Nelson Mandela Metropolitan University

for tomorrow

“WIRELESS INDUSTRIAL INTELLIGENT CONTROLLER FOR A NON-LINEAR SYSTEM”

Prepared by

JM Fernandes (Student No. 20616207)

Declaration

This Project Proposal is submitted in fulfilment of the requirements towards:

Master of Engineering: Mechatronics

NMMU Promoter: Prof. TI van Niekerk

Submission Date: 9/01/2015

AUTHORS DECLARATION

I, John Fernandes, hereby declare the work reported in this dissertation to be my own, and that the dissertation has not previously been submitted in full or partial fulfilment of the requirements for another qualification.

Author's Signature: . . . .

Date: . . . .

ACKNOWLEDGEMENT

I would firstly like to acknowledge my late father, Remedious Manuel Fernandes (1936 - 2014) whose constant love and watchful guidance pushed me beyond personal boundaries. I would also like to acknowledge my mother, Patricia Fernandes and my siblings Gabriel and Gladys for their encouragement and for always being a pillar of support especially in those times of weakness.

I would further like to acknowledge my dear friends Mr Oluwole Olufayo and Dr. Michael Ayomoh for all the assistance rendered in the formatting and proof reading of this document as well as my promoter, Prof. Theo van Niekerk for the moral and financial support given over the duration of the research. Without his guidance and direction over the years, I would not be where I am today.

Last but not least I would like to extend my deepest gratitude to my Lord and Saviour, Jesus Christ, without whom I am nothing and can do nothing. Dear Master, I give back to you what you gave to me in your goodness; use this for your Glory.

ABSTRACT

Modern neural network (NN) based control schemes have surmounted many of the limitations found in the traditional control approaches. Nevertheless, these modern control techniques have only recently been introduced for use on high-specification Programmable Logic Controllers (PLCs) and usually at a very high cost in terms of the required software and hardware. This ‘intelligent’ control in the sector of industrial automation, specifically on standard PLCs thus remains an area of study that is open to further research and development.

The research documented in this thesis examined the effectiveness of linear traditional control schemes such as Proportional Integral Derivative (PID), Lead and Lead-Lag control, in comparison to non-linear NN based control schemes when applied on a strongly non-linear platform. To this end, a mechatronic-type balancing system, namely, the Ball-on-Wheel (BOW) system was designed, constructed and modelled. Thereafter various traditional and intelligent controllers were implemented in order to control the system. The BOW platform may be taken to represent any single-input, single-output (SISO) non-linear system in use in the real world. The system makes use of current industrial technology including a standard PLC as the digital computational platform, a servo drive and wireless access for remote control.

The results gathered from the research revealed that NN based control schemes (i.e. Pure NN and NN-PID), although comparatively slower in response, have greater advantages over traditional controllers in that they are able to adapt to external system changes as well as system non-linearity through a process of learning. These controllers also reduce the guess work that is usually involved with the traditional control approaches where cumbersome modelling, linearization or manual tuning is required. Furthermore, the research showed that online-learning adaptive traditional controllers such as the NN-PID controller which maintains the best of both the intelligent and traditional controllers may be implemented easily and with minimum expense on standard PLCs.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

AC	-	Alternating Current
ANN	-	Artificial Neural Network
BP	-	Back Propagation
BOW	-	Ball on Wheel
CB	-	Communication Board
CP	-	Communications Processor
DB	-	Data Block
DC	-	Direct Current
ER-NN	-	Error Recurrent-Neural Network
FB	-	Function Block
FC	-	Function
HMI	-	Human Machine Interface
I/O	-	Input/ Output
LAN	-	Local Area Network
LM	-	Levenburg-Marqardt
LQR	-	Linear Quadratic Regulator
LS	-	Linear System
LTI	-	Linear Time Invariant
MANN	-	Multi-Applicable Neural Network
MB	-	Memory Byte
MC	-	Motor Control
MD	-	Memory Double
MIMO	-	Multiple Input-Multiple Output
MPC	-	Model Predictive Controller
MPI	-	Multi-Point Interface

