Matthew James Reaney

MEng Final Year Project Report:

Analysing the impact of cyber-attacks in cyber-physical systems

April 14th, 2023

Supervisor: Dr Nikolaos Athanasopoulos

QUEEN'S UNIVERSITY BELFAST

School of Electronics, Electrical Engineering and Computer Science

Abstract

Cyber-attacks are becoming a major concern as cyber-physical systems govern more of modern infrastructure. Continuous research is required to ensure safe and continual operation of these systems even in the presence of a cyber-attacks. The aim of this project is to continue developing methods for risk assessment of the possible cyber-security threats through the use of control theory to create a software simulation tool. This tool models a cyber-physical system with the implementation of various cyber-attacks. The goal is to enable a platform which can be used to estimate the impact of different cyber-threats.

The proposed design of the simulation tool is a three-tank water system created and run through MATLAB & Simulink with GUI made in App Designer available at https://github.com/mattr862/three-tank-water-system. The design features the ability to simulate attacking inputs and/or outputs using six different attack types, as well as a novel approach to stealthy attack scenarios called state dependant attacks.

This report seeks to provide context on the topic through literary review, explanation of the control theory used to model cyber-physical systems and attacks, presentation of software design, analysis of simulation results, discussion on findings, areas for future work and lastly, evaluate against the project specification.

Specification

Title: Analysing the impact of cyber-attacks in cyber-physical systems

Supervisor: Dr Nikolaos Athanasopoulos

Cyber-security is becoming a major challenge in every aspect of everyday life. Systems engineered and controlled by software, hardware, and communication networks, becoming increasingly vulnerable to attacks, that become themselves more and more sophisticated; covert attacks, replay attacks, stealthy attacks, data injection, bias injection, denial of service, are some of the so far identified and documented attack strategies that have been attempted in safety-critical infrastructure involving electricity networks, power generation, manufacturing, military, transportation etc.

This project aims to analyse the impact of attacks, specifically on dynamical systems controlled by a ‘cyber’ component (hardware, software, communication network), also known as cyber-physical systems. Our aim will be to properly categorise and unify the different types of attacks in a rich model borrowed from hybrid systems theory, namely, linear hybrid automata. The goal is to use mature tools from modern control theory to analyse the impact of different types of attacks on a dynamical system.

Objectives

1. Familiarize with attack strategies for dynamical systems, and modelling of cyber-physical systems.
2. Adapt safety analysis algorithms for dynamical systems so they can incorporate attack models.
3. Analyse impact of attacks on a specific benchmark problem, to be agreed.

MEng Extension

4. Define proper attack impact metrics for the developed algorithms.

Learning Outcomes

Upon completion of the project, you will expect to:

1. Learn about attack modelling and simulation for dynamical systems.
2. Learn about formal modelling methods in control.
3. Gain experience in programming and simulation.
4. Develop good mathematical and algorithmic reasoning.
5. Be exposed and become comfortable with multidisciplinary problems that require analytical and innovative thinking.

Acknowledgments

I would like to show my deepest appreciation for my supervisor, Nikolaos Athanasopoulos for dedicating their time and patience to mentor me throughout this project, your feedback has been invaluable. Additionally, my support tutor Elaine Hogg, who has been a massive help throughout the years for their editing, advice and always keeping me on track. Thanks also go out to all the university staff for facilitating and inspiring my academic growth.

I also want to thank my friends and classmates for the laughs, the numerous late-night or early-morning discussions and constant moral support. Lastly, I of course must mention my family, who have always believed in me and supported me in everything I do.

Declaration of Originality

I declare that this report is my original work except where stated.

Electronically Signed: M Reaney Dated: 14/4/2023.

…………………………………………………………………………….

Table of Contents

Section 1 – Introduction

This section of the report details the goal of the project, motivations for the research and the different approaches used within key literature, as well as the contribution this work hopes to provide.

1.1 Project Goal

The goal of the project is to further research in analysing the impact of cyber-attacks in cyber-physical systems. To achieve this will require application of control theory within a software environment, enabling simulation of a benchmark cyber-physical system. This system will integrate an attacker that can implement attacks, imitating cyber-security risks. Finally using the software tool to investigate various attack scenarios results, enabling analysis of the impact of the various cyber-threats.

1.2 Research Motivation

A key piece of inspiration for this project is an annual review by Sandberg et al. that covers lots of the literature on the cyber security of control systems, it states *"As many of these systems form the backbone of critical infrastructure ... it is of the utmost importance to develop methods that allow system designers and operators to do risk analysis and develop mitigation strategies"* [1]. The paper provides context to the problem, framing it using game theory, outlining important security technologies and core requirements of control systems. It also highlights the key design assumption that the attacker has complete knowledge of the system, *"the defender should implement security algorithms under the assumption that the algorithms are public and thus known to the adversary"* [1].

The paper then provides research motivation with the real-world examples of cyber security breaches: Stuxnet industry attack 2010 [2]; which resulted in multiple industrial plants in Iran, including a nuclear facility, being infected with a virus that caused malfunctions. Miller & Valasek take over the 2014 jeep Cherokee [3] [4]; where two researchers outlined the vulnerabilities within Wi-Fi connected vehicles, resulting in over a million cars being recalled. Ukraine Power Grid attack 2015 [5] [6]; this cyber-attack left over 200,000 consumers without power for up to six hours. As these cyber-security vulnerabilities can lead to devastating outcomes its important research continues to evolve its methods to manage the risk of cyber-attacks and develop security measures to mitigate their effects.

1.3 Approaches in other literature

After Stuxnet in 2010 there was an increase in activity in this area of research. One of the earlier papers in the area by Cárdenas et al. [7] outlines the future research challenges and wanted to *"initiate the discussion between control and security practitioners"* [7]. The majority of literature presents different strategies for impact estimation for the following attacks: false data injection (FDI) [8], bias injection [9], denial of service (DoS) [10] [11], sign alternation attacks [12], rerouting [13] or replay attacks [14] [15]. Some additional papers that seek to evaluate the impact of attacks include: [16] [17] [18].

Another key piece of inspiration for this project is a standout paper by Milošević et al. [8] which presents all the attacks mentioned before. The paper provides a fantastic numerical example of a water tank system which was adapted for use within this project. It also discusses stealthy constraints a more complex form of attacking that uses knowledge of the complete system, specifically the anomaly detector, to attack un-noticed by the system's safety mechanism. Across literature these types of attacks are primarily referred to as 'stealthy' attacks [7] and are of particular focus due to the dangers they pose if the system cannot react to them. A common approach to this scenario is modelling the maximum damage the attacker can do while meeting the stealthy constraints of the system [19]. Another common approach is to find the entire range of possible attacks that satisfy the stealthy constraints using complex high-order linear equations, sometimes referred to as the stealthy set. Using this stealthy set, analysis can be completed to estimate the worst possible impact on the system. Some additional papers that investigate stealthy attacks include: [20] [21] [22] [23].

Another interesting direction taken by research is 'combination' [24] or 'coordinated' [25] attacks, for this report they will be referred to using the former. These seek to combine different attack types simultaneously to produce a more complex attack strategy. Research focuses on using DoS attack or Replay attacks on the output of a system to hide injection attacks on the input. This approach results in a unique way to preform 'stealthy' attacks as the system will be unable to detect the injection attack if the anomaly detector is fed the incorrect output information.

With regards to the programming of the simulation tool, inspiration was taken from the Sim3Tanks model [26], which presents a large MATLAB/Simulink system which simulates faults and noise within cyber-physical system. However, there is no implementation of cyber-attacks. Verica Radisavljevic-Gajic [27] provides instruction for linear control system design within MATLAB/Simulink.

1.4 Contribution

This project hopes to contribute to current academic research by providing a software tool capable of simulating a cyber-physical system complete with a range of different attack types. The goal of the tool is to help facilitate ways to estimate the impact of various cyber-threats. Although other pieces of research implement their mathematical modelling with the help of software, a configurable model capable of simulating multiple attack types is, to the best of our knowledge, a novel contribution. The hope is that the tool can be adapted to investigate other cyber-physical systems other than just the chosen example under the same set of attack types. This means the software tool design should be easily editable and interchangeable for others to customise it to suit their needs. For these reasons it will be publicly available on GitHub and feature a GUI making it easy for anyone to use.

With this tool, this project seeks to evaluate a novel approach to implementing stealthy attacks which will be referred to a ‘state dependant’ attacks. Mentioned in the previous sub-section, commonly the impact of stealthy attacks is estimated using linear inequalities from a set of stealthy constrains. The new method still uses the underpinning methodology of stealthy constraints but instead of solving to find all possible, or worst, attacks for a given scenario, enable the attacker to evaluate each attack they wish to send based on how the states of the system would react. If the system will be disrupted to the point of rising the alarm the attack should not be applied that iteration as it violates the stealthy constraints. Due to this dependence on the state’s reaction to an attack they are referred to as ‘state dependant’ attacks. Additionally, the tool will aim to incorporate combination attacks of all attack types leading to the possibility of estimating the impact of new attack combinations.

Section 2 - Reflection against Interim Report

This section of the report details how the second half of the project progressed with respect to the tasks and plan laid out in the interim report, written January. Comparison against the specification objectives directly can be found in *Section 8.1*.

2.1 Project Tasks

Within the Interim report several tasks were created to achieve the project objectives. Below is the original set of seven main tasks and their status at the time of writing in January:

1. Learn about attack strategies and modelling of cyber-physical systems – Complete.
2. Designing a control system model of a cyber-physical system – in progress.
3. Incorporate attack models to cyber-physical system design – in progress.
4. Creation of an anomaly detector – in progress.
5. Analysing impact of attacks using proper attack impact metrics – planned.
6. Implementation and analysis of state dependant attacks – planned.
7. Simulation of multiple attack scenarios – planned.

All these tasks have now been completed additionally two extra tasks were added. Below is a list showing in which part of the report each of the tasks were fulfilled.

1. Learning about attack strategies and modelling was achieved through literary review in *Sections 1.2 & 1.3*.
2. The design of the control system model is presented in *Section 3* with implementation in *Section 5*.
3. The incorporation of attacks is presented in *Section 4* with implementation in *Section 5*.
4. The anomaly detector design is presented in *Section 3.3* with implementation in *Section 5*.
5. Analysing impact of attacks is presented in *Sections 6 & 7*.
6. The implementation and analysis of state dependant attacks, presented in *Section 4.4* with implementation in *Section 5*, and results in *Section 6*.
7. Results of simulations presented in *Section 6*.
8. Implementation of combination attacks in *Section 4* with implementation in *Section 5*, and results in *Section 6*.
9. Creation of a graphical user interface implemented and presented in *Section 5.4*.

2.2 Project Schedule

The Gantt chart below in *Figure 2.2.1* was taken from the interim report and used the tasks outlined in the previous subsection to create the project schedule.

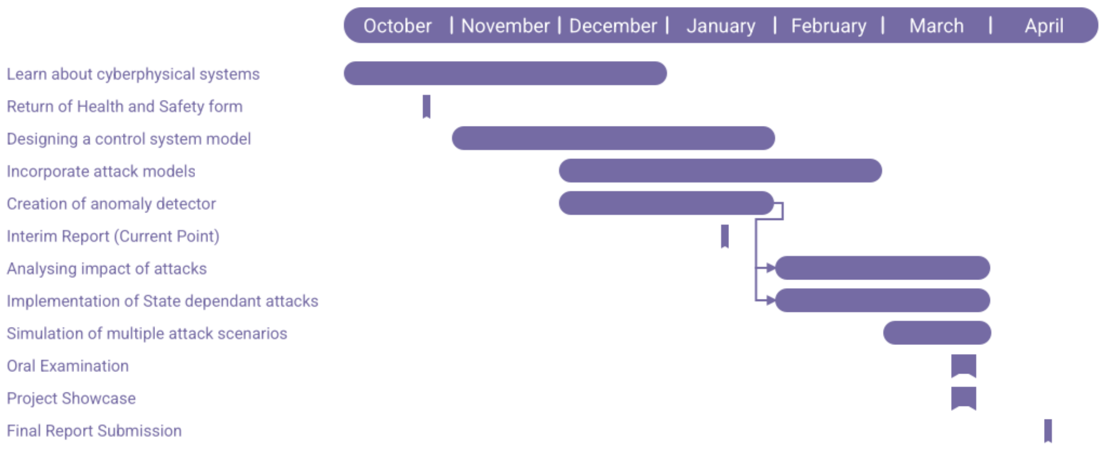


Figure 2.2.1 – Interim Report Schedule Gantt Chart

Below in *Figure 2.2.2* is an updated Gantt chart to reflect the actual schedule of the research.

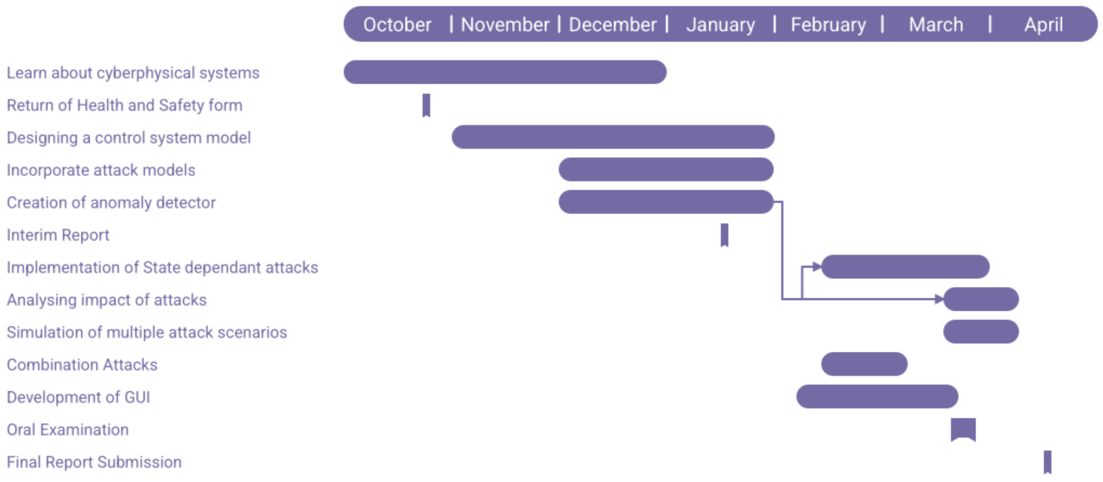


Figure 2.2.2 - Updated Schedule Gantt Chart

2.3 Change in Project Direction

Around the end of January there was a change in the direction for the outcome of the project. This shift moved the focus from creating complex impact metrics and towards creating a sophisticated software tool that could facilitate future analysis. For this reason, development of a customisable tool and a user-friendly GUI became a bigger focus within the project.

Section 3 - Control System Model

This section details the equations which describe the cyber-physical system which will be the foundation of the software simulation. The networked control system being modelled is adapted from the example used in a paper by Milošević et al. [12] which was originally described within the book: Diagnosis and Fault-Tolerant Control [28].

3.1 Three-Tank Water System

The chosen cyber-physical system is a three-tank water system, with four inputs (u values) and three states (x Values): *"The states are the volume in Tank 3 (x1), the volume in Tank 2 (x2), and the temperature in Tank 2 (x3). The control signals are the flow rate of Pump 2 (u1), the openness of the valve (u2), the flow rate of Pump 1 (u3), and the power of the heater (u4). We assume that the control objective is to keep a constant temperature in Tank 2. The objective is achieved by injecting hot water from Tank 1 and cold water from Tank 3"* [12]. Below in *Figure 3.1.1* is visual representation of the networked control system.

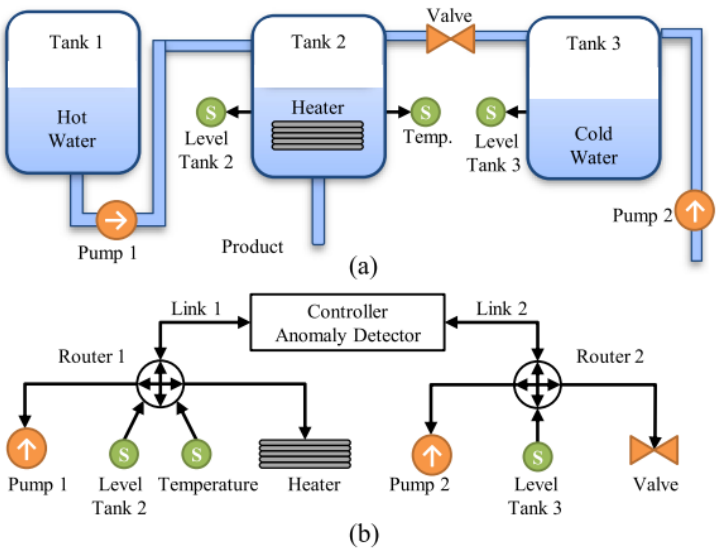


Figure 3.1.1 - tank model "(a) Physical part of a chemical process with four actuators (two pumps, one heater, and one valve) and three sensors (two level sensors and one temperature sensor). (b) Cyber part of the process" [12]

3.2 Mathematical Modelling

The system contains a plant, this models the dynamics of the water tank system. An estimator, this generates a set of expected output values given a set of inputs to enable feedback control. A controller, this takes the estimator values and generates inputs with the aim of creating a stable system. The system is discrete as some functions rely on information from specific time intervals. *t* is used to denote an interval of time.

Plant - The equations below describe the plant:

$$x(t+1) = Ax(t) + Bu(t) + v(t) \tag{3.2.1}$$

$$y(t) = Cx(t) + w(t) \tag{3.2.2}$$

Where, $x(t)$ represents the plant state, $y(t)$ are the outputs, $u(t)$ are the inputs. $v(t)$ and $w(t)$ represent process and measurement uncertainties, these are implemented within *section 5* and are independent sources of gaussian noise with zero mean. All states in the system are observable so the terms outputs $y(t)$ and states $x(t)$ are interchangeable, $x(t) = y(t)$.

Estimator - The equations below describe the estimator:

$$\hat{x}(t+1) = (A - LC)\hat{x}(t) + [B\ L]\begin{bmatrix} u \\ y \end{bmatrix}(t) \tag{3.2.3}$$

$$\hat{y}(t) = C\hat{x}(t) \tag{3.2.4}$$

Where, $\hat{y}(t)$ represents the estimated output value, $\hat{x}(t+1)$ the plant state one iteration ahead of $x(t)$. $\begin{bmatrix} u \\ y \end{bmatrix}(t)$ is a vector containing both input and output signals. The estimator gain L is defined so that $(A - LC)$ and $[B\ L]$ will drive the error of the system to zero.

Controller – The equation below describes the controller:

$$u(t) = -\mathrm{k}\,\hat{x}(t) \tag{3.2.6}$$

Where, k represents the controller gain. This value is found using a linear-quadratic regulator which ensures in the absence of attacks the system will converge to equilibrium.

Numerical Values - The numerical values used to describe the plant are taken from the example used in a paper by Milošević et al. [12]:

$$A = \begin{bmatrix} 0.96 & 0 & 0 \\ 0.04 & 0.97 & 0 \\ -0.04 & 0 & 0.90 \end{bmatrix} \tag{3.2.7}$$

$$B = \begin{bmatrix} 8.8 & -2.3 & 0 & 0 \\ 0.20 & 2.2 & 4.9 & 0 \\ -0.21 & -2.2 & 1.9 & 21 \end{bmatrix} \tag{3.2.8}$$

$$C = I_3 \tag{3.2.9}$$

3.3 Anomaly Detector

Below is the equation used to define the anomaly detector:

$$\mathrm{e(t)} = \|y(t) - \hat{y}(t)\|_\infty \tag{3.3.1}$$

Where, $\mathrm{e(t)}$ represents the error of the system, $y(t) - \hat{y}(t)$ represents the difference between the real and estimated output values. The error is using the infinity norm, the reason for its selection is that it selects the largest magnitude of the difference values in each vector. This is important in the context of cyber-security as the detector will be able to spot error within a single value in the vector which may correspond to an attack aimed at a single part of the network. This design choice is supported by an existing paper by Nabil H. Hirzallah and Petros G. Voulgaris [29].

Alarm Flag – The system is designed so that an alarm flag is raised if the error exceeds a detector threshold defined by the user. In its simplest form this can be defined as:

$$\text{Alarm flag is raised } if\ e(\mathrm{t}) > \text{Detector threshold} \tag{3.3.2}$$

As described in *Section 5* a more complex implementation was introduced within the software implementation which enables the user to define more complex requirements for raising the alarm flag based on historical error values. This can be described as the following.

1. Exceeding several sequential iterations where $e(\mathrm{t}) >$ Detector threshold within a period.
2. Exceeding a total number of iterations where $e(\mathrm{t}) >$ Detector threshold within a period (average rate of errors).

Section 4 - Attack Implementation

This section details the equations which describe the cyber-attacks which will be implemented within the software tool. As the sensors and actuators are connected in a network using link 1 and link 2, as shown in *Figure 3.3.1 (b)*, these network links represent the points in which the attacker can gain access to manipulate the inputs and outputs of the system. For this reason, the attacker is modelled as an additive signal as they effect the system by adding their attack signal.

4.1 Attack Modelling

The equations below describe the attacked input (4.1.1) and attacked output (4.1.2) signals.

$$u_\alpha(t) = u(t) + v(t) + \alpha_x(t) \tag{4.1.1}$$

$$y_\alpha(t) = y(t) + w(t) + \alpha_y(t) \tag{4.1.2}$$

Where, $u_\alpha(t)$ represents the resulting input signal, $u(t)$ the original input signal, $v(t)$ process uncertainties and $\alpha_x(t)$ the input attack signal. $y_\alpha(t)$ represents the resulting output signal, $y(t)$ the original output signal, $w(t)$ measurement uncertainties and $\alpha_y(t)$ the output attack signal. As both equations have the attacks take the same form, a general attack equation can be created:

$$ts_\alpha(t) = ts(t) + \mathrm{\alpha}(t) \tag{4.1.3}$$

Where, $ts_\alpha(t)$ represents the resulting target signal, $ts(t)$ represents the target signal including its uncertainty and $\mathrm{\alpha}(t)$ represents the attack signal. So, through the attack signal $\mathrm{\alpha}(t)$ the attacker can influence the resulting signal $ts_\alpha(t)$.

4.2 Attack Types

Six different attacks have been implemented within this project all based on the additive modelling equations from the previous subsection, these include:

False data injection (FDI) – *"false data injection attack affects the data integrity of packets by modifying their payloads"* [8]. This is implemented by adding an attack signal belonging to a uniformly distributed random set between 0 and 1 ($\mathrm{R}(t)$), with its magnitude scaled by the attacker (S). Modelled as:

$$\alpha(t) = \mathrm{S} * \mathrm{R}(t) \tag{4.2.1}$$

Bias injection (Bias) - *"the attacker's goal is to increase the mean square estimation error by adding a constant bias to some of the sensor measurements"* [9] This is implemented by adding a constant value defined by the user (Z) to the target signal. Modelled as:

$$\alpha(t) = Z \tag{4.2.2}$$

Denial of service (DoS) – *"attacks that prevent delivery of control and measurement data packets"* [11] This is implemented by adding a negative version of the target signal to itself, resulting in the target signal being equal to zero. Modelled as:

$$\alpha(t) = -ts(t) \tag{4.2.2}$$

$$ts_\alpha(t) = 0 \tag{4.2.3}$$

Sign alternation – *"This attack can turn negative feedback into positive and potentially destabilize the system"* [12]. This is implemented by adding a double the negative of the target signal to itself, resulting in a change of the target signals polarity. Modelled as:

$$\alpha(t) = -2 * ts(t) \tag{4.2.4}$$

$$ts_\alpha(t) = -ts(t) \tag{4.2.5}$$

Rerouting – *"the adversary is able to re-route the measurements transmitted by the sensors"* [13]. This is a more complex implementation as it requires swapping one target signal for another. This is implemented by first removing the original signal by adding a negative version of the target signal to itself, then adding the other signal over itself. Modelled as:

$$\alpha_1(t) = -ts_1(t) + ts_2(t) \tag{4.2.6}$$

$$ts_\alpha(t) = ts_2(t) \tag{4.2.7}$$

Where, the result is replacing the target signal $ts_1(t)$ with another signal $ts_2(t)$.

Replay Attacks – "the attacker will hijack the sensors, observe and record their readings for a certain amount of time and repeat them afterwards" [14] This is implemented by first removing the original signal by adding a negative version of the target signal to itself, then adding a previously recording signal value to itself. Modelled as:

$$\alpha(t) = -ts(t) + ts(t-1) \tag{4.2.8}$$

$$ts_\alpha(t) = ts(t-1) \tag{4.2.9}$$

Where, the result is replacing the target signal $ts(t)$ with a value of that signal from a previous iteration $ts(t-1)$.

4.3 Combination Attack Modelling

As described in *section 3,* the control system being modelled in this project has several inputs and outputs, each of which can be independently targeted by the attacker. As discussed in the introduction (*section 1.3*), other research has evaluated at the combination of injection and DoS attacks. However, this project aims to be able to simulate any combination of the six attack types listed in the previous sub-section.

4.4 Stealthy/State Dependant Attack Modelling

Stealthy attacks can be defined as attacks that go undetected by the anomaly detector. This can be modelled by adapting equation *3.3.1/3.32*:

$$e_\alpha(\mathrm{t}) = \| y_\alpha(t) - \hat{y}(t) \|_\infty \tag{4.4.1}$$

$$\text{Attack is stealthy } if\ e_\alpha(\mathrm{t}) < \text{Detector threshold} \tag{4.4.2}$$

Where, $e_\alpha(\mathrm{t})$ represents the attacked error value and $y_\alpha(t)$ the attacked system's outputs. The estimate $\hat{y}(t)$ is not directly affected by the attacker. Similarly, by adapting equation *3.3.2* used to define the alarm conditions.

As explained in the introduction this approach is novel as it seeks to determine if an attack is stealthy for a single iteration using the state of the plant $y_\alpha(t)$, instead of finding all the possible stealthy values belonging to a set of constraints (Soled using linear inequalities). So far, a proof of concept for this form of state dependent attack has been implemented in software for the input attacks. The implementation for the input attacks has the attacker, for each iteration they want to 'stealthily' attack, follow the instructions:

State dependant input attack process:

1. Using $u(t)$ Calculate $u_\alpha(t)$.
2. Calculate a $y_\alpha(t)$ using $u_\alpha(t)$ within the mock plant mock plant.
3. Calculate a mock $\hat{y}(t)$ using $u(t)$ and $y_\alpha(t)$ within the mock estimator.
4. Calculate the $e_\alpha(\mathrm{t})$ value using $y_\alpha(t)$ and $\hat{y}(t)$ with equation *4.4.1*
5. Evaluate $e_\alpha(\mathrm{t})$ against the detector threshold using equation *4.4.2*:

   If $e_\alpha(\mathrm{t}) < \mathrm{Detector\ threshold}$, it is safe to send the attack as it will be undetected.

   If $e_\alpha(\mathrm{t}) > \mathrm{Detector\ threshold}$, do not send the attack as it will be detected.

Currently the state-dependent output attacks are still a work in progress in software implementation. However, it would fundamentally follow a similar structure, for each iteration the attacker wants to 'stealthily' attack, follow the instructions:

State dependant output attack process:

1. Using $y(t)$ Calculate $y_\alpha(t)$.
2. Calculate a mock $\hat{y}(t)$ using $u(t)$ and $y_\alpha(t)$ within the mock estimator.
3. Calculate the $e_\alpha(\mathrm{t})$ value using $y_\alpha(t)$ and $\hat{y}(t)$ with equation *4.4.1*
4. Evaluate $e_\alpha(\mathrm{t})$ against the detector threshold using equation *4.4.2*:

   If $e_\alpha(\mathrm{t}) < \mathrm{Detector\ threshold}$, it is safe to send the attack as it will be undetected.

   If $e_\alpha(\mathrm{t}) > \mathrm{Detector\ threshold}$, do not send the attack as it will be detected.

Section 5 - Software Design

This section details the design of the software tool. A natural choice for programming environments were MATLAB & Simulink as they provide excellent tools for mathematical modelling. Simulink offers ways to express models in a more visual manor which is helpful for system attacks and analysis. Lastly, MATLAB app design enables the creation of simple user interfaces that integrate directly with the MATLAB workspace environment. The tool is available at https://github.com/mattr862/three-tank-water-system.

5.1 MATLAB Scripts

The MATLAB script *intialise_tank_model.m* manages the initialisation of system values for the Simulink model. This implements parts of the control theory mentioned in *Section 3* of the report: the plant and estimator dynamics and the gain values for the controller and estimator using linear quadratic regulator, while developing the system MATLAB was helpful for ensuring the system had a working design. Shown in *Figure 5.1.1* is the state's step response, as the states are converging to equilibrium, this means the design of the system is stable.

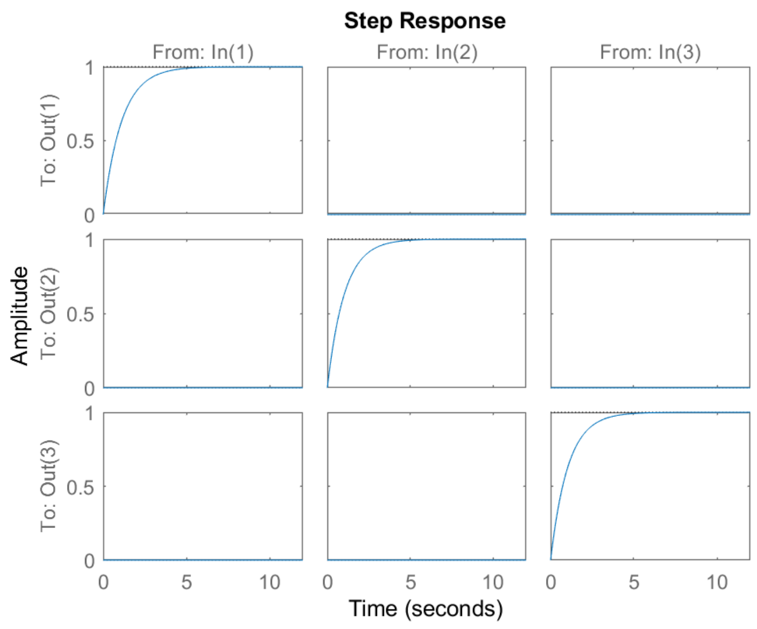


Figure 5.1.1 - Step Response

Additionally, there is a MATLAB script *tank_model_config*.m, the user can modify this to enable control of all system parameters without having use the GUI. This file can be used to save important attack configurations that users may wish to revisit.

Within the Simulink model there is also several MATLAB function blocks that implement scripts, these include the anomaly detector, the input and output attack signal generators, and the stealthy attack filters.

5.2 Simulink Model

The Simulink model is within *tank_model.slx*, the top-level structure can be seen in *Figure 5.2.1*. This structure is comprised of four main subsystems: the attacker, system (plant), estimator & controller, detector. Additionally. there is a simple display that shows the status and trigger time of the alarm flag. Throughout the model, scopes and 'to workspace' blocks are placed to automatically plot graphs and record variables.

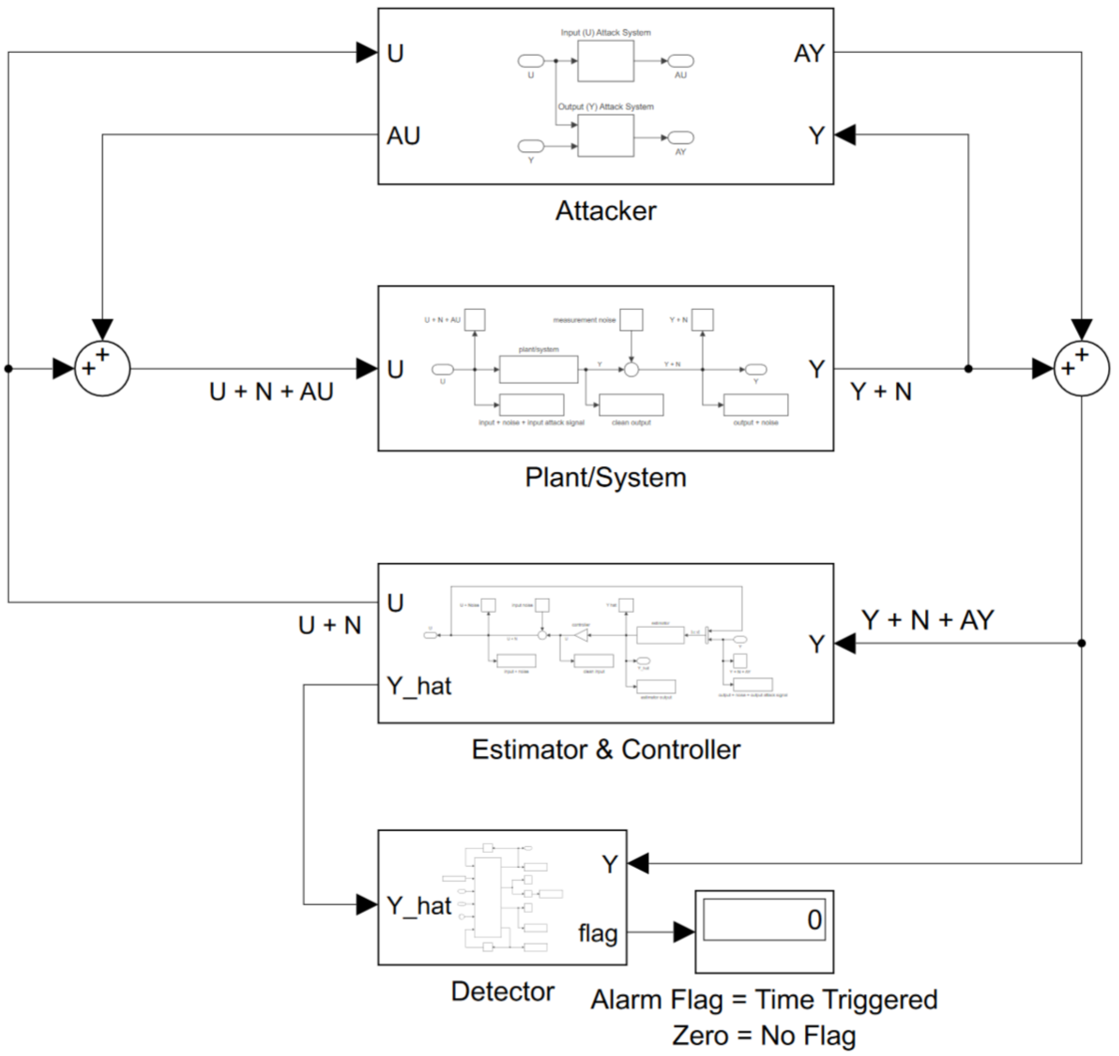


Figure 5.2.1 – Top-Level Simulink Model

Attacker Subsystem – This subsystem, shown in *Figure 5.2.2*, represents the attacker and it made up of two further subsections the input attack system (U attack system) and the output attack system (Y attack system). As described in *Section 4*, the attack signals are additive, this is represented in the model by having the attack signals added onto either the system inputs or outputs as seen in *Figure 5.2.1*.

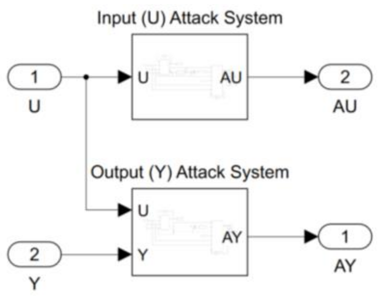


Figure 5.2.2 - Attacker Subsystem

System (Plant) Subsystem – This subsystem, shown in *Figure 5.2.3*, implements the dynamics of the plant and adds the measurement noise to simulate measurement uncertainties.

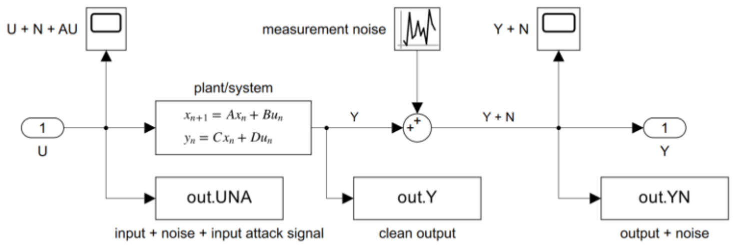


Figure 5.2.3 - System (Plant) Subsystem

Estimator & Controller Subsystem - This subsystem, shown in *Figure 5.2.4*, implements the estimator and controller and adds input noise to simulate process uncertainties.

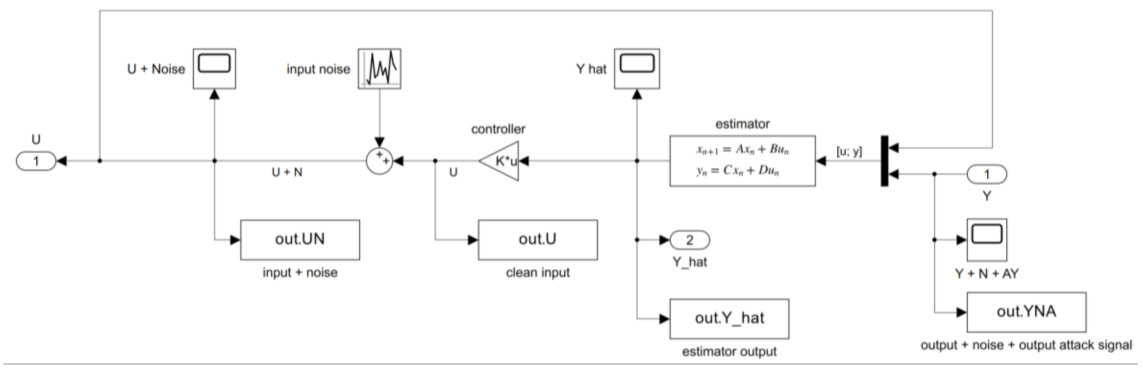


Figure 5.2.4 - Estimator & Controller Subsystem

Detector Subsystem - This subsystem, shown in *Figure 5.2.5*, implements the anomaly detector through the use of a MATLAB script function, code for this can be found in *detector.m*. To ensure the anomaly detector is not flagged while the system is initialising the system only starts recording error after 0.25 seconds.

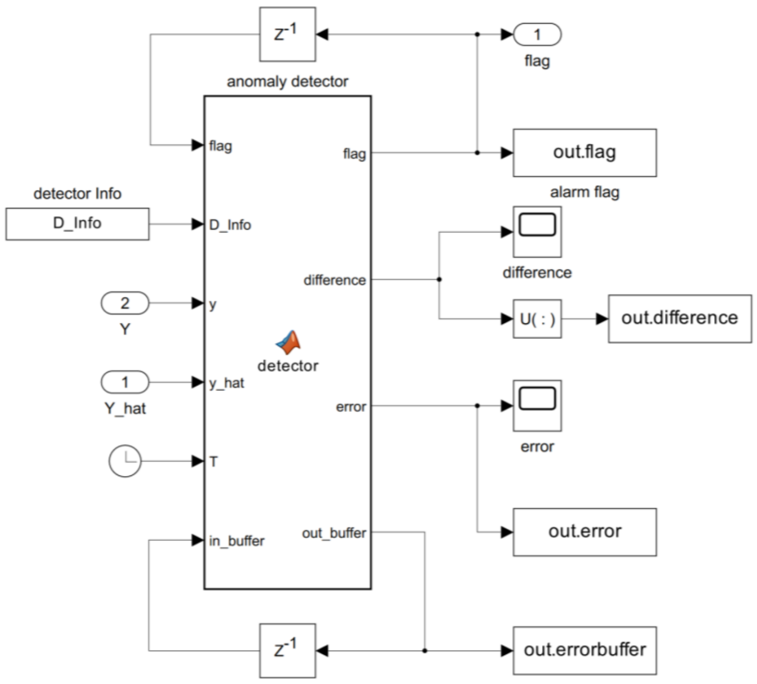


Figure 5.2.5 - Detector Subsystem

Input Attacker Subsystem - This subsystem, shown in *Figure 5.2.6*, implements the input attack signals through the use of the through the use of a MATLAB script function, code for this can be found in *U_attack_generator.m.* Additionally, if enabled it feeds these signals through the mock system used to implement the state dependant input attack as described *in section 4.4*. So, of this is done through the use of another MATLAB script function, code for it be found in *U_state_dependant_filter.m.*

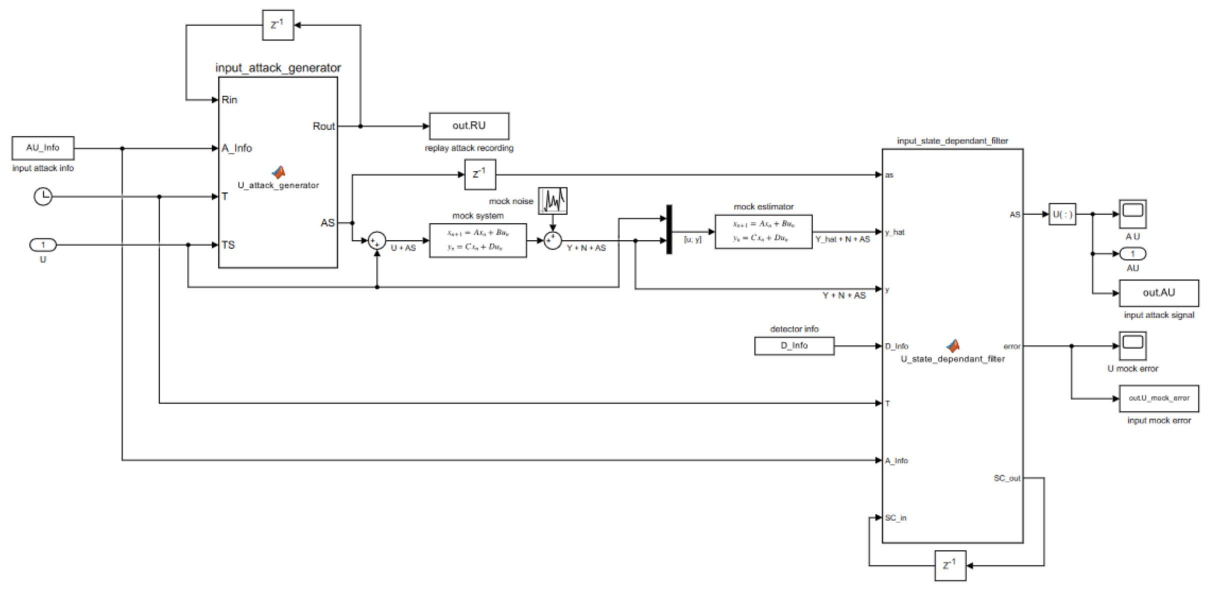


Figure 5.2.6 - Input Attacker Subsystem

Output Attack Subsystem - This subsystem, shown in *Figure 5.2.7*, implements the output attack signals, it works in a similar way to the input attacks. Using the two MATLAB script functions, code for these can be found in *Y_attack_generator.m* and *Y_state_dependant_filter.m.* This subsystem also features the current work-in-progress implementation for state dependant output attacks as described *in section 4.4*.

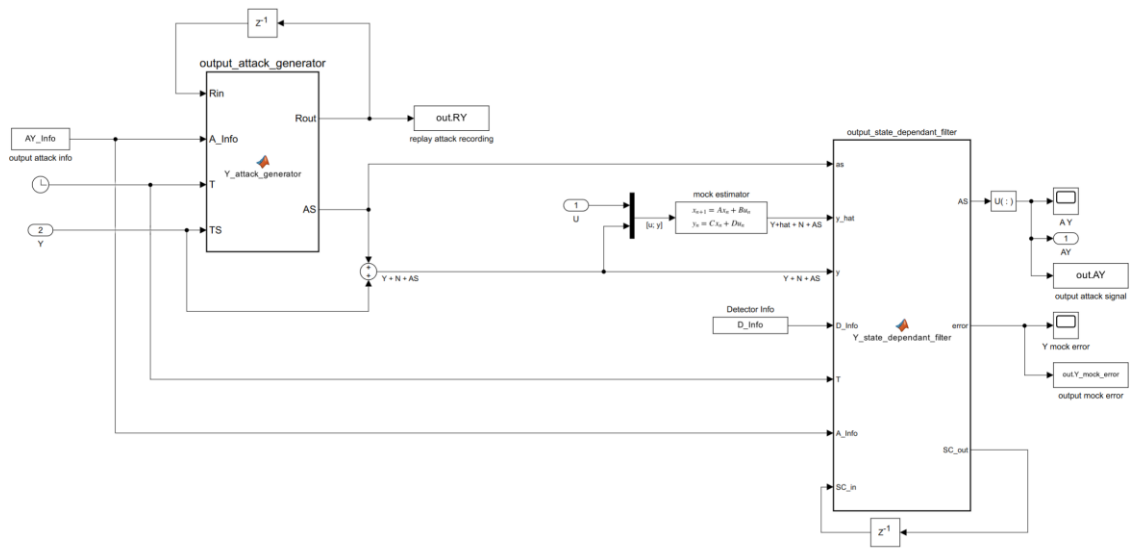
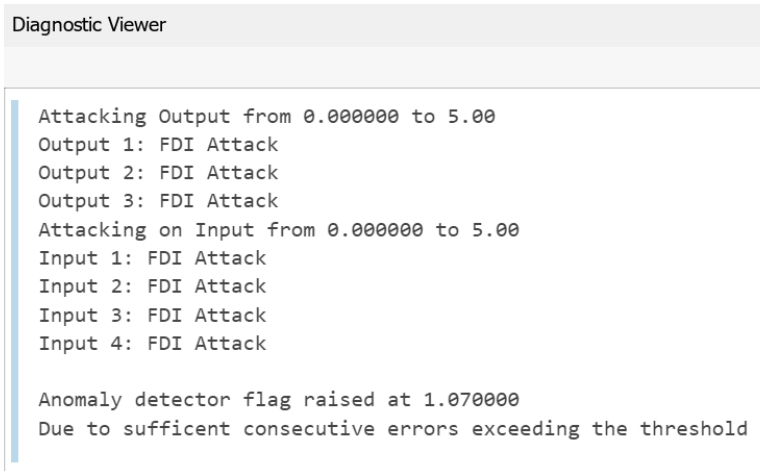


Figure 5.2.7 - Output Attacker Subsystem

Also contained within the Simulink model, specifically the MATLAB scripts, are several debug messages which paste information in the diagnostic viewer. This is to aid in the analysis and creation of the model, these incudes information in attacks selected and the time at which the anomaly detector was raised. An example of these messages can be seen in *Figure 5.2.8* below:



```
Attacking Output from 0.000000 to 5.00
Output 1: FDI Attack
Output 2: FDI Attack
Output 3: FDI Attack
Attacking on Input from 0.000000 to 5.00
Input 1: FDI Attack
Input 2: FDI Attack
Input 3: FDI Attack
Input 4: FDI Attack

Anomaly detector flag raised at 1.070000
Due to sufficent consecutive errors exceeding the threshold
```

Figure 5.2.8 - Simulink Debug Messages

5.3 System Parameters

As the software tool is designed to easily configured to preform different types of analysis, for this reason there are a number of key parameters which can be controlled by the user through the MATLAB workspace using scripts or the GUI. These parameters fall into five main categories listed below. The default values have also been listed; these represent the model running for five seconds without the presence of any attacks.

General properties – This includes:

- Stop time – total duration of the simulation, in seconds. Default - 5 seconds
- Initial state values (X1, X2, X3) – These represent the plant values at the start of the simulation. Default: 10, 15, 20
- Initial Estimator values (X1, X2, X3) – These represent the estimator values at the start of the simulation. Default: 9.5, 14.5, 19.5.

Noise/Uncertainties – These values govern the noise generator components that model uncertainties of the system. This includes:

- Measurement noise (min, max, seed) – This represents measurement uncertainties. Default: -0.15, 0.15, randi(1000) – random integer between 1-1000.
- Input noise (min, max, seed) – This represents process uncertainties. Default: -0.05, 0.05, randi(1000) – random integer between 1-1000.

For both of these above, the min and max values govern the distribution of the noise. The seed value represents a unique noise distribution and can be used to replicate the same results. However, the seed value is normally randomised.

Anomaly detector – These values govern the anomaly detector; all values are stored within a vector of length 4. This includes:

- Detector threshold – This represents the value above which the alarm flag can be raised. Default: 1.
- Buffer size (max 100) – This represents the number of error values stored within memory, this defines the period for more complex detector design. Default: 5.
- Consecutive errors requirement – This represents the number of consecutive errors above the threshold within the period defined by the buffer required to set off the alarm. Default: 3.
- Total errors requirement - This represents the number of total errors above the threshold within the period defined by the buffer required to set off the alarm, average rate of errors within the buffer. Default: 3.

Input attacks - These values govern the input attack configuration; all values are stored within a vector of length 9. By default, all values are zero, as no attacks are present. This includes:

- Attack Type on U1, the flow rate of pump 2.
- Attack Type on U2, Openness of the valve.
- Attack Type on U3, the flow rate of pump 1.
- Attack Type on U4, power of the heater.
- Attack start time, in seconds.
- Attack end time, in seconds.

- Stealthy Attack filter, 0 = off. Used to denote if attack should be stealthy.
- FDI attack scaler for random number distribution. This is value *S* in *equation 4.2.1.*
- Bias attack offset value. This is value *Z* in *equation 4.2.2.*

Output attacks - These values govern the input attack configuration; all values are stored within a vector of length 9. By default, all values are zero, as no attacks are present. This includes:

- Attack Type on Y1, tank 3 volume.
- Attack Type on Y2, tank 2 volume.
- Attack Type on Y3, tank 2 temperature.
- Unused (vector lengths kept the same for simplicity of software design).
- Attack start time, in seconds.
- Attack end time, in seconds.
- Stealthy Attack filter, 0 = off. Used to denote if attack should be stealthy.
- FDI attack scaler for random number distribution. This is value *S* in *equation 4.2.1.*
- Bias attack offset value. This is value *Z* in *equation 4.2.2.*

The Numerical value for each attack type are as follows:

1. = FDI
2. = Bias
3. = Dos
4. = Sign Alternation
5. = Rerouting
6. = Replay

Any other value represents no attack.

5.4 GUI (Graphical User Interface)

The GUI was created using MATLAB app designer, featuring three tabs accessible on the top left. It reads and writes all variables to the standard MATLAB workspace. As the Simulink model also shares this workspace that is how the two are able to communicate.

Output Plots - This tab, shown in *Figure 5.4.1*, contains a 'Refresh Graphs' button, used to load in graphs when a new simulation is run. The alarm trigger time and flag light, it goes red if the alarm is raised (white if no alarm). Eight different plots, all of which contain MATLAB's build in suite of tools to interact, save, edit. These eight plots include:

- Input.
- Input attack signal.
- Attacked input.
- Output.
- Output Attack signal.
- Attacked output.
- Estimator values.
- Error.

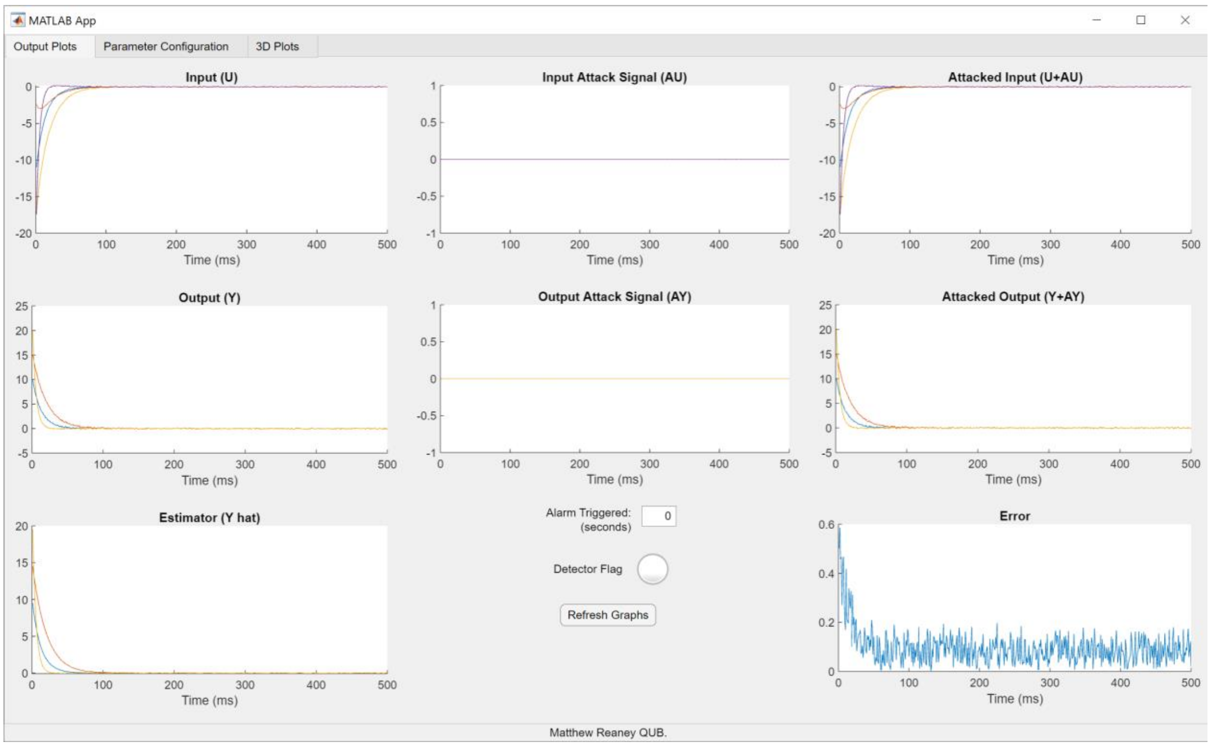
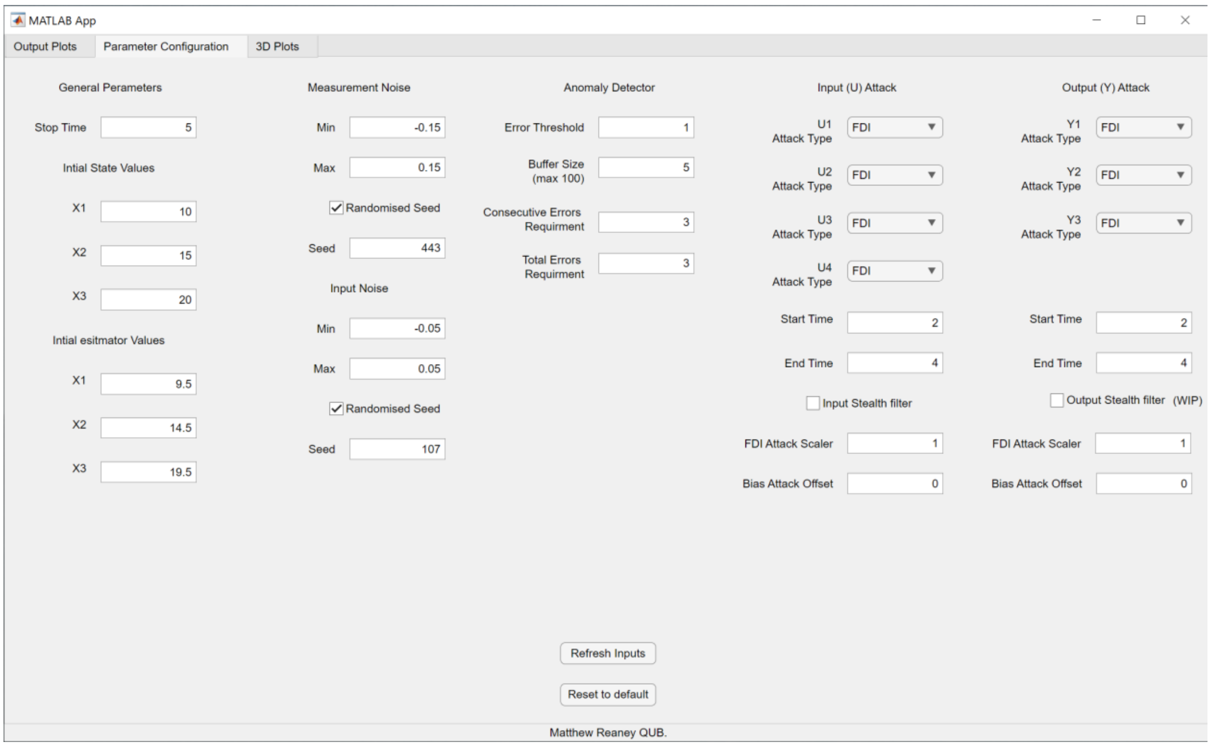


Figure 5.4.1 - GUI, Output plots

Parameter Configuration - This tab, shown in *Figure 5.4.2*, contains ways to edit the system parameters discussed in the previous subsection of the report. Additionally, there two buttons, a 'Refresh Inputs' button, used to load any changes to the parameters made within the workspace. A reset to default button that sets all parameters to the default also specified in the previous subsection.



General Perameters
Stop Time 5
Intial State Values
X1 10
X2 15
X3 20
Intial esitmator Values
X1 9.5
X2 14.5
X3 19.5

Measurement Noise
Min -0.15
Max 0.15
☑ Randomised Seed
Seed 443
Input Noise
Min -0.05
Max 0.05
☑ Randomised Seed
Seed 107

Anomaly Detector
Error Threshold 1
Buffer Size (max 100) 5
Consecutive Errors Requirment 3
Total Errors Requirment 3

Input (U) Attack
U1 Attack Type FDI
U2 Attack Type FDI
U3 Attack Type FDI
U4 Attack Type FDI
Start Time 2
End Time 4
☐ Input Stealth filter
FDI Attack Scaler 1
Bias Attack Offset 0

Output (Y) Attack
Y1 Attack Type FDI
Y2 Attack Type FDI
Y3 Attack Type FDI
Start Time 2
End Time 4
☐ Output Stealth filter (WIP)
FDI Attack Scaler 1
Bias Attack Offset 0

Figure 5.4.2 - GUI, Parameter Configuration

3D Plots - This tab, shown in *Figure 5.4.3*, similar in design to the output tabs also contains a 'Refresh Graphs' button, used to load in graphs when a new simulation is run. Five different 3D plots, all of which contain MATLAB's build in suite of tools to interact, save, edit. These five plots include:

- Output 3D.
- Output Attack signal 3D.
- Attacked output 3D.
- Estimator values 3D.
- Error 3D. This uses the difference vector, which has not yet been put through the infinity norm.

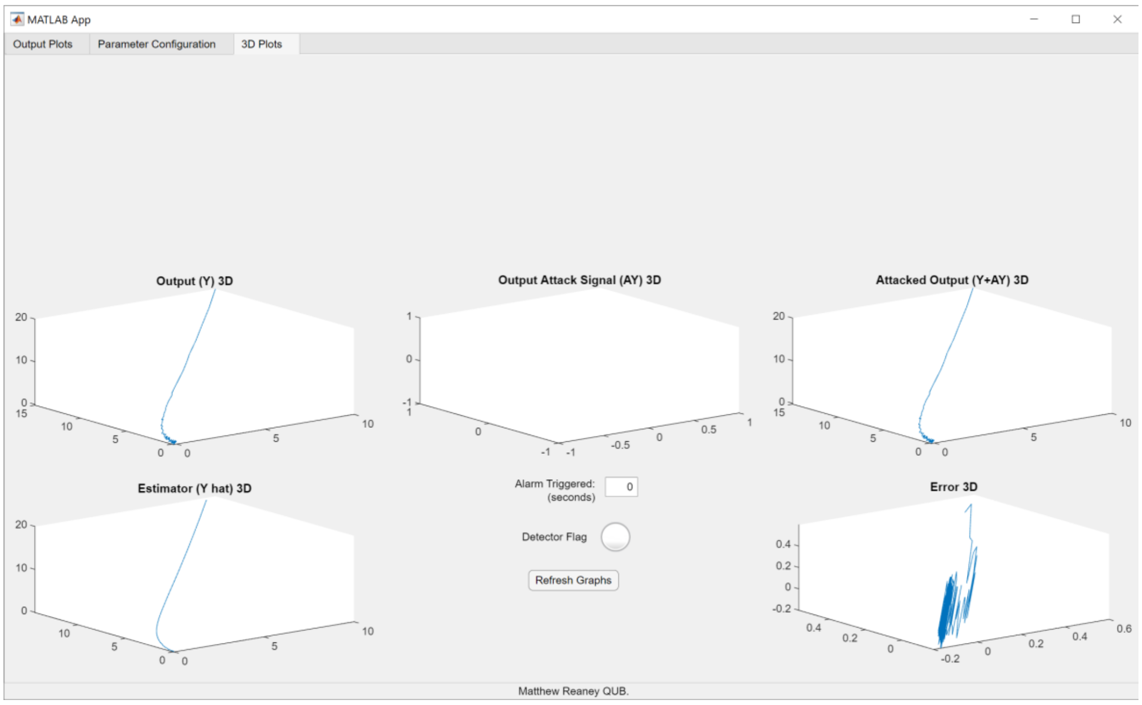


Figure 5.4.3 - GUI, 3D Plots

5.5 Instructions for Use

The MATLAB project was made using the 2022B release and the files available online at https://github.com/mattr862/three-tank-water-system, it shouldn't require any additional MATLAB addons other than Simulink and App designer. Outline of tool operation:

1. Open the project folder within MATLAB.
2. Initialise the model parameters by running the *intialise_tank_model.m* script, this should populate your MATLAB workspace with various objects.
3. Open and run the Simulink model within *tank_model.slx*, upon opening the model scopes to key signals should open in your MATLAB workspace. Numerical outputs can be found within the workspace under the *out* object.
4. If you wish to use the GUI you will need to right click and select run on the file *tank_model_GUI.mlapp*, double clinking will open the file in edit mode.
    a. Alternatively, you can use the file *tank_model_config.m* to change parameters. This can be used as a template to quickly load in scenarios.
5. Upon editing parameters, go back to the Simulink model and hit run again. This should update the Numerical outputs and graphs within in your MATLAB workspace.
6. Going back to the GUI use the 'Refresh Graphs' button to load the new information into the GUI. You can now repeat steps 5 and 6 for other parameters.

Section 6 - Testing

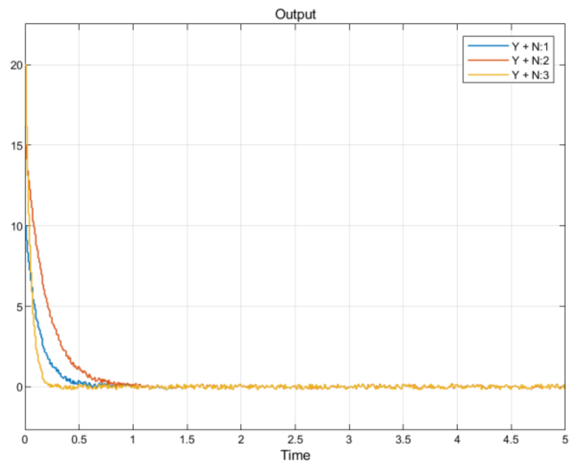
This section details the testing, results, and analysis for five different attack simulation scenarios and laying out the testing methodology and metrics. The goal of these tests is to find the most disruptive attack type and most vulnerable signal to target. Then analyse the impact of combination attacks and Stealthy/State Dependant Attacks.

6.1 Methodology and Metrics

Metrics - To quantify the impact of attacks on the system five metrics will be used. The first two are the mean absolute error (MAE) and the maximum absolute error value (MaxAE). These two metrics will measure the attack's impact on the systems estimator, which feeds the anomaly detector. The third will be the maximum absolute output value (MaxAY). This metric will measure the largest output value from equilibrium (zero for this system) allowing estimation of the maximum disruption caused by a given attack. The first three values will only use data during the chosen attack window, for attacks in steady state this is 2 – 3 seconds, for attacks during initialisation this is 0.25 – 1.25 seconds. A window of one second was chosen as the buffer for replay attack's max size is limited to one second of recording. The fourth metric is time of failure (ToF), this measures the time taken for an output value to exceed a set of bounds, these bounds represent failure states for the system. No upper and lower bounds were provided with the example water tank model so bounds of +/- 10 were selected, given the context of water volumes and temperature. Lastly, the alarm flag raised time (AFT) metric is based on the operation of the anomaly detector and represents when the system is able to identify it is under attack.

For the first three test sets the focus is on MAE, MaxAE, MaxAY. These quantify the extent of the disruption, additionally ToF identifies if the system was pushed into a state representative of failure. For the combination attacks and state dependant attacks the focus shifts towards leveraging the ToF and AFT metrics to analyse the impact on the system before its able to identify it is under attack.

Baseline - To aid analysis an un-attacked baseline will be included alongside results. This baseline will use majority of the default parameters recorded in *Section 5.3*, with the exception of the noise seeds which will be fixed to 500. Below in *Figure 6.1.1* is a graph with the output information and in *Figure 6.1.2* is a graph containing the error information. These can be used as a reference for comparison of other output and error graphs.



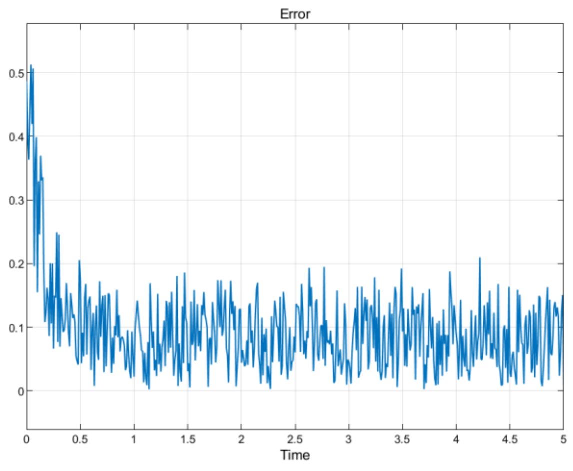


Figure 6.1.1 – Baseline Output

Figure 6.1.2 – Baseline Error

6.2 Attacks During Steady State

The first set of tests seeks to identify the most disruptive attack once the system has reached a steady state. This evaluation is split into two halves, the inputs, and outputs. Each scenario is using the same parameters as the baseline but will have attacks active for one second between, 2 – 3. For input attacks target signals include: U1 – U4. For outputs attacks target signals include: Y1 – Y3.

Input attacks – Shown in *Table & Figure 6.2.1,* is the results for the input attacks during a steady state. Using the mean of MAE, MaxAE and MaxAY, the most disruptive attack on the inputs can be ranked (most to least):

1. Sign Alternation
2. Bias
3. FDI
4. Rerouting
5. Replay
6. DoS

The most disruptive attack is sign alternation having a much larger impact than the rest and it was the only attack to push the system to a point of failure within this set of tests. Another observation is that DoS and replay had a small impact, comparable to that of the baseline.

Table 6.2.1 – Steady State Input Attacks

Attack Type	MAE	MaxAE	MaxAY	Mean	ToF (s)	AFT(s)
Baseline	0.081	0.195	0.167	0.148	-	-
FDI (S=1)	0.554	0.853	1.379	0.929	-	-
Bias (Z=1)	1.072	1.384	2.506	1.654	-	2.12
DoS	0.084	0.204	0.173	0.154	-	-
Sign Alternation	18.702	158.831	271.133	149.555	2.65	2.43
Rerouting	0.170	0.551	0.956	0.559	-	-
Replay	0.116	0.227	0.301	0.214	-	-

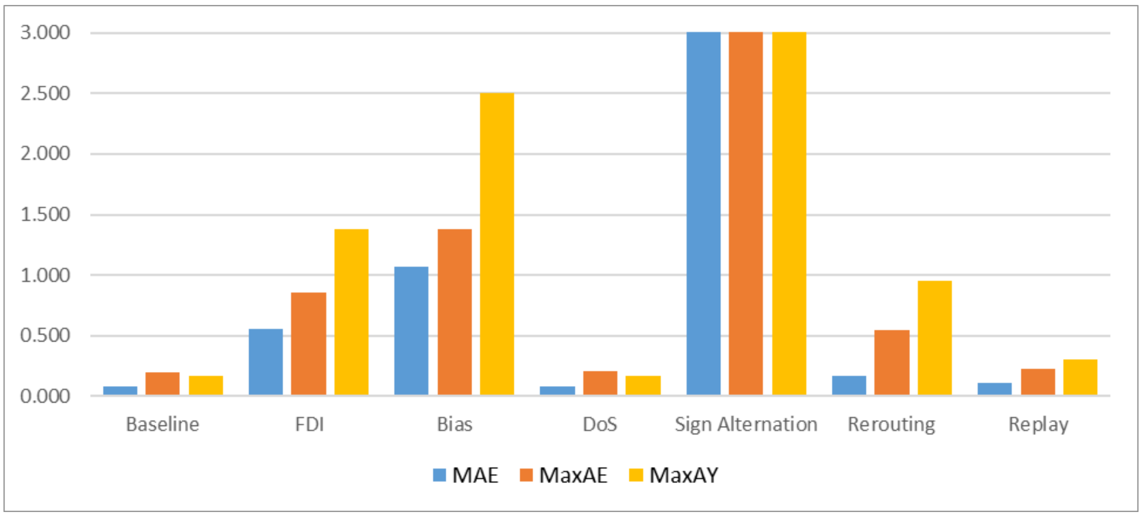


Figure 6.2.1 – Steady State Input Attacks (Sign alternation goes beyond bounds of plot)

Output attacks – Shown in *Table & Figure 6.2.2*, is the results for the output attacks during a steady state. Using the mean of MAE, MaxAE and MaxAY, the most disruptive attack on the outputs can be ranked (most to least):

1. Sign Alternation
2. Bias
3. FDI
4. Replay
5. Rerouting
6. DoS

This produces the similar rankings as the input attacks, with just replay and rerouting swapping places, additionally rerouting is joining DoS and replay in the group of results sharing a small impact, comparable to that of the baseline. The same observation can be made about the most disruptive attack with sign alternation being much larger than the rest. However, it did not push the system to a point of failure within this set of tests.

Table 6.2.2 – Steady State Output Attacks

Attack Type	**MAE**	**MaxAE**	**MaxAY**	**Mean**	**ToF (s)**	**AFT(s)**
Baseline	0.081	0.195	0.167	0.148	-	-
FDI (S=1)	0.434	0.947	0.807	0.729	-	-
Bias (Z=1)	0.239	0.997	1.439	0.892	-	3.03
DoS	0.009	0.129	0.206	0.115	-	-
Sign Alternation	0.356	2.623	4.510	2.497	-	2.91
Rerouting	0.087	0.195	0.193	0.158	-	-
Replay	0.088	0.180	0.231	0.166	-	-

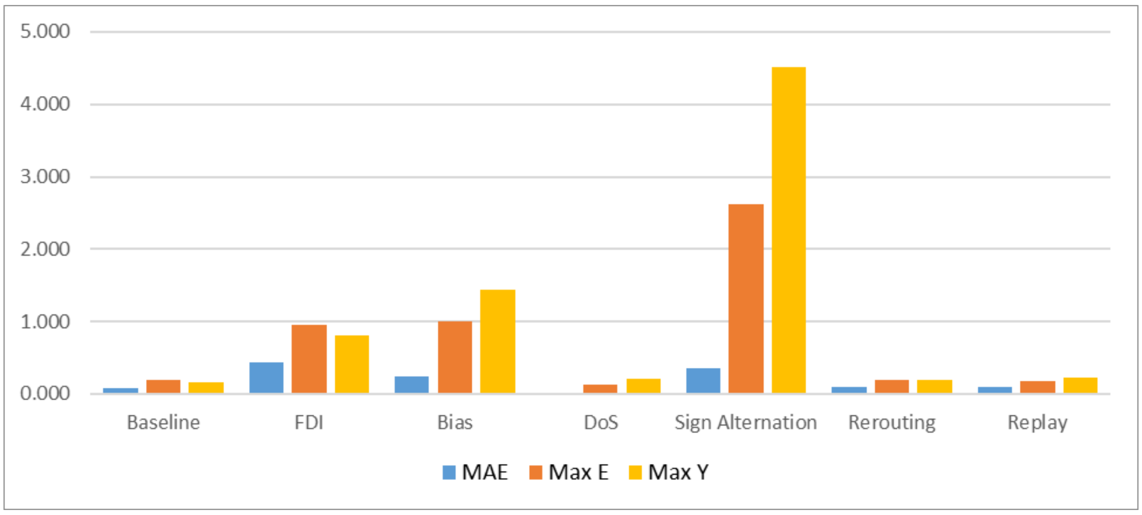


Figure 6.2.2 – Steady State Output Attacks

6.3 Attacks During Initiation

The second set of tests seeks to identify the most disruptive attack before the system has reached its steady state, representing times it may be in an unstable state. This evaluation is split into two halves, the inputs, and outputs. Each scenario is using the same parameters as the baseline but will have attacks active for one second between, 0.25 – 1.25. For input attacks target signals include: U1 – U4. For outputs attacks target signals include: Y1 – Y3.

Input Attacks – Shown in *Table & Figure 6.3.1*, is the results for the input attacks during Initiation. Using the mean of MAE, MaxAE and MaxAY, the most disruptive attack on the inputs can be ranked (most to least):

1. Sign Alternation
2. Rerouting
3. Replay
4. DoS
5. Bias
6. FDI

The most disruptive attacks are the sign alternation and rerouting attacks both with larger impacts than the rest. Comparing against the tests while in steady state, from the previous sub-section, the impact is much larger for all attacks, with four making the system enter a failure state.

Table 6.3.1 - Initiation Input Attacks

Attack Type	**MAE**	**MaxAE**	**MaxAY**	**Mean**	**ToF (s)**	**AFT(s)**
Baseline	0.094	0.249	4.299	1.548	-	-
FDI (S=1)	0.578	0.824	4.299	1.900	-	-
Bias (Z=1)	1.089	1.357	4.299	2.248	-	0.39
DoS	2.547	4.236	10.613	5.799	1.2	0.32
Sign Alternation	49.755	372.432	634.914	352.367	0.48	0.29
Rerouting	56.181	286.327	521.964	288.158	0.53	0.31
Replay	4.153	12.433	24.631	13.739	1.07	0.32

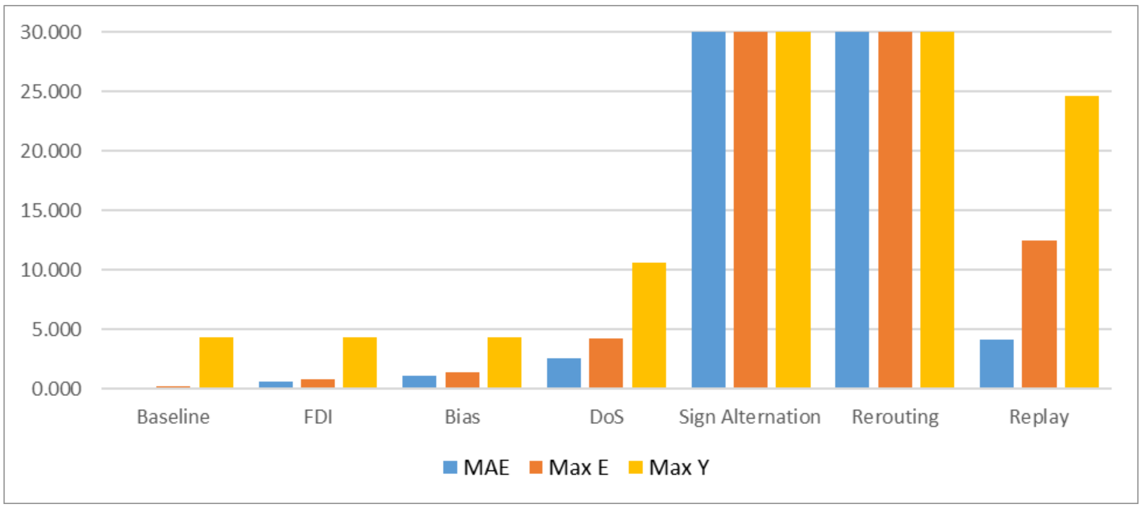


Figure 6.3.1 - Initiation Input Attacks (Sign alternation and rerouting go beyond bounds of plot)

Output Attacks – Shown in *Table & Figure 6.3.2*, is the results for the output attacks during Initiation. Using the mean of MAE, MaxAE and MaxAY, the most disruptive attack on the outputs can be ranked (most to least):

1. Sign Alternation
2. Rerouting
3. Replay
4. DoS
5. FDI
6. Bias

This produces similar rankings when compared against input attacks with just FDI and Bias swapping. The most disruptive attack again is sign alternation having a much larger impact than the rest. Comparing against the tests while in steady state again the impact is much larger for all attacks, and this set has three scenarios where the system enters a failure state.

Table 6.3.2 - Initiation Output Attacks

Attack Type	**MAE**	**MaxAE**	**MaxAY**	**Mean**	**ToF (s)**	**AFT(s)**
Baseline	0.094	0.249	4.299	1.548	-	0.27
FDI (S=1)	0.430	1.096	4.299	1.942	-	0.27
Bias (Z=1)	0.250	1.087	4.299	1.879	-	0.27
DoS	0.267	3.964	6.558	3.597	-	0.27
Sign Alternation	19.697	124.975	212.917	119.196	0.64	0.27
Rerouting	8.267	32.899	48.725	29.964	0.88	0.27
Replay	1.307	19.979	12.502	11.263	1.16	0.27

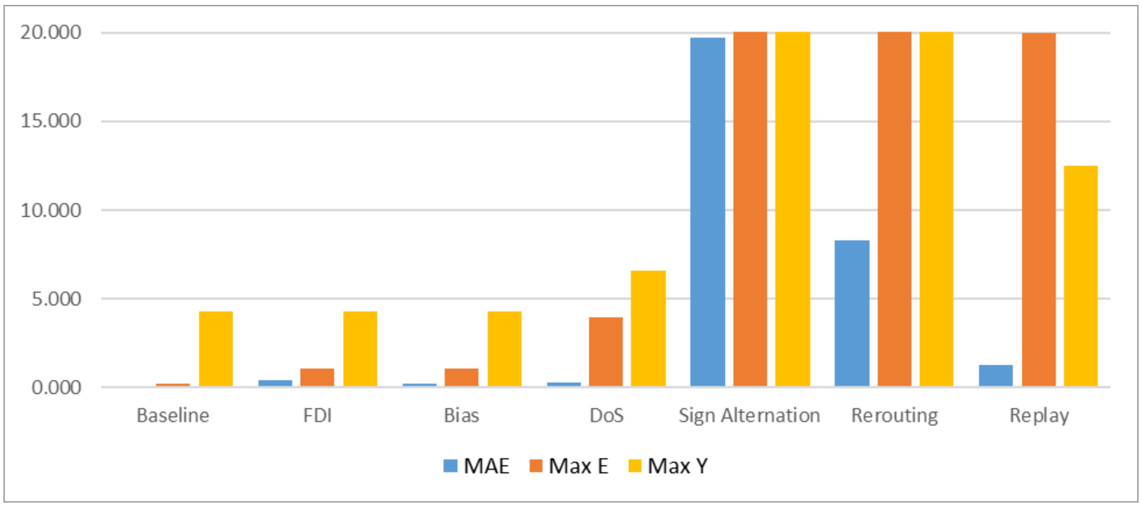


Figure 6.3.2 - Initiation Output Attacks (Sign alternation, rerouting and replay go beyond bounds of plot)

6.4 Single Signal Attacks

The third set of tests seeks to identify the most vulnerable signal to attack. The attack will target each signal once the system is in a stable state to enable fair evaluation for the individual impact of each signal. Each scenario is using the same parameters as the baseline but will have a single attack (bias attack Z=1) active for one second between, 2 – 3.

Shown in *Table and Figure 6.4.1* is the results for the single signal attacks. Using the mean of MAE, MaxAE and MaxAY, the most vulnerable signals can be ranked (most to least):

1. U1
2. U4
3. U3
4. Y2
5. Y1
6. Y3
7. U2

An observation is that the inputs showed a much larger variance for their effect on the output and error of the system compared to targeting the system outputs. Overall, the inputs also showed a larger disruption, with the exception of U2, compared to the outputs.

Table 6.4.1 Single Signal Attacks

Target Signal	**MAE**	**MaxAE**	**MaxAY**	**Mean**	**TTF (s)**	**AFT(s)**
Baseline	0.081	0.195	0.167	0.148	-	-
U1	1.320	1.811	2.973	2.035	-	2.2
U2	0.383	0.578	0.861	0.607	-	-
U3	0.484	0.740	1.710	0.978	-	-
U4	1.023	1.300	2.414	1.579	-	2.13
Y1	0.240	0.997	1.415	0.884	-	3.03
Y2	0.191	0.987	1.421	0.866	-	-
Y3	0.132	0.956	1.244	0.777	-	-

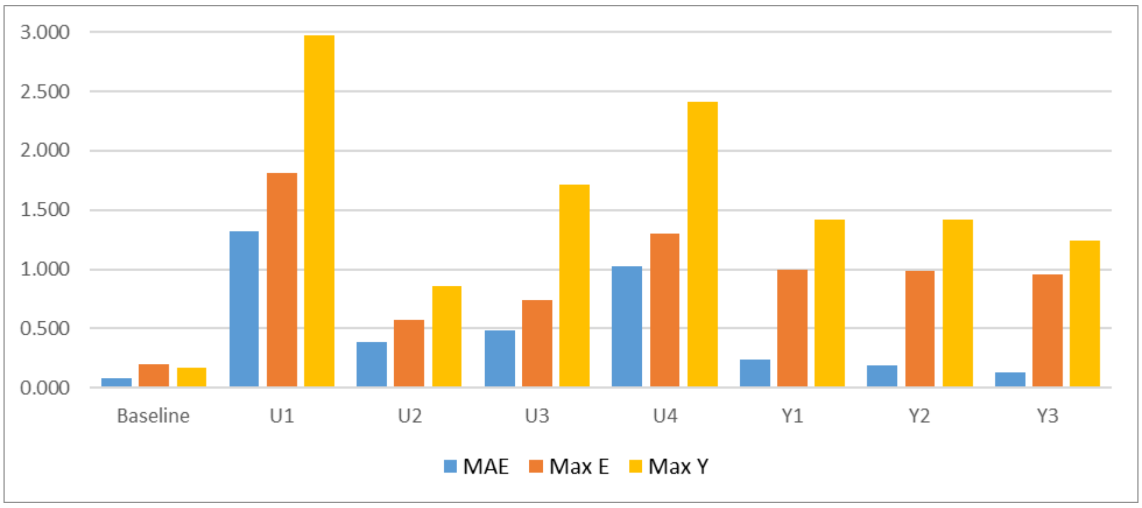


Figure 6.4.1 Single Signal Attacks

6.5 Combination Attacks

The fourth set of tests seeks to access the viability of different attack combinations with the focus on seeing what attack combinations can cause the system to fail before being caught by the anomaly detector. The scenarios will target the system in steady state between, 2 – 3. Each scenario is using the same parameters as the baseline but will have attacks (bias attack Z=1) on all input signals with the different combinations being applied on the output signals.

Shown in *Table and Figure 6.5.1* are the test results. There were three instances where the attacks pushed the system into a failure state, these included DoS, sign alternation and Replay. However, out of these three only DoS and Replay managed to cause the system to fail before being caught by the anomaly detector. This makes them especially dangerous combinations as the system would not be able to react before it is too late.

Table 6.5.1 Combination Attacks

Attack Type	**MAE**	**MaxAE**	**MaxAY**	**TTF (s)**	**AFT(s)**
No combo	1.072	1.384	2.506	-	2.12
FDI (S=1)	1.193	1.690	1.847	-	2.06
Bias (Z=1)	1.018	1.226	1.248	-	2.1
DoS	**0.009**	**0.129**	**31.832**	**2.45**	**3.03**
Sign Alternation	724.099	6159.808	10515.326	2.25	2.1
Rerouting	1.791	4.274	6.619	-	2.13
Replay	**0.088**	**0.180**	**31.949**	**2.45**	**3.03**

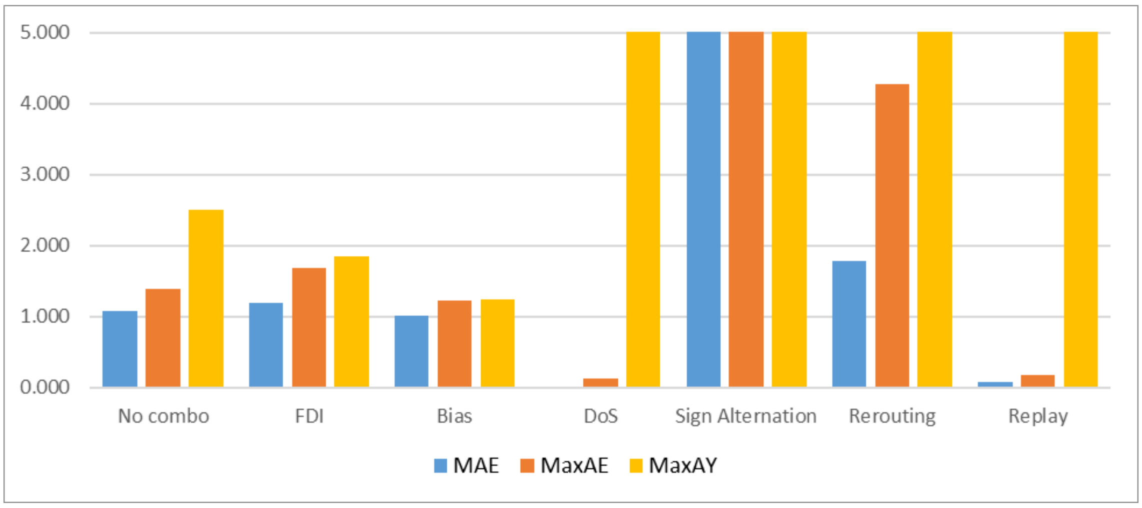
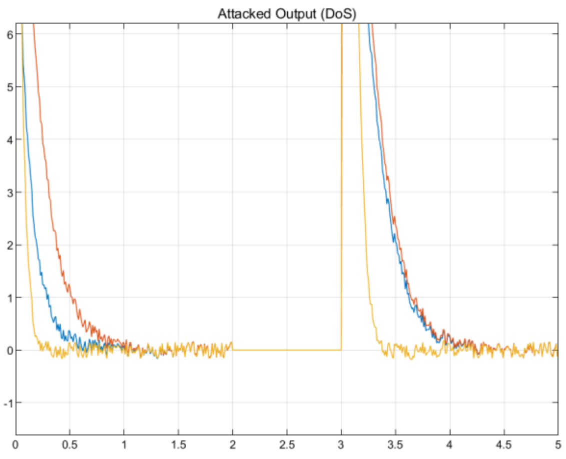


Figure 6.5.1 Combination Attacks (DoS, Sign alternation, rerouting and replay go beyond bounds of plot)

Below in *Figures 6.5.2 and 6.5.3* is the attacked output signals for the DoS and Replay attacks. These are the signals that the estimator is able to see. The mentioned attacks mask any form of disruption on the output which explains why it is unable to detect any form of attack on the inputs. *Appendix section 1* contains additional graphs for these two attack scenarios in *Figures A.1.1 – A.1.4* providing additional context to the attack. The actual output shows the extent of the disruption caused by the undetected bias attack and the error graphs show the sudden spike when the masking of the system output is disabled causing the anomaly detector to raise the alarm at 3.03.



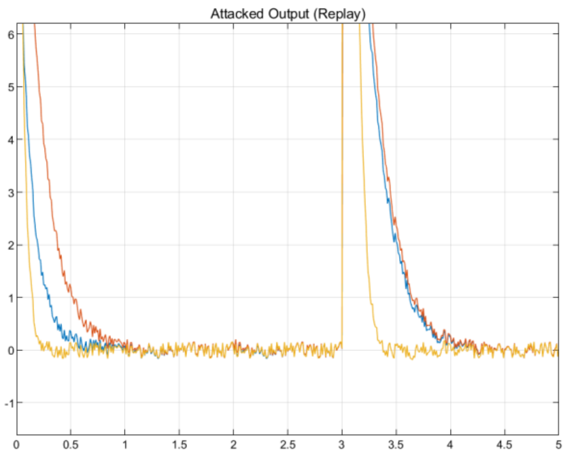


Figure 6.5.2 – DoS Attacked Output

Figure 6.5.3 – Replay Attacked Output

6.6 Stealthy/State Dependant Attacks

The fifth set of tests seeks to determine the viability of the proposed implementation of state dependant attacks. Similar to the combination attacks in the previous section this will assess the ability to disrupt the system without raising the alarm. The scenarios will target the system in steady state between, 2 – 3 and include results with and without the stealthy constraints. Within the results another metric will be added which measures the number of attacks (NoA) sent to the system within the attack window, the maximum number of attacks possible is 100 (this value is printed to the Simulink model's diagnostic viewer). The current implementation of the system only works for input attacks so in each scenario the attacker is targeting all inputs signals: U1 – U4. Attacks with the stealthy constrains active will have the word stealthy included in their name. The last change to the test scenarios is that the anomaly detector will be changed to raise the flag on a single error above the threshold, detector parameters are all equal to one.

Shown in *Table and Figure 6.6.1* are the results of the test scenarios, which can be split into four pairs of attacks with or without the stealthy constrains active. The most notable overall trend is that using the stealthy attacks decreases the disruption towards the system. This is expected as a lower disruption is required to evade detection.

Table 6.6.1 Stealthy/State Dependant Attacks

Attack Type	**MAE**	**MaxAE**	**MaxAY**	**ToF (s)**	**AFT(s)**	**NoA**
FDI (S=1)	0.554	0.853	1.379	-	-	100
Stealthy FDI (S=1)	0.554	0.853	1.379	-	-	100
FDI (S=1.5)	0.829	1.133	1.954	-	2.49	100
Stealthy FDI (S=1.5)	0.710	0.994	1.700	-	-	86
Bias (Z=1)	1.072	1.384	2.506	-	2.12	100
Stealthy Bias (Z=1)	0.255	0.970	1.399	-	-	17
Sign Alternation	18.702	158.831	271.133	2.65	2.43	100
Stealthy Sign Alternation	0.324	1.111	1.811	-	2.82	73

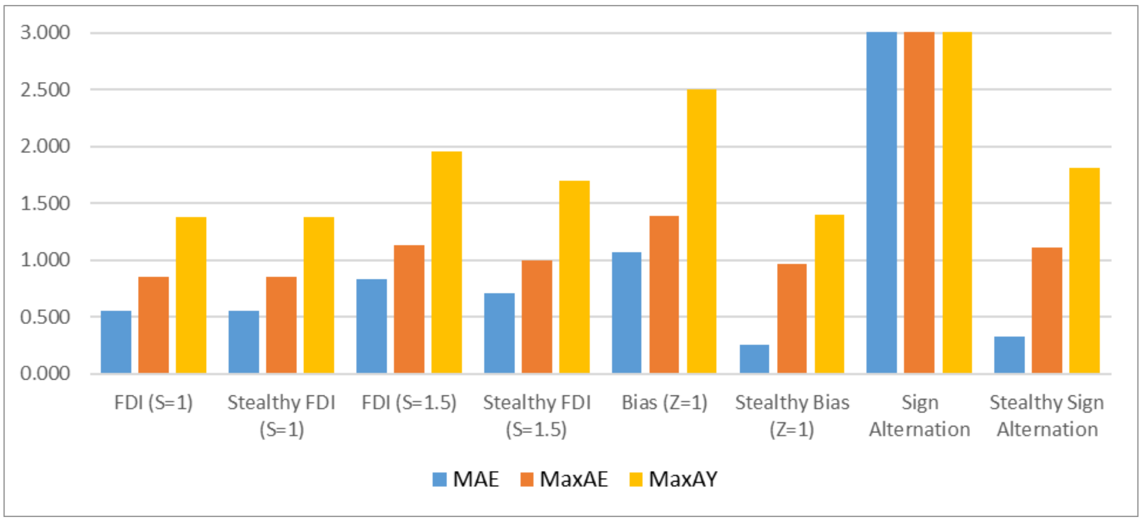
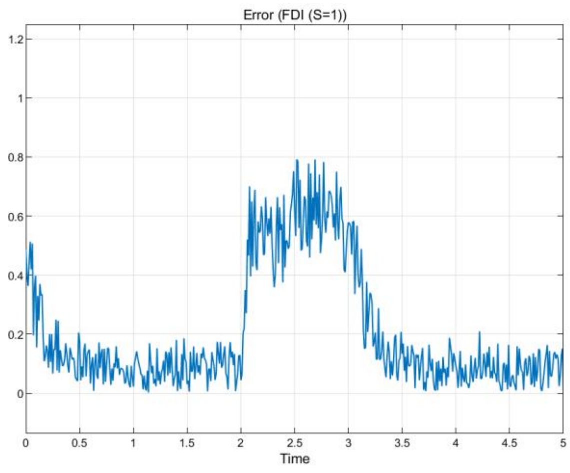


Figure 6.6.1 Stealthy/State Dependant Attacks

The first pair of results, FDI (S=1) represents a scenario where the attacks are all already undetected by the system. Activating the stealthy filter therefore allows all 100 attacks through, this is reflected by the same error values being obtained, as shown in *Figures 6.6.2 and 6.6.3*.



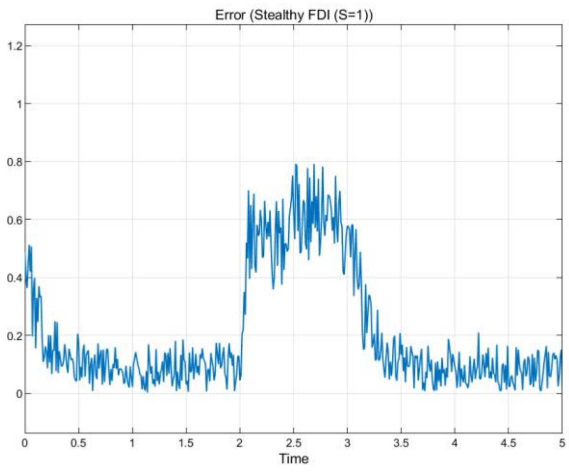


Figure 6.6.2 – FDI (S=1) Error *Figure 6.6.3 – Stealthy FDI (S=1) Error*

The next two pairs of results, FDI (S=1.5) and Bias (Z=1) both represent scenarios where the system is working as intended limiting the number of attacks in such a way that the alarm is never raised. FDI (S=1.5) represents a scenario where majority of the attacks are still able to be performed, *Figures 6.6.4 and 6.6.5* show the similar error graphs with the stealthy system stopping each time before the error exceeded the threshold of one. Bias (Z=1) represents a scenario where only minority of the attacks can be sent. This results in two quite different error value graphs as shown in *Figures 6.6.6 and 6.6.7*.

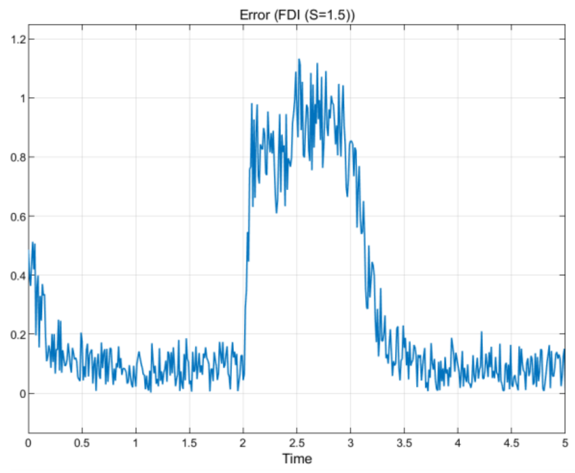


Figure 6.6.4 – FDI (S=1.5) Error

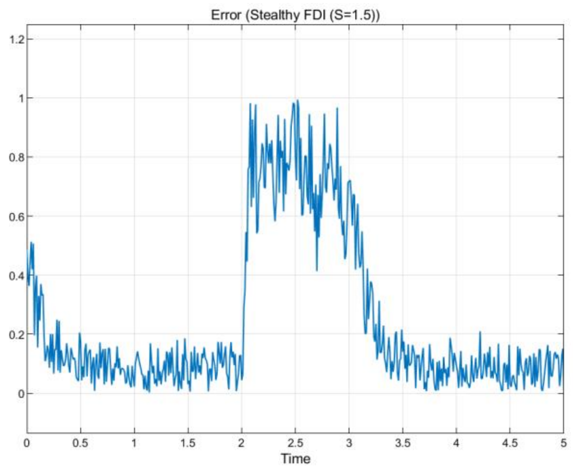


Figure 6.6.5 – Stealthy FDI (S=1.5) Error

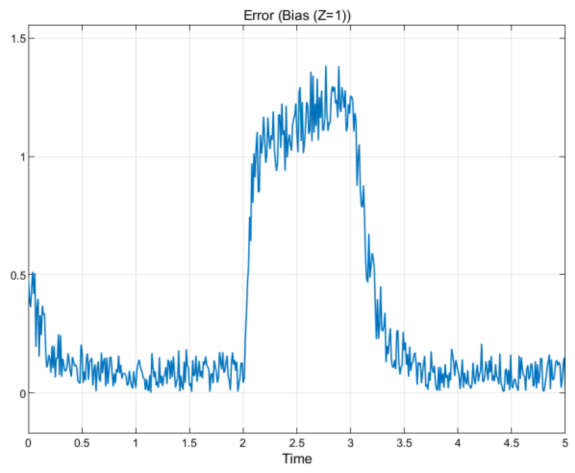


Figure 6.6.6 – Bias (Z=1) Error

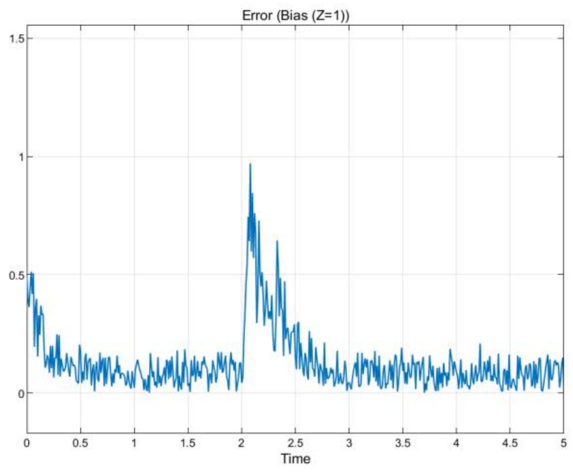


Figure 6.6.7 – Stealthy Bias (Z=1) Error

Lastly, the sign alternation attack, represents a scenario in which the system fails to remain stealthy. *Figure 6.6.8* shows the single iteration where the systems error goes above the threshold setting off the alarm. The cause for this will be discussed further within *Section 7.4*.

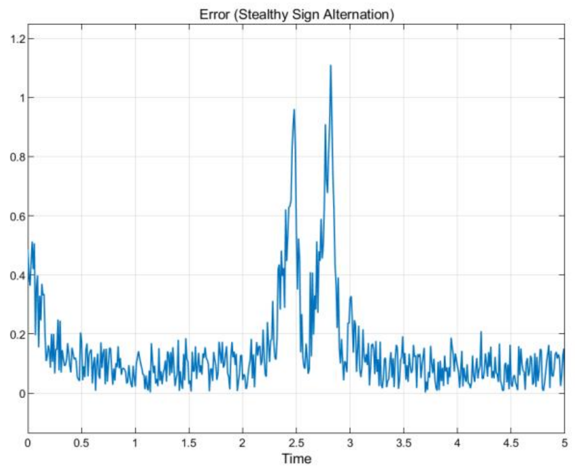


Figure 6.6.8 Stealthy Sign Alternation Error

Section 7 – Discussion and Future Work

This section details the discussion and outcomes from the tests conducted in the previous section and includes suggestions for future work. It will discuss the most disruptive attack types and their relationship with the stability of the system, then the most vulnerable signals. Next it will assess the viability of attack combinations and state dependant attacks with relation to remaining undetected. Lastly, general points about future additions to the software tool.

7.1 Most Disruptive Attack Type

From evaluating the results and observations within *sections 6.2 and 6.3.* The most disruptive attack is the sign alternation attack. The severity of the attack type may stem from the negative feedback loop created by its ability to invert all feedback that pushes the system towards equilibrium to do the opposite. This results in the system trying to send larger corrective signals which instead make the situation worse. In this given simulation where the inputs are unbounded this makes for an extremely devastating attack which was often magnitudes higher than the others. However, in the future if the tool allowed the user to bound the inputs to reflect a real system representing hard-coded limits or mechanical fail safes which remain in place even when the network is under attack. Tests could then be conducted to develop an understanding of the relationships between attacks and different sets of parameter bounds.

Evaluation of the four sets of disruption rankings in *sections 6.2 and 6.3* yields two subgroups of attack can be formed, with the exception of sign alternation. The grouping factor is if the attack uses the target signal as part of the attack signal, this includes DoS, rerouting, replay. This group has next to no effect during steady state but large effect during unstable periods. This means their impact has a connection to the system stability. The opposite can be said for the injection attacks of FDI and Bias, their impact was consistent between the two sets as they are injecting their own signals irrelevant to the state of the system. Future analyse could be done to see if this statement holds true by repeating the tests using systems with differing stabilities. This could be done by using the software with different noise/uncertainties parameters. Evaluating attack impacts for each attack type across a range of different system stabilities.

7.2 Most Vulnerable Signal

From the results and observations in *Section 6.4,* The most vulnerable signal is U1, this corresponds to the flowrate of pump 2. Another notable observation is the input signals posed a large threat to the system when compared against the output, with the exception of U2. This is due to the system's ability to feed output attacks through the estimator and controller before they enter the plant, limiting the extent of their impact. This analysis was completed using Bias attacks, it might be of value in future work to repeat it using other attack types.

7.3 Viability of Combination Attacks

From the results and observations in *Section 6.5,* DoS and Replay can both be used to create situations where the system is pushed to a point of failure before the detector was alerted. However, all other attacks had no special interactions. These results are likely why the other approaches to combination attacks as mentioned within *Section 1.3* stick to these two attack combinations. The tool also has the capability to explore more complicated combination attacks with each individual signal capable of being targeted by a unique attack type. Future work could be done to exhaustively evaluate each individual signal combination. However, this would mean recording over a hundred sets of test data. Although possible by hand this would benefit from automation, making scripts to assist testing would therefore be a great feature to implement in the future.

7.4 Viability of State Dependant Attacks

Section 6.6 demonstrates four pairs of attack scenarios using the state-dependant approach. The FDI and Bias scenarios show promising results for the implementation. However, when evaluating at the sign alternation attack the error exceeds the threshold for a single iteration. Diving deeper with additional testing and use of the raw numerical data, the software implementation is correctly implementing the theory. On the iteration of the alarm no attack signal was sent as it would violate the stealthy constraint. The alarm comes from an attack from a previous iteration that was so disruptive it pushed the system out of the safe set for more than one iteration. This is an interesting finding as it poses a problem for an attacker that only has knowledge of their attack's impact on the current iteration. Future work on this could explore an implementation where an attacker can check if the attack satisfies current and future stealthy constraints and only then will they attack, knowing it will always be safe.

7.5 Smarter Stealthy Attacks

With the more complicated design of the anomaly detector described in *Section 3.3*. This theoretically enables a more complicated form of stealthy attack that deliberately violates the threshold when permitted by the alarm settings. For example, the detector allows a total of three exceeds of the threshold within a period, the attack would deliberately exceed twice. Alternatively, the attacker could target a system that only raises the alarm if there are two consecutive threshold violations, meaning they could deliberately violate the error threshold every other iteration. These approaches theoretically allow the attacker to cause additional disruption without ever setting the alarm off. Within other literature this approach has been referred to as ‘hidden attacks’ [30]. A potential route for future work could involve creating an implementation of these smarter stealthy attacks within the software tool.

7.6 Additions To The System

Lastly, this point seeks to discuss some of the probable future work to expand the functionality of the software tool. Currently the implementation is hard coded to work with a four input, three state/output system. This can allow a user to create a custom script to initialise any system that also has this 4/3 split. However, it could be a good addition to allow the tool to dynamically adapt to the number of inputs and outputs, making the atter automatically apply attacks to all signals without specifics being supplied by the tool user. Other than just making the tool scalable, there are several core pieces of the system that could be implemented using different methods. This may include changing the estimator to use something like a Kalman filters or PID control. Additionally, the anomaly detector could make use of different detectors such as the Euclidean norm. All of these additions would make the tool more flexible in its application for risk assessment.

Section 8 – Conclusion

This project set out to further research in analysing the impact of cyber-attacks in cyber-physical systems, through development of a software tool. This report has detailed the control theory which describes the system and attack modelling. The implementation of this theory within a software tool and demonstrated how the tool can be used to investigate the impact of various attack scenarios. Lastly, discussing the research completed using the tool and future work that could be completed. In particular a focus was placed on assessing the viability of a novel form of stealthy attacks called state-dependant attacks.

Section 2 focused on a reflection of the research process and evaluating how the plan from the interim report was executed in the following months. Similarly, the last sub-section of the report seeks to provide reflection on the project and the achievement of the objectives laid out within the specification.

8.1 Assessment Against Objectives

This section will focus on the reflection on achieving the four project objectives and will follow a rough timeline of events of how the project progressed. More context on the shift in project direction, as mentioned in *Section 2.3,* will be provided here and how it affected the project.

1. Familiarize with attack strategies for dynamical systems, and modelling of cyber-physical systems.

This knowledge is the backbone of the project and majority of the time in the first semester and was achieved through literary review, see *section 1.* Learning the underpinning control theory, see *section 3,* was done using a 'learn through doing' approach which involved applying the learnt knowledge using MATLAB scripts and Simulink models. This meant development of the first versions of the software were as early as a month after starting the project. Once a benchmark system was chosen for the project a simple system including just the plant and estimator was completed by the end of November.

2. Adapt safety analysis algorithms for dynamical systems so they can incorporate attack models.

Once the core system was created the next step was understanding not only the attack modelling, see *section 4*, but also the security feature of the anomaly detector. These were then added to the software model. With five attacks being implemented by the end of December, an early version of the Simulink model submitted within the interim report can be seen in *Appendix A.2*. Following the interim report in January, the final attack was added as well as increasing the complexity of the anomaly detector. Within February, combination and stealthy attacks were then implemented.

It was during February when the project took a change in direction from the spec to now focus on the software as a customisable tool and test bed rather than just focusing on purely analysis of the benchmark problem. One task that stemmed from that change in direction was changing the software to have a wide range of parameters for test set customisation. The largest change was the development of a sophisticated GUI with the intention that others could easily interact with the tool as a platform to perform their own analysis, see *Section 5*.

3. Analyse impact of attacks on a specific benchmark problem, to be agreed.
4. Define proper attack impact metrics for the developed algorithms.

These last two points can be combined as they are both completed simultaneously. Once the tool was finished in March, the last and most important parts of the project can be completed. *Section 6* provides evidence of the completion of this analysis, which involved over 40 different test scenarios, only counting those recorded within this report. The discussion in *Section 7*, highlights many of the outcomes provided from the testing and provides lots of suggestions for work which could help further research.

References

[1] Henrik Sandberg, Vijay Gupta, Karl H. Johansson, “Secure Networked Control Systems,” *Annual Review of Control, Robotics, and Autonomous Systems,* vol. 5, no. 1, pp. 445-464, 2022.

[2] David Kushner, “The real story of stuxnet,” *IEEE Spectrum,* vol. 50, no. 3, pp. 48-53, 2013.

[3] Dr. Charlie Miller, Chris Valasek, “Remote Exploitation of an Unaltered Passenger Vehicle,” Paper presented at Black Hat USA, Las Vegas, NV, Aug. 1–6, Extended report available at https://illmatics.com/Remote%20Car%20Hacking.pdf, 2015.

[4] Andy Greenberg, “Hackers Remotely Kill a Jeep on the Highway—With Me in It,” Wired, 21 July 2015. [Online]. Available: https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/. [Accessed 29 March 2023].

[5] Kim Zetter, “Inside the Cunning, Unprecedented Hack of Ukraine's Power Grid,” Wired, 3 March 2016. [Online]. Available: https://www.wired.com/2016/03/inside-cunning-unprecedented-hack-ukraines-power-grid/. [Accessed 29 March 2023].

[6] “ICS alert (IR-ALERT-H-16-056-01): cyber-attack against Ukrainian critical infrastructure,” Cybersecurity & Infrastructure Security Agency, 20 July 2016. [Online]. Available: https://www.cisa.gov/news-events/ics-alerts/ir-alert-h-16-056-01. [Accessed 29 March 2023].

[7] Alvaro A. Cárdenas, Saurabh Amin, Zong-Syun Lin, Yu-Lun Huang, Chi-Yen Huang, Shankar Sastry, “Attacks against Process Control Systems: Risk Assessment, Detection, and Response,” in *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security,* New York, NY, USA, 2011.

[8] Y. Mo, E. Garone, A. Casavola, B. Sinopoli, “False data injection attacks against state estimation in wireless sensor networks,” in *49th IEEE Conference on Decision and Control (CDC),* Atlanta, GA, USA, 2010.

[9] Jezdimir Miloševič, Takashi Tanaka, Henrik Sandberg, Karl Henrik Johansson, “Analysis and Mitigation of Bias Injection Attacks Against a Kalman Filter,” *IFAC-PapersOnLine,* vol. 50, no. 1, pp. 8393-8398, 2017.

[10] A. A. C.´. S. S. S. Amin, “Safe and Secure Networked Control Systems Under Denial-of-Service Attacks,” in *Hybrid Systems: Computation and Control*, Springer Berlin / Heidelberg, Lecture Notes in Computer Science, 2009, pp. 31-45.

[11] A. Cetinkaya, H. Ishii, T. Hayakawa, “An Overview on Denial-of-Service Attacks in Control Systems: Attack Models and Security Analyses,” *Entropy,* vol. 21, no. 2, p. 210, 2019.

[12] J. Milošević, H. Sandberg, K. H. Johansson, “Estimating the Impact of Cyber-Attack Strategies for Stochastic Networked Control Systems,” *IEEE Transactions on Control of Network Systems,* vol. 7, no. 2, pp. 747-757, 2020.

[13] Riccardo M.G. Ferrari, André M.H. Teixeira, “Detection and isolation of routing attacks through sensor watermarking,” in *American Control Conference (ACC),* Seattle, WA, USA, 2017.

[14] Y. Mo, B. Sinopoli, “Secure control against replay attacks,” in *47th Annual Allerton Conference on Communication, Control, and Computing*, Monticello, IL, USA, 2009.

[15] Rohan Chabukswar, Yilin Mo, Bruno Sinopoli, “Detecting Integrity Attacks on SCADA Systems,” *IFAC Proceedings Volumes,* vol. 44, no. 1, pp. 11239-11244, 2011.

[16] Nicola Forti, Giorgio Battistelli, Luigi Chisci, Suqi Li, Bailu Wang, Bruno Sinopoli, “Distributed Joint Attack Detection and Secure State Estimation,” *IEEE Transactions on Signal and Information Processing over Networks,* vol. 4, no. 1, pp. 96-110, 2018.

[17] G. B. L. C. S. L. B. W. B. S. N. Forti, “Distributed Joint Attack Detection and Secure State Estimation,” *IEEE Transactions on Signal and Information Processing over Networks,* vol. 4, no. 1, pp. 96-110, 2018.

[18] J. P. H. B. S. Y. Mo, “Resilient Detection in the Presence of Integrity Attacks,” *IEEE Transactions on Signal Processing,* vol. 62, no. 1, pp. 31-43, 2014.

[19] V. G. F. P. C. -Z. Bai, “On Kalman Filtering with Compromised Sensors: Attack Stealthiness and Performance Bounds,” *IEEE Transactions on Automatic Control,* vol. 62, no. 12, pp. 6641-6648, 2017.

[20] Yilin Mo, Bruno Sinopoli, “On the Performance Degradation of Cyber-Physical Systems under Stealthy Integrity Attacks,” *IEEE Transactions on Automatic Control,* vol. 61, no. 9, pp. 2618-2624, 2016.

[21] Carlos Murguia, Iman Shames, Justin Ruths, Dragan Nešić, “Security metrics and synthesis of secure control systems,” *Automatica,* vol. 115, p. 108757, 2020.

[22] B. S. Y. Mo, “Secure Estimation in the Presence of Integrity Attacks,” *IEEE Transactions on Automatic Control,* vol. 60, no. 4, pp. 1145-1151, 2015.

[23] S. D. L. S. E. Kung, “The performance and limitations of Stealthy Attacks on Higher Order Systems,” *IEEE TRANSACTIONS ON AUTOMATIC CONTROL,* vol. 62, no. 2, pp. 941-947, 2017.

[24] A. M. H. T. M. C. P. P. K. Pan, “Cyber Risk Analysis of Combined Data Attacks Against Power System State Estimation,” *IEEE Transactions on Smart Grid,* vol. 10, no. 3, pp. 3044-3056, 2019.

[25] V. G. F. P. R. Anguluri, “Periodic coordinated attacks against cyber-physical systems: Detectability and performance bounds,” in *IEEE 55th Conference on Decision and Control (CDC),* Las Vegas, 2016.

[26] Arllem O. Farias, Gabriel Alisson C. Queiroz, Iury V. Bessa; Renan Landau P. Medeiros, Lucas C. Cordeiro, Reinaldo M. Palhares, “Sim3Tanks: A Benchmark Model Simulator for Process Control and Monitoring,” *IEEE Access,* vol. 6, pp. 62234-62254, 2018.

[27] Verica Radisavljevic-Gajic, “Linear observers design and implementation,” in *American Society for Engineering Education*, Bridgeport, CT, USA, 2014.

[28] M. Blanke, M. Kinnaert, J. Lunze, M. Staroswiecki, J. Schroder, Diagnosis and Fault-Tolerant Control, New York, NY, USA: Springer, 2006.

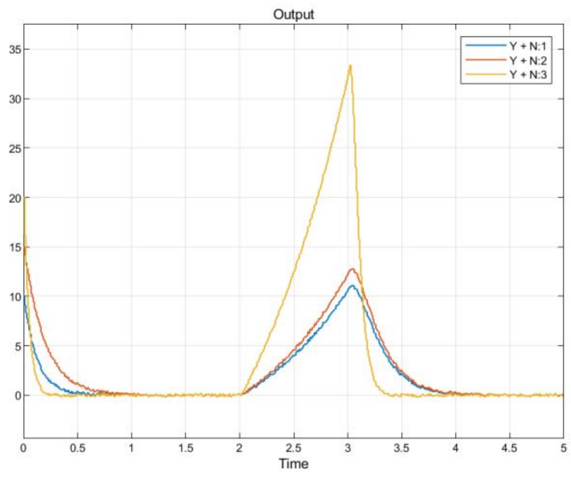
[29] Nabil H. Hirzallah, Petros G. Voulgaris, “On the Computation of Worst Attacks: A LP Framework,” *2018 Annual American Control Conference (ACC),* pp. 4527-4532, 2018.

[30] N. v. d. W. J. R. Carlos Murguia, “Reachable Sets of Hidden CPS Sensor Attacks: Analysis and Synthesis Tools,” *IFAC-PapersOnLine,* vol. 50, no. 1, pp. 2088-2094, 2017.

[31] Yilin Mo, Bruno Sinopoli, “Secure Estimation in the Presence of Integrity Attacks,” *IEEE Transactions on Automatic Control*, vol. 60, no. 4, pp. 1145-1151, 2015.

Appendix

A.1 Extra Combination Attack Graphs



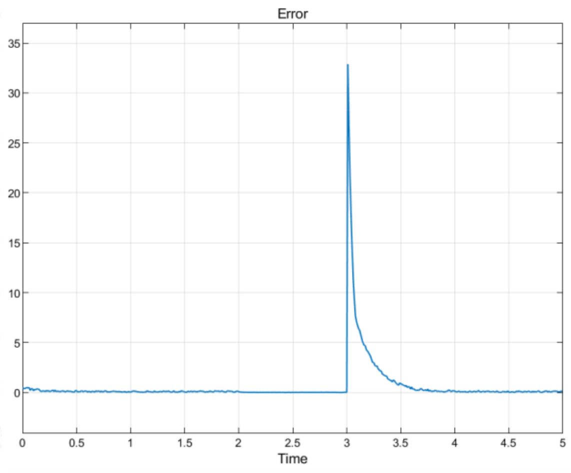
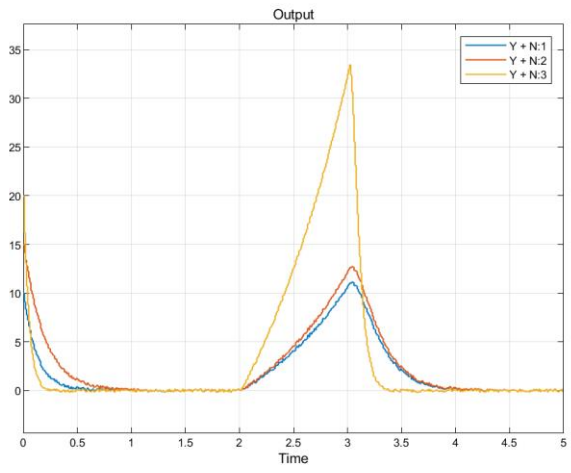


Figure A.1.1 – DoS Output

Figure A.1.2 – DoS Error



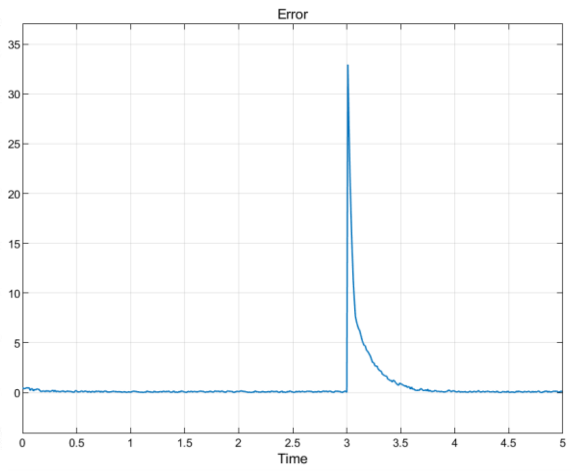


Figure A.1.3 – Replay Output

Figure A.1.4 – Replay Error

A.2 Interim report software design

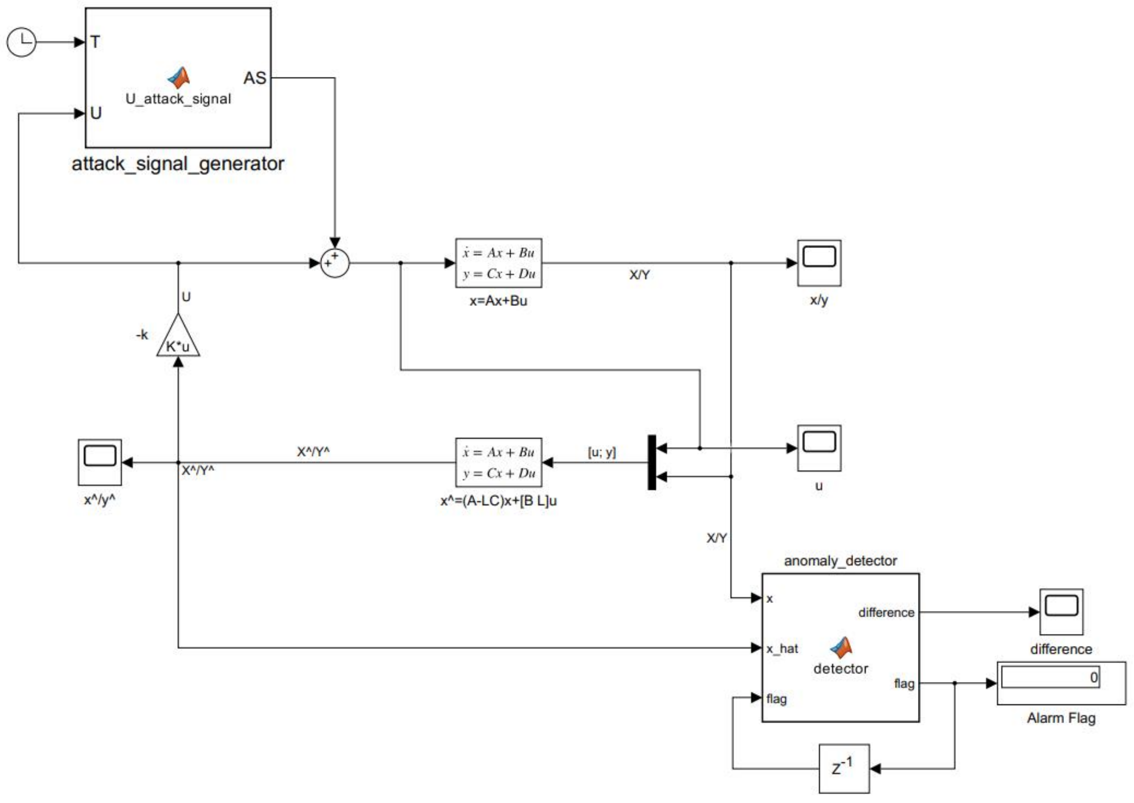


Figure A.2.1 Simulink Model from interim report.