.
- Dawson, M., Patros, P., & Kent, K. B. (2021). Multi-tenant cloud elastic garbage collector (U.S. Patent No. 10,990,519).
- de Silva, B., Champion, K., Quade, M., Loiseau, J.-C., Kutz, J., & Brunton, S. (2020). PySINDy: A Python package for the sparse identification of non-linear dynamical systems from data. *Journal of Open Source Software*, 5(49), 2104. doi:10.21105/joss.02104
- Demo, N., Tezzele, M., & Rozza, G. (2018). PyDMD: Python Dynamic Mode Decomposition. *The Journal of Open Source Software*, 3(22), 530. doi:https://doi.org/10.21105/joss.00530
- Deutsche Telekom. (n.d.-a). Sicherheitstacho (security dashboard). Retrieved August 10, 2020, from <https://www.sicherheitstacho.eu>
- Deutsche Telekom. (n.d.-b). T-pot. Retrieved August 10, 2020, from <https://dtag-dev-sec.github.io/>
- DinoTools et al. (n.d.). Dionaea. Retrieved August 10, 2020, from <https://github.com/DinoTools/dionaea>
- Donoho, D. L. (2006). Compressed sensing. *IEEE Transactions on Information Theory*, 52(4), 1289–1306. doi:10.1109/TIT.2006.871582
- Eckhart, M., & Ekelhart, A. (2018a). A Specification-Based State Replication Approach for Digital Twins. In *Proceedings of the 2018 workshop on*

- cyber-physical systems security and privacy* (pp. 36–47). CPS-SPC '18. doi:10.1145/3264888.3264892
- Eckhart, M., & Ekelhart, A. (2018b). Towards Security-Aware Virtual Environments for Digital Twins. In *Proceedings of the 4th acm workshop on cyber-physical system security* (pp. 61–72). CPSS '18. doi:10.1145/3198458.3198464
- Eckhart, M., Ekelhart, A., & Weippl, E. (2019). Enhancing Cyber Situational Awareness for Cyber-Physical Systems through Digital Twins. In *2019 24th ieee international conference on emerging technologies and factory automation (etfa)* (pp. 1222–1225). doi:https://doi.org/10.1109/ETFA.2019.8869197
- Feng, C., Li, T., & Chana, D. (2017). Multi-level Anomaly Detection in Industrial Control Systems via Package Signatures and LSTM Networks. In *2017 47th annual ieee/ifip international conference on dependable systems and networks (dsn)* (pp. 261–272).
- Filieri, A., Maggio, M., Angelopoulos, K., D'ippolito, N., Gerostathopoulos, I., Hempel, A. B., ... Vogel, T. (2017). Control Strategies for Self-Adaptive Software Systems. *ACM Trans. Auton. Adapt. Syst.* 11(4). doi:10.1145/3024188
- Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A Survey on Concept Drift Adaptation. *ACM Comput. Surv.* 46(4). doi:10.1145/2523813
- Gao, H., Peng, Y., Jia, K., Dai, Z., & Wang, T. (2013). The Design of ICS Testbed Based on Emulation, Physical, and Simulation (EPS-ICS Testbed). In *2013 ninth international conference on intelligent information hiding and multimedia signal processing* (pp. 420–423).
- Gazula, M. B. (2017). *Cyber warfare conflict analysis and case studies* (Doctoral dissertation, Massachusetts Institute of Technology).

- Gehrmann, C., & Gunnarsson, M. (2020). A Digital Twin Based Industrial Automation and Control System Security Architecture. *IEEE Transactions on Industrial Informatics*, *16*(1), 669–680. doi:10.1109/TII.2019.2938885
- Ghahramani, Z. (2004). Unsupervised learning. In O. Bousquet, U. von Luxburg, & G. Rätsch (Eds.), *Advanced lectures on machine learning: Ml summer schools 2003, canberra, australia, february 2 - 14, 2003, tübingen, germany, august 4 - 16, 2003, revised lectures* (pp. 72–112). doi:10.1007/978-3-540-28650-9_5
- Glaessgen, E., & Stargel, D. (n.d.). The digital twin paradigm for future NASA and US Air Force vehicles. In *53rd aiaa/asme/asce/ahs/asc structures, structural dynamics and materials conference 20th aiaa/asme/ahs adaptive structures conference 14th aiaa* (p. 1818). doi:10.2514/6.2012-1818
- Government Communications Security Bureau. (2020). New Zealand Information Security Manual v3.3. Retrieved from <https://www.nzism.gcsb.govt.nz/>
- Green, B., Lee, A., Antrobus, R., Roedig, U., Hutchison, D., & Rashid, A. (2017). Pains, Gains and PLCs: Ten Lessons from Building an Industrial Control Systems Testbed for Security Research. In *10th USENIX workshop on cyber security experimentation and test (CSET 17)*, Vancouver, BC: USENIX Association. Retrieved from <https://www.usenix.org/conference/cset17/workshop-program/presentation/green>
- Gunning, D. (2017). Explainable artificial intelligence (xai). *Defense Advanced Research Projects Agency (DARPA), nd Web*, *2*(2).
- Hayes, G., & El-Khatib, K. (2013). Securing modbus transactions using hash-based message authentication codes and stream transmission control protocol. In *2013 third international conference on communications and information technology (iccit)* (pp. 179–184).
- Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Comput.* *9*(8), 1735–1780. doi:10.1162/neco.1997.9.8.1735

- Holm, H., Karresand, M., Vidström, A., & Westring, E. (2015). A Survey of Industrial Control System Testbeds. In S. Buchegger & M. Dam (Eds.), *Secure it systems* (pp. 11–26). Cham: Springer International Publishing.
- Hutchins, E. M., Cloppert, M. J., & Amin, R. M. (2011). Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. *Leading Issues in Information Warfare & Security Research*, 1(1), 80.
- Iannucci, S., Barba, O. D., Cardellini, V., & Banicescu, I. (2019). A Performance Evaluation of Deep Reinforcement Learning for Model-Based Intrusion Response. In *2019 IEEE 4th International Workshops on Foundations and Applications of Self* Systems (FAS*W)* (pp. 158–163).
- Jang, J. S. R. (1992). Self-learning fuzzy controllers based on temporal back-propagation. *IEEE Transactions on Neural Networks*, 3(5), 714–723. doi:10.1109/72.159060
- Jones, D., Snider, C., Nassehi, A., Yon, J., & Hicks, B. (2020). Characterising the Digital Twin: A systematic literature review. *CIRP Journal of Manufacturing Science and Technology*. doi:https://doi.org/10.1016/j.cirpj.2020.02.002
- Juang, J.-N., & Pappa, R. S. (1985). An eigensystem realization algorithm for modal parameter identification and model reduction. *Journal of guidance, control, and dynamics*, 8(5), 620–627.
- Juang, J.-N., Phan, M., Horta, L. G., & Longman, R. W. (1993). Identification of observer/Kalman filter Markov parameters-Theory and experiments. *Journal of Guidance, Control, and Dynamics*, 16(2), 320–329.
- Karampidis, K., Panagiotakis, S., Vasilakis, M., Markakis, E. K., & Papadourakis, G. (2019). Industrial CyberSecurity 4.0: Preparing the Operational Technicians for Industry 4.0. In *2019 IEEE 24th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)* (pp. 1–6).

- Kritzinger, W., Karner, M., Traar, G., Henjes, J., & Sihn, W. (2018). Digital Twin in manufacturing: A categorical literature review and classification. *IFAC-PapersOnLine*, *51*(11), 1016–1022. 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018. doi:<https://doi.org/10.1016/j.ifacol.2018.08.474>
- Lai, Y., Liu, Z., Song, Z., Wang, Y., & Gao, Y. (2016). Anomaly detection in Industrial Autonomous Decentralized System based on time series. *Simulation Modelling Practice and Theory*, *65*, 57–71. Analyzing and Visual Programming Internet of Things. doi:<https://doi.org/10.1016/j.simpat.2016.01.013>
- LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. doi:[10.1038/nature14539](https://doi.org/10.1038/nature14539)
- Liang, C., Zhang, Y., & Song, Q. (2010). Decision tree for dynamic and uncertain data streams. In *Proceedings of 2nd asian conference on machine learning* (pp. 209–224). JMLR Workshop and Conference Proceedings.
- Lightwire. (n.d.). Trusted Rural Broadband for Waikato & BOP, Lightwire. Retrieved August 10, 2020, from <https://www.lightwire.co.nz/>
- Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, ... Xiaoqiang Zheng. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from [tensorflow.org](https://www.tensorflow.org/). Retrieved from <https://www.tensorflow.org/>
- Microsoft Corporation. (n.d.). Hackers hit Norsk Hydro with ransomware. The company responded with transparency. Retrieved August 13, 2020, from <https://news.microsoft.com/transform/hackers-hit-norsk-hydro-ransomware-company-responded-transparency/>
- Ministry of Business, Innovation & Employment. (2019a). Process heat in New Zealand. Available at: <https://www.mbie.govt.nz/assets/8c89799b73/process-heat-current-state-fact-sheet.pdf>.
- Ministry of Business, Innovation & Employment. (2019b). Process Heat in New Zealand: Opportunities and barriers to lowering emissions. Available at:

<https://www.mbie.govt.nz/dmsdocument/4292-process-heat-in-new-zealand-opportunities-and-barriers-to-lowering-emissions>.

Ministry of Business, Innovation & Employment. (2020). Process heat in New Zealand. Available at: <https://www.mbie.govt.nz/science-and-technology/science-and-innovation/funding-information-and-opportunities/investment-funds/strategic-science-investment-fund/ssif-funded-programmes/university-of-waikato/>. Accessed: 22/Mar/2021.

Ministry of Business, Innovation & Employment. (2021). Process heat in New Zealand. Available at: <https://www.mbie.govt.nz/building-and-energy/energy-and-natural-resources/low-emissions-economy/process-heat-in-new-zealand/>.

MushMush Foundation. (n.d.). Conpot. Retrieved August 10, 2020, from <https://github.com/mushorg/conpot>

Oliveira-Neto, F. M., Han, L. D., & Jeong, M. K. (2013). An Online Self-Learning Algorithm for License Plate Matching. *IEEE Transactions on Intelligent Transportation Systems*, *14*(4), 1806–1816. doi:10.1109/TITS.2013.2270107

Oosterhof, M. et al. (n.d.). Cowrie. Retrieved August 10, 2020, from <https://github.com/cowrie/cowrie>

Parliamentary Counsel Office of New Zealand. (2019). Climate Change Response (Zero Carbon) Amendment Act 2019. Available at: <https://www.legislation.govt.nz/act/public/2019/0061/latest/whole.html>.

Patros, P., Kent, K. B., & Dawson, M. (2017). SLO Request Modeling, Reordering and Scaling. In *Proceedings of the 27th annual international conference on computer science and software engineering* (pp. 180–191). CASCON '17. Markham, Ontario, Canada: IBM Corp.

Pauna, A. (2012). Improved self adaptive honeypots capable of detecting rootkit malware. In *2012 9th international conference on communications (comm)* (pp. 281–284).

- Pauna, A., Bica, I., Pop, F., & Castiglione, A. (2019). On the rewards of self-adaptive IoT honeypots. *Annals of Telecommunications*, 74(7-8), 501–515.
- Pauna, A., Iacob, A., & Bica, I. (2018). QRASSH - A Self-Adaptive SSH Honeypot Driven by Q-Learning. In *2018 international conference on communications (comm)* (pp. 441–446).
- Peng Wen, Zhang Dianhua, & Gong Dianyao. (2012). Optimization of roll-gap self-learning algorithm in tandem hot rolled strip finishing mill. In *2012 24th chinese control and decision conference (ccdc)* (pp. 3947–3950). doi:10.1109/CCDC.2012.6243107
- Podolskiy, V., Mayo, M., Koay, A., Gerndt, M., & Patros, P. (2019). Maintaining SLOs of Cloud-Native Applications Via Self-Adaptive Resource Sharing. In *2019 IEEE 13th international conference on self-adaptive and self-organizing systems (saso)* (pp. 72–81). doi:10.1109/SASO.2019.00018
- Podolskiy, V., Patrou, M., Patros, P., Gerndt, M., & Kent, K. B. (2020). The Weakest Link: Revealing and Modeling the Architectural Patterns of Microservice Applications. In *Proceedings of the 30th annual international conference on computer science and software engineering* (pp. 113–122). CASCON '20. Toronto, Ontario, Canada: IBM Corp.
- Proctor, J. L., Brunton, S. L., & Kutz, J. N. (2016). Dynamic mode decomposition with control. *SIAM Journal on Applied Dynamical Systems*, 15(1), 142–161.
- Rabiner, L., Schafer, R., & Rader, C. (1969). The chirp z-transform algorithm. *IEEE Transactions on Audio and Electroacoustics*, 17(2), 86–92. doi:10.1109/TAU.1969.1162034
- Reuters. (n.d.). UPDATE 1-Hydro's Norwegian aluminium plant faces months of reduced output. Retrieved August 13, 2020, from <https://www.reuters.com/article/norsk-hydro-outages/update-1-hydros-norwegian-aluminium-plant-faces-months-of-reduced-output-idUSL8N21K4L7>

- Sarco, S. (2018). *The Steam and Condensate Loop: Effective Steam Engineering For Today*. Spirax Sarco Limited.
- Schluse, M., Atorf, L., & Rossmann, J. (2017). Experimentable digital twins for model-based systems engineering and simulation-based development. In *2017 annual ieee international systems conference (syscon)* (pp. 1–8). doi:10.1109/SYSCON.2017.7934796
- Schroeder, G. N., Steinmetz, C., Rodrigues, R. N., Henriques, R. V. B., Retberg, A., & Pereira, C. E. (2021). A Methodology for Digital Twin Modeling and Deployment for Industry 4.0. *Proceedings of the IEEE, 109*(4), 556–567. doi:10.1109/JPROC.2020.3032444
- Shilling, S. A. (1972). *A study of the chirp Z-transform and its applications* (Doctoral dissertation). Masters. Retrieved from <http://hdl.handle.net/2097/7844>
- Shodan. (n.d.). Shodan.io. Retrieved August 10, 2020, from <https://www.shodan.io>
- Su, J., Vargas, D. V., & Sakurai, K. (2019). One Pixel Attack for Fooling Deep Neural Networks. *IEEE Transactions on Evolutionary Computation, 23*(5), 828–841. doi:10.1109/TEVC.2019.2890858
- Tan, S. C., Ting, K. M., & Liu, T. F. (2011). Fast Anomaly Detection for Streaming Data. In *Proceedings of the twenty-second international joint conference on artificial intelligence - volume volume two* (pp. 1511–1516). IJCAI'11. Barcelona, Catalonia, Spain: AAAI Press.
- The Boeing Company. (2019). Welcome to Boeing's factory of the future. Retrieved February 3, 2021, from <https://www.boeing.com/features/innovation-quarterly/feb2019/btj-global.page>
- The Guardian. (2021). Natanz 'sabotage' highlights Iran's vulnerability to cyber-attacks. Available at: <https://www.theguardian.com/world/2021/apr/12/natanz-nuclear-facility-sabotage-iran-vulnerability-to-cyber-attacks>.

- Trendmicro. (2015a). GasPot Integrated Into Conpot, Contributing to Open Source ICS Research. Retrieved August 10, 2020, from <https://blog.trendmicro.com/trendlabs-security-intelligence/gaspot-integrated-into-conpot-contributing-to-open-source-ics-research/>
- Trendmicro. (2015b). The GasPot Experiment: Unexamined Perils in Using Gas-Tank-Monitoring Systems. Retrieved August 10, 2020, from https://documents.trendmicro.com/assets/wp/wp_the_gaspot_experiment.pdf
- United Nations Framework Convention on Climate Change. (2015). Paris Agreement. Available at: <https://unfccc.int/process-and-meetings/the-paris-agreement/the-paris-agreement>.
- Urias, V., Van Leeuwen, B., & Richardson, B. (2012). Supervisory Command and Data Acquisition (SCADA) system cyber security analysis using a live, virtual, and constructive (LVC) testbed. In *Milcom 2012 - 2012 ieee military communications conference* (pp. 1–8).
- van Zijl, M. (2020). *Model checking for cloud autoscaling using WATERS* (Doctoral dissertation, Hamilton, New Zealand). Masters. Retrieved from <https://hdl.handle.net/10289/13602>
- Wade, K. C. (1995). Steam generator degradation and its impact on continued operation of pressurized water reactors in the United States. *Energy Information Administration/Electric Power Monthly*, 66, 36.
- Wallace, N., & Atkison, T. (2013). Observing Industrial Control System Attacks Launched via Metasploit Framework. In *Proceedings of the 51st acm southeast conference*. ACMSE '13. doi:10.1145/2498328.2500067
- Wang, K., Du, M., Maharjan, S., & Sun, Y. (2017). Strategic Honeypot Game Model for Distributed Denial of Service Attacks in the Smart Grid. *IEEE Transactions on Smart Grid*, 8(5), 2474–2482.
- Wang, W., & Lu, Z. (2013). Cyber security in the smart grid: Survey and challenges. *Computer networks*, 57(5), 1344–1371.

- Wei, Q., Liao, Z., Song, R., Zhang, P., Wang, Z., & Xiao, J. (2021). Self-Learning Optimal Control for Ice-Storage Air Conditioning Systems via Data-Based Adaptive Dynamic Programming. *IEEE Transactions on Industrial Electronics*, *68*(4), 3599–3608. doi:10.1109/TIE.2020.2978699
- Weyns, D. (2018). Engineering Self-Adaptive Software Systems – An Organized Tour. In *2018 IEEE 3rd International Workshops on Foundations and Applications of Self* Systems (FAS*w)* (pp. 1–2). doi:10.1109/FAS-W.2018.00012
- Weyns, D. (2021a). *An Introduction To Self-Adaptive Systems: A contemporary software engineering perspective*. Wiley.
- Weyns, D. (2021b). *An Introduction To Self-Adaptive Systems: A contemporary software engineering perspective*. Wiley.
- Zamiri-Gourabi, M.-R., Qalaei, A. R., & Azad, B. A. (2019). Gas What? I Can See Your GasPots. Studying the Fingerprintability of ICS Honeypots in the Wild. In *Proceedings of the fifth annual industrial control system security (icss) workshop* (pp. 30–37). ICSS. doi:10.1145/3372318.3372322
- Zhao, Z., Shen, L., Yang, C., Wu, W., Zhang, M., & Huang, G. Q. (2021). IoT and digital twin enabled smart tracking for safety management. *Computers and Operations Research*, *128*, 105183. doi:https://doi.org/10.1016/j.cor.2020.105183
- Zhou, C., Huang, S., Xiong, N., Yang, S., Li, H., Qin, Y., & Li, X. (2015). Design and Analysis of Multimodel-Based Anomaly Intrusion Detection Systems in Industrial Process Automation. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *45*(10), 1345–1360.

Appendix

Strategy	DMD _c	OKID	SINDY _c
No Retraining	5622.5	1.36e9	3320.3
Step	5622.5	2.45e9	3320.3
Signed	2545.3	2.45e9	3320.3
Sign., Step	2877.8	2.45e9	3320.3
Absolute	2699.0	2.45e9	3320.3
Abs., Step	2629.4	2.45e9	3320.3
Abs., Sign.	2683.4	2.45e9	3320.3
Abs., Sign., Step	2636.1	2.45e9	3320.3
Time	2804.9	2.92e9	3320.3
Time, Step	2668.8	2.45e9	3320.3
Time, Sign.	2742.7	2.45e9	3320.3
Time, Sign., Step	2610.1	2.45e9	3320.3
Time, Abs.	2766.7	2.45e9	3320.3
Time, Abs., Step	2591.1	2.45e9	3320.3
Time, Abs., Sign.	2706.1	2.45e9	3320.3
All	2610.1	2.45e9	3320.3

Table 8.1: Complete Temperature Error Across All Retraining Strategies with minimum time before retrain

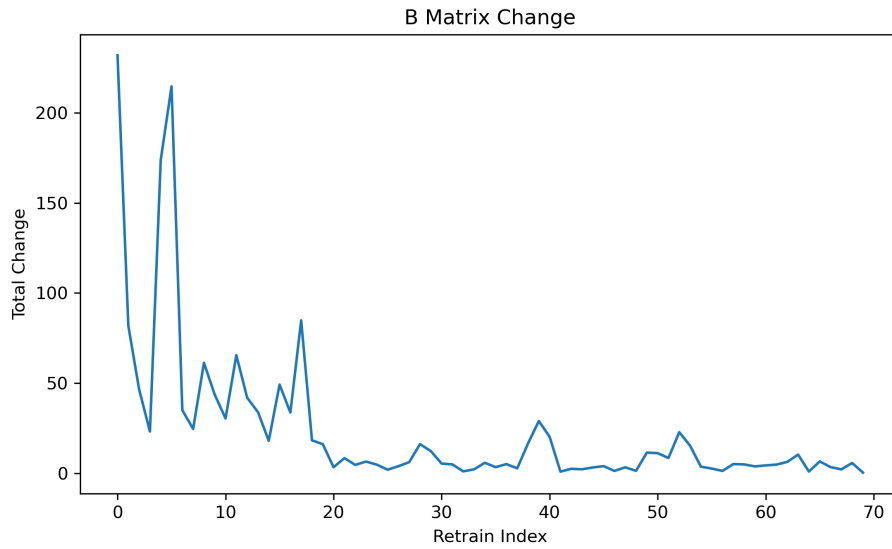


Figure 8.1: B Matrix Change

Longer Run Retraining Images

The following images are generated from longer, 1,048,576 step sequences to show the system at steady state for an elongated period of time. Do note that the following figures were generated using the minimum retrain time setting, whereas the other images shown in Section 5.1.1 were able to retrain immediately if the retrain strategies kicked in.

Bode Plots

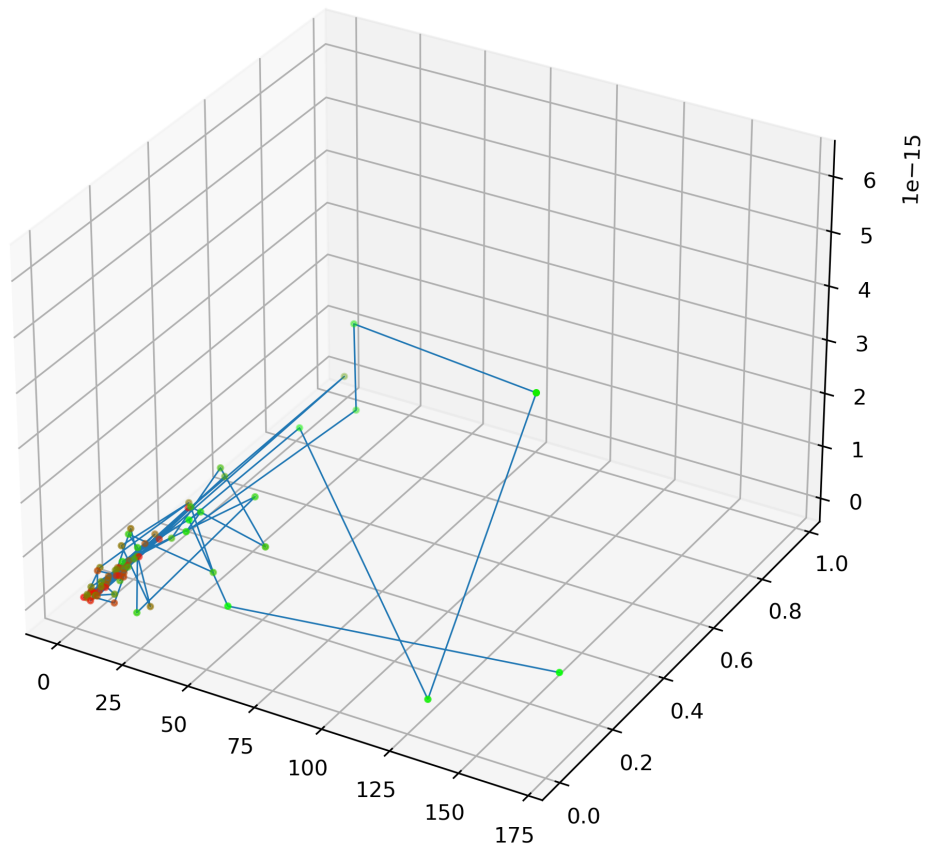


Figure 8.2: Principal Component Analysis of B matrix change over time

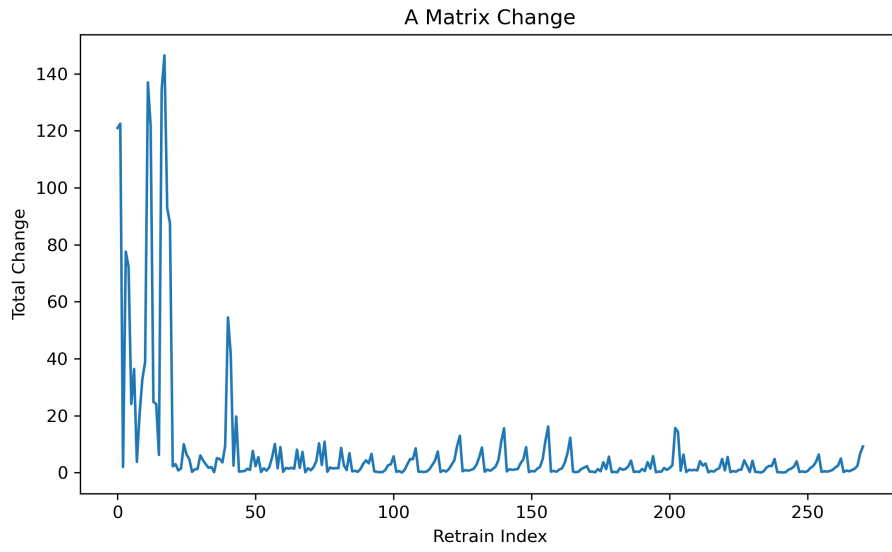


Figure 8.3: Absolute change in A matrix values between retraining

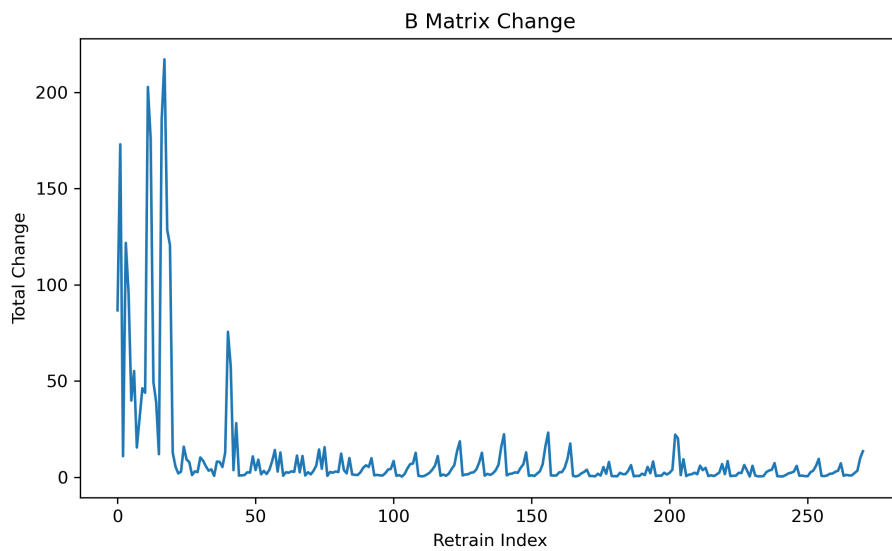


Figure 8.4: Absolute change in A matrix values between retraining

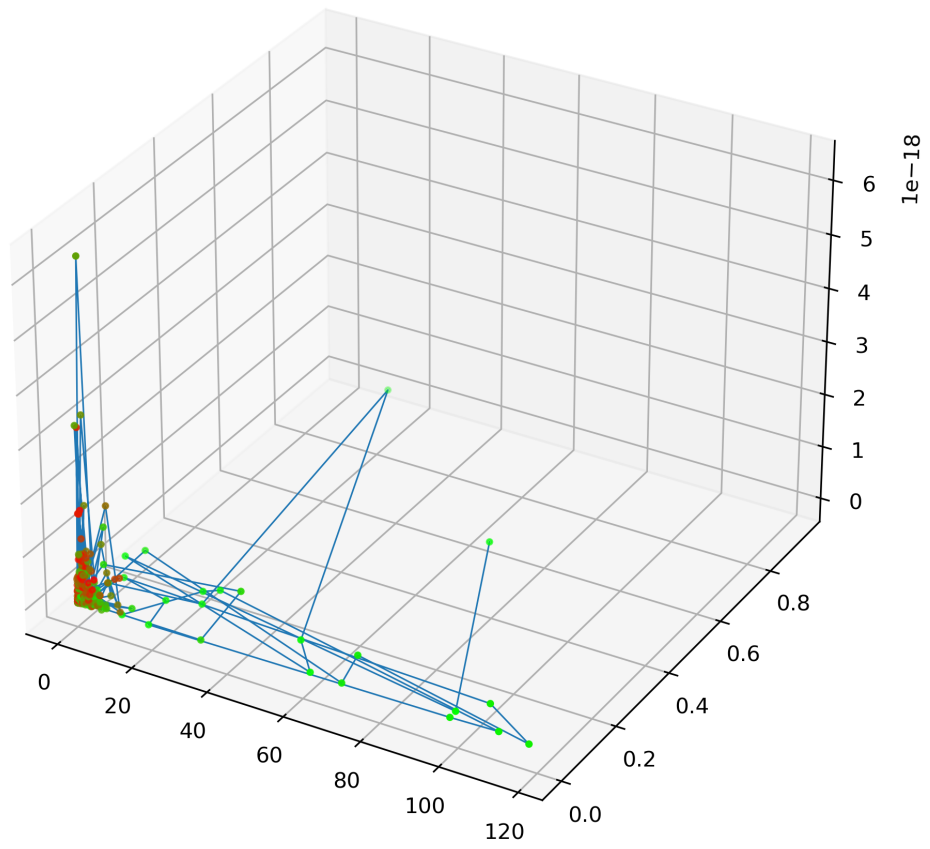


Figure 8.5: Principal Component Analysis of A matrix change over time

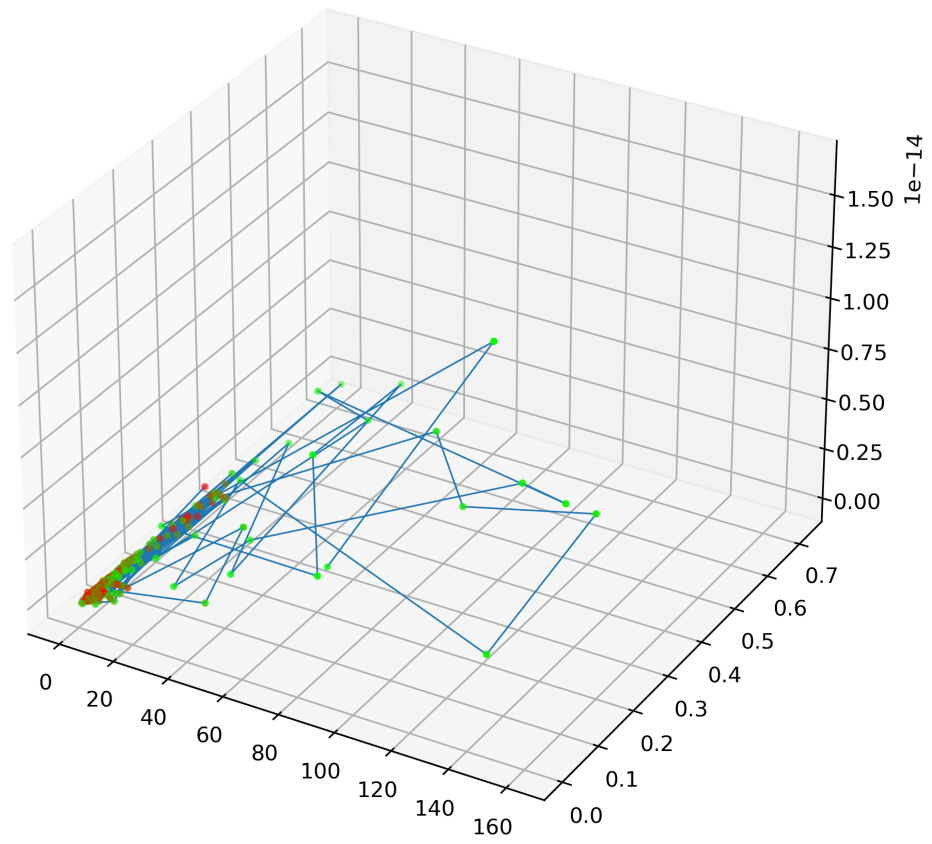


Figure 8.6: Principal Component Analysis of B matrix change over time

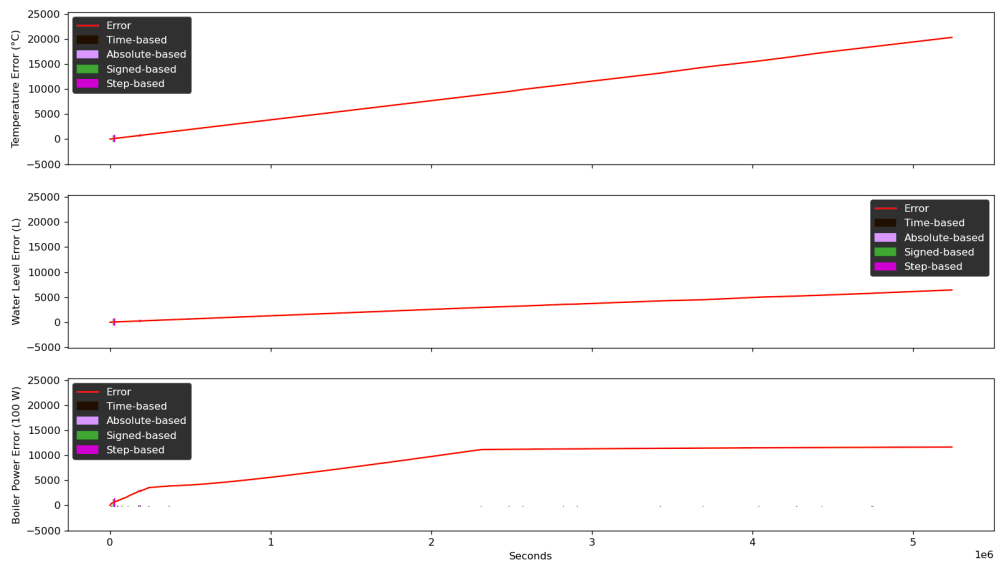


Figure 8.7: Integral of Absolute Error for DMDc Retrains with the Signed strategies

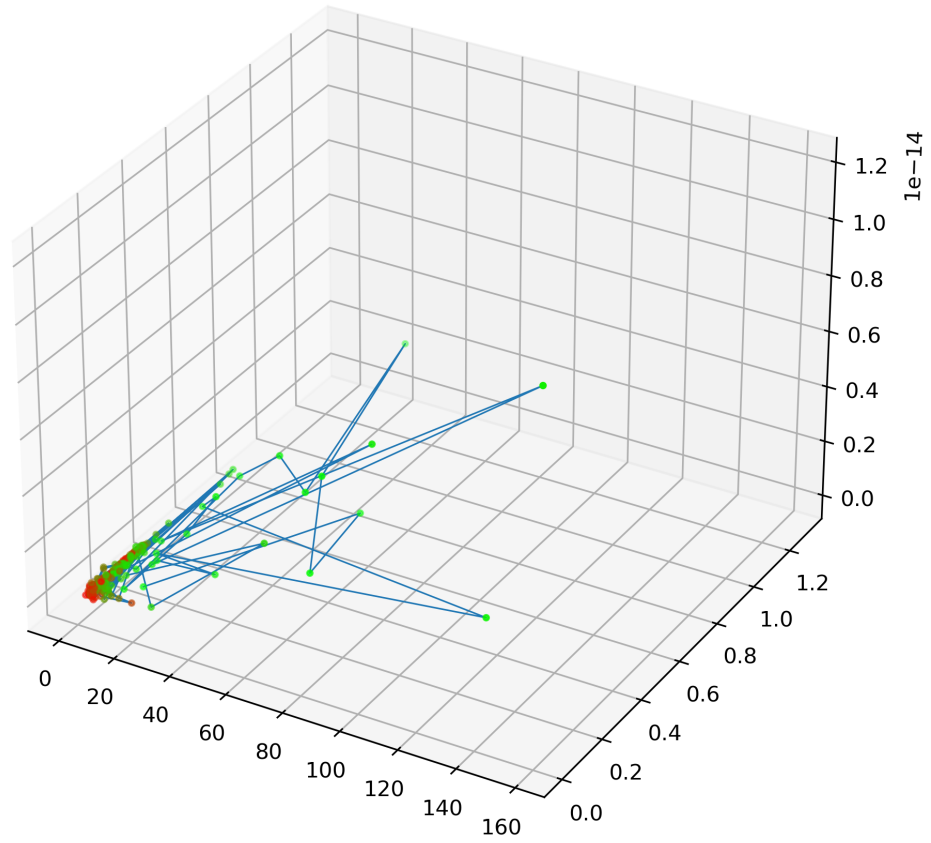


Figure 8.8: Principal Component Analysis of B matrix over time

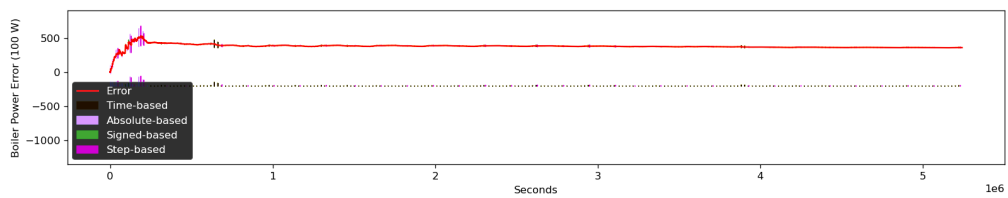


Figure 8.9: Heater error only for Octuple length run

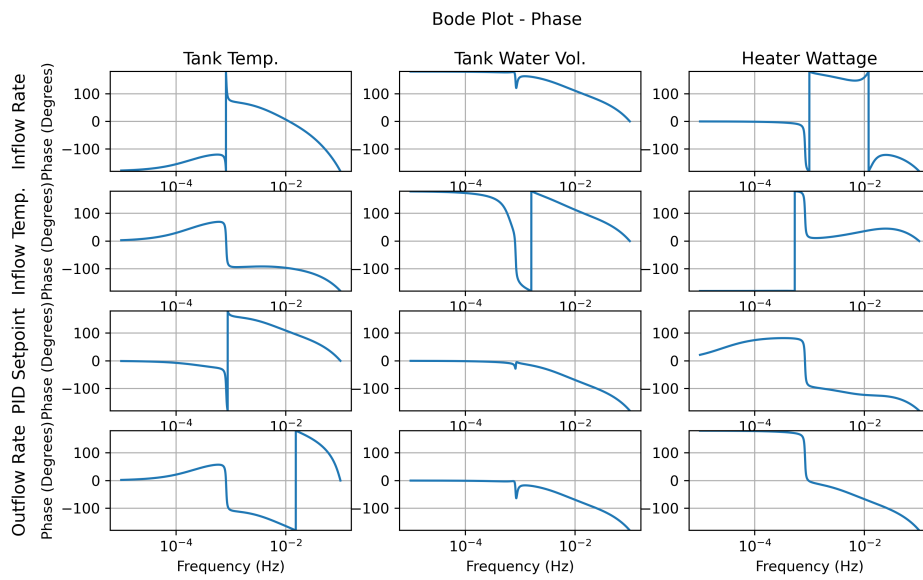


Figure 8.10: DMDc First Retrain Bode Plot — Phase