Preemptive Detection of Cyber Attacks in Industrial Control Systems

Omkar Anand Harshe

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Electrical Engineering

April 29, 2015

Blacksburg, Virginia

Keywords: Industrial control systems, malware resilience, real-time prediction,
classifier-based intrusion detection

Preemptive Detection of Cyber Attacks in Industrial Control Systems

Omkar Anand Harshe

ABSTRACT

Industrial Control Systems (ICSes), networked through conventional IT infrastructures, are vulnerable to attacks originating from network channels. Perimeter security techniques such as access control and firewalls have had limited success in mitigating such attacks due to the frequent updates required by standard computing platforms, third-party hardware and embedded process controllers. The high level of human-machine interaction also aids in circumventing perimeter defenses, making an ICS susceptible to attacks such as reprogramming of embedded controllers. The Stuxnet and Aurora attacks have demonstrated the vulnerabilities of ICS security and proved that these systems can be stealthily compromised.

We present several run-time methods for preemptive intrusion detection in industrial control systems to enhance ICS security against reconfiguration and network attacks. A run-time prediction using a linear model of the physical plant and a neural-network based classifier trigger mechanism are proposed for preemptive detection of an attack. A standalone, safety preserving, optimal backup controller is implemented to ensure plant safety in case of an attack. The intrusion detection mechanism and the backup controller are instantiated in configurable hardware, making them invisible to operating software and ensuring their integrity in the presence of malicious software. Hardware implementation of our approach on an inverted pendulum system illustrates the performance of both techniques in the presence of reconfiguration and network attacks.

Acknowledgments

I would like to thank my adviser, Dr. William Baumann, for his valuable guidance on academics and research without which I could not have accomplished this study. Thanks to Dr. Cameron Patterson for providing me an opportunity to be a part of this project. Thanks to Dr. Steve Southward for being part of my advisory committee. I would also like to thank my colleagues, Teja Chiluvuri, Christopher J. McCarty and Pallavi Deshmukh for helping me in making this study a success.

Thanks to my parents and other family members for their support and guidance. A special thanks to my fiance Sayali, for her constant support and encouragement. Thanks to my friends, Pratik Joshi, Jaydeep Deshpande, Abhijit Khare and Surabhi Ramdasi who made Blacksburg a place I will always remember fondly.

This material is based upon work supported by the National Science Foundation under Grant Number CNS-122256. Any opinions, findings, conclusions and recommendations expressed in this material are those of the author and do not necessarily reflect the views of National Science Foundation.

The donation of a Zedboard and design tools by Xilinx, Inc is gratefully acknowledged.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

In the last few years, Industrial Control Systems (ICSes) have passed through a significant transformation from proprietary, isolated systems to open architectures and standard technologies highly interconnected with other corporate networks and the Internet. Current ICSes utilize embedded platforms and commercial off the shelf software. This has helped in reducing costs and has enabled control and monitoring of processes from remote locations. However, ICS communication protocols were not designed with security in mind. Many of these are serial protocols which lack security measures such as encryption and built-in authentication. Nowadays, many of these protocols are either connected with the TCP/ IP protocol suite or replaced by standard open protocols, making them vulnerable to attacks executed from remote locations using the internet.

There is an important distinction between protecting personal information and safeguarding physical infrastructure, and lumping them together as *cyber security* implies a common solution [1]. Perimeter defenses such as firewalls and access control can safeguard information in personal devices, however they are not sufficient for protecting process controllers controlling critical infrastructures. Thus, we address the control system protection as a fundamentally different security problem, requiring a distinct method of protection, rather than trying to apply techniques designed for personal devices.

1.2 Literature Survey

The term *industrial control system* refers to the electronic devices used to automate the physical processes (plants) in industries such as power, oil and gas, water and manufacturing. ICSes consist of sensors and actuators at the lowest level, which communicate through network protocols, such as DeviceNet and ControlNet, with a Programmable Logic Controller (PLC) and a Remote Terminal Unit (RTU). A supervisory controller monitors, collects, and analyzes the process data to reduce waste and improve efficiency. ICSes are networked through IT infrastructures to allow for convenient updating of software components and to enable a timely flow of data between sensors, actuators, and controllers.

While networking provides many advantages, it also makes ICS vulnerable to malicious attacks such as reconfiguration or network integrity attacks. A reconfiguration attack on one or more controllers could cause large-scale disruption with potential for irreparable harm to humans and physical systems. With the level of human-machine interaction within these systems, perimeter security techniques such as firmware, access control are not sufficient. The Stuxnet worm which attacked the PLCs controlling centrifuges in a nuclear plant in Iran demonstrated the vulnerabilities of such a practice [2]. Cárdenas et. al. discuss the vulnerabilities of ICSes and how control and security practitioners can work together to develop secured control systems to survive such attacks [3].

Cyber attacks on ICSes can be broadly classified as either integrity attacks on network channels or reconfiguration attacks on embedded controllers [4]. An integrity attack on a network channel modifies the sensor, actuator or command signal to a malicious value. A reconfiguration attack, on the other hand, is a malicious software update to the controller and has the potential to affect the firmware or control code. Such an attack can be accomplished through multiple avenues, such as insiders, third-party code, manufacturer's maintenance ports, and personal computing device updates.

1.2.1 Network Integrity Attacks

Teixeira et al. have characterized the attack space for attacks on network channels using the resources available to the intruder [5]. The resources can be of three types:

- *Disruption* resources enable an intruder to affect the process operation by either vi-

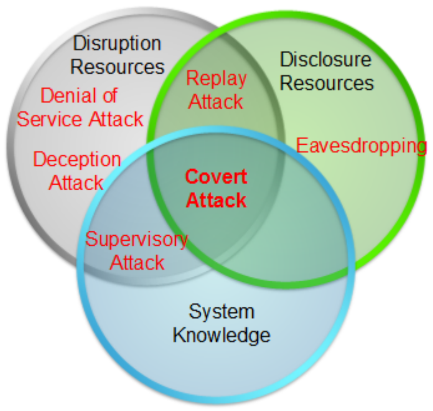

Figure 1.1: Networked control system attack space

olating the integrity of the data through a sensor or actuator signal modification or jamming the sensor or actuator resulting in data unavailability.

- *Disclosure* resources allow an adversary to find sensitive information about the process without affecting it. Analysis of the collected data can be used to model the process behavior. These resources are also helpful during the attack to ensure effective usage of disruption resources.
- *System knowledge* about the process behavior can be used by an intruder to increase the stealthiness of the attack.

We assume that the primary goal of the attacker is to disrupt the physical process while making it hard for the defensive mechanisms to detect it. With more than one resource available, the adversary has a better chance of concealing the attack. Defense mechanisms for detecting the network integrity attacks captured in Fig. 1.1 is a well-researched topic except for the covert attack scenarios.

- **Eavesdropping Attack**: In an eavesdropping attack the intruder has access to one or more communication channels in ICS. With minimal understanding of the process behavior, the intruder can gather intelligence about the process and violate data confidentiality. Bernabeu et. al. utilizes data mining and decision tree analysis in phase monitoring units for online state estimation [6]. Similar techniques for collecting data allow an adversary to model the overall behavior of the plant. Better understanding of the plant behavior allows an adversary to create stealthier subsequent attacks.

- **Deception Attack**: In a deception attack the intruder modifies one or more control signals and/or sensor measurements from their calculated or measured values to corrupt values. Deception attacks are easily detectable since, without system knowledge, it is difficult for the intruder to conceive an attack of long duration. These attacks can be concealed if the intruder has access to multiple channels and makes small changes to the process affecting performance without endangering the plant safety in the near future.

- **Denial of Service Attack**: A DoS attack prevents the actuator and sensor data from reaching their respective destinations and is typically modeled as the absence of data. This could mean unavailability of a sensor to provide the process state information or an actuator to influence the process in a specific way. Amin et. al. tackles DoS attacks by developing a safety constrained optimal control law with a Bernoulli packet drop model for DoS attacks on the measurement packet [7].

- **Replay Attack**: A replay attack is typically initiated on processes which operate at steady-state. In a replay attack, the adversary has both disruption and disclosure resources. The attacker initiates a disclosure or an eavesdropping attack to gather a sequence of data for a definite period of time. The attack is executed by replaying valid sensor or control data from the past instead of current measurements. The stealthiness of the attack varies depending upon the number of compromised sensor/actuator channels. The plant or supervisor responds to past data that does not reflect the current state of the physical process causing disruptive operation. Mo et. al. employs an LQG controller and an improved χ^2 detector to detect the attack at the cost of control performance [8].

- **Supervisory Attack**: In a supervisory attack the intruder has knowledge about the process and has disruptive resources to affect it. These attacks can be broadly classified into two types: zero dynamics attack and bias injection attack.

  - A *Zero Dynamics* attack may go undetected for a significant amount of time for processes with transmission zeros. Teixeira et. al. have used the effect of initial condition mismatch for observable systems to modify system structure in terms of outputs, inputs and states to detect these attacks [9].

  - A slowly converging *Bias Injection* attack drives a process away from steady state slowly, without causing an alarm. Yilin et. al. have used an ellipsoid approximation to define an inner and outer approximation of reachable sensor values. A point outside the reachable zone indicates the bias attack [10].

- **Covert Attack**: In a covert attack the intruder has all three resources required to affect the process. Typical attack scenarios would include enhancing the stealthiness of the attacks using knowledge of the process or creating new attack scenarios with combinations of attacks.

There are other viable techniques for intrusion detection that depend upon run-time monitoring and reporting of security threats making them reactive and necessitating a corrective action. However, all these techniques trust the controller and monitoring software to detect attacks through network channels. Thus, if these trusted components are compromised, run-time monitoring or any other intrusion detection methods discussed above, will fail to detect the threat.

1.2.2 Reconfiguration Attacks

Figure 1.2 illustrates a conventional ICS architecture. ICS architecture has 3 layers.

- The lowermost level in the hierarchy is the *Infrastructure layer* that consists of physical processes with sensors and actuators that communicate through network protocols such as DeviceNet and ControlNet, with the regulatory layers.
- The *Regulatory layer* includes PLCs and RTUs, governing the process behavior by gathering information from sensors and providing appropriate control action to the process by controlling the actuators. These controllers interact with the supervisory layer for operational set-pint commands and to report the current process status.
- The *Supervisory layer*, that consists of Supervisory Control and Data Acquisition System (SCADA) or a Distributed Control System (DCS), monitors, collects, and analyzes the plant data to reduce wastage and improve efficiency.

The defense mechanisms are typically implemented in embedded controllers in regulatory layer. These controllers require software updates for normal operation. A malignant software update is capable of tampering with the defense mechanisms, as well as with the control

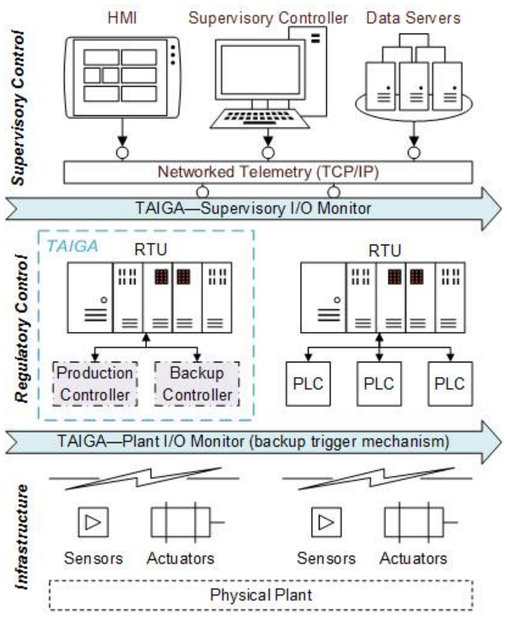

Figure 1.2: TAIGA integration in an ICS Architecture

strategy governing the process behavior, making the overall system insecure. A reconfiguration attack coupled with a network integrity attack could cause havoc in the absence of protective defense mechanisms.

1.3 Contributions

Considering all the attack scenarios, it is clear that to protect critical infrastructures from cyber attacks requires:

1. An intrusion detection scheme for preemptive detection of attacks
2. A trusted controller free from network updates to take over the process in case of an attack

We propose to use a trustworthy autonomic guardian interface architecture (TAIGA), to implement both intrusion detection schemes and the backup controller. TAIGA works under the assumption that the controller, as well as network channel for the controller may be compromised. TAIGA monitors the communication between the supervisory layer and the infrastructure layer, allowing it to detect and respond to cyber attack.

This thesis builds on previous work that introduced formally verified application-specific configurable hardware for monitoring process operation at the lowest I/O pin level [11]. The erroneous behavior of the controller was identified by running a linear model of the plant and the production controller in accelerated mode to forecast the future behavior in case of a reconfiguration attack [12]. TAIGA, which separates the trusted components by implementing them in configurable hardware and protects them against latent malware and network reconfiguration, is discussed in [13].

In this thesis, we use a thorough understanding of the process behavior to develop intrusion detection mechanisms for preemptive detection of a network or reconfiguration attack before the attack compromises process stability. A slightly different version of the online prediction discussed in our previous work, along with classifiers based on three different techniques, are proposed and their performance is compared. A safety preserving optimal backup controller is developed to maintain process stability in case of an attack. In the TAIGA framework, the safety preserving backup controller and the trigger mechanism are instantiated in programmable logic, to protect them from malicious software updates. We demonstrate a hardware implementation of the new trigger mechanisms, using a Xilinx Zynq-7000 All Programmable SoC, on a Quanser rotary inverted pendulum that embodies many of the concerns of typical process control systems: stability, non-linearity, actuation limits, and disturbances.

1.4 Thesis Organization

This thesis is organized as follows: Chapter 1 provides a survey of the existing literature and notes the contributions of this work. Chapter 2 demonstrates mathematical modeling of the rotary inverted pendulum. Chapter 3 defines the control law for inverted pendulum balance control. Chapter 4 discusses the simulation and experimental results for the inverted pendulum balance control experiment. Chapter 5 provides a brief overview of TAIGA and its adaptability in an ICS environment. Chapter 6 discusses the implementation of TAIGA for the rotary inverted pendulum balance control experiment. Chapter 7 presents and evaluates the results of the given implementation in presence of few different attack scenarios. Chapter 8 provides conclusions and outlines future work and improvements.

Chapter 2

Mathematical Modeling of the Rotary Inverted Pendulum

This chapter follows the modeling of the rotary inverted pendulum, shown in Fig. 2.1, as discussed in the Rotary Pendulum Workbook by Quanser Inc. [14].

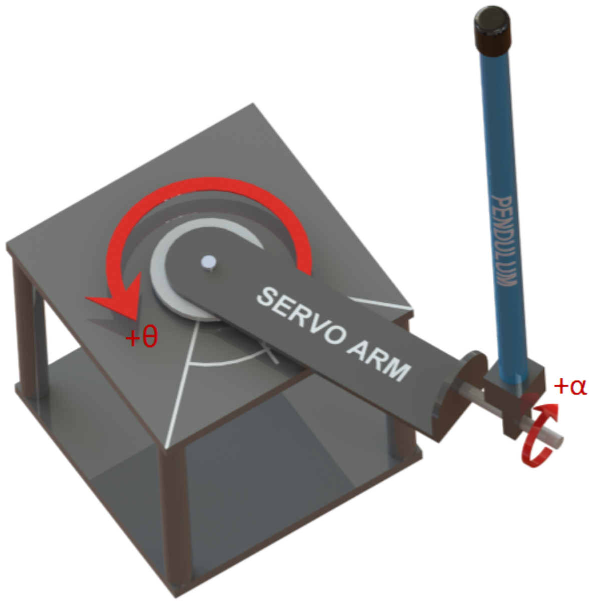

Figure 2.1: Quanser Rotary Inverted Pendulum

2.1 Model Conventions

The rotary arm is attached to a servo system and actuated. The arm has a length of L_r, a moment of inertia of J_r, and we assume its angle, θ, increases positively when it rotates counter-clockwise (CCW). The servo rotates CCW when the control voltage is positive, i.e. $V_m > 0$. The pendulum link is connected to the end of the rotary arm. It has a total length of L_p and its center of mass is $\frac{L_p}{2}$. The moment of inertia about the center of mass is J_p. The inverted pendulum angle α, is zero when the pendulum is perfectly upright in the vertical position and increases positively when rotated CCW. Figure 2.2 captures the system sign conventions.

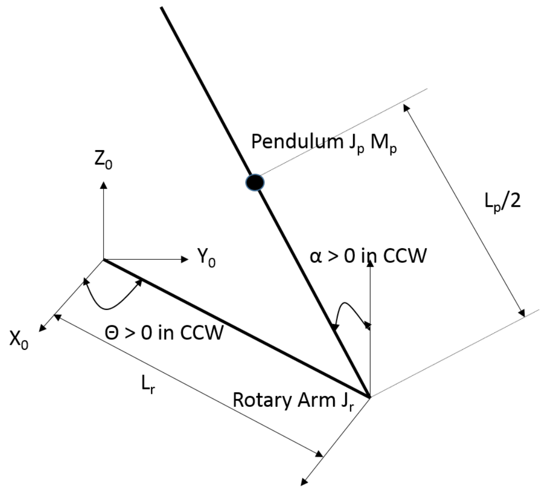

Figure 2.2: Rotary Inverted Pendulum Sign Conventions

2.2 Dynamic Equations of Motion

The equations that describe motion of the arm and pendulum in response to the servo motor voltage are nonlinear and will be obtained using the Euler - Lagrange equation.

$$\frac{\partial^2 L}{\partial t \partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = Q_i \tag{2.1}$$

The variables q_i are called generalized coordinates. For this system the pendulum angle α and rotary arm angle θ are the generalized coordinates.

$$q(t)^T = \left[\begin{array}{cc} \theta(t) & \alpha(t) \end{array} \right]$$

The corresponding velocities are

$$\dot{q}(t)^T = \left[\begin{array}{cc} \frac{\partial \theta(t)}{\partial t} & \frac{\partial \alpha(t)}{\partial t} \end{array} \right]$$

Note: Throughout this document, the time variable will be dropped from θ and α i.e. $\theta = \theta(t)$ and $\alpha = \alpha(t)$.

With the defined generalized coordinates, the Euler - Lagrange equations for the rotary pendulum system are

$$\frac{\partial^2 L}{\partial t \partial \dot{\theta}} - \frac{\partial L}{\partial \theta} = Q_1 \tag{2.2}$$

$$\frac{\partial^2 L}{\partial t \partial \dot{\alpha}} - \frac{\partial L}{\partial \alpha} = Q_2 \tag{2.3}$$

The Lagrangian of the system is

$$L = T - V \tag{2.4}$$

where T is the total kinetic energy of the system and V is the total potential energy of the system. The generalized forces Q_i account for the non-conservative forces (e.g. friction) applied to a system with respect to the generalized coordinates. Thus the generalized force acting on the rotary arm is

$$Q_1 = \tau - B_r \dot{\theta} \tag{2.5}$$

and the force acting on the pendulum is

$$Q_2 = -B_p \dot{\alpha} \tag{2.6}$$

The torque τ acting on the arm is generated by the input servo motor voltage, V_m. Opposing this applied torque is viscous damping B_r. Since the pendulum is not actuated directly, the only force acting on the link is the damping. The viscous damping coefficient of the pendulum is denoted by B_p.

Table 2.1: Servo Motor Coefficients

Parameter	Description
R_m	Motor Armature resistance
k_t	Motor torque constant
k_m	motor back - emf constant
K_g	total gear ratio
η_g	Gearbox efficiency
η_m	Motor efficiency

Once the kinetic and potential energy are obtained and the Lagrangian is found, then the task is to compute various derivatives to get the equations of motion. The nonlinear equations of motion for the rotary inverted pendulum are:

$$\begin{aligned}\tau - B_r\dot{\theta} = {} & (m_pL_r^2 + \frac{1}{4}m_pL_p^2 - \frac{1}{4}m_pL_p^2cos(\alpha)^2 + J_r)\ddot{\theta} - (\frac{1}{2}m_pL_pL_rcos(\alpha))\ddot{\alpha} \\ & + (\frac{1}{2}m_pL_p^2\sin(\alpha)\cos(\alpha))\dot{\theta}\dot{\alpha} + (\frac{1}{2}m_pL_pL_r\sin(\alpha))\dot{\alpha}^2\end{aligned} \tag{2.7}$$

$$\begin{aligned}-B_p\dot{\alpha} = {} & -(\frac{1}{2}m_pL_pL_rcos(\alpha))\ddot{\theta} + (J_p + \frac{1}{4}m_pL_p^2)\ddot{\alpha} - (\frac{1}{4}m_pL_p^2\sin(\alpha)\cos(\alpha))\dot{\theta}^2 \\ & - \frac{1}{2}m_pL_pg\sin(\alpha)\end{aligned} \tag{2.8}$$

The torque applied at the base of the rotary arm is generated by a servo motor as described by the equation.

$$\tau = \frac{\eta_gK_g\eta_mk_t(V_m - K_gk_m\dot{\theta})}{R_m} \tag{2.9}$$

Table 2.1 lists the servo parameters.

2.3 Non-linear Pendulum Model

A non-linear model of the pendulum is required to simulate the inverted pendulum setup. This is created by removing the algebraic dependencies in equations 2.7 and 2.8, and solving the simultaneous differential equations for $\ddot{\theta}$ and $\ddot{\alpha}$.

Rearranging equation, 2.7 and 2.8, we get

$$\begin{bmatrix} a1 & a2 \\ b1 & b2 \end{bmatrix} \begin{bmatrix} \ddot{\theta} \\ \ddot{\alpha} \end{bmatrix} = \begin{bmatrix} c1 \\ c2 \end{bmatrix}$$

where,

$$\begin{aligned}
a1 &= m_p L_r^2 + \frac{1}{4} m_p L_p^2 - \frac{1}{4} m_p L_p^2 cos(\alpha)^2 + J_r \\
b1 &= -\frac{1}{2} m_p L_p L_r cos(\alpha) \\
a2 &= -\frac{1}{2} m_p L_p L_r cos(\alpha) \\
b2 &= J_p + \frac{1}{4} m_p L_p^2 \\
c1 &= \tau - B_r \dot{\theta} - \frac{1}{2} m_p L_p^2 \sin(\alpha)\cos(\alpha)\dot{\theta}\dot{\alpha} - \frac{1}{2} m_p L_p L_r \sin(\alpha)\dot{\alpha}^2 \\
c2 &= -B_p \dot{\alpha} + \frac{1}{4} m_p L_p^2 \sin(\alpha)\cos(\alpha)\dot{\theta}^2 + \frac{1}{2} m_p L_p g \sin(\alpha)
\end{aligned} \tag{2.10}$$

This equation is of the form

$$AX = B \tag{2.11}$$

Thus, $\ddot{\theta}$ and $\ddot{\alpha}$ can be uniquely determined from the equation if A^{-1} exists.

2.4 Linearizing the nonlinear model

Consider the non-linear function $f(z)$ with

$$z^T = \begin{bmatrix} \theta & \alpha & \dot{\theta} & \dot{\alpha} & \ddot{\theta} & \ddot{\alpha} \end{bmatrix}$$

that is to be linearized about the operating point

$$z_0^T = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Since the left-hand side of the Eqn. 2.7 is already linear, setting the right-side equal to $f(z)$ gives,

$$\begin{aligned}
f(z) = {} & (m_p L_r^2 + \frac{1}{4} m_p L_p^2 - \frac{1}{4} m_p L_p^2 cos(\alpha)^2 + J_r)\ddot{\theta} - (\frac{1}{2} m_p L_p L_r cos(\alpha))\ddot{\alpha} \\
& + (\frac{1}{2} m_p L_p^2 \sin\alpha\cos\alpha)\dot{\theta}\dot{\alpha} + (\frac{1}{2} m_p L_p L_r \sin(\alpha))\dot{\alpha}^2
\end{aligned} \tag{2.12}$$

The linearized function is in the form,

$$\begin{aligned} f_{lin}(z) = f(z_0) &+ \left.\frac{\partial f(z)}{\partial \ddot{\theta}}\right|_{z=z_0} \ddot{\theta} + \left.\frac{\partial f(z)}{\partial \ddot{\alpha}}\right|_{z=z_0} \ddot{\alpha} + \left.\frac{\partial f(z)}{\partial \dot{\theta}}\right|_{z=z_0} \dot{\theta} \\ &+ \left.\frac{\partial f(z)}{\partial \dot{\alpha}}\right|_{z=z_0} \dot{\alpha} + \left.\frac{\partial f(z)}{\partial \theta}\right|_{z=z_0} \theta + \left.\frac{\partial f(z)}{\partial \alpha}\right|_{z=z_0} \alpha \end{aligned} \tag{2.13}$$

Computing the partial derivatives yields,

$$\begin{aligned} \left.\frac{\partial f(z)}{\partial \ddot{\theta}}\right|_{z=z_0} &= m_p L_r^2 + \frac{1}{4} m_p L_p^2 - \frac{1}{4} m_p L_p^2 \cos(0) + J_r \\ &= m_p L_r^2 + J_r \end{aligned} \tag{2.14}$$

$$\begin{aligned} \left.\frac{\partial f(z)}{\partial \ddot{\alpha}}\right|_{z=z_0} &= \frac{1}{2} m_p L_p L_r \cos(0) \\ &= \frac{1}{2} m_p L_p L_r \end{aligned} \tag{2.15}$$

All other terms are as follows:

$$\begin{aligned} \left.\frac{\partial f(z)}{\partial \dot{\theta}}\right|_{z=z_0} &= 0 \\ \left.\frac{\partial f(z)}{\partial \dot{\alpha}}\right|_{z=z_0} &= 0 \\ \left.\frac{\partial f(z)}{\partial \theta}\right|_{z=z_0} &= 0 \\ \left.\frac{\partial f(z)}{\partial \alpha}\right|_{z=z_0} &= 0 \\ f(z_0) &= 0 \end{aligned}$$

The linearized equation, is therefore

$$f_{lin}(z) = (m_p L_r^2 + J_r)\ddot{\theta} - \frac{1}{2} m_p L_p L_r \ddot{\alpha} \tag{2.16}$$

and the linearized version of Eqn. 2.7 is

$$(m_p L_r^2 + J_r)\ddot{\theta} - \frac{1}{2} m_p L_p L_r \ddot{\alpha} = \tau - B_r \dot{\theta} \tag{2.17}$$

Applying the same technique to linearize Eqn. 2.8,

$$\begin{aligned} f(z) = &-\frac{1}{2} m_p L_p L_r \cos(\alpha)\ddot{\theta} + (J_p + \frac{1}{4} m_p L_p^2)\ddot{\alpha} - \frac{1}{4} m_p L_p^2 \sin\alpha \cos\alpha \dot{\theta}^2 \\ &- \frac{1}{2} m_p L_p g \sin(\alpha) \end{aligned} \tag{2.18}$$

Computing the partial derivatives yield,

$$\left.\frac{\partial f(z)}{\partial \ddot{\theta}}\right|_{z=z_0} = -\frac{1}{2}m_pL_pL_r \tag{2.19}$$

$$\left.\frac{\partial f(z)}{\partial \ddot{\alpha}}\right|_{z=z_0} = J_p + \frac{1}{4}m_pL_p^2 \tag{2.20}$$

$$\left.\frac{\partial f(z)}{\partial \dot{\alpha}}\right|_{z=z_0} = -\frac{1}{2}m_pL_pg \tag{2.21}$$

The other $\dot{\theta}$, $\dot{\alpha}$ and θ based derivatives are zero and $f(z_0) = 0$.

The linearized function is,

$$f_{lin}(z) = -\frac{1}{2}m_pL_pL_r\ddot{\theta} + (J_p + \frac{1}{4}m_pL_p^2)\ddot{\alpha} - \frac{1}{2}m_pL_pg\alpha \tag{2.22}$$

and Eq. 2.8 becomes,

$$-\frac{1}{2}m_pL_pL_r\ddot{\theta} + (J_p + \frac{1}{4}m_pL_p^2)\ddot{\alpha} - \frac{1}{2}m_pL_pg\alpha = -B_p\dot{\alpha} \tag{2.23}$$

2.5 Linear state-space model

The linearized equations of the inverted pendulum can be written as,
State Equation:

$$\dot{x} = Ax + Bu \tag{2.24}$$

Output Equation:

$$y = Cx + Du \tag{2.25}$$

where

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{1}{4J_T}m_p^2L_p^2L_rg & -\frac{1}{J_T}(J_p + \frac{1}{4}m_pL_p^2)B_r & -\frac{1}{2J_T}m_pL_pL_rB_p \\ 0 & \frac{1}{2J_T}m_pL_pg(J_r + m_pL_r^2) & \frac{1}{2J_T}m_pL_pL_rB_r & -\frac{1}{J_T}(J_r + m_pL_r^2)B_p \end{bmatrix}$$

and

$$B = \begin{bmatrix} 0 \\ 0 \\ \frac{1}{J_T}(J_p + \frac{1}{4}m_p L_p^2 \\ \frac{1}{2J_T}m_p L_p L_r \end{bmatrix}$$

$$x^T = \begin{bmatrix} \theta & \alpha & \dot{\theta} & \dot{\alpha} \end{bmatrix}$$

is the state and

$$y^T = \begin{bmatrix} \theta & \alpha \end{bmatrix}$$

is the output.

The C & D matrices in the output equation are,

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

and

$$D = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

The velocities of the servo and pendulum angles can be computed either by taking the derivative and filtering the result through a low-pass filter or by using a state estimator, for example a Kalman filter.

Chapter 3

Rotary Inverted Pendulum Balance Control

In this chapter, we will use the LTI model to investigate the properties of the system such as stability, controllability and observability and then use it to define a linear quadratic regulator, control law and a Kalman filter state estimator to balance the pendulum in the inverted position.

3.1 Linear state-space model

The state equation for the LTI system is,

$$\dot{x} = Ax + Bu \tag{3.1}$$

Substituting the values of the coefficients and constants in the A and B matrices defined earlier we get,

$$\begin{bmatrix} \dot{\theta} \\ \dot{\alpha} \\ \ddot{\theta} \\ \ddot{\alpha} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1.0000 & 0 \\ 0 & 0 & 0 & 1.0000 \\ 0 & 81.4033 & -45.8570 & -0.9319 \\ 0 & 122.0545 & -44.1266 & -1.3972 \end{bmatrix} \begin{bmatrix} \theta \\ \alpha \\ \dot{\theta} \\ \dot{\alpha} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 83.4659 \\ 80.3162 \end{bmatrix} u$$

3.2 Stability

The two main notions of stability are,

- Internal stability, which is the qualitative behavior of the zero-input state response.
- Bounded Input Bounded Output stability, which involves the zero-state output response for a bounded input signal.

Internal stability focuses on the stability of equilibrium states for homogeneous state equations and can be defined in terms of **asymptotic stability** or **exponential stability** [15]. For the linear state Eqn. 3.1, asymptotic stability always implies bounded-input, bounded-output stability.

An LTI system is asymptotically stable if and only if $\Re\mathrm{e}(\lambda) < 0$ for all eigenvalues λ of A. Eigenvalues of A are the roots of the characteristic equation

$$det(sI - A) = 0 \tag{3.2}$$

For the linear state-space model developed in the previous section, open loop poles of the system are located at -48.72, 7.06, -5.86 and 0.

An asymptotically stable system must have all the eigenvalues with strictly negative real part. The rotary inverted pendulum has a pole in right-half plane making it unstable. From the practical standpoint, this makes perfect sense as the balanced condition is an unstable equilibrium state and requires a constant control effort to ensure stability.

3.3 Controllability

The linear state-space system is controllable if and only if

$$Rank([B \;\; AB \;\; A^2B \;\; .....A^nB]) = n \tag{3.3}$$

where, the matrix on the left is called the *controllability* matrix.

For the rotary inverted pendulum,

$$C_M = \begin{bmatrix} 0 & 83.5 & 3900 & 189000 \\ 0 & 80.3 & -3902.3 & 187300 \\ 83.4659 & -3902.3 & 189020 & -9151600 \\ 80.3162 & -3795.3 & 187300 & -9066000 \end{bmatrix}$$

and rank(C_M) = 4; hence the rotary inverted pendulum is controllable.

3.4 Observability

The linear state-space system is observable if and only if

$$n = rank \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix}$$

where the matrix on the right is called the *observability* matrix.

For the rotary inverted pendulum,

$$O_M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 81.4033 & -45.8570 & -0.9319 \\ 0 & -3846.7.3 & 2144 & 125.4 \end{bmatrix}$$

and rank(O_M) = 4; hence the rotary inverted pendulum is observable.

3.5 Design of an Optimal Feedback Control Law

Now that we have analyzed the properties of the system, our focus is on the design of an optimal feedback control law that yields desirable closed-loop performance in terms of both, the transient and steady-state response characteristics. Since we are operating on a digital platform, we need to design our controllers and estimators in the discrete domain. In the case of LQR, the controller is defined in continuous time to match the full state feedback gain values with the pole placement values provided by Quanser. After finding a good match the poles in continuous time were mapped to discrete time. For estimator design, we implement the Kalman filter in discrete domain.

3.5.1 State Variable Feedback Control

We begin this section with the linear-time invariant state equation

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

$$y = Cx + Du \tag{3.4}$$

with $x(t) \in \Re^n$ the internal state, $u(t) \in \Re^m$ the control input, and $y(t) \in \Re^p$ the measured output. Matrix A is the state matrix, B is the control input matrix, C is the output and measurement matrix, and D is the direct feed through or disturbance matrix. If all the states are measured as outputs, we can define a state variable feedback control law as,

$$u = -Kx + r \tag{3.5}$$

with the goal of achieving desired performance characteristics for the closed-loop state equation,

$$\dot{x} = (A - BK)x + Br \tag{3.6}$$

The effect of state feedback on the open-loop plant is shown in Fig. 3.1.

As illustrated by Eqn. 3.6, the state feedback control law features a constant state feedback gain matrix K. This is called a *regulator* as it is designed to deliver desirable transient response for nonzero initial conditions and/or attenuate disturbances to maintain the equilibrium state $\tilde{x} = 0$.

3.5.2 Control Design Specification

Apart from the obvious expectation of closed loop asymptotic stability, a specific characteristic behavior in terms of transient and steady-state response is expected of the process

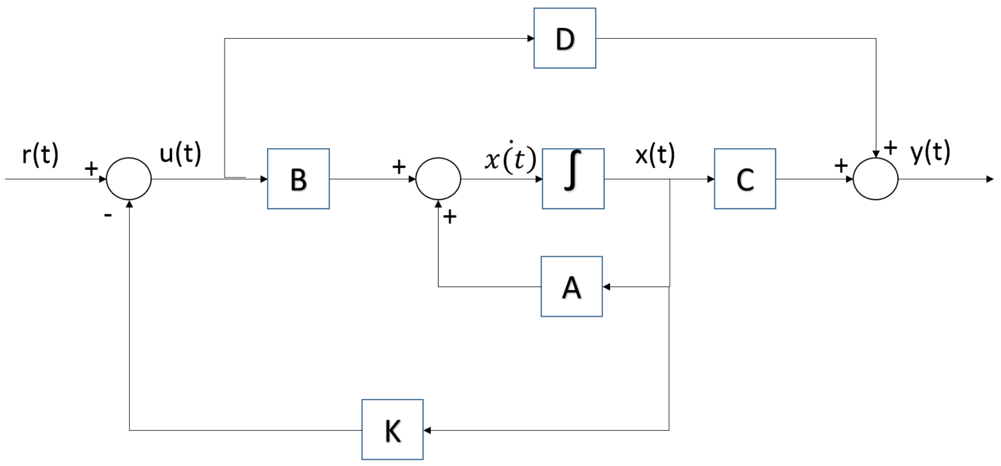

Figure 3.1: Closed Loop Full State Feedback Block Diagram

response. The behavior is characterized in terms of timing characteristics such as rise time or settling time and magnitude response characteristics such as percentage overshoot.

For the inverted pendulum balance control, the following response specifications are defined:

- **Specification 1 :** Maximum Overshoot: 5%
- **Specification 2 :** Maximum Settling Time: 1.5 Seconds
- **Specification 3 :** Maximum Pendulum angle deflection: $|\alpha| = 15°$
- **Specification 4 :** Maximum control effort (Voltage): $|V_m| = 5$ Volt

3.5.3 Optimal Feedback Control Law

In this section, the optimal full-state feedback, linear-quadratic regulator (LQR) problem is formulated and solved. Since the system under consideration is LTI, an infinite horizon regulator problem is defined where,

- Cost is time independent
- No terminal cost i.e. we assume that as $t \to \infty, x(t) \to 0$.

LQR Problem Statement

The basic optimization problem under consideration is that of finding a state-feedback control law of the form $u = -Kx$ that minimizes a performance index of the form

$$J = \int_0^\infty (x^T Q x + u^T R u) dt \tag{3.7}$$

for the system dynamics captured by Eqn. 3.1. Q is typically a positive semi-definite matrix and R is a positive definite matrix. The control goal is to keep the state x as close to the state $x_r = 0$ as possible. The term $(x^T Q x)$ in Eqn. 3.7 is then a measure of control accuracy and the term $(u^T R u)$ is a measure of control effort.

The performance index J can be interpreted as an energy function so that keeping it small ensures that the total energy of the closed-loop system remains small. Since, both the state $x(t)$ and the control input $u(t)$ are weighted in J, if J is small neither $x(t)$ nor $u(t)$ can be large. The two matrices Q and R are tuning parameters to achieve the desired performance. Generally speaking, a large Q means that to keep J small, state $x(t)$ must be smaller. On the other hand selecting R large means that the control input $u(t)$ must be smaller to keep J small. This means that a larger value of Q generally results in poles of the closed-loop system matrix $(A - BK)$ being further left in the s-plane so that the state decays faster to zero. On the other hand, a larger R means that less control effort is used resulting in slower poles and larger values of the state $x(t)$. Since the plant is linear and the performance index is quadratic, the problem of determining state variable feedback gain K to minimize J is called the Linear Quadratic Regulator (LQR) problem.

LQR Problem Solution

For the LTI system in Eqn. 3.1 , with initial condition $x(0) = x_0$ and performance index Eqn. 3.7, if (A, B) is controllable and (A, C) is observable, then

- The limit,

$$\bar{P} = \lim_{t_f \to \infty} P(t, t_f) \tag{3.8}$$

  the solution of the Riccati differential equation exists. This is a constant matrix (independent of time) and is the unique, positive definite solution to the Algebraic Riccati Equation,

$$A^T \bar{P} + \bar{P} A + Q - \bar{P} B R^{-1} B^T \bar{P} = 0 \tag{3.9}$$

- The closed-loop system matrix $A_{cl} = A - BR^{-1}B^T\bar{P}x$ is *stable*
- The infinite horizon control law is $u = -R^{-1}B^T\bar{P}x$

LQR Design Procedure

The design procedure for finding the LQR feedback gain K is

- Select design parameter matrices Q and R
- Solve algebraic Riccati equation for $\bar{P}$

$$A^T\bar{P} + \bar{P}A + Q - \bar{P}BR^{-1}B^T\bar{P} = 0 \tag{3.10}$$

- Find the state variable feedback gain using, $K = R^{-1}B^T\bar{P}$
- Verify the design. If not suitable, try again with different weighing matrices, Q,R.

Limitations of LQR

LQR yields good results when it comes to robustness, but it has certain limitations.

- It requires knowledge of all the states of the physical system. In real world processes, not all the states are measurable. A state estimator is used in our design to estimate the velocity states based on the measurements of the servo and the pendulum position, providing LQR the full state feedback it needs.
- The performance of the controller gets affected by nonlinearities of the plant. In our design, the plant model is linearized and operated close to the linearization point, minimizing the impact of nonlinearities.
- LQR heavily relies on plant model accuracy. A poor dynamic model could affect its performance. For our design, the plant model is derived from physical principles with accurate values of the individual components. The model can be validated in the future using system identification techniques.
- LQR tuning is not intuitive, as the tuning matrices Q and R are not directly related to the transient behavior of the control law. Hence an iterative process needs to be followed to achieve a desired response.

3.5.4 State Estimator Design

Background

For a controllable plant, we can implement a state feedback controller if the plant states are known. However, for the rotary inverted pendulum system the output equation only provides information about two states, namely rotary arm angle θ and inverted pendulum angle α. The Kalman filter enables us to determine the velocity states associated with the arm and the pendulum.

An alternative approach to the Kalman filter is to use either a simple numerical derivative that is then low-pass filtered, or an observer. A simple derivative filter though useful, results in noisy velocity values affecting the controls performance. An observer, for example a Luenberger observer [16], can be effectively used for state estimation, however the Kalman filter is more theoretically attractive as it ensures minimization of the variance of the estimation error.

Design Specifications

The objective of the estimator design is to develop an estimator which will give an accurate estimate of the true state as we cannot measure it [17]. The objective can be quantified in terms of the following two criteria.

- Expected value of the estimate should be equal to the true value, that is the estimate should not be biased.
- The estimator should have the smallest possible error variance.

Kalman Filter Problem Statement

The Kalman filter addresses the general problem of trying to estimate the state x $\in \Re^n$ of a discrete-time controlled process that is governed by the linear stochastic difference equation.
State Equation:

$$x_{k+1} = Ax_k + Bu_k + w_k \tag{3.11}$$

with a measurement $y \in \Re^m$
Output Equation:

$$y_k = Cx_k + v_k \tag{3.12}$$

In the above equations A, B, C are matrices; k is the time index; x is called state of the system; u is a known input to the system; y is the measured output; and w and v are the noises. The variable w is called the process noise while v is called the measurement noise. We assume that w and v are zero mean purely random processes with,
Process noise covariance:

$$E\{ww^T\} = W \tag{3.13}$$

Measurement noise covariance:

$$E\{vv^T\} = V \tag{3.14}$$

The vector x contains all the information about the present state of the plant, but we cannot measure x directly. Instead, we measure y which is a function of x corrupted by noise v. We can divide the equations of the Kalman filter into two groups:

- The time update equations are responsible for projecting forward in time to obtain a priori estimate for the next time step.
- The measurement update equation incorporates the new measurement in the priori estimate to obtain an improved posteriori estimate.

The time update equation can be thought of as a *predictor* equation while the measurement update equation can be thought of as a *corrector* equation. Thus, the final estimation algorithm resembles a *predictor-corrector* algorithm.

We define $\hat{x_k}^- \in \Re^n$ to be the priori state estimate at step k given knowledge of measurement prior to step k and $\hat{x_k}^+ \in \Re^n$ to be our posteriori state estimate at step k given measurement y_k. We can define the priori and posteriori errors as

$$e_k^- = x_k - \hat{x_k}^- \tag{3.15}$$

$$e_k^+ = x_k - \hat{x_k}^+ \tag{3.16}$$

The priori estimate covariance is then

$$P_k^- = E[e_k^- \ e_k^{-T}] \tag{3.17}$$

and the posteriori estimate covariance is

$$P_k^+ = E[e_k^+ \ e_k^{+^T}] \tag{3.18}$$

In deriving the equation for the Kalman filter, we compute posteriori state estimates $\hat{x_k}^+$ as a linear combination of a priori state estimate $\hat{x_k}^-$ and a weighted difference between actual measurement y_k and a measurement prediction $C\hat{x_k}^-$.

$$\hat{x_k}^+ = \hat{x_k}^- + K_f(y_k - C\hat{x_k}^-) \tag{3.19}$$

The difference $(y_k - C\hat{x_k}^-)$ is called the *residual* which reflects the discrepancy between the prediction and the actual measurement. A zero residual means that both are in complete agreement.

Kalman Filter Solution

For a *detectable* linear time invariant system described in Eqn. 3.11 and Eqn. 3.12, one may obtain a constant estimator gain K_f by considering the limit $t_0 \to -\infty$. In steady-state, the Riccati equation for the co-variance P_k becomes an algebraic Riccati equation (ARE).

$$PA^T + AP - PC^TV^{-1}CP + W = 0 \tag{3.20}$$

The optimal estimator gain satisfying the ARE is given by,

$$K_f = PC^TV^{-1} \tag{3.21}$$

Since we consider the limit $t_0 \to -\infty$, the closed-loop state matrix $A - K_fC$ should be *Hurwitz*. To achieve that, the pair (A, D) must be *stabilizable* for any full column rank matrix D satisfying $W = DD^T$. This requirement is analogous to the requirement in LQ controller design that, any unstable modes should be reflected in the cost function. Though W represents the spectral density of the process disturbance, in practice it must be modified to satisfy the stabilizability requirement. Otherwise, the state estimate may diverge from the true value.

Based on Eqn. 3.19, we could also comment that as the measurement covariance V approaches zero, the actual measurement y_k is trusted more while the predicted measurement $C\hat{x_k}^-$ is trusted less. On the other hand as the priori estimate covariance error P_k^- approaches zero the actual measurement y_k is trusted less and the predicted state $C\hat{x_K}^-$ is trusted more.

3.5.5 Linear Quadratic Gaussian Control

Combining the linear quadratic *control* problem for the deterministic LTI system,

$$x_{k+1} = Ax_k + Bu_k \tag{3.22}$$

and the minimum variance *estimation* problem for the stochastic LTI system

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + w_k \\ y_k &= Cx_k + v_k \end{aligned} \tag{3.23}$$

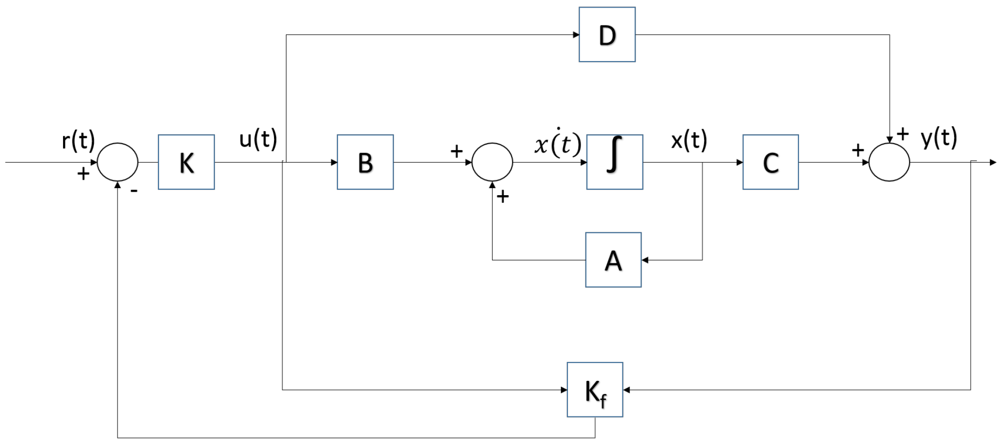

Figure 3.2: Closed Loop Block Diagram for Linear Quadratic Gaussian Control

subject to zero-mean, Gaussian disturbance processes, leads to the following *linear- quadratic gaussian (LQG)* control problem: Find an output feedback controller for the latter system which minimizes the cost function

$$\lim_{t\to\infty} E\{x^TQx + u^TRu\} \tag{3.24}$$

Because of the special structure of the problem, the solution satisfies a ***separation principle:*** The minimizing feedback control law is linear state estimate feedback $u = -K\hat{x}$, where the control gain K and the estimator gain K_f are determined *separately* by solving, respectively,

- The LQR problem for Eqn. 3.1, with state penalty matrix **Q** and control penalty **R**.
- The minimum variance estimation problem for Eqn. 3.23, with disturbance spectral density **W** and noise spectral density **V**.

The resulting controller is an n-dimensional dynamic compensator. The $2n$-dimensional closed-loop system is

$$\begin{aligned} \dot{x} &= Ax + B(-K\hat{x}) + w \\ \dot{\hat{x}} &= A\hat{x} + B(-K\hat{x}) - K_f(C\hat{x} - y) \end{aligned} \tag{3.25}$$

The closed loop control block diagram is shown in Fig. 3.2.

The eigenvalues of closed-loop system are n eigenvalues of **A - BK** and the n eigenvalues of **A** **-** K_f**C** . This observation underscores the separation principle, which says that the controller and observer dynamics may be designed separately according to our known optimal design procedures and, moreover, that the resulting combination remains optimal, i.e. it solves the LQG problem.

Chapter 4

Simulation and Experimentation

An inverted pendulum balance control example is used to demonstrate the effectiveness of the Linear Quadratic Gaussian Control law. In this implementation, the servo and pendulum positions are measured as output feedback and used to estimate the velocity states using a minimum variance estimator. Xilinx's Zynq-7000 All Programmable SoC is used as a programmable controller for control software. The encoders for position measurement are sampled every 1 millisecond.

4.1 Control Law Implementation

For implementing the control law for the inverted pendulum balance control, a modular design strategy is followed where every task is defined and tested in a standalone environment before integrating all of them together [18].

- Create a mathematical model of the plant, either from physical laws or by acquiring and analyzing the real data.
- Use the plant model to synthesize an effective control algorithm.
- Simulate the time response of the system to inputs such as unit step changes.
- Deploy the controller on an embedded platform using the code that is automatically generated from the simulated controller.

We have followed the first three steps in model based design strategy. For the last step, instead of using the auto code generated from the simulated controller an embedded C code is manually created. This is necessary as currently no tool exists to synthesize auto-generated C code into configurable hardware.

4.1.1 LQR Specifications

On account of the special structure of the LQG problem, the separation principle can be followed to design the LQR and the Kalman Filter separately to meet the desired specifications. For LQR design, the design procedure defined in Section 3.5.3 is followed.

Since the primary goal of the experiment is to balance the pendulum, control law should ensure that the pendulum deviation from its upright position is minimal. Any motion of the servo arm results in pendulum oscillations. Thus, it is imperative that control law should ensure minimum deviation of servo position from the set-point. To meet these goals, the Q matrix is tuned with a high penalty on both, the servo position, θ, and the pendulum position, α.

$$Q = \begin{bmatrix} 6.3 & 0 & 0 & 0 \\ 0 & 6.3 & 0 & 0 \\ 0 & 0 & 0.0002 & 0 \\ 0 & 0 & 0 & 0.0006 \end{bmatrix}$$

Since the inverted pendulum setup is actuated only by a servo motor, the R matrix has just 1 element. As discussed in Section 3.5.3, selecting large R results in smaller values of control input u. To limit the control voltage to 5 Volts, an iterative loop is used to simulate the pendulum behavior for different values of R and based on the results 0.38 is selected as the weight for control input. Simulation and experimental results confirm that this setting ensures that u remains within its bounds.

$$R = 0.38 \tag{4.1}$$

The state feedback gain vector to achieve the desired performance is

$$K_c = \begin{bmatrix} -4.0275 & 21.1205 & -2.4906 & 2.8588 \end{bmatrix}$$

The LQR control law places the closed loop poles for the plant at $-4.7354 \pm 1.8644i$ (dominant poles), -11.9257 and -47.5858. Since all the poles are in the left half plane the control law ensures closed-loop stability. The poles of the LQR are transformed into discrete domain using transformation $z_0 = e^{s_0 T}$ where s_0 is the pole location in continuous time and T is the sampling time for analog to digital conversion. A pole placement algorithm is then used to determine the gain coefficients for the discrete domain poles. This gives the closed loop poles for the plant as, $0.9953 \pm 0.0018j$, (dominant poles), 0.9536 and 0.9880. Since all the poles are inside the unit-circle, the closed loop system is stable.

4.1.2 Discretization of the Plant

To implement Kalman filter on a digital platform, the continuous-time plant model is converted to discrete-time using the zero-order hold technique.

The discretized linear model of the physical plant is,

$$A = \begin{bmatrix} 1.0000 & 0.0001 & 0.0012 & -0.0000 \\ 0.0000 & 1.0001 & -0.0000 & 0.0012 \\ 0.0000 & 0.0988 & 0.9443 & -0.0011 \\ 0.0000 & 0.1497 & -0.0536 & 0.9984 \end{bmatrix} \quad B = \begin{bmatrix} 0.0001 \\ 0.0001 \\ 0.1013 \\ 0.0975 \end{bmatrix}$$

The C and D matrices remain unchanged on the discretization.

The discrete domain open-loop eigenvalues of the inverted pendulum are 1.0000, 0.9412, 1.0087 and 0.9929. Since there is a pole outside the unit circle, the pendulum is inherently unstable.

4.1.3 Kalman Filter Specification

To find the Kalman filter gain, the covariance matrices for the process noise and measurement noise spectral density should be defined. Since we are using encoders for both servo and pendulum angle measurement, the standard deviation for the encoder is selected to be half a count. The encoder resolution is 0.0015 *rad/count*. Therefore,

$$V = \begin{bmatrix} 0.00076^2 & 0 \\ 0 & 0.00076^2 \end{bmatrix}$$

W is tuned to ensure the process stability. The process noise is selected based on the difference between maximum and minimum values of each state(δ). A small percentage of the range of operation is selected as the noise for the particular state. We assume a small noise of the order of 10^{-4} on position states. In case of the pendulum, δ for velocity states is smaller than δ for the position states when we consider the state trajectory close to the balancing position. Hence, velocity state noise is set at $1/5^{th}$ of the position state noise.

$$W = \begin{bmatrix} 0.0001 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0001 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.00002 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.00002 \end{bmatrix}$$

Using Matlab, Kalman filter gains K_f are computed for the selected design parameters.

$$K_f = \begin{bmatrix} 0.9943 & 0.0000 \\ 0.0000 & 0.9943 \\ 0.0023 & -0.0071 \\ -0.0078 & 0.2988 \end{bmatrix}$$

The poles of the Kalman filter are located at 0.0059, 0.0057, 0.9544 and 0.9593. All poles of Kalman filter are inside unit circle and hence stable.

4.1.4 Discrete Kalman Filter Implementation

The state estimate at time $k+1$ given data at time k is given by the prediction equation

$$\hat{x}_{k+1}^{-} = A\hat{x}_k^{+} + Bu_k \tag{4.2}$$

This state equation gets updated based on the measurement at time $k+1$ by correction equation.

$$\hat{x}_{k+1}^{+} = \hat{x}_{k+1}^{-} + K_f(y_{k+1} - C\hat{x}_{k+1}^{-}) \tag{4.3}$$

This can be simplified to

$$\begin{aligned} \hat{x}_{k+1}^{+} &= A\hat{x}_k^{+} + Bu_k + K_f(y_{k+1} - C(A\hat{x}_k^{+} + Bu_k)) \\ &= (I - K_fC)A\hat{x}_k^{+} + (I - K_fC)Bu_k + K_fy_{k+1} \end{aligned} \tag{4.4}$$

The first two terms on the right-hand side of Eqn. 4.4, are precomputed at the end of every scheduled execution as they depend on the state and control data at time k. The last term depends on the measured output at time $k+1$ and is calculated in real time.

4.1.5 Validation of Control Law

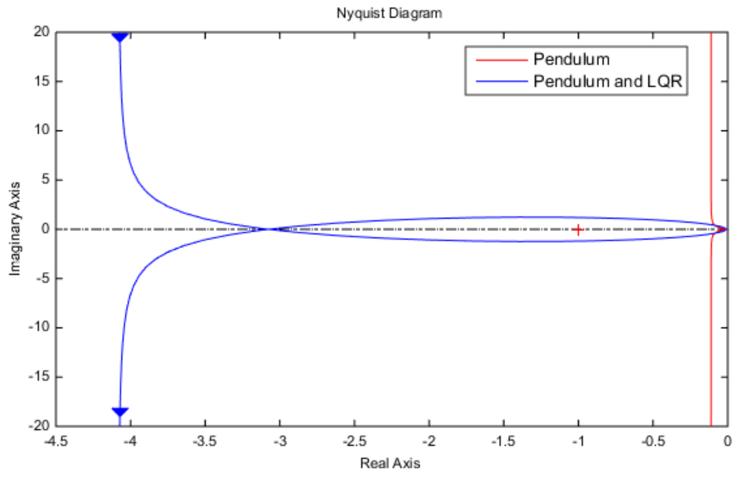

Figure 4.1: Frequency Response of Plant and Controller: Nyquist Plot

A good way to validate the developed control law is to analyze its frequency response of the loop transfer function. Figure 4.1 and 4.3 captures Nyquist and Bode plot for both the physical plant and compensated physical plant. As per the Nyquist criterion, if the loop transfer function $L(z)$ has p poles outside unit circle, for closed loop stability, Nyquist plot of $L(z)$ must encircle the point $-1 + j0$ exactly p times in counter-clockwise direction. The pendulum has 1 pole outside unit circle. Figure 4.1 shows that the loop transfer function for physical plant do not encircle the point $-1 + j0$ at all while the loop transfer function for the physical plant and the controller encircles it once in CCW direction. This proves that the control law is stabilizing. An open loop analysis is performed to determine the stability margins of the LQG Controller. Figure 4.2 captures the closed loop LQG controller. A red cross indicates the position at which closed loop is broken to analyze the open loop response. The *2n* dimensional control loop considered for this analysis is,

$$\begin{bmatrix} \dot{x} \\ \dot{\hat{x}} \end{bmatrix} = \begin{bmatrix} A & 0 \\ K_f C & A + BK_c - K_f C \end{bmatrix} \begin{bmatrix} x \\ \hat{x} \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u$$

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

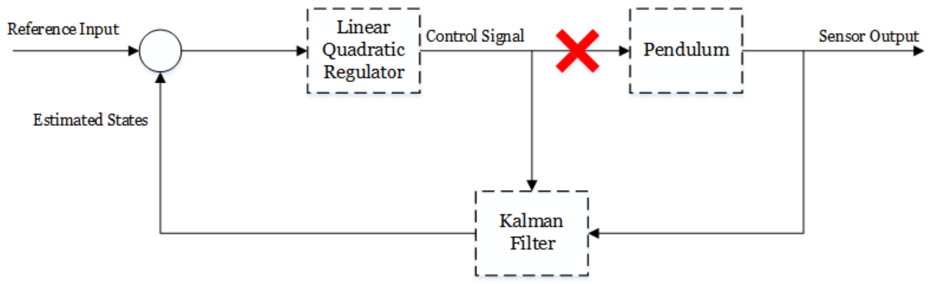

Figure 4.2: Open Loop Analysis for LQG Controller

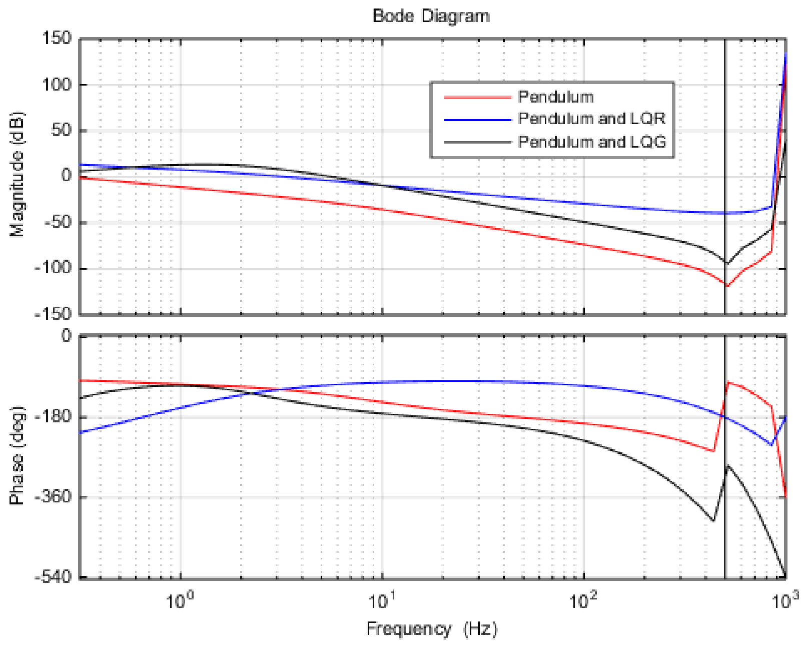

Figure 4.3: Frequency Response of Plant and Controller: Bode Plot

The stability margins can be derived from the bode plot in fig. 4.3 in terms of phase margin and increasing and decreasing gain margins. The Nyquist frequency on the bode plot corresponds to the +1 on Z-grid. Since, the open loop plant contains a pure integrator, which on a Nyquist plot is placed at +1, it results in increasing magnitude response on the bode plot beyond Nyquist frequency.

- The *increasing gain margin* is the positive factor $\alpha > 1$ by which feedback gain can be increased before the closed loop system becomes unstable. In our design increasing gain margin for LQR is 39 dB and for LQG, it is 17.4 dB .
- The *decreasing gain margin* is the factor $0 < \alpha \leq 1$ by which the feedback gain can be decreased before the closed loop system becomes unstable. In our design decreasing gain margin for LQR is -9.77 dB while, for LQG it is -4.01 dB.
- The *phase margin* is the amount of additional phase lag at the gain crossover frequency required to bring the closed loop system to the verge of instability. For LQR, the phase margin in our design is 64.6°. Thus the feedback system can survive a time delay of 53 milliseconds before loosing stability. However, with Kalman filter for state estimation, the phase margin goes down to 21.6°, meaning the control loop can survive a phase delay of 10 milliseconds.

4.1.6 Feedback Control

For the feedback control loop that balances the rotary pendulum the reference state is defined as

$$x_d = [\theta_d \;\; 0 \;\; 0 \;\; 0] \tag{4.5}$$

where θ_d is the desired rotary arm angle. The controller is

$$u = K_c(x_d - x) \tag{4.6}$$

where K_c is the LQR controller gain.

While designing the control law, the inverted pendulum is linearized near the reference state. Hence, the control law will be stabilizing only if the pendulum operates close to the reference state. To ensure effective control, the control law kicks in only when the pendulum is brought close to its upright vertical position.

$$\begin{aligned} u &= K_c(x_d - x) \quad for \;\; |x_2| < \epsilon \\ &= 0 \qquad\qquad\quad otherwise \end{aligned} \tag{4.7}$$

where ϵ is the angle about which the controller should engage. We select $\epsilon = 15\,^\circ$

4.2 Results

The simulation response of the linear pendulum model, to a step input of 20° for the servo position, is captured in Fig. 4.4.

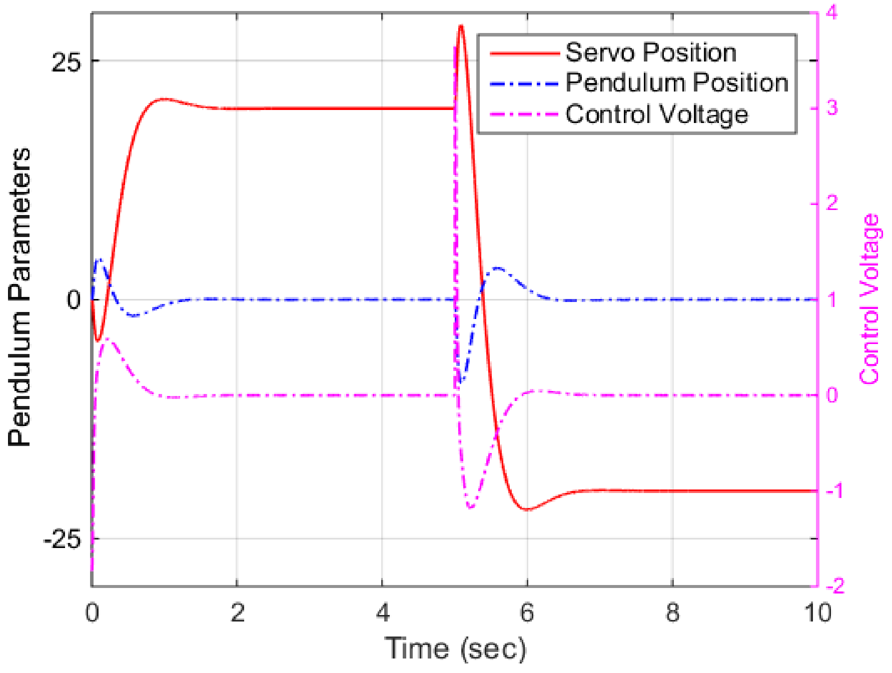

Figure 4.4: Simulation of Dynamic response of Linearized Inverted Pendulum

Settling time is the time it takes for the response curve to reach and stay within a range of 2% from the desired value. For the response in Fig. 4.4 it is 1.1337 seconds. The percentage overshoot is about 4.75%. The control voltage stays within ±5 Volt. The pendulum stays within 15°, meeting all the desired requirements.

The response of the nonlinear model of the pendulum described in section 2.3 for the same control law is captured in Fig. 4.5. The transient response of nonlinear model deviates from the linear model on account of the nonlinearity. Figure 4.6 captures the Simulink model developed for analysis. Applying the same control law on the physical plant yields the response captured in Fig. 4.7. For the physical plant, an operational set-point of $\theta = 25°$ is commanded at $T = 30$ sec.

To move the pendulum in a counter-clockwise direction, the governing controller first applies

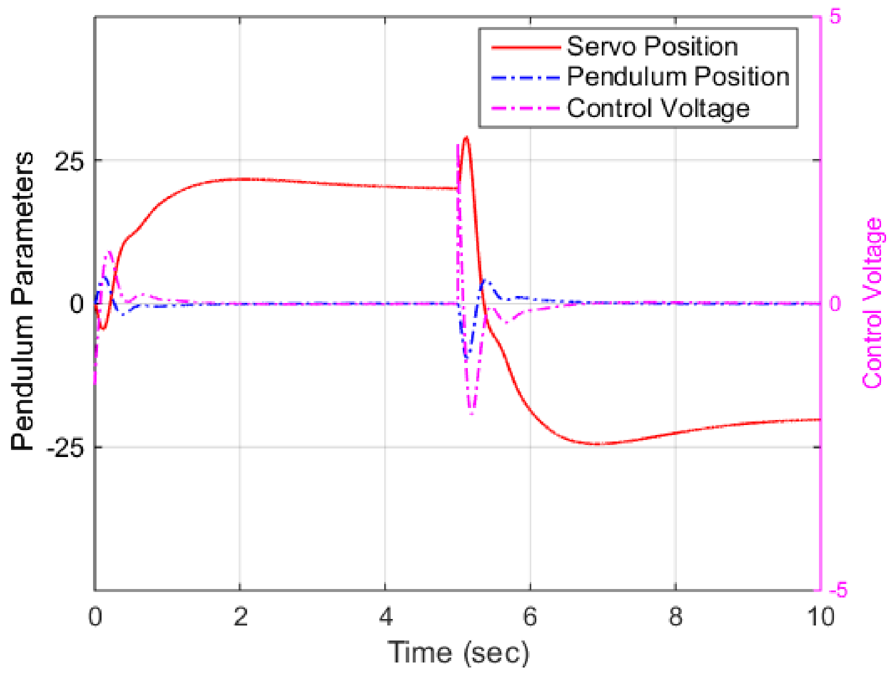

Figure 4.5: Simulation of Dynamic Response of Nonlinear Inverted Pendulum

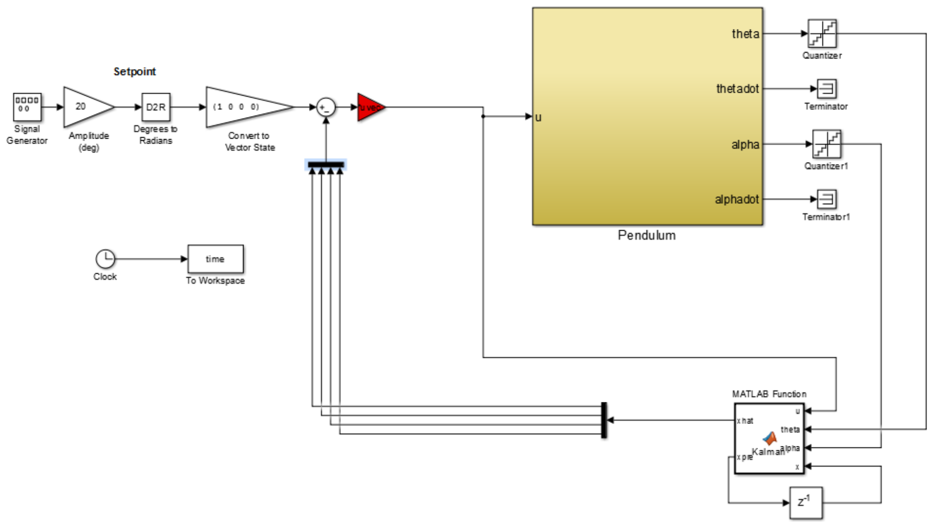

Figure 4.6: Nonlinear Model of Inverted Pendulum

a small negative voltage to tilt the pendulum. A positive control voltage signal takes the tilted pendulum from the current servo position to the desired position. As the desired servo position is reached, the pendulum is again balanced in an upright position by applying a negative voltage. The steady-state response of the pendulum in the physical system exhibits a low-frequency oscillation in steady state from a limit cycle caused by static and Coulomb friction in the servo.

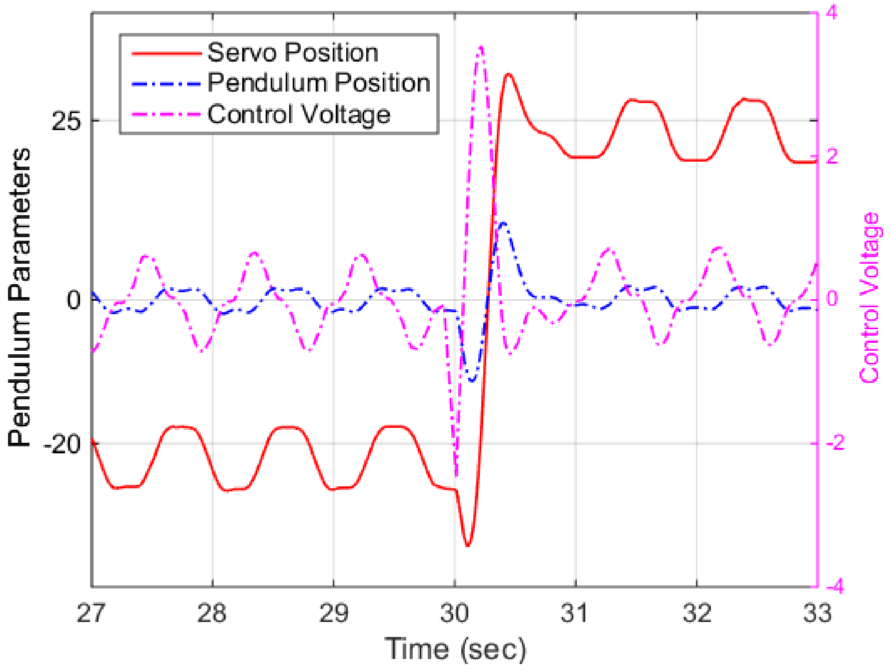

Figure 4.7: Dynamic response of the Physical system

Figure 4.8 captures frictions contributing to the steady-state oscillation in the response.

- The torque due to static friction S is that which is required to produce motion in the stationary rotary arm.
- The torque due to Coulomb friction F is that which in a moving rotary arm is independent of the velocity of the arm and which always opposes the motion.
- The torque due to viscous friction is that which is dependent upon the velocity of the rotary arm.

To model the frictions in the servo we use the friction model discussed in [19] [20].

$$F_{fric} = \begin{cases} F_{static} & \text{if} \quad \dot{\theta} = 0 \\ F_{coulomb} + F_{viscous} & \text{if} \quad \dot{\theta} \neq 0 \end{cases} \tag{4.8}$$

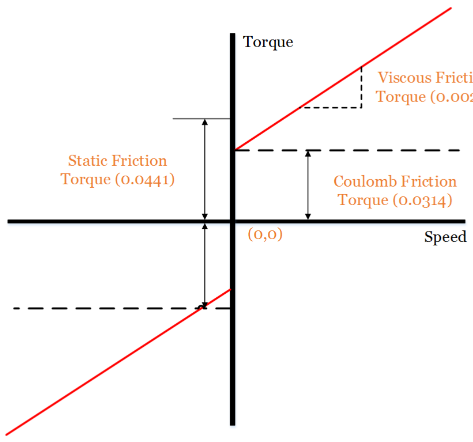

Figure 4.8: Friction Model for Rotary Inverted Pendulum

The friction torques opposing the motor torque are further modeled as:

$$\tau_{stick} = \begin{cases} \tau_{static} & \text{if} \quad |\tau_{applied}| \geq \tau_{static} \\ \tau_{applied} & \text{if} \quad |\tau_{applied}| < \tau_{static} \end{cases} \tag{4.9}$$

$$\tau_{coulomb} = \mu_c \, sgn(\dot{\theta}) \tag{4.10}$$

$$\tau_{viscous} = B_r \dot{\theta} \tag{4.11}$$

where, μ_c and B_r are the coefficients of Coulomb and viscous friction respectively. The coefficient of viscous and Coulomb friction were provided by Quanser. The static friction is tuned to match the simulation response with the experiment. The values of all the parameters are listed in Table 4.1.

Table 4.1: Friction model coefficients

Parameter	Description	Value
τ_{static}	Static Friction Torque	0.0441
μ_c	Coefficient of Coulomb Friction	0.0314
B_r	Coefficient of Viscous Friction	0.0024

The rotary inverted pendulum balance is an unstable equilibrium and requires constant small corrections to maintain balance. The LQG control law is designed using a linear model of the plant and does not take into account the nonlinear friction model. The small control effort applied by the control law is not sufficient to overcome the opposing friction torque

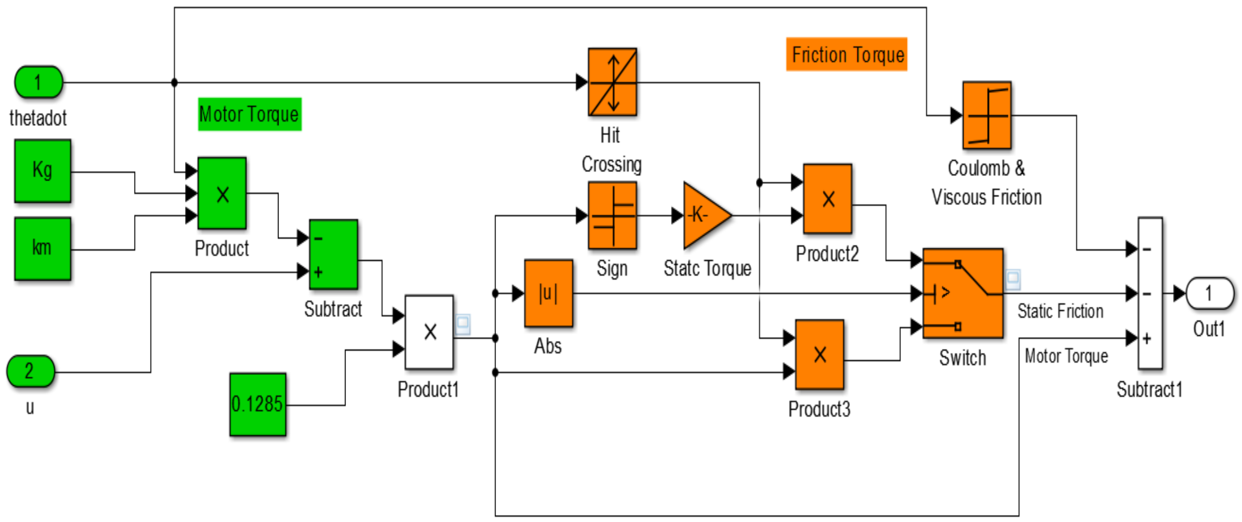

Figure 4.9: Addition of Friction Model in Applied Torque Calculations

and achieve steady-state balance. Therefore, an increase in control voltage is necessary to overcome static friction τ_{static}. A constant coulomb friction μ_c and a velocity dependent viscous friction B_r, opposes the moving pendulum arm requiring over correction by the linear control law. By the time, control law achieves an equilibrium condition with the opposing friction torque, the pendulum overshoots its commanded set-point. To move the pendulum from its equilibrium condition, the control law has to overcome all the friction forces again resulting in overshoot in the opposite direction. This results in a steady state limit cycle generation in the pendulum response.

To model the steady-state oscillations, we start with the assumption that the rotary arm and pendulum joint have minimal static and Coulomb friction opposing its motion. For the rotary arm and servo joint, the modeled friction torque opposes the motor torque. Thus we modify the Eq. 2.7 by introducing the Coulomb and static friction terms.

The static friction torque opposes the motion at zero servo velocities only. As per the stick-slip friction phenomenon, stiction force is more than the slip or kinetic friction. Thus stiction torque tuning is performed by selecting slightly higher values than the known values of Coulomb and viscous friction until the simulation response matches with the physical plant response.

$$\begin{aligned}\tau - B_r\dot{\theta} - \mu_c \; sgn(\dot{\theta}) - \tau_{static} = {} & (m_pL_r^2 + \frac{1}{4}m_pL_p^2 - \frac{1}{4}m_pL_p^2cos(\alpha)^2 + J_r)\ddot{\theta} - (\frac{1}{2}m_pL_pL_rcos(\alpha))\ddot{\alpha} \\ & + (\frac{1}{2}m_pL_p^2\sin(\alpha)\cos(\alpha))\dot{\theta}\dot{\alpha} + (\frac{1}{2}m_pL_pL_r\sin(\alpha))\dot{\alpha}^2\end{aligned} \tag{4.12}$$

Figure 4.9 captures the torque model implementation in servo torque calculations. The components marked in green models the torque equation 2.9. Blocks in orange represent the friction model in equation 4.12. Kg and Km are constant values provided by Quanser. u is the calculated control voltage. The hit crossing block in stiction calculation detects the instances at which the servo velocity is zero. The block outputs 1 for $\dot{\theta} = 0$ and 0 for all other servo velocities. The sign block outputs +1 for positive velocities and -1 for negative velocities. This ensures that the stiction torque is applied to oppose the physical pendulum and acts only if the velocity is zero. The comparison block compares the absolute value of applied force against the stiction force. If the applied force is less than the stiction firce it is applied as the opposing friction torque while an applied force greater than stiction results in τ_{static} as the opposing static torque.

Figure 4.10 indicates the simulated response using the tuned friction model. On account of the frictions listed above, the pendulum oscillates with a constant amplitude and frequency exhibiting a stable *limit cycle*. The frequency of the oscillation can be determined from root locus analysis of the loop transfer function for the plant and the LQG controller. Figure 4.11 captures all the open-loop zeros and poles of the pendulum and Kalman Filter. The loop transfer function with pendulum and Kalman filter has 6 zeros and 8 poles. There are a couple of poles outside unit circle making system unstable at zero gains. As the gain increase, these poles enter unit circle making the physical system stable. The stick-slip friction however, prevents the physical system from achieving the perfect balance.

The frequency of the oscillation can be calculated based on the point at which pole crosses the unit circle, which occurs at approximately $1 \pm 0.00689j$. This crossover point provides us information about the frequency of oscillation of the nonlinear response. Since +1 on Z-grid corresponds to the Nyquist frequency ($samplingfrequency/2$), the damped frequency of oscillations is

$$w_d = 0.00689/pi \times (fs/2) = 1.0965 \;\; Hz \tag{4.13}$$

The measured frequency of oscillation is 1.12 Hz while simulated model frequency is 1.02 Hz.

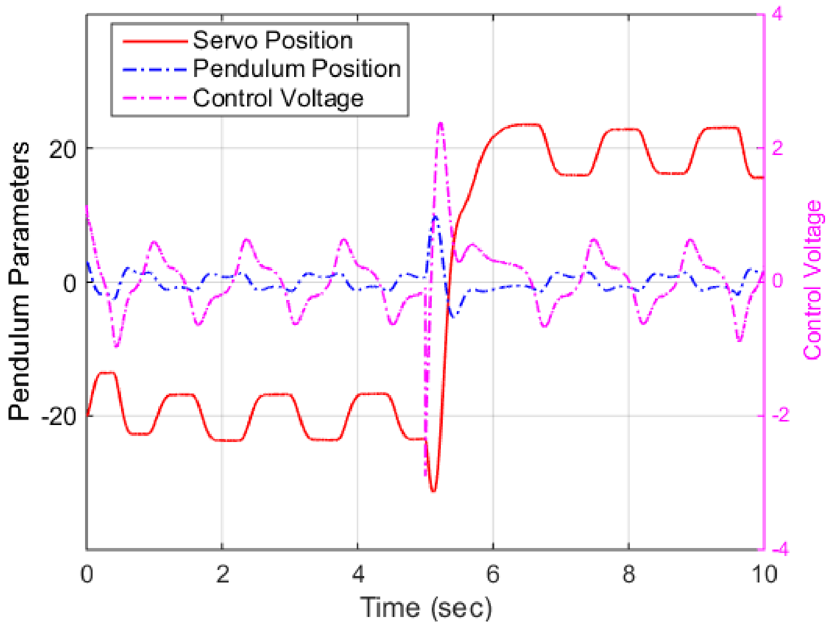

Figure 4.10: Simulated response of Nonlinear Pendulum

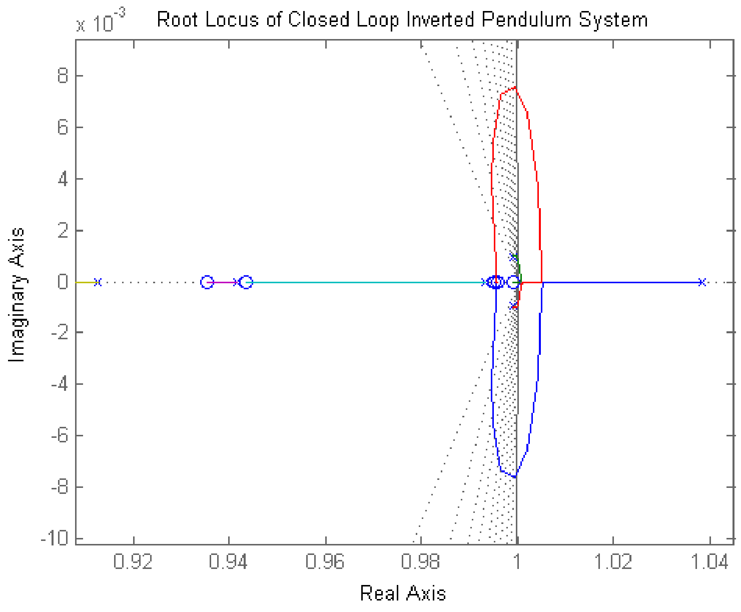

Figure 4.11: Root Locus of Loop Transfer Function of Pendulum and LQG Controller

Chapter 5

TAIGA Overview

Trustworthy Autonomic Interface Guardian Architecture is an expansion of traditional control systems. TAIGA introduces a trusted backup controller and an I/O intermediary module (IOI). The IOI module acts as a trusted I/O processor for both supervisory as well as physical inputs to the embedded controllers.

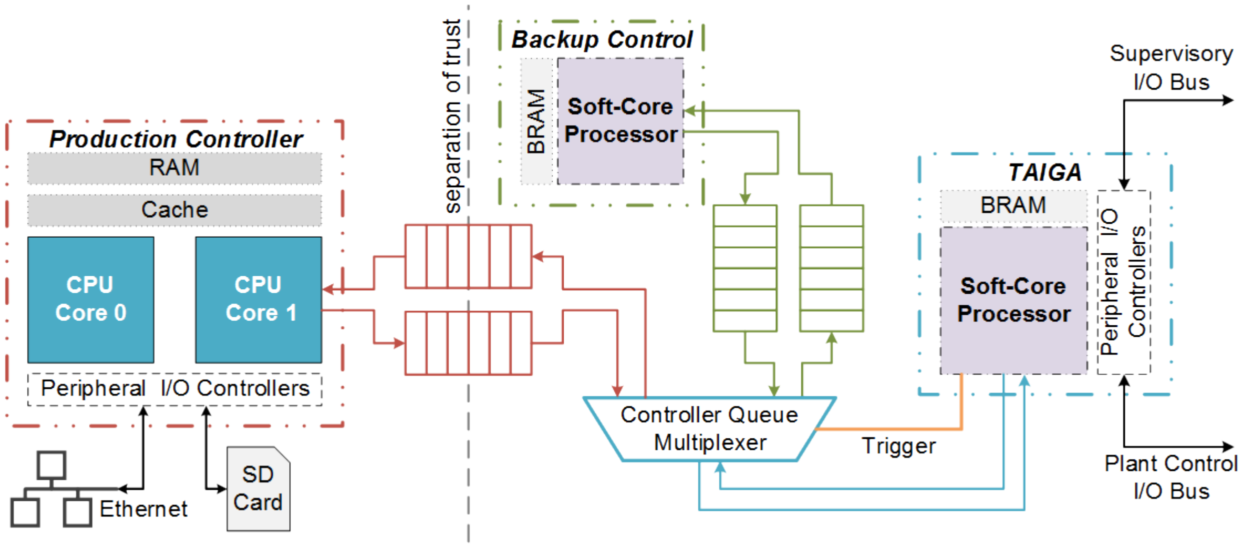

Figure 5.1: TAIGA implementation on a configurable SoC

Figure 5.1 illustrates the TAIGA architecture realized on a configurable SoC. TAIGA enhances ICS security by segregating the trusted components in configurable hardware, restricting the inter-module communication through hardware implemented I/O queues and ensuring invisibility of different modules to each other [21].

5.1 Trust Requirements in TAIGA

TAIGA components meet the following trust requirements (TRs) [22]:

TR1 The source code and implementation for the entire component are analyzed.

TR2 The component uses private hardware resources for computation, internal communication, and memory, and does not invoke external components as sub-functions.

TR3 All external communication with untrusted components is through hardware-implemented, bounded, and isolated queues.

TR4 The component cannot be bypassed or disabled, and has a fixed repertoire of essential services, such as I/O or cryptography.

TR5 Critical functionalities of the component, such as rule checking logic, cannot be updated without provably secure or physical access.

TAIGA establishes a high level of trust in its components matched only by a Trusted Platform Model (TPM), typically used as a secure cryptoprocessor [23].

5.2 Architecture Overview

TAIGA, represented in Fig. 5.1, is entirely invisible to the existing process and the control system. The IOI module has 2 main functions:

1. It monitors the communication between the supervisory controller and the RTU in the production controller. All the network traffic between these two nodes is directed through the supervisory I/O bus allowing the IOI module to monitor and detect any attack through the supervisory control system. Communication via the supervisory bus is bidirectional, to receive supervisory set-point commands and to report process critical parameters to human-machine interfaces in the supervisory layer.

2. The IOI module hosts the trigger mechanism, which monitors the plant behavior based on the sensory feedback received from the physical process. Monitoring the bidirectional communication on the plant control I/O bus allows TAIGA to detect attacks originated from infrastructure layer.

In order to ensure stability of the process, a reliable and accurate actuation of the process is necessary. TAIGA incorporates a backup controller in its architecture which ignores the supervisory commands and actuate the system towards a pre-defined safe condition.

The IOI module communicates with both the production and the backup controller through restrictive message queues. Both read and write channels for the production and backup controllers are FIFO queues implemented directly in the programmable logic. FIFO queues ensure a client/server relation between the IOI module and the controller preventing untrusted components from re-configuring parameters of the trusted modules.

The architecture also incorporates a bi-directional controller multiplexer to select the active plant controller based on the monitored data from the supervisory and plant I/O buses. The controller multiplexer is implemented directly in the programmable logic and it operates based on the trigger input from the IOI module for controller selection.

5.3 Control Strategy

Figure 5.2 elaborates on the security measures in TAIGA. The objective of the production controller is to keep the pendulum in a balanced condition at a supervisor-specified servo position. The backup controller takes over the control of the pendulum whenever a malware attack is detected. Since the backup controller is not susceptible to reconfiguration, safety of the physical plant can be assured. A number of possible trigger mechanisms to switch from the production to the backup controller are considered to assess their ability to keep the specification guards from being violated in case of an attack.

The proposed TAIGA framework constitutes following modules:

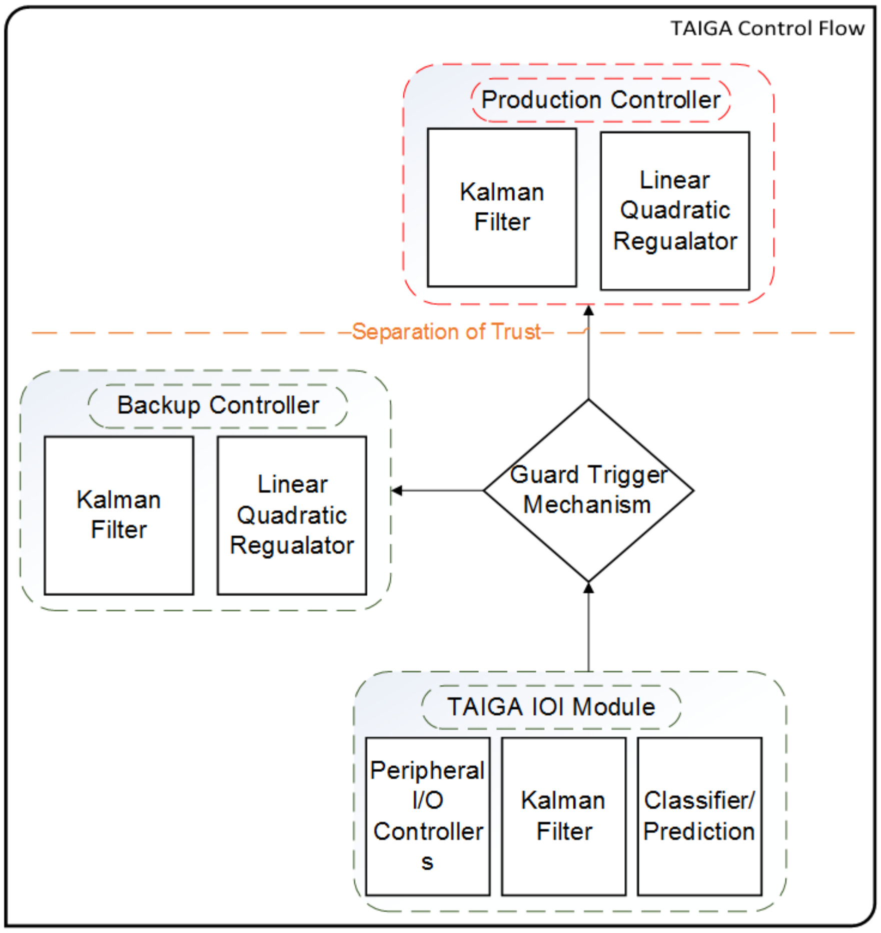

Figure 5.2: Separation of Trust within TAIGA

5.3.1 Production Controller

The production controller is responsible for controlling one or more processes and is tuned to meet high-performance demands. The production controller receives its operational set-point from the supervisory controller and sensory values from the physical plant via the IOI module. The Kalman filter in the production controller computes the velocity states and then a full state feedback control law is applied to the physical plant to ensure pendulum balance at the commanded set-point. A dedicated Ethernet connection for network updates makes it susceptible to reconfiguration attacks. As a result, we consider it an untrusted component in TAIGA framework.

5.3.2 Backup Controller

We envision the backup controller as a legacy version of the production controller, thoroughly verified for correct operation, that can act upon the plant in the event that the production controller fails or is compromised by malware. The backup controller's integrity can be assured as it is not accessible through the network, does not share any resources with the production controller, and requires physical access for updates.

The backup controller operates in the same manner as the production controller except that it ignores the supervisory command and ensure the pendulum balance at a predefined set-point. As long as the production controller's output is maintaining the pendulum in a balanced condition, without violating any of the specification guards on the position states, the backup controller's outputs are ignored by the IOI module. However, if the production controller fails, the backup controller supersedes it and ensures sustained pendulum balance.

To ensure a smooth transition from the production to the backup controller, the backup controller requests for the current state vector from the TAIGA intermediary module before switching. This ensures that the backup controller is initialized at the current state values, avoiding any transient during the switch-over from the production controller.

5.3.3 Specification Guards

The specification guards are of two types:

- *Operational guards* are typically operating limits for the sensor measurements and control signals.
- *Safety guards* ensure that the process remains stable and protects the plant from safety hazards. In case of a safety guard violation, defensive mechanisms, such as relief valves and auxiliary power for electro-mechanical systems, bring the process back to safety.

5.3.4 I/O Intermediary Module

The I/O intermediary module hosts the trigger mechanism, which is responsible for maintaining the process operation within specified guard bands even in the presence of malicious software. The production controller and the backup controller receive all of the sensory responses and supervisory control commands only after they are vetted by the IOI Module.

A key part of the IOI module is a bidirectional controller multiplexer which is synthesized from C directly into the programmable logic using High-Level Synthesis (HLS). This ensures transparency between the IOI module and the actuating controller. The controller is oblivious to the fact that its response is verified by the trigger mechanism while the trigger mechanism is unaware of which controller is governing the plant. Thus, the trigger mechanism treats a read or write request from either of the controllers in exactly the same way. Separation of trust within different components reduces the susceptibility of TAIGA to malicious attacks.

Trigger mechanisms ensure a swift transition to the backup controller to prevent a violation of the specification guards. The trigger mechanism is equipped with an instance of Kalman filter to estimate velocity states from the sensed positions. This ensures that the state estimates are not corrupted by the malicious software. We consider three different trigger mechanisms in this paper:

Trivial Trigger

A *trivial* trigger compares the measured sensor values against the operational guards and switches to the backup controller if a guard is violated. Its principle of operation makes it reactive rather than preemptive. On account of its reactive nature, switching from the production to the backup controller with a trivial trigger necessarily results in operational guard violation and may result in safety guard violations depending upon the attack. Hence, in our control strategy, we use trivial triggers as the last line of defense in case of failure of other trigger mechanisms to detect threats.

Prediction Trigger

A *prediction* trigger uses a linearized plant model and a copy of the controller to predict in real time, whether the controller is capable of safely rescuing the system from its current state without causing a guard violation. The online prediction can be done with either the production controller or the backup controller.

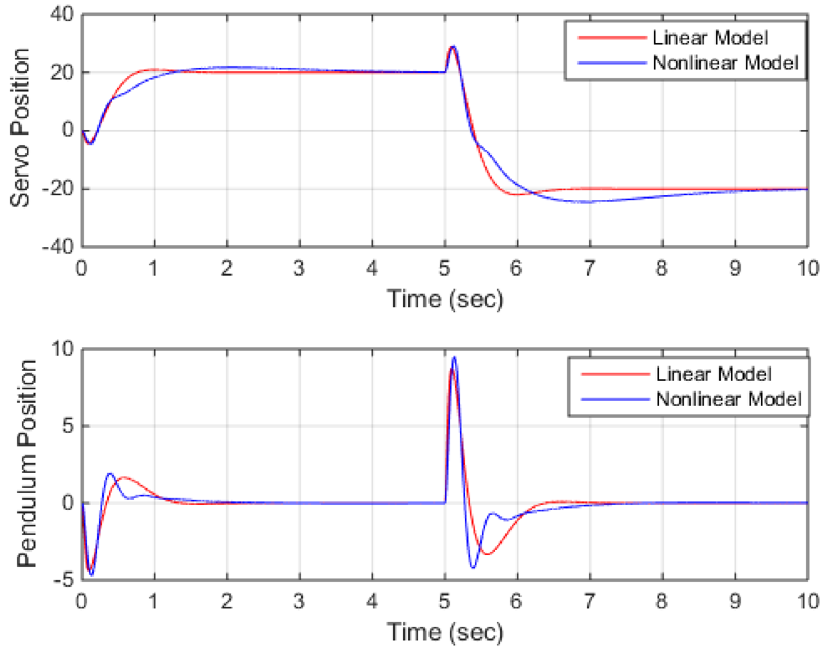

Figure 5.3: Comparison of Pendulum Linear and Nonlinear Model Response

- Using the production controller, the linear model runs in an accelerated loop using the current set-point value. The initial condition for the prediction is obtained by synchronizing the model state with the current pendulum state. The set-point command from the supervisory controller is accepted during synchronization and retained until an entire state trajectory is estimated. After completion of the estimation cycle, the pendulum states are again synced with the physical plant and the cycle continues.

- Using the backup controller, the linear model follows a similar synchronization routine except for the set-point synchronization. The backup controller operates using a pre-defined set-point and does the prediction to check the backup controller's capability to bring the pendulum to a safe state without guard violations.

The prediction trigger has two drawbacks:

1. The prediction trigger is based on a linearized process model which is an approximation of the actual process behavior. The linearized model cannot capture the exact transient and steady state behavior of the physical system as it does not account for the non-linearity in the process response. Figure 5.3 captures the pendulum states, θ and

α, during a transient response of the pendulum, for both the linear and non-linear models. As is easily visible from the plot, the nonlinear model exhibits more overshoot compared to the linear model for the same set-point. Thus, for some of the malicious conditions, the prediction trigger using a linear model might categorize a particular state as a good state, while the physical plant with the same set-point and initial condition might violate the guards.

2. Predicting in real time requires significant computational resources, limiting the trigger's capability to do the entire prediction in one controller cycle. Thus, the entire trajectory prediction might require multiple scheduler instances, affecting the speed of the response of the trigger in detecting the attack.

In the case of an inverted pendulum, the impact of the nonlinearity is more on the steady-state than the transient response as discussed in Chapter 3. Hence a prediction trigger using the production controller proves to be a reasonable technique in certain situations.

Classification Trigger

A *classifier* trigger classifies any specified system state into either a safe state, meaning that if needed the backup controller can bring the system to a safe state without violating the operational guards, or a danger state, meaning that even if the backup controller is activated immediately, the operational guards will be reached or violated.

Since the purpose of the trigger mechanism is to decide when to switch from the production to the backup controller, it is natural to view it as a classification problem: given the current state should we switch or not. We investigated three different classifiers to make an intelligent decision in real time, based on the current sensor measurements.

The data required for offline tuning of classifiers is collected by running the non-linear model of the pendulum for different sets of initial conditions. The backup control law is applied to the model of the plant with a set-point $\theta = 0°$ for different sets of initial conditions corresponding to the current state: servo position (θ), pendulum position (α), servo velocity ($\dot{\theta}$), and pendulum velocity ($\dot{\alpha}$). The transient response of the model is monitored for guard violations. A guard is set on θ and α restricting them to a specified range. An algorithm is developed to monitor the plant states and check if transitioning to the backup controller at that instant can be done without violating a guard. A guard violation is set as 0 while a non-violation is set as 1.

- *Neural Network*: A generic linear model fitting analysis of the experimental data indi-

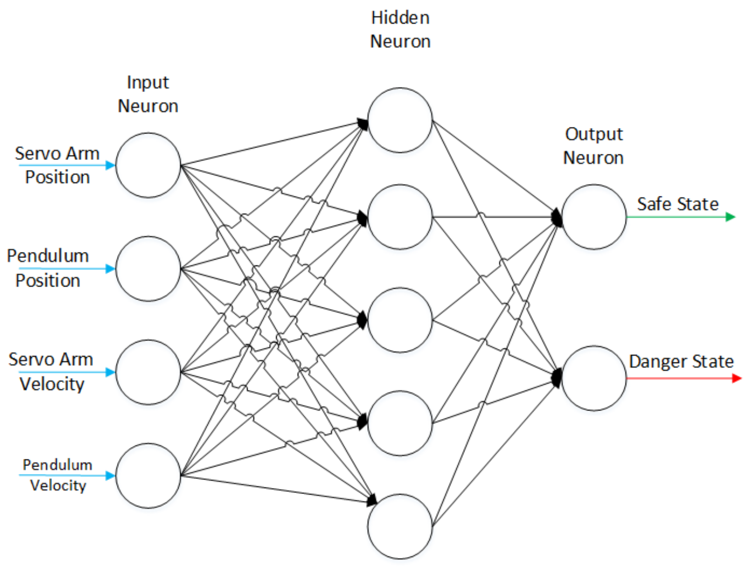

Figure 5.4: Multi-Layer Perceptron Model for Classification

cates that a linear model cannot be fit using just 4 basis vectors. Thus we will use a multilayer perceptron (MLP), a feed-forward artificial neural network model that maps a set of inputs to outputs using a nonlinear activation function. The back-propagation technique is used to calculate the gradient of the loss function with respect to all the weights of the neural network followed by an optimization method to reduce the loss function [24].

Figure 5.4 captures the neural network. It consists of 4 input neurons for each state of physical pendulum, followed by a hidden layer which maps inputs to the propagation function. An activation function is used at both the hidden neurons and the output neurons, to map the propagation function to the corresponding outputs. The classifier has 1 output with 1 for a good state and 0 for a bad state.

Using the training data set, x_i, and targets, y_i, the MLP is trained with the Levenberg-Marquardt optimization technique, which employs a non-linear least squares method. The algorithm finds weights for the network, β, to minimize the mean square error given by Eq. 5.1.

$$S(\beta) = \frac{1}{m}\sum_{i=1}^{m}[y_i - f(x_i, \beta)]^2 \tag{5.1}$$

An hyperbolic tangent function is used as the activation function to map the weighed input vectors to the outputs.

$$\hat{y}_i = \frac{e^{\sum x_i\beta_i} - e^{-\sum x_i\beta_i}}{e^{\sum x_i\beta_i} + e^{-\sum x_i\beta_i}} \tag{5.2}$$

- *Support Vector Machines:*

  A support vector machine (SVM) classifies data by finding the best hyperplane that separates all data points of one class from that of the other class. The SVM classifier is trained using sequential minimal optimization method to identify support vectors s_i, weights w_i and bias b that are used to classify vectors x according to the following equation:

  $$y_i(k(w_i, x_i) + b) \geq 1 \tag{5.3}$$

  For the linear Kernel function considered in this thesis, k is the dot product of data vector x and the weights w_i. Support vectors s_i are the data vectors x_i for which $y_i(< w_i, x_i > +b) = 1$ The optimal solution of the separating hyperplanes for the training data set $x_i \in R^d$ If $c \geq 0$ then x is classified as a member of the first group, otherwise it is considered as part of the second group [25].

  The data for training is a set of points $x_i \in R^d$, along with their categories $y_i \pm 1$. The equation for separating hyper-planes is

  $$y = \text{sign}(w_i.s_i + b) \tag{5.4}$$

- *Logistic Regression:*

  Logistic regression is a technique for describing how an output variable is functionally related to one or more predictor or regressor variable [26]. Logistic regression can be used to deal with multinomial, as well as binomial response variables. Nurunnabi et. al. discuss the use of logistic regression for the classification of an output variable for binomial response [27].

  The logistic regression can be considered as a special case of generalized linear modeling. The generalized linear model in the logistic regression is of the form

  $$f(y) = Xb \tag{5.5}$$

where,

- y is the response variable with the binomial distribution
- X is the training data set of input vectors
- b is a coefficient vector which defines a linear combination Xb of the predictors X
- f is a linking function that links the output variable to the linear combination Xb. For logistic regression, the linking function is called *logit(y)* or $log[\frac{y}{(1-y)}]$.

Therefore,

$$log[\frac{y}{(1-y)}] = Xb \tag{5.6}$$

Solving the equation for y,

$$y = \tfrac{e^{(Xb)}}{1+e^{(Xb)}} \tag{5.7}$$

$$= \tfrac{1}{1+e^{-(Xb)}} \tag{5.8}$$

Because of the form of the equation, the response variable is limited between 0 to 1. Negative values of linear combination Xb results in y assuming values close to 0 while positive values results in values close to 1. Unlike support vector machine and neural network, logistic regression output assumes values in the 0-1 range and in order to classify them we use 0.5 as the threshold. $y < 0.5$ is classified as bad state while $y > 0.5$ is a good state.

Chapter 6

Implementation of Pendulum Balance Control in TAIGA

6.1 Hardware Overview

The Quanser SRV02 (rotary inverted pendulum) contains a flat arm with a pivot at one end and a metal shaft at the other. The pivot end is mounted on top of SRV02 load gear shaft and fastened with screws. The actual pendulum link is fastened onto the metal shaft. The result is a horizontally rotating arm with a pendulum end. Figure 6.1 shows the pendulum in a balanced condition.

A DC motor is used to actuate the pendulum and two encoders are used for servo and pendulum position feedback. A single-channel linear voltage amplifier, Quanser VoltPAQ-X1 drives the electro-mechanical pendulum system. A hardware DAC is implemented to interface the linear voltage amplifier to the ZedBoard such that the data acquisition and control can be conducted with digital signals. An SPI bus is used to communicate with the encoder/DAC data acquisition interface.

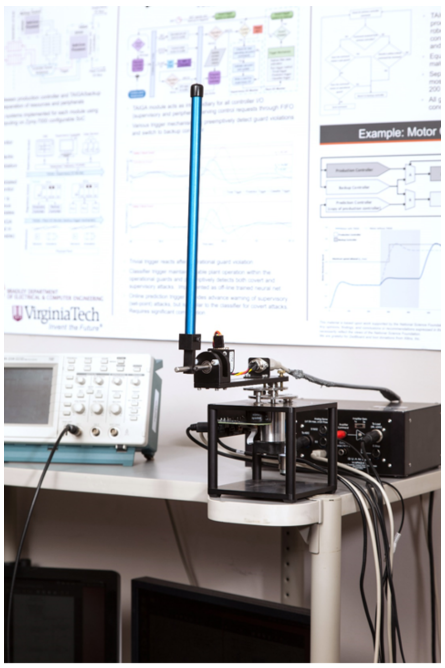

Figure 6.1: Rotary inverted pendulum in balanced condition

6.2 Software Overview

TAIGA is implemented on a Xilinx Zynq-7000 programmable SoC. The Zynq-7000 contains a dual-core ARM processor as well as configurable FPGA fabric all in the same chip. The FPGA fabric on Zynq can host multiple instances of Microblaze soft processors with independent resources such as Block RAM and I/O channels.

6.2.1 Production Controller

The production controller represents the functionality of a PLC and an RTU in an ICS architecture. It is implemented on an embedded processor (Zynq PS). It consists of two CPU cores, an Ethernet controller, and external storage. The primary functionality of a PLC is to communicate with sensors and actuators in the regulatory layer for process control. The RTU is responsible for telemetry with the supervisory layer to report the current process status and receive supervisory commands. In this implementation, both the responsibilities are handled by a single core of ARM processor hosting FreeRTOS. Linux, hosted on the other core, acts as a reconfiguration node for software updates. To report the process status to the supervisory layer, FreeRTOS writes the state history in the shared memory blocks between the two cores. Linux uses this information to print it to the user interface.

The second core hosting Linux uses the Ethernet Controller for software updates to the production controller. Updates to the production controller are of two types:

- *Controls* updates for optimizing the control algorithm such as altering the control parameters or restructuring of control sequence to optimize the process response.
- *Firmware* updates modify the procedural aspects such as service routines.

Linux interacts with FreeRTOS using shared memory to update the controller code parameters to improve the performance of the existing control algorithm. The same channel can be used by an adversary to introduce malicious software in to the production controller.

6.2.2 Backup Controller

The backup controller is instantiated in a MicroBlaze soft processor in the programmable logic. The backup controller follows the same control routine followed by the production controller. It utilizes a hardware timer module instantiated in the FPGA fabric to maintain a one millisecond control cycle. From the ICS perspective, the backup controller is invisible to the operator as it cannot be commanded through the supervisory layer.

6.2.3 I/O Intermediary module

The I/O intermediary (IOI) module is implemented as a second MicroBlaze processor in the programmable logic. It is responsible for monitoring communication on both, the supervisory and plant control bus. As discussed in TAIGA overview, FIFO queues instantiated in the configurable hardware are used for communication between the IOI module and the controller. IOI module ensures that the messages coming from the FIFO queue are transferred to controllers only if they follow a pre-defined syntax. Figure 6.2 describes the operation of IOI module.

The IOI module is responsible for initializing the physical process and the position sensors (encoders) to their default value before the normal operation. The encoders used for

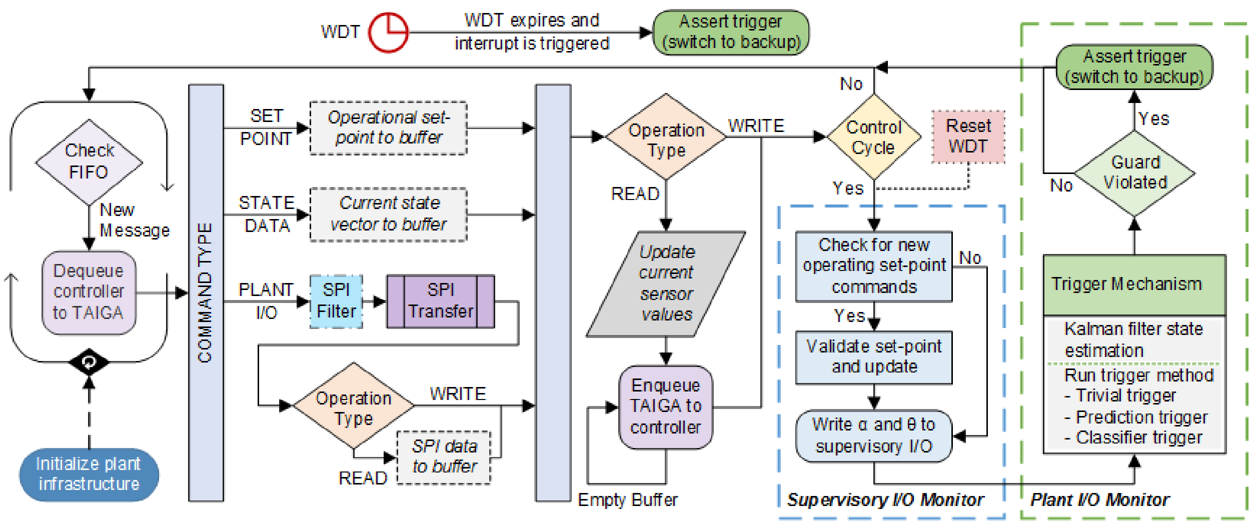
WDT
WDT expires and interrupt is triggered
Assert trigger (switch to backup)
Check FIFO
New Message
Dequeue controller to TAIGA
Initialize plant infrastructure
COMMAND TYPE
SET POINT
Operational set-point to buffer
STATE DATA
Current state vector to buffer
PLANT I/O
SPI Filter
SPI Transfer
Operation Type
WRITE
READ
SPI data to buffer
Operation Type
WRITE
READ
Update current sensor values
Enqueue TAIGA to controller
Empty Buffer
No
Control Cycle
Yes
Reset WDT
Check for new operating set-point commands
No
Yes
Validate set-point and update
Write α and θ to supervisory I/O
Supervisory I/O Monitor
Assert trigger (switch to backup)
Yes
Guard Violated
No
Trigger Mechanism
Kalman filter state estimation
Run trigger method
- Trivial trigger
- Prediction trigger
- Classifier trigger
Plant I/O Monitor

Figure 6.2: Software implementation and flow of TAIGA's IOI module

position measurement are relative encoders and an initialization before normal operation ensures that they are initialized to a good value. Performing this routine in the production controller endangers pendulum stability as a malicious update of the production controller can request an initialization routine midway through the normal operation. This could lead to hazardous results as the relative encoders could be initialized to bad values affecting the controls performance.

A command filter ensures that both the controllers are provided limited privileges in that they are allowed to execute only a limited set of commands. They can request encoder values or write voltage commands to the servo through the SPI bus. Any other requests, such as re-initialization of encoders, would be filtered out and not processed. The IOI module command filter processes following set of commands.

- A *set-point* command provides the value of current set-point for servo position.
- A *state-data* command returns the estimated state-values based on the current sensor measurement provided by the Kalman filter hosted in IOI module.
- A *Plant I/O* command interacts with the physical process by reading sensor values and writing the actuator commands via the SPI bus.

In each scheduler cycle, the production controller requests the operational set-point command from the supervisory controller, reads the pendulum and servo position and write a voltage command to actuate the servo. The backup controller operates on a pre-defined set-point and requests only measurement data every millisecond. The IOI module is responsible for serving all I/O requests from both controllers. It checks the supervisory bus for a new operating set-point and updates the operational set-point if it does not violate specification guard. The IOI module also reads the plant control bus for sensor values and uses the Kalman filter instantiated in the trigger mechanism to compute the state vector requested by the backup controller. A control sequence is defined in each scheduler cycle to ensure smooth operation.

1. A read request for the pendulum encoder position, α.
2. A read request for the servo encoder position, θ.
3. A write request for servo motor actuation.

The IOI module is also responsible for limiting the servo motor actuation command from both actuating controllers to the actuator limits. Any commanded control voltage above the limit is limited to the actuator limits by the IOI module before applying it to the servo motor.

To ensure invisibility of the TAIGA module to the controller, it is necessary to execute the TAIGA operations only after the control sequence is completed. Hence, the control sequence is the highest priority task followed by monitoring of both the supervisory and the plant I/O buses. A supervisory set-point command is accepted only after completion of the control sequence execution to ensure synchronization of control law with the sensor measurements within each scheduler cycle.

A watchdog timer (WDT), is instantiated in the programmable logic to monitor the execution of the control sequence. The WDT has a calibration parameter, set at a slightly higher value than the scheduler time. The timer is reset after every successful execution of a control sequence. A WDT is critical in TAIGA to protect the physical plant against Denial of service (DoS) attacks. A denial of service attack could be executed in few different ways by the intruder and the stealthiness of the attack would depend on the criticality of the denied service for ensuring pendulum balance. Any of the following attack scenarios could be selected by the adversary by performing a reconfiguration attack on the production controller.

- The production controller stops requesting set-point command and operates on a past command. This attack will not impact the plant stability but affect the performance.

- The production controller stops requesting the sensor values. The control command is executed based on the measured sensor values. In absence of the sensor readings the controller will hang in the loop affecting the system stability.

6.3 Control Logic

Figure 6.3 captures the control logic developed for balancing the inverted pendulum. The following control sequence is followed:

- The pendulum is initialized to its initial condition by the IOI module.
- The pendulum position and servo position are measured at every scheduled invocation of the TAIGA module, and a supervisory command from the supervisory controller is accepted to update the operational set-point.
- A classifier is built offline and instantiated in TAIGA to determine whether a switch to the backup controller at the current instant would violate a guard.
- The classifier outputs either a 0 or 1. For an output of 0, a trigger service routine provides the most recent measurements and estimated velocities to the backup controller.
- The production controller command is the governing controller for a classifier output of 1 while the backup controller is the governor for a 0.

6.4 Trigger Mechanism

The trigger mechanism is a part of the Plant I/O monitor process and is responsible for the pre-emptive switch to the backup controller in case of a malware attack. We have considered 2 different trigger mechanisms: an online prediction and a classifier.

6.4.1 Online Prediction

Online prediction uses an instance of either the production or the backup controller to do prediction based on the current sensor measurements. Since the controller is operating on

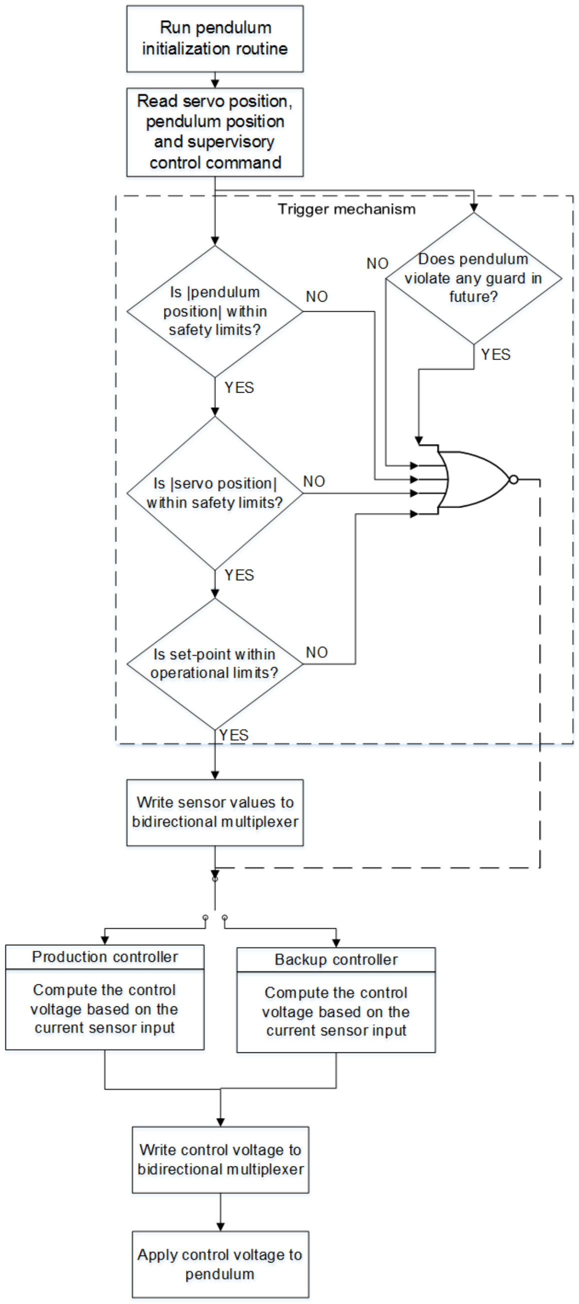
Run pendulum initialization routine
Read servo position, pendulum position and supervisory control command
Trigger mechanism
Is |pendulum position| within safety limits?
NO
YES
Does pendulum violate any guard in future?
NO
YES
Is |servo position| within safety limits?
NO
YES
Is set-point within operational limits?
NO
YES
Write sensor values to bidirectional multiplexer
Production controller
Compute the control voltage based on the current sensor input
Backup controller
Compute the control voltage based on the current sensor input
Write control voltage to bidirectional multiplexer
Apply control voltage to pendulum

Figure 6.3: Control strategy for inverted pendulum balance control

a 1 millisecond schedule, it allows a maximum window of about 500 microseconds to do prediction. In the available window the closed loop with the controller and the discretized version of the linear model of the physical plant, can perform about 60 iterations. The settling time of pendulum is about 1.2 seconds. Since we are interested in only the transient response to look for a guard violation, we set a prediction window of 900 milliseconds. With 60 iteration per scheduler cycle, the prediction requires 15 scheduler cycles to complete the prediction. Thus with a prediction trigger, TAIGA can preemptively respond about 885 milliseconds in advance to protect the pendulum from guard violations. At the end of every prediction cycle, the prediction states sync with the physical plant states before initiating the next prediction cycle.

The performance of the prediction trigger depends upon the selection of a controller for prediction. Online prediction with the production controller at times, can detect a malicious attack earlier than the classifier trigger. However, a malicious update to the production controller could lead to the prediction trigger not responding to the malicious threats. Prediction with the backup controller is secured against malicious updates but responds slower to the attack compared to the classification trigger due to the computational overhead.

6.4.2 Classifiers

We have investigated 3 classification techniques.

Neural Network

The neural net is tuned using using the Matlab function *train*. The available data set is divided into 3 sub data sets for tuning, testing and validating the designed classifier. The accuracy of the neural net depends on the number of data-points and number of neurons in the net. Adding neurons increases the computational time for the network. Thus, a trade-off needs to be made between execution time and accuracy. In this approach, the neural net is trained with approx. 50000 data points for 5 and 10 hidden-layer neurons. The results of the training are listed in Table 6.1.

Table 6.1: Classification Results

Classification Technique	Parameters	Accuracy	Execution Time (mS)
Neural Net	5	97.2	2.448
Neural Net	10	99.1	2.621
Logistic Regression	14	95.6321	2.156
Logistic Regression	31	95.7038	2.492
Support Vector Machine	14	86.69	0.908
Support Vector Machine	31	87.91	1.127

Linear Regression Model

The accuracy of classification using logistic regression or a support vector machine depends on how closely the guard violations are related to the state vectors. A classifier tuned with either of these techniques using just the states as the basis vectors leads to a poor classification. Thus 2^{nd} and 3^{rd} order terms with combination of the physical plant states were used for classification.

Adding more states results in more computation, hence a statistical analysis was performed to check the dependence of the guard violations on all basis vectors. This analysis was performed by fitting a generic linear model with the available data [28]. The basis vectors that have significant impact on the output are retained while the others are ignored.

$$Z_k = H_k\theta + V_k \tag{6.1}$$

where,

- Z_k is the measurement vector or the parameter which decides guard violation (θ_{max} during transient response).
- H_k is the observation vector or the coefficients of the basis vector that leads to a particular measurement.
- θ is the unknown parameter vector or the basis vector.
- V_k is the measurement noise vector or the constant value affecting the measurement.

In our analysis we use the generic linear model for function approximation, that is given a set of data $(x_1, f(x_1))$,$(x_2, f(x_2))$ .... $(x_N, f(x_N))$, find an approximate function

$$f(x) \approx \hat{f}(x) = \sum_{j=1}^{n} \theta_j \phi_j(x) \tag{6.2}$$

where $\phi_j(x)$ are a set of n pre-specified basis functions.

The analysis is performed for 3 sets of values: only 1^{st} order terms, 1^{st} and 2^{nd} order terms which includes squares of all the state variables (for ex. α^2) and cross combinations of state variables (for ex. $\alpha\theta$, $\dot{\theta}\dot{\alpha}$), and 1^{st}, 2^{nd} and 3^{rd} order terms, which includes cubes (for ex. θ^3) and combinations (for ex. $\theta\dot{\theta}^2$, $\theta\alpha\dot{\theta}$) of all state variables. Assuming a normal distribution in the measurements for the maximum value of θ during a transient response, a generic model is fitted to the available data. Our analysis shows that with 4 and 14 basis vectors all the terms are statistically significant in estimation of θ_{max}. When the same analysis was performed with 34 basis vectors 2 terms, $\alpha\dot{\theta}$ and $\theta^2\dot{\theta}$ had high p-values making them statistically insignificant. Hence, these two basis vectors were removed from the generic linear model. Figure 6.4 captures the performance of the models for models with 4,14 and 31 basis vectors.

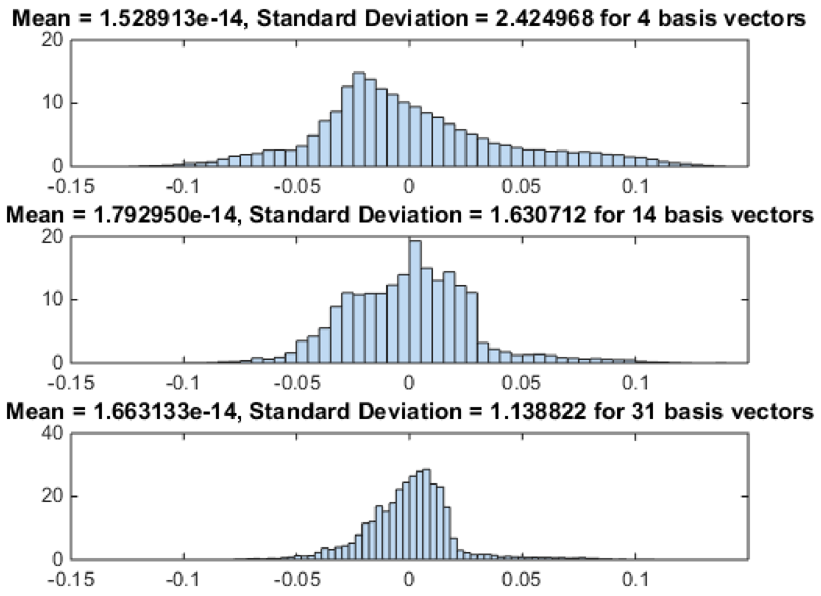

Figure 6.4: Residual Plot for generic linear model fitting

Logistic Regression

The logistic regression classifier is tuned with the collected data and basis vectors discussed in previous section. The classification is performed for 2 different sets of basis vectors, 14 and 32 basis vectors respectively. Matlab function *mnrfit* is used to perform logistic regression. The matlab funtion provides coefficients for the basis vectors. The *logit* function maps the linear combination of predictor variables and their coefficients to the estimated outputs. As can be seen from the Table 6.1, adding basis vectors result in significant increases in computation with a minor increase in accuracy.

Support Vector Machine

The support vector machine technique uses the same basis vectors as logistic regression and performs a linear mapping between data inputs and outputs. The matlab function *svmtrain* is used to train the classifier while the matlab function *svmclassify* is used to assess the accuracy of the trained classifier. As seen from the Table 6.1, the SVM is the quickest algorithm of the three at the cost of classification accuracy. Increasing the basis vectors results in only a minor improvement in accuracy, making any attempts to add 4th order terms seem futile.

Based on these results, a decision was made to use a neural network with 5 neurons as it provides significant accuracy with reasonable computational time. The computational times included in the table are based on executions of low-level C code in Matlab. Running them under real-time operating systems is significantly faster.

Chapter 7

Preemptive Detection of Simulated Cyber Attacks

A rotary inverted pendulum balance control experiment is used to demonstrate the effectiveness of TAIGA. The rotary arm is connected to a servo and its angle, θ, increases positively when rotated counter-clockwise. The pendulum is connected to the end of the rotary arm. In its upright position, the inverted pendulum angle, α, is zero and increases positively in the counter-clockwise direction. Chapter 4 discussed the inverted pendulum balance control experiment. This chapter focuses on how TAIGA ensures the pendulum balance in case of network and reconfiguration attack scenarios. We discusses different attack scenarios and how trigger mechanisms instantiated in TAIGA protect the ICS from such attacks.

7.1 Eavesdropping Attack

A simple eavesdropping attack in the TAIGA framework is possible only at a supervisory level as all the communication within TAIGA is either wired or happen through hardware implemented FIFO queues, making them impossible to intercept. Any eavesdropping attack at the supervisory level can be used to gather information about the process, however, it will not affect system stability in the absence of disruptive resources.

A covert attack as a combination of reconfiguration along with eavesdropping could result in the production controller transmitting the plant states on its reconfiguration channel. This could allow an adversary to get all the information about the process such as sensor values, velocity states, supervisory commands and control voltages to improve the stealthiness of future attacks. Making the reconfiguration channel unidirectional in future work could protect the system against this attack.

7.2 Deception Attack

The deception attack corrupts the control signals u_k and sensor measurements y_k. In TAIGA framework, deception attacks are not possible as both sensors read and control voltage write instructions are handled by the IOI module and are processed through FIFO queues.

However, a combination of a deception and reconfiguration attack could result in the production controller writing corrupted control voltages to the physical plant. Another deception attack scenario would be a malicious update to the production controller reading the sensor values, but ignoring them for the controls calculation. In the absence of sensory feedback, the controller will run open loop with potentially disastrous results. In the absence of disclosure resources and system knowledge, the stealthiness of either of these two attack scenarios is minimal. A corrupted control command will affect all the states of the physical plant. A classifier tuned with the nonlinear model of the plant will detect such an attack and ensure process stability by switching back to the backup controller.

7.3 Replay Attack

In a replay attack, the adversary first performs an eavesdropping attack to gather a sequence of data and then begins replaying the recorded data at a later time until the end of the attack. As discussed above, the adversary cannot utilize disruption resources without performing a reconfiguration attack. So a replay attack is possible only with reconfiguration privileges.

We assume that the physical plant is secure and trust the sensor values coming from the process. Since all the I/Os are handled by the IOI module, a replay attack on sensors is not possible. A replay attack coupled with reconfiguration could result in the production controller writing old control signals. Such an attack will be detected by the trigger mechanism and a switch to the backup controller will ensure process stability.

7.4 Bias Injection Attack

A bias injection attack is a covert version of a deception attack. It is executed when the adversary has developed system knowledge by performing an eavesdropping attack. The attacker's goal in such a situation is to inject a constant bias into the system without being detected. The attack is executed in open loop and does not require any disclosure resources.

The TAIGA framework is developed to ensure process stability and will not result in any corrective action until the added bias voltage has an impact on plant stability. It will result in a switch to the backup controller only if the intrusion detection schemes predict a future guard violation due to added bias.

7.5 Denial of Service Attack

A denial of service attack stops the sensor and/or actuator commands from reaching their respective destinations. A covert DoS attack coupled with a reconfiguration attack prevents the production controller from transmitting control commands on the FIFO queue. The Watchdog timer implemented in IOI module times out if a pre-defined control sequence does not execute. If the WDT is exhausted, it creates an interrupt which enables the backup controller to take control of the physical plant preventing any guard violations.

7.6 Supervisory Attack

Figure 7.1 demonstrates the performance of all three trigger mechanisms in the presence of a supervisory attack. Initially the pendulum is balanced in the upright position at $\theta = 10°$. The operational guard for θ is set at $\theta = 35°$, while the safety critical guard is set at $\theta = 50°$. The operational guard for α is set at $\alpha = 15°$. To simulate the attack, the supervisory controller commands a malicious set-point change to $\theta = 35°$ at time $T_{attack} = 60$ sec.

The trivial trigger does not detect the malicious behavior until the operational guard is violated. The backup controller is triggered at $T_t = 60.36$ sec when $\theta_t = 35°$. The transient induced as the backup controller attempts to drive the servo to $\theta = 0°$ causes the system to violate the safety critical guard on θ while staying within the guard on α.

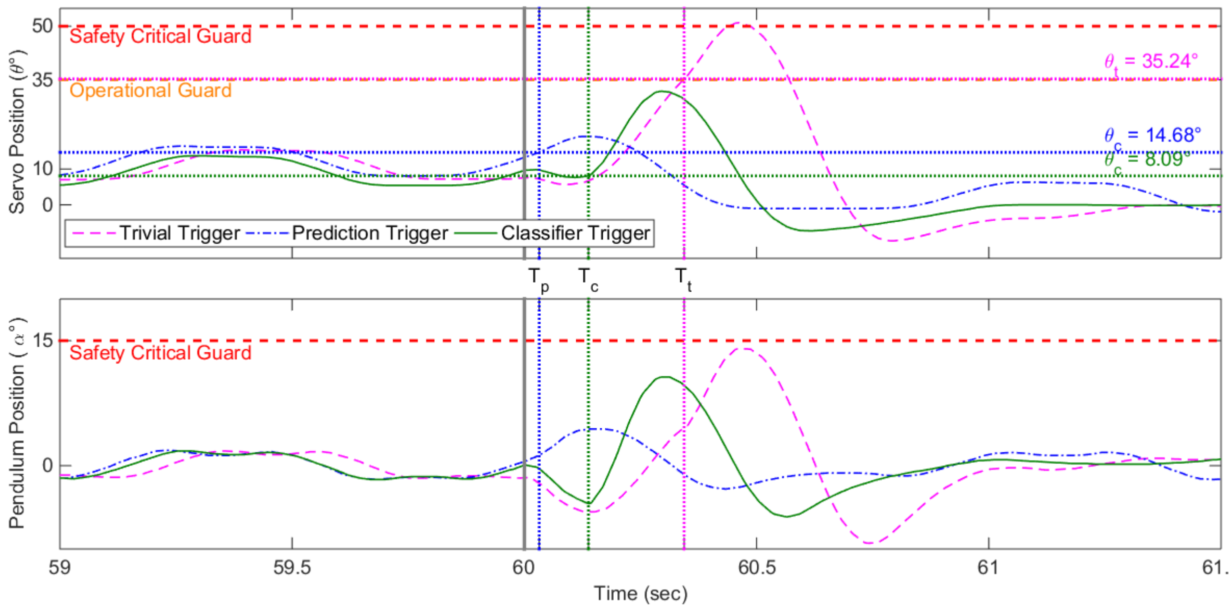

Figure 7.1: Plant response to a simulated supervisory attack at time $t = 60$ sec, and a trigger mechanism to switch plant control to a backup controller

The classifier trigger, however, detects the malicious behavior at $T_c = 60.1138$ sec when $\theta = 8.09°$ and $\dot{\theta} = 103.1°/\text{sec}$. A transient is again generated as the backup controller attempts to bring the plant to a safe position, but there are no violations of the operational guards.

A prediction trigger doing real time prediction with the production controller and a commanded set-point of $\theta = 35°$ is the first to detect such an attack. The prediction trigger detects the attack at $T_p = 60.032$ sec when $\theta = 14.68°$. If this attack gets coupled with a reconfiguration attack, the attack may change the controls code of production controller, limiting the trigger mechanism's capability for attack detection. A prediction trigger with the backup controller will predict whether the backup controller is capable of rescuing the physical plant from its current state without a guard violation. However, due to the limitations of linear model in capturing the physical plant response and the computational overhead associated with prediction, the attack detection will be slightly delayed in comparison with the classifier trigger and may result in a guard violation.

7.7 Covert Attack

Covert attacks are typically stealthier versions of the previously discussed attacks, created by combining multiple attacks, making it hard for the control system to detect them preemptively. In a covert reconfiguration attack, the production controller itself can drive the system to an unsafe position in the absence of any supervisory set-point change. For example, assume the malicious production controller moves the pendulum at a constant servo velocity towards the operational guard while holding the pendulum position and velocity constant ($\alpha = \dot{\alpha} = 0$).

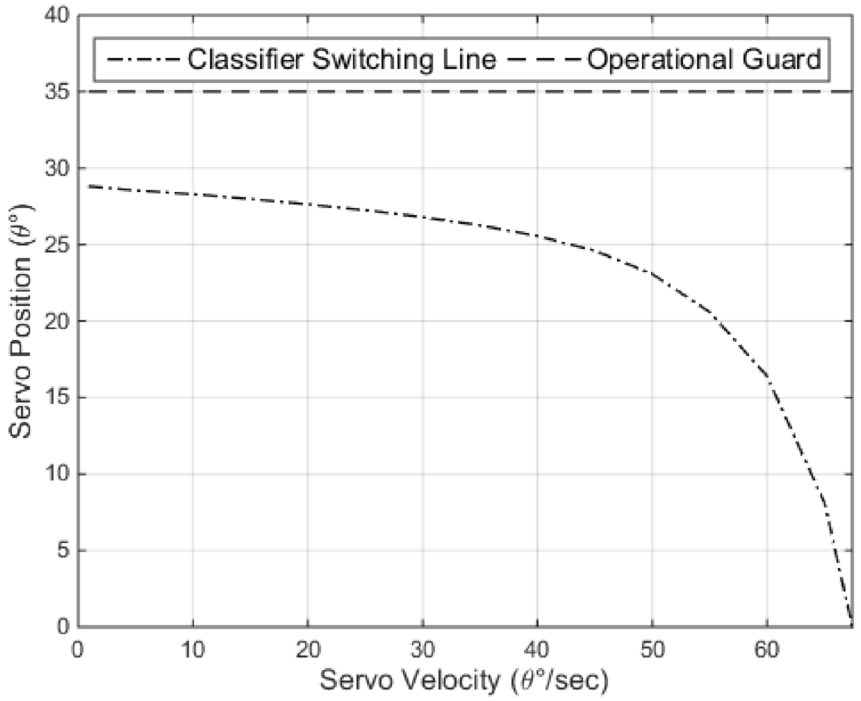

Figure 7.2: Switching line for a constant $\dot{\theta}$ and zero $\dot{\alpha}$

The trivial trigger has the same problem as before since it reacts only after the physical plant violates the guard. The prediction trigger using the compromised production controller might not detect the attack preemptively resulting in a guard violation. The prediction trigger with the backup controller continues to predict what the backup controller will do using the current plant state as its initial condition for prediction. Assuming infinitely fast prediction and close proximity of the linear model response with the nonlinear, it will trigger a switch at exactly the same time as the classifier trigger. In reality, slight divergence of the linear model from the nonlinear and the computational overhead of prediction will cause a switch after the classifier has switched, causing a slight violation of the operational guard. The classifier trigger requires very little computation (since the prediction work has been done offline) and will trigger the switch as the system crosses the boundary between a safe return and a guard violation.

Figure 7.2 shows this boundary, the switching surface, for the constant servo velocity case. If the servo velocity is low, switching to the backup controller can be done closer to the operational guard. As the servo velocity increases, the switching point for the safe return of the system shifts away from the guard.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

A comprehensive rather than a case specific solution will go a long way in protecting ICSes against emerging cyber attacks. Such a solution requires a trusted platform for implementing intrusion detection schemes and safety preserving controllers and thorough understanding of the process behavior to identify any discrepancies in the response and a mechanism to quickly react to the threat. Our proposed solution, using TAIGA framework for meeting trust requirements and a neural network classifier based intrusion detection scheme can protect ICSes against both, reconfiguration and network security attacks.

Existing approaches to run-time fault detection use redundancy, with either a cold or hot standby controller used for comparing present outputs or an approximate linear prediction model used to predict the future behavior based on the past and present outputs. However, all of these solutions trust the underlying software, which could be compromised in a cyber attack.

Assuming the physical plant security, the attacks on ICS can originate from either the supervisory layer or the regulatory layer. The attack can be in terms of a compromised network channel providing falsified information to a controller or a malicious update to the controller affecting the controller performance. Our solution to this problem is a two-fold approach.

The supervisory node of an ICS is connected to internet and provides a channel for real-time monitoring of process operation through HMIs and networked channels. Compromising the supervisory layer allows an adversary to utilize disruptive and disclosure resources to attack the physical plant. The supervisory I/O monitoring node of TAIGA monitors all supervisory commands to differentiate between normal and malicious operating commands and protect the physical plant from operating on malicious commands.

The second level of defense in TAIGA is the trigger mechanism. The trigger mechanism monitors the behavior of the plant and detects stealthier supervisory and reconfiguration attacks. Online prediction with the production controller is the fastest detection technique in case of supervisory attack. However, it fails in the presence of a reconfiguration attack. Also, ability of the production controller to safely rescue the physical plant does not necessarily mean that the backup controller can rescue the physical plant. Online prediction with the backup controller is protected against malicious updates but results in a slower response as it does not predict using the supervisory set-point command. A classifier trigger is a faster and more accurate approach for preemptive detection of malicious threats. Since it is trained with a nonlinear model of the plant, its response matches closely that of the physical plant. Furthermore, the classifier is tuned offline and hence requires significantly less computation compared to the prediction trigger and can execute in every scheduled instance. The neural network based classifier was selected based on a trade-off between computational time and prediction accuracy. However, other techniques such as logistic regression or support vector machine can also be used for classification depending on the computational resources and the need for accuracy of a particular application.

The backup controller is based on an LQG optimal control law and ensures pendulum balance after guard trigger. The backup controller and the trigger mechanism are instantiated in the hardware and cannot be reconfigured through the Ethernet channel. They are trusted components in TAIGA and safeguard the rotary inverted pendulum even when the supervisory network or the production controller is compromised.

Our approach to security does not improve perimeter security but rather acts as the last line of defense to protect the physical system from attacks by malicious internal software and external commands transmitted through the network. Thus, the intrusion detection schemes in TAIGA works along with conventional perimeter defenses to make systems more resilient to intruder attacks.

8.2 Discussion

Current work is based on thorough understanding of the physical plant behavior in terms of a state-space based plant model for both, controls development and intrusion detection schemes. However, LQG control law is sensitive to parametric uncertainties and modeling inaccuracies could result in significant performance degradation. Since the control gains are static constants obtained offline, the current technique does not offer ways to tune the system in real-time. Hence, most industrial control systems use either, variants of proportional-integral-derivative (PID) controllers or, adaptive controllers that can adapt to the changes in process parameters to meet the performance specifications.

A PID controller can be used effectively in our TAIGA framework. We have envisioned the backup controller as a legacy version of the production controller. Therefore, both the production and the backup controllers can be PID controllers tuned with different gain coefficients. The IOI module will keep a track of internal states for both integral and derivative states and initialize the backup controller with them for smoother transition. In case of an adaptive controller, the production controller can have self-updating capabilities, however, to allow any updates to the backup controller, we need a mechanism to differentiate between genuine updates due to process parameters changes, and malicious updates due to cyber attacks. This limits use of adaptive controllers in TAIGA framework.

The TAIGA framework primarily attempts to improve security at each leaf node in an ICS. It can be effectively used if, the process loop controlled by that particular node can be modeled as a function of physical sensor inputs interfaced with the node. However, if a control action for the process loop also depends on the sensory data coming from another leaf node, an intruder can attack the communication channel between the two leaf nodes compromising TAIGA's ability to safeguard the process.

8.3 Future Work

Our focus in this work is to protect the ICS from network and reconfiguration attacks and ensure process stability. However, not all attacks on ICSes attempt to destabilize the process. Some of them might just attempt to degrade the process performance. Our intrusion detection schemes are currently not tuned for detection of such attacks.

Since the TAIGA IOI module is the primary interface to the process, we can assume that the communication between sensors and the IOI module is secure. However, the same cannot be said for the communication at the supervisory node. The configuration node for the production controller is an untrusted link and, so is the communication channel between TAIGA and the supervisory node. The link between TAIGA and a supervisory node can be protected to a certain extent by adding authentication and some sort of cryptography. However, the reconfiguration channel communicates directly with the production controller making it hard to implement any security measures within the scope of TAIGA. Thus, an adversary can compromise this channel to attack the current framework.

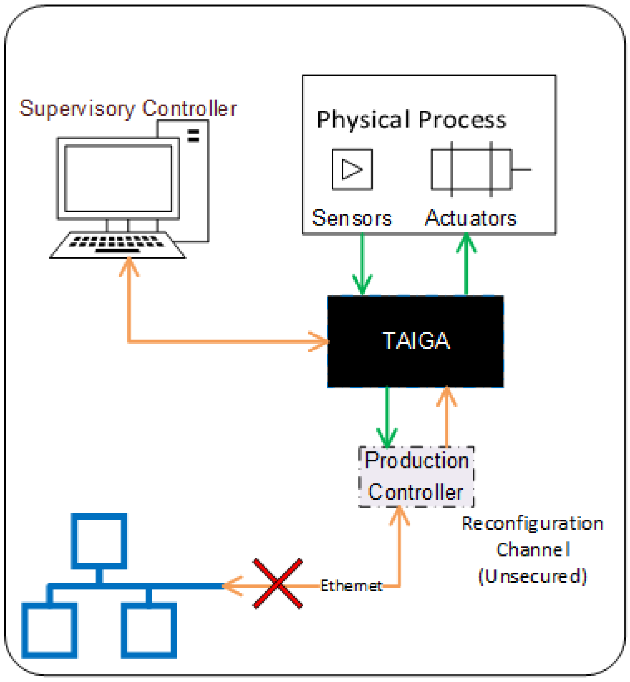

Figure 8.1: Limitations of TAIGA

In Figure 8.1 unsecured channels are marked in orange while secured connections are marked in green. Unsecured channels in TAIGA are the sources of attacks which attempt to affect the process performance. If the intruder manages to enter the communication channel between the supervisory node and TAIGA, then false data can be injected into the channel and reported to the plant operator. This class of attacks, executed with process performance degradation as the objective, can be further subdivided into two types:

1. *False Positive Attacks*: In this attack, the process is operating well within its limits,

however the adversary reports a false alarm to the operator showing that the performance is degraded. The natural reaction of the operator would be to start safety shut-down. A shut-down command overrides TAIGA and results in plant shut-down and loss of productivity.

2. *Bias Injection and Replay Attack*: In this attack, the adversary will add a bias voltage to the controller in such a way that it degrades the performance without compromising process stability. By getting into the reporting channel to the supervisory node, the adversary will perform a replay attack. Thus, the operator will be unaware of the degradation in the physical plant performance, resulting in the loss of revenues.

Thus, the future work can involve performance monitoring of the closed loop physical process and production controller in real time. If the degraded performance is worse than the backup controller, the control of the physical plant will be handed over to the backup controller.

In case of a supervisory attack, our current approach checks if the commanded set-point input violates operational guards. Our current specification guards are primitive and the trigger mechanisms are tuned for these guards. However, some of these specification guards can be made intuitive. For example, in our current work supervisory command is compared with the operational guards and passed to the physical plant only if it does not violate the guard. However, a classifier can be developed to check if, for a given supervisory set-point command and process states, the backup controller is able to bring the system to a safe state without violating the operational guards. If the backup controller classifies the state as good state, the set-point is considered as good, if not, the set-point command should be ignored and the backup controller should actuate the physical plant.

While TAIGA is implemented successfully on the inverted pendulum, the scaling to more complex ICSes needs to addressed. Future work will focus on the feasibility of the TAIGA framework for automotive systems. Koscher et. al. highlight the vulnerabilities of intra-Electronic Unit (ECU) CAN communication in automobiles and the potential risks associated with it. Potentially TAIGA can be used to mitigate such attacks by monitoring the CAN communication to detect abnormal behavior [29].

Bibliography

[1] J. Weiss, *Protecting Industrial Control Systems from Electronic Threats.* Momentum Press, 2010. [Online]. Available: http://books.google.com/books?id=0R0yngEACAAJ

[2] N. Falliere, "W32.stuxnet dossier," *in Cyberwar Resources Guide*, no. Item 108, 2011. [Online]. Available: http://www.projectcyw-d.org/resources/items/show/108

[3] A. A. Cárdenas, S. Amin, Z.-S. Lin, Y.-L. Huang, C.-Y. Huang, and S. Sastry, "Attacks against process control systems: Risk assessment, detection, and response," in *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, ser. ASIACCS '11. New York, NY, USA: ACM, 2011, pp. 355–366. [Online]. Available: http://doi.acm.org/10.1145/1966913.1966959

[4] A. A. Cárdenas, S. Amin, and S. Sastry, "Research challenges for the security of control systems," in *Proceedings of the 3rd Conference on Hot Topics in Security*, ser. HOTSEC'08. Berkeley, CA, USA: USENIX Association, 2008, pp. 6:1–6:6. [Online]. Available: http://dl.acm.org/citation.cfm?id=1496671.1496677

[5] A. Teixeira, D. Pérez, H. Sandberg, and K. H. Johansson, "Attack models and scenarios for networked control systems," in *Proceedings of the 1st International Conference on High Confidence Networked Systems*, ser. HiCoNS '12. New York, NY, USA: ACM, 2012, pp. 55–64. [Online]. Available: http://doi.acm.org/10.1145/2185505.2185515

[6] E. Bernabeu, J. Thorp, and V. Centeno, "Methodology for a security/dependability adaptive protection scheme based on data mining," *Power Delivery, IEEE Transactions on*, vol. 27, no. 1, pp. 104–111, Jan 2012.

[7] S. Amin, A. Cárdenas, and S. Sastry, "Safe and secure networked control systems under denial-of-service attacks," in *Hybrid Systems: Computation and Control*, ser. Lecture

Notes in Computer Science, R. Majumdar and P. Tabuada, Eds. Springer Berlin Heidelberg, 2009, vol. 5469, pp. 31–45.

[8] Y. Mo and B. Sinopoli, "Secure control against replay attacks," in *Proceedings of the 47th Annual Allerton Conference on Communication, Control, and Computing*, ser. Allerton'09. Piscataway, NJ, USA: IEEE Press, 2009, pp. 911–918. [Online]. Available: http://dl.acm.org/citation.cfm?id=1793974.1794131

[9] A. Teixeira, I. Shames, H. Sandberg, and K. Johansson, "Revealing stealthy attacks in control systems," in *Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on*, Oct 2012, pp. 1806–1813.

[10] Y. Mo, E. Garone, A. Casavola, and B. Sinopoli, "False data injection attacks against state estimation in wireless sensor networks," in *Decision and Control (CDC), 2010 49th IEEE Conference on*, Dec 2010, pp. 5967–5972.

[11] L. Lerner, Z. Franklin, W. Baumann, and C. Patterson, "Application-level autonomic hardware to predict and preempt software attacks on industrial control systems," in *Dependable Systems and Networks (DSN), 2014 44th Annual IEEE/IFIP International Conference on*, June 2014, pp. 136–147.

[12] L. W. Lerner, M. M. Farag, and C. D. Patterson, "Run-time prediction and preemption of configuration attacks on embedded process controllers," in *Proceedings of the First International Conference on Security of Internet of Things*, ser. SecurIT '12. New York, NY, USA: ACM, 2012, pp. 135–144. [Online]. Available: http://doi.acm.org/10.1145/2490428.2490447

[13] Z. Franklin, C. Patterson, L. Lerner, and R. Prado, "Isolating trust in an industrial control system-on-chip architecture," in *Resilient Control Systems (ISRCS), 2014 7th International Symposium on*, Aug 2014, pp. 1–6.

[14] M. Roman, E. Bobasu, and D. Sendrescu, "Modelling of the rotary inverted pendulum system," in *Automation, Quality and Testing, Robotics, 2008. AQTR 2008. IEEE International Conference on*, vol. 2, May 2008, pp. 141–146.

[15] R. Williams and D. Lawrence, *Linear State-Space Control Systems*. John Wiley & Sons, 2007. [Online]. Available: http://books.google.com/books?id=UPWAmAXQu1AC

[16] D. Luenberger, "An introduction to observers," *Automatic Control, IEEE Transactions on*, vol. 16, no. 6, pp. 596–602, Dec 1971.

[17] D. Simon, *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches.* Wiley-Interscience, 2006.

[18] R. C. Dorf and R. H. Bishop, *Modern Control Systems*, 9th ed. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2000.

[19] M. K. Campbell S, Crawford S, "Friction and the inverted pendulum stabilization problem," *Journal of Dynamic Systems, Measurement, and Control*, vol. 130, no. 5, 2008. [Online]. Available: http://dx.doi.org/10.1115/1.2957631

[20] C. J. Radcliffe and S. C. Southward, "A property of stick-slip friction models which promotes limit cycle generation," in *American Control Conference, 1990*, May 1990, pp. 1198–1205.

[21] N. Chiluvuri, O. Harshe, W. Baumann, and C. Patterson, "Using heterogeneous computing to implement a trust isolated architecture for cyber-physical control systems," in *Proceedings of the 2015 ACM Symposium on Information, Computer and Communications Security*, 2015.

[22] L. W. Lerner, "Trustworthy embedded computing for cyber-physical control," Ph.D. dissertation, Virginia Tech, 2015.

[23] *TPM Main Specification Level 2 Version 1.2, Revision 116 Part 1 Design Principles*, Trusted Computing Group, Incorporated, Mar 2011.

[24] M. Syiam, H. Klash, I. Mahmoud, and S. Haggag, "Hardware implementation of neural network on FPGA for accidents diagnosis of the multi-purpose research reactor of Egypt," in *Microelectronics, 2003. ICM 2003. Proceedings of the 15th International Conference on*, Dec 2003, pp. 326–329.

[25] K. Muller, S. Mika, G. Ratsch, K. Tsuda, and B. Scholkopf, "An introduction to kernel-based learning algorithms," *Neural Networks, IEEE Transactions on*, vol. 12, no. 2, pp. 181–201, Mar 2001.

[26] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics).* Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[27] A. Nurunnabi and G. West, "Outlier detection in logistic regression: A quest for reliable knowledge from predictive modeling and classification," in *Data Mining Workshops (ICDMW), 2012 IEEE 12th International Conference on*, Dec 2012, pp. 643–652.

[28] J. M. Mendel, *Lessons in estimation theory for signal processing, communications, and control.* Pearson Education, 1995.

[29] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage, "Experimental security analysis of a modern automobile," in *Security and Privacy (SP), 2010 IEEE Symposium on*, May 2010, pp. 447–462.

Appendix A

Controls Development and Validation

Contents

Build LQG controller for inverted pendulum

```
setup_rotpen; % initialization file provided by Quanser
```

```
Balance control gain:

K =

   -5.2612   28.1568   -2.7576    3.2190

Swing-up Parameters:
```

```
Er = 0.42 J
a_max = 11.5752 m/s^2
```

First, fix up mistakes in Quanser model

State Space Representation

```
Jt = Jr*Jp + Mp*(Lp/2)^2*Jr + Jp*Mp*Lr^2;
A = [0 0 1 0;
     0 0 0 1;
     0 Mp^2*(Lp/2)^2*Lr*g/Jt -Dr*(Jp+Mp*(Lp/2)^2)/Jt -Mp*(Lp/2)*Lr*Dp/Jt;
     0 Mp*g*(Lp/2)*(Jr+Mp*Lr^2)/Jt -Mp*(Lp/2)*Lr*Dr/Jt -Dp*(Jr+Mp*Lr^2)/Jt];

B = [0; 0; (Jp+Mp*(Lp/2)^2)/Jt; Mp*(Lp/2)*Lr/Jt];
C = eye(2,4);
D = zeros(2,1);

% Add actuator dynamics
B = Kg * kt * B / Rm;
%fixed Quanser errors.  Extra Kg*kt/Rm factor removed.  A(4,4) replaced by
%A(4,3)
A(3,3) = A(3,3) - Kg*kt*B(3);
A(4,3) = A(4,3) - Kg*kt*B(4);
```

LQR development

```
sys = ss(A,B,C,D); % continuous- time linear model of inverted pendulum
q = [6.3 0 0 0;0 6.3 0 0;0 0 0.0002 0; 0 0 0 0.0006]; % LQR tuning
r = 0.38;
n = 0;
K = lqr(sys,q,r,n)
```

```
K =

   -4.0717   21.2785   -2.5099    2.8804
```

Converting LQR poles from CT to DT

```
Tsamp=.001; % sampling time
dpoles=exp(1).^(eig(A-B*K)*Tsamp)  %convert desired poles to DT

%discretize continuous model
sysd=c2d(ss(A,B,C,D),Tsamp);
[Ad,Bd,Cd,Dd]=ssdata(sysd);

Kc=acker(Ad,Bd,dpoles) % DT full state feedback gain


dpoles =

   0.9536 + 0.0000i
   0.9880 + 0.0000i
   0.9953 + 0.0018i
   0.9953 - 0.0018i


Kc =

   -4.0275   21.1205   -2.4906    2.8588
```

Kalman Filter Development

```
G=eye(4);   % process noise matrix
```

```
H=zeros(2,4);  % process noise effect on measurements

RN = diag([(K_ENC/2)^2,(K_ENC/2)^2]); % measurement noise defined as half count

QN = eye(4)*1e-4; % process noise
QN(3,3) = QN(3,3)/5;
QN(4,4) = QN(4,4)/5;

% compute Kalman update gain
[~,L,P,M,Z]=kalman(ss(Ad,[Bd G],Cd,[Dd H],Tsamp),QN,RN);
Kf=M
```

```
Kf =

    0.9942   -0.0000
   -0.0000    0.9942
    0.0023   -0.0072
   -0.0077    0.2965
```

Control Law validation of Discrete Time LQR

```
[b1,a1] = ss2tf(Ad,Bd,Cd,Dd);
H_open_DT = tf(b1(1,:),a1,Tsamp);

Atemp = Ad;
Btemp = Bd;
Ctemp = Kc;
Dtemp = 0;

[b2,a2] = ss2tf(Atemp,Btemp,Ctemp,Dtemp)
H_comp_DT = tf(b2,a2,Tsamp)
```

```
figure(1)
p = bodeoptions; % to set frequency unit to Hz
p.FreqUnits = 'Hz';
w = (2*pi())*logspace(-0.5,3);
 bode(H_open_DT,w,p,'r')
hold on;
bode(H_comp_DT,w,p,'b');
legend('Pendulum','Pendulum and LQR')
grid on;

figure(2)
n = nyquistoptions; % to set frequency unit to Hz
n.FreqUnits = 'Hz';
% n.Grid = 'on';
n.Xlim = ([-4.5 0]);
n.Ylim = ([-20 20]);
nyquist(H_open_DT,n,'r');
hold on
nyquist(H_comp_DT,n,'b')
legend('Pendulum','Pendulum and LQR')
grid on;


b2 =

         0    0.0218   -0.0643    0.0631   -0.0206


a2 =

    1.0000   -3.9539    5.8617   -3.8616    0.9538
```

```
H_comp_DT =

  0.02184 z^3 - 0.06429 z^2 + 0.06306 z - 0.02062
  -----------------------------------------------
  z^4 - 3.954 z^3 + 5.862 z^2 - 3.862 z + 0.9538

Sample time: 0.001 seconds
Discrete-time transfer function.
```

open Loop Analysis for Frequency and Amplitude of limit cycle

```
Aop = [Ad zeros(4,4);Kf*Cd (Ad  +Bd*Kc -Kf*Cd)]; %Bd*Kc
Bop = [Bd;zeros(4,1)];
Cop = [zeros(1,4),Kc];
Dop = zeros(1,1);
[b,a] = ss2tf(Aop,Bop,Cop,Dop);
H_comp = tf(b,a,Tsamp);

figure(3)
p = bodeoptions; % to set frequency unit to Hz
p.FreqUnits = 'Hz';
w = (2*pi())*logspace(-0.5,3);
 bode(H_open_DT,w,p,'r')
hold on;
bode(H_comp_DT,w,p,'b');
bode(H_comp,w,p,'k');

legend('Pendulum','Pendulum and LQR','Pendulum and LQG')
grid on;

figure(4)
n = nyquistoptions; % to set frequency unit to Hz
```

```
n.FreqUnits = 'Hz';
% n.Grid = 'on';
n.Xlim = ([-4.5 0]);
n.Ylim = ([-20 20]);
nyquist(H_open_DT,n,'r');
hold on
nyquist(H_comp,n,'b')
legend('Pendulum','Pendulum and LQR')
grid on;
```

Appendix B

Classification Scripts

B.1 Neural Network

This code is for a trained classifier with 5 neurons in hidden layer

Contents

First Hidden Layer Calculation

```
H1 = [-0.4915 -1.5304 0.2741 -0.2712;-5.0208 7.4966 -0.0595 0.1440;1.7308 -2.9727 ...
    1.9770 -2.0926; -0.7712 -0.1702 -1.4878 1.4008;0.5753 0.6799 5.7288 -5.8389];
B1 = [3.6342;6.4743;-2.4650;-1.3560;-4.9857];
```

Second Hidden Layer Calculation

```
H2 = [-7.7722 11.7027 -10.1851 3.8138 -10.8677];
B2 = -6.2977;
```

Limits on Inputs

xmax and xmin define the limit of actual data which is scaled between -1 and 1 specified by matlab

```
xmin = [0.01745329; -0.26179938; -1.52359877; -0.82359877]
xmax = [0.52359877; 0.24434609; 1.50614548; -0.80614548];

ymax = 1;
ymin = -1;
```

```
xmin =

    0.0175
   -0.2618
   -1.5236
   -0.8236
```

Solving for a given input

```
tic;
x = [0.1571 -0.2618 -0.4887 -0.5236]; % input signal to classify
for i=1:1:4

y(i) = ((ymax - ymin)*(x(i) - xmin(i))/(xmax(i) - xmin(i)) + ymin);
end
```

```
for i = 1:1:5
    z(i) = H1(i,1)*y(1) + H1(i,2)*y(2) + H1(i,3)*y(3) + H1(i,4)*y(4) + B1(i);
    z_1(i) = 2/(1+exp(-2*z(i)))-1;
 end

p = H2(1)*z_1(1) + H2(2)*z_1(2) + H2(3)*z_1(3) + H2(4)*z_1(4) + H2(5)*z_1(5) + B2;
p_1 = 2/(1+exp(-2*p)) - 1;

out_max = 1;
out_min = 0;

final = ((out_max - out_min)*(p_1-ymin)/(ymax - ymin) + out_min) % result of classificat
toc;
```

```
final =

     1

Elapsed time is 0.000223 seconds.
```

B.2 Logistic Regression

Contents

logistic regression c code

This code is for a logistic regression based classifier with 14 basis vectors

fitted model for 14 basis vectors

```
 model_regression_offset = 2.8646;
input = [0.5235,0.2617,-0.43633,-1.3962]; % input signal to classify

model_regression_coeff = [-0.0078 1.3475 0.0530 -0.4278 -7.3287 -62.0800 ...
     -1.4356 -1.3978 28.2279 -1.7626 2.9192 2.9251 -7.9267 2.7756];

tic;
```

Defining the basis vectors

```
M1 = input(1);
M2 = input(2);
M3 = input(3);
M4 = input(4);
M5 = input(1)*input(1);
M6 = input(2)*input(2);
M7 = input(3)*input(3);
M8 = input(4)*input(4);
M9 = input(1)*input(2);
M10 = input(1)*input(3);
M11 = input(1)*input(4);
M12 = input(2)*input(3);
M13 = input(2)*input(4);
M14 = input(3)*input(4);

M = [M1 M2 M3 M4 M5 M6 M7 M8 M9 M10 M11 M12 M13 M14];

 temp = model_regression_offset;
for i=1:1:length(M)
    temp = temp + M(i)*model_regression_coeff(i);
end
final_ans = 1/(1+exp(-temp));
```

```
if final_ans > 0.5
    answer = 1;
else
    answer = 0;
end
toc;
```

```
Elapsed time is 0.012204 seconds.
```

B.3 Support Vector Machine

This code is for support vector machine based classifier with 14 basis vectors

Contents

- fitted model for 14 basis vectors
- Defining the basis vectors
- Classification to determin the expected output

fitted model for 14 basis vectors

```
 SVM_offset = 2.5130;
input = [0.5235,0.2617,-0.4363,-1.3962];

SVM_coeff = [-0.8777 2.9884 0.0280 -0.3870 -2.8116 -42.4812 -1.8951 -2.10003 ...
    14.7573 -2.4160 3.4318 4.7279 -8.9392 3.9347]; % tuned classifier coefficients

tic;
```

Defining the basis vectors

```
M1 = input(1);
M2 = input(2);
M3 = input(3);
M4 = input(4);
M5 = input(1)*input(1);
M6 = input(2)*input(2);
M7 = input(3)*input(3);
M8 = input(4)*input(4);
M9 = input(1)*input(2);
M10 = input(1)*input(3);
M11 = input(1)*input(4);
M12 = input(2)*input(3);
M13 = input(2)*input(4);
M14 = input(3)*input(4);

M = [M1 M2 M3 M4 M5 M6 M7 M8 M9 M10 M11 M12 M13 M14];
```

Classification to determine the expected output

```
 temp = SVM_offset;
for i=1:1:length(M)
    temp = temp + M(i)*SVM_coeff(i);
end

if temp > 0
    answer = 1;
else
    answer = 0;
end
toc;
```

```
Elapsed time is 0.025604 seconds.
```