MRAC	-	Model Reference Adaptive Control
MW	-	Memory Word
NN	-	Neural Network
NN-PID	-	Neural Network – Proportional-Integral-Derivative
OB	-	Organization Block
OLE	-	Object Linking and Embedding
OPC	-	OLE for Process Control
PD	-	Proportional-Derivative
PID	-	Proportional-Integral-Derivative
PI	-	Proportional-Integral
PLC	-	Programmable Logic Controller
PKW	-	Parameter Channel
PZD	-	Process Data
RMS	-	Root Mean Square
SCADA	-	Supervisory Control and Data Acquisition
SCL	-	Structured Code Language
SISO	-	Single Input-Single Output
SP	-	Set-point
SS	-	State Space
STL	-	Statement list
	-	

LIST OF FIGURES

LIST OF TABLES

Chapter 1.
INTRODUCTION

1.1 Control Systems Engineering

Control Systems Engineering is a growing field of study that has gained widespread attention over the past century. This has mostly been due to the outstanding advancements in technology that have taken place over a relatively short period of time. The advent of the microchip in particular, has enabled research in this field to progress further than ever before [1]. A control system is essentially an interconnection of components that forms a system configuration that produces the desired system response [2]. Control Systems Engineering analyses the individual components of a system and their interactions and effects on each other from input to output. It also involves the development of a strategy that suitably imposes complete or partial control over the system. The study of control systems is by no means limited to the field of engineering. As the dynamics of social, economic and political systems increase, so too will the ability to model and control these systems [2].

An automatic control system is a system that does not necessitate human intervention to function but operates through a compensation process that makes use of a controlling device, a sensing device for feedback and an actuating device in order to produce the desired output. Figure 1.1 depicts the fundamental building blocks of a typical feedback control system.

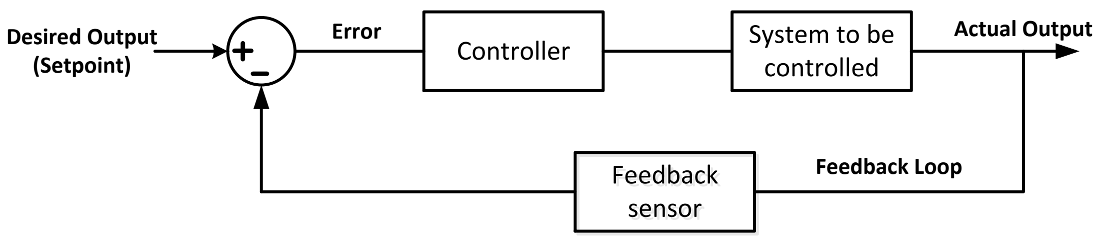


Figure 1.1: Feedback control system

All automatic systems, including those found in nature (for example biological systems), incorporate some kind of control system.

Today, automatic control systems are literally found everywhere; from common household systems such as the coffee machine in the kitchen to the elaborate climate and lighting systems found in most modern living rooms. They are also found in more complex systems such as aircraft, guided missiles, rockets, industrial robots, motor vehicles and many industrial processes [2, 3]. In the competitive world of manufacturing, advanced methods of control are mandatory. Without the aid of automatic control systems, it would be virtually impossible to meet the high demands of production whilst maintaining quality through precision manufacturing.

1.2 Traditional vs. Intelligent control systems

Most processes cannot be controlled by simple feedback systems. Such processes must be well understood before they can be controlled. Firstly, a mathematical function that adequately describes the process in question must be determined. Thereafter, a suitable control algorithm may be designed, simulated and implemented to control the process as desired. Many standard control algorithms and techniques have been devised, studied and tested in order to solve a variety of control problems. The Proportional-Integral-Derivative (PID) controller is commonly used because of its simplicity and excellent ability on linear systems [4]. Other popular control algorithms that have been successfully implemented for the control of linear systems include: Lead, Lag, Lead-Lag, Proportional-Derivative (PD), Proportional-Integral (PI) and Linear Quadratic Regulators (LQR). These controllers are often referred to as 'traditional' controllers because of their linear problem-solving approach.

In recent years, "intelligent" – neural network (NN) based controllers and/ or adaptive controllers have been used extensively in solving the more complex industrial control problems. The complexity found in such systems comes primarily from their non-linear nature. As a result, their solutions are often cumbersome and difficult to implement in the control environment. It may be worth noting at this stage that most industrial processes (and all chemical processes) are in fact non-linear [5]. If a traditional control approach is to be used, non-linear systems are treated as linear systems (through a process of linearization) within a limited operational range close to their equilibrium point/s. If pushed outside the 'linear' region of operation, stability may be lost. Traditional controllers also fall short when the specialized skills and tools required for accurate system modelling and controller design are lacking. Furthermore, because traditional controllers are unable to adapt to changes in their immediate environment, they often have to be re-tuned to compensate for the parameter variation that may naturally occur in a system. Failure to accurately re-tune these controllers could manifest in reduced plant efficiency, damage to equipment or worst of all, complete process and system failure.

Unlike the traditional control approaches, integrated intelligent control with its parallel (non-linear) computational ability has the benefit of being able to solve any non-linear problem through learning and can therefore adapt to an ever changing environment. This makes it desirable for use in the control of dynamic, non-linear systems [6].

1.3 Programmable Logic Controller’s

Programmable Logic Controller’s (PLCs) are the most widely used computer platform in the sphere of industrial automation. They are used to automate a wide range of processes, including many processes found in chemical plants, oil refineries and power stations just to name a few. Because of their wide range of application, PLCs are manufactured in different performance grades. Generally, the more capable a PLC is in terms of memory size, processing power and communication ability, the more costly it will be. Traditional control algorithms have successfully been implemented on PLCs in order to control dynamic processes. More recently, intelligent NN or fuzzy logic based control has been introduced to solve complex manufacturing and process control problems but still remains an expensive and unchartered area that is open to further research and development.

1.4 Aim of the research

The aim of this research is to compare the performance of linear traditional control schemes such as the conventional PID controller to intelligent non-linear NN based control schemes on a strongly non-linear mechatronic-type platform whilst subjected to various physical system parameter variations. The control schemes will be implemented on a standard, medium specification PLC with the option of wireless control and monitoring from a remote PC.

1.5 Objectives

The following objectives must be met in order to attain the goal of this research:

- Develop a non-linear platform upon which the research can be carried out. This system will be mechatronic in nature, having both electrical and mechanical aspects. It will be adjustable to allow for induced parameter variation and it will represent any similar non-linear system found in industry.
- Determine the mathematical model and hence the transfer-function of the designed non-linear platform.
- Linearize the non-linear system about its equilibrium point/s.
- Design and implement at least one traditional controller (e.g. PID) to control the non-linear system based on the linearized plant model.
- Design and implement at least one intelligent, NN based controller to control the non-linear system through a training process.
- Monitor, analyse and update control algorithms remotely over a wireless connection.
- Compare the performance of the traditional controller/s with that of the intelligent controller/s by varying plant parameters.

1.6 Hypothesis

The implementation of an intelligent, NN based controller on a non-linear system will both simplify (if not eliminate completely) the modelling and design processes found with traditional controllers and will also allow the system to adapt to parametric changes in the plant. The overall performance and operational range of the system will thus be increased. Such control can be achieved wirelessly using standard industrial control platforms such as the PC and PLC.

1.7 Delimitations

This research will be limited to the following:

- The development of a suitable non-linear platform upon which the research can be carried out.

- The design and implementation of at least one traditional controller on the designed platform.
- The design and implementation of at least one intelligent controller on the designed platform.
- A performance comparison between the above mentioned controllers.
- The incorporation of wireless access to the system for remote monitoring and control of the designed platform. Existing wireless technology will be used as a means of implementing the wireless control.
- The project is limited to a single PC and PLC only with all the necessary hardware/ software components.

1.8 Significance of the research

- The research platform may be used as a Control Systems teaching aid to demonstrate how intelligent algorithms can be used as an alternative means of solving complex control problems.
- The control concepts covered in this research will be applicable to any PLC controlled linear or non-linear processing system.
- The research highlights an intelligent NN based solution that may reduce process implementation costs as well as maintenance costs when used in a real-world application. The need for expensive equipment to accurately determine the physical plant parameters required for system modelling would be eliminated. Also, the need for specialized skills to compute plant models or re-tune existing controllers would be minimized.
- The integration of wireless control and monitoring would reduce costs further when used in a real-world application as much less physical wiring would be necessary. Fault finding and troubleshooting would be made easier. Wireless network infrastructure would also allow for greater flexibility and plant scalability.

- The research, where applied, could result in a significant increase in process efficiency, operability and accuracy due to the ability of the intelligent controllers to learn and adapt to changing environments. The trial and error approach found with traditional controllers would be eliminated.

1.9 Structure of the Thesis

The dissertation maintains the following structure:

Chapter 1: This chapter introduces Control Systems Engineering and its applicability in the world today. The scope of the research, the objectives, the hypothesis and the significance of the study are also covered in this section.

Chapter 2: This chapter reviews aspects of control theory that are relevant to the study including system modelling, linearization and controller design techniques. Existing studies pertaining to the control of non-linear systems using traditional and modern control approaches is also looked at. The study goes further to investigate the implementation of these controllers on PLCs with the option of remote control.

Chapter 3: In this chapter, the research platform (the Ball-on-Wheel (BOW) balancing system) is modelled and linearized. Traditional and intelligent controllers are then designed and simulated on the model in Matlab/ Simulink.

Chapter 4: This chapter looks at the design aspects of the BOW system from a mechanical, electrical and software point of view. It details the hardware components of the system in terms of their functionality, limitations and interaction. The chapter discusses the experimental setup and strategy in significant detail.

Chapter 5: In this chapter, the designed traditional and intelligent controllers are implemented and then analysed and compared in terms of system stability after the introduction of a controlled disturbance. Their ability to meet the design specifications even after plant parameters have been altered is also evaluated.

Chapter 6: In this chapter the research is concluded by examining the hypothesis in light of the obtained results. Recommendations to improve and hence expand the research are also made.

Chapter 2.
LITERATURE REVIEW

Owing to decades of research, there is a tremendous amount of literature available pertaining to traditional and intelligent control systems. However, this literature review focuses only on those aspects of automatic control that are relevant to the study. Apart from establishing fundamental theoretical control concepts and methodologies such as system modelling, system analysis and controller design, numerous works are categorically cited that explore various modern (intelligent) control techniques, particularly in their ability to overcome non-linear dynamical problems.

2.1 Introduction to control systems

Closed-loop control systems are also commonly referred to as feedback systems [3]. In feedback systems, the variable being controlled is measured by a sensor. The measured information is fed back to the controller to influence the controlled variable. One of the simplest examples of feedback control is the household furnace which is controlled by a thermostat. Figure 2.1 depicts this temperature control system in the form of a block diagram [7].

The room temperature can be influenced by external factors, e.g. a window or door being opened. In control systems this external influence is referred to as a disturbance. The aim of any controller is to reject external disturbances and bring the systems output back to the desired set-point as soon as possible.

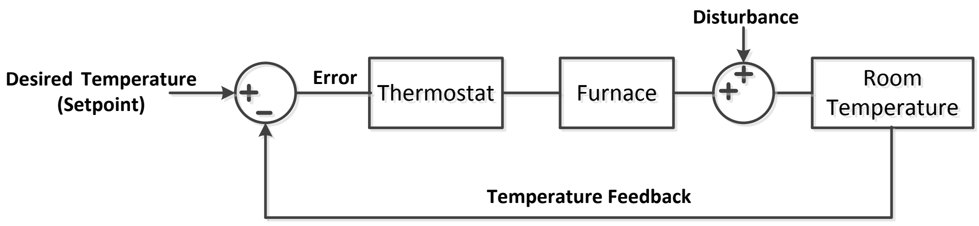


Figure 2.1: Room temperature control system

Feedback control can be extended beyond this basic system to incorporate multiple feedback loops, controlling devices and actuators.

There are two main types of feedback control systems, namely: negative feedback and positive feedback systems. In positive feedback systems, the set-point and output values are added. In negative feedback control the set-point and output values are subtracted. Negative feedback systems have been proven to be more stable than positive feedback systems [8].

The main purpose of feedback control is to compare the set-point to the actual output. The difference is referred to as the system error. It is then the job of the controller (i.e. the thermostat in Figure 2.1) to adjust the output in such a way that the error is minimized as far as possible. In real systems it is often impossible to eliminate the error completely. However, below a certain threshold defined by the control application, the error can be considered small enough to be ignored [3].

Consider the block diagram of an automobile cruise control system as shown in Figure 2.2:

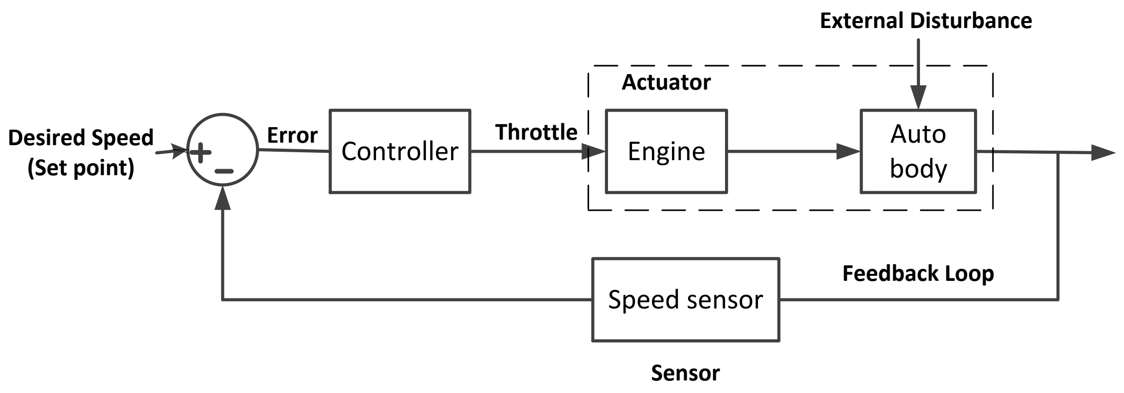


Figure 2.2: Block diagram of automobile cruise control

The combination of the process and actuator is referred to as the plant (highlighted by the broken square in Figure 2.2). The component that computes the desired control signal is the controller. Key external disturbances in the automobile cruise control application would include wind resistance, road surface texture, and road incline.

2.2 Dynamic system modelling

The purpose of deriving a system model is to obtain a mathematical description of the system being observed [7]. The determined model then assists in simulating the systems response to a set of pre-defined inputs. System modelling seeks to understand the interaction of system components from input to output. The process of system modelling is also sometimes referred to as system identification.

The fundamental step in building a mathematical model for a particular system is to determine its dynamic equations. These equations, once defined, can be expressed in the state variable form to enhance the ability to analyse the system. In many real-world cases the modelling of complex processes is difficult and expensive and often requires specialized skills. Once a model has been formulated, advanced engineering tools such as Matlab and Simulink are available for further analysis, simulation and design. Testing a system under varying conditions may prove to be very difficult in the real world due to the cost and difficulty of implementation. However, in the simulation environment, there is no limit to the extent of testing that is possible. In essence, a system can be simulated and tested even before it is built. No mathematical model will ever be an exact representation of a real system due to the lack of precise knowledge and the need to make assumptions but will usually suffice for the study of a specific systems response within a wide operating range. The essential equations required in system modelling are the continuity equations of mass, momentum and energy as well as other basic physical rules [3].

Physical systems can be purely electrical or purely mechanical systems, however, most are found to be of an electro-mechanical nature.

2.2.1 Mechanical Systems

Motion can either be rotary, translational or a combination of both. The translational motion of a mechanical system is characterized by a set of energetically interacting components. The interaction between these components depends on the applied forces and their reactions [9].

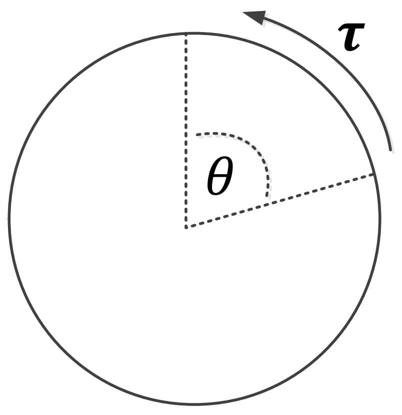


Figure 2.3: Properties of rotational motion

Referring to Figure 2.3, rotational motion about a fixed axis is governed by a set of torques and angular velocities. Torque is essentially defined as the moment of a force about a point [8] and is proportional to the angular acceleration. The relationship is made clear by Eq. 2.1.

$$\alpha(t) = \frac{\tau(t)}{J} \tag{2.1}$$

Where α is the angular acceleration, τ is the applied torque and J is the mass moment of inertia of the rotating body. The angular acceleration can be integrated to find the angular velocity which in turn can be integrated to find the angular displacement. The mass moment of inertia determines an object's resistance to acceleration.

In translational motion, Newton's law of motion (Eq. 2.2) forms the basis for obtaining a mathematical model for any mechanical system.

$$F = ma \tag{2.2}$$

Where "F" is the applied force (N), "m" is the mass (Kg) on which the force is applied and "a" is the resultant acceleration (m/s^2) on the mass [7].

Application of this law usually involves defining coordinates to account for the body's motion. For this reason, a "free-body" diagram assists in visualizing the system in its entirety.

Physical systems generally vary in the complexity of their dynamics and depending on the nature of the complexity, usually require very specific treatment. Fortunately, countless years of research into control theory have produced standard methods of dealing with most types of control problems.

Example 1: Linear Spring-Damper system

Consider a simple, mass, spring and damper system, seated upon a pair of frictionless wheels that has been fixed to a wall via a spring and damper, with a force applied in the direction of the displacement, 'x'.

Mass-Spring-Damper System:

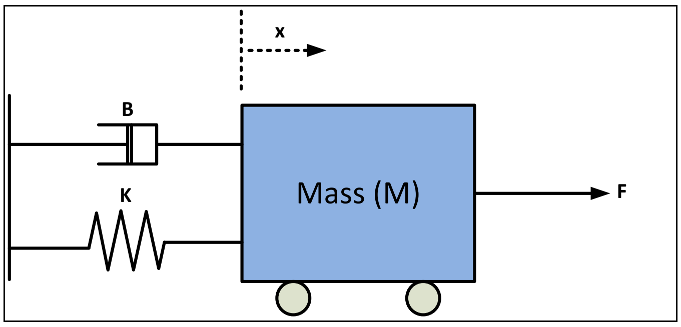


Figure 2.4: Mass, spring and damper system

Free-body diagram of mass, spring and damper system:

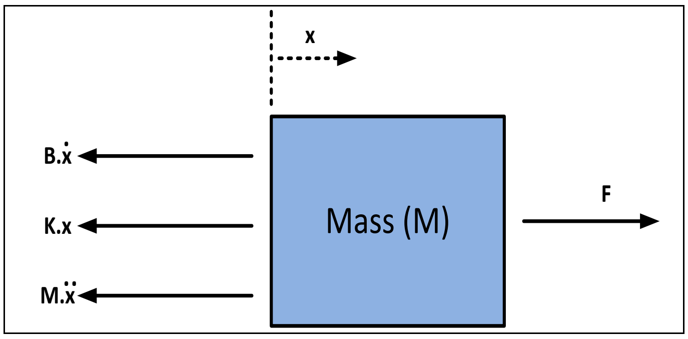


Figure 2.5: Free-body diagram of Mass, spring and damper system

From the free-body diagram, the system equations can be deduced by equating the acting forces as follows:

$$F = M\ddot{x} + B\dot{x} + Kx \qquad (2.3)$$

Rearranging in terms of the highest order term (acceleration) yields:

$$\ddot{x} = \frac{F}{M} - \frac{B}{M}\dot{x} - \frac{K}{M}x \tag{2.4}$$

This may be expressed as:

$$\ddot{x} + 2\varepsilon\omega_n\dot{x} + {\omega_n}^2 x = {\omega_n}^2 F' \tag{2.5}$$

$$F' = \frac{F}{K} \tag{2.6}$$

Where the natural, un-damped frequency of the system is given by:

$$\omega_n = \sqrt{\frac{K}{M}} \tag{2.7}$$

And the damping coefficient:

$$\varepsilon = \frac{\gamma}{2M\omega_n} = \frac{\gamma}{2}\sqrt{\frac{1}{KM}} \tag{2.8}$$

The actual form of the response will depend on the input force F. If F is 0, then Eq. 2.5 describes the behaviour of the system if released from a position ‘x’ away from its natural equilibrium condition.

Taking the Laplace transform of Eq. 2.5 we get:

$$(s^2 + 2\varepsilon\omega_n s + {\omega_n}^2) = {\omega_n}^2 F'(s) \tag{2.9}$$

This yields the input-output transfer function:

$$\frac{X(s)}{F\prime(s)} = \frac{{\omega_n}^2}{s^2+2\varepsilon\omega_n s + {\omega_n}^2} \tag{2.10}$$

If M = 1kg and K = 1 N/m, then the time response to a step input would be as follows:

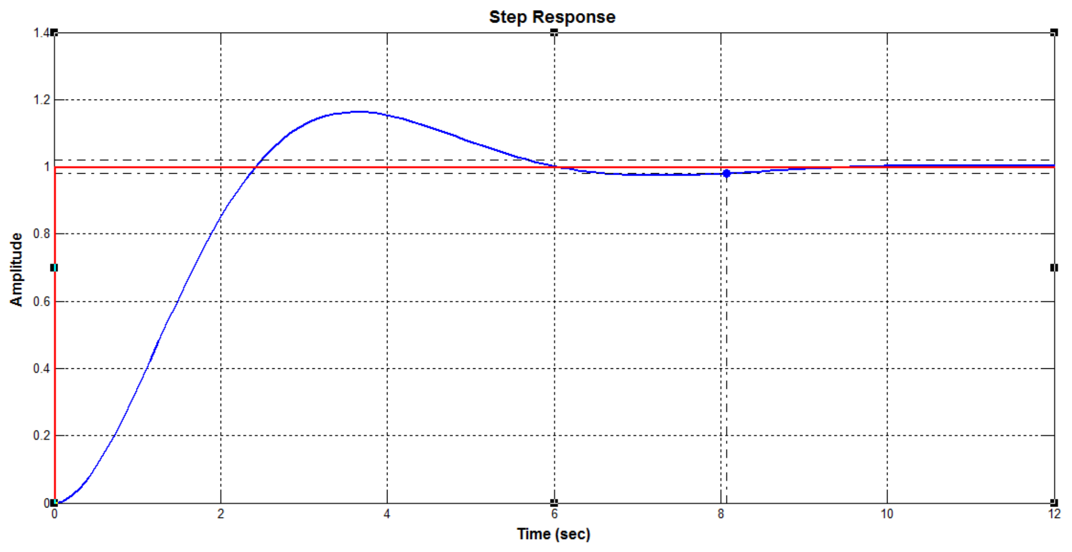


Figure 2.6: Step response of Mass, spring and damper system

It can be seen from Figure 2.6 that the system response generally follows the input step command (shown in red) but takes a relatively long time to rise and then settle at the commanded position. To improve or alter this natural response, a control system would be needed.

2.2.2 Electrical Systems

As with mechanical systems, fundamental principles and laws are used to build mathematical models of electrical systems from input (power source) to output (load). There are a wide range of simple electronic components used in electrical circuits including resistors, capacitors and inductors. These are also called passive components. More complex components, called active components, include op-amps and transistors. Active components as opposed to passive components are capable of changing their behaviour [8].

2.2.3 Electromechanical Systems

Many systems in use today have a combination of mechanical and electrical components. As such, energy is converted from mechanical to electrical energy or vice versa depending on the nature of the system. The simple DC motor is a good example of a common system in which electrical energy is converted into mechanical energy in the form of rotation. Consider the DC motor shown in Figure 2.7 [2, 7, 8]:

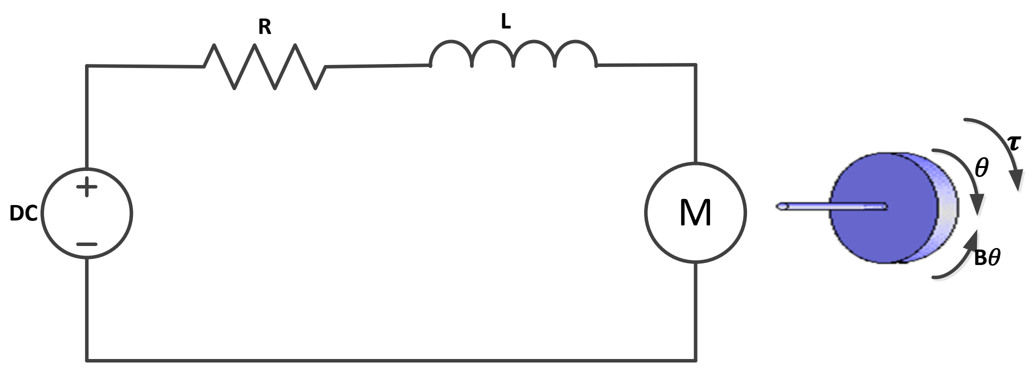


Figure 2.7: Free-body diagram for DC motor

The motor torque is given by:

$$\tau = K_t i \tag{2.11}$$

Where K_t is the armature constant and i is the armature current. The back emf, is given by:

$$e = K_e \dot{\theta} \tag{2.12}$$

Where K_e is the motor constant and $\dot{\theta}$ is the angular velocity of the rotor. The final system equations can be determined from Newton's law combined with Kirchhoff's law as follows:

$$J\ddot{\theta} + B\dot{\theta} = Ki \tag{2.13}$$

$$L\frac{di}{dt} + Ri = v - K\dot{\theta} \tag{2.14}$$

In the Laplace domain, these system equations can be expressed in terms of s, yielding the following open-loop input to output transfer function as a relation between input voltage and rotor speed:

$$\frac{\dot{\theta}}{V} = \frac{K}{(Js+B)(Ls+R)+K^2} \tag{2.15}$$

There are various standard techniques used for analysing electro-mechanical systems and designing appropriate controllers for them. The section that follows addresses some of these.

2.3 Dynamic system analysis and design

As already expressed above, the first step in analysing an electro-mechanical system is to generate its time domain differential equations that represent the dynamic behaviour of the physical system. These equations are derived from the physical laws governing the system behaviour. The second step is to determine and designate the inputs and outputs of the system and then formulate the transfer function characterizing its input to output behaviour. By studying the transfer function, the dynamic properties of the system can be determined. One way of retrieving useful information from a systems transfer function is simply to examine the positions of its pole-zero locations. Another way is to examine the time-domain properties of a system by determining the response of the system to typical excitation signals such as impulses, steps, ramps and sinusoids. There are various methods employed for system analysis and design. The choice of method ultimately depends on whether the system is linear or non-linear.

2.3.1 Linear vs. Non-Linear systems

Strictly speaking, all systems or processes possess some degree of non-linearity. A non-linear system is a system in which the output is not directly proportional to the input. Depending on the degree of non-linearity, such systems are usually unpredictable in their response and pose serious challenges to control engineers. Non-linearity can easily be seen from a system's dynamic equations; particularly if trigonometric or high order terms exist [10]. However, such systems can be approximated by linear models within a distinct operating range. In an angular-type system for example, if θ is small then $sin\theta \approx \theta$ and $cos\theta \approx 1$. Lyapunov showed in his study that if the linear approximation of a system is stable near an equilibrium point, then the truly non-linear system will be stable in the neighbourhood of the equilibrium point [7]. This process of linear approximation is known as linearization. Various linearization techniques exist, namely: Jacobian Linearization, Carleman Linearization, Lie Series, iteration technique and feedback linearization [10, 11].

For systems that have non-linear characteristics, most 'traditional' control techniques will only work on an equivalent linear approximation of the non-linear system and only within a small operating range about the equilibrium point/s [8, 12].

Depending on the complexity of the plant, linearization techniques can be mathematically challenging. The existence of a unique solution is also never guaranteed and in many cases, numerical approximations for such systems are not always sufficient. A more intelligent approach to non-linear systems is therefore required [7, 13].

Once a non-linear system has been modelled and linearized, Matlab or Simulink may be used to simulate the system's response to various types of input, e.g. a step, ramp or parabolic input, and then also design an appropriate controller [13]. Only after this step has been taken can a suitable control algorithm be implemented on the actual process in order to control the system as desired. For the purpose of analysis and design, a particular system may be studied in the time or frequency domains. Each of these is discussed in the following sections.

Numerous existing studies investigate the control of non-linear systems. Notably, Deng [14] investigates the feasibility of applying advanced control strategies to a mixing tank process. In this work a non-linear mixing tank process is modelled and linearized about its equilibrium point before an appropriate controller is designed and implemented.

2.3.2 Time response

Most control systems are designed to be stable (i.e. all the poles of the transfer function have negative real parts), so that if specific forcing inputs are used the response either settles to some steady state value or repeats itself after a certain time. If a range of values of time t are used, say from $t = 0$ to $t = \infty$, then a plot of the time response is obtained. The resulting plot is useful in the design process and gives a clear picture of how the system responds with reference to performance criteria [1, 15]. Forcing inputs or excitation signals include - but are not limited to - the following:

I. Step Input

Mathematically, the step input can be described as:

$$u(t) = \begin{cases} 0 \; for \; t < 0 \\ A \; for \; t > 0 \end{cases} \tag{2.16}$$

Where ‘A’ is the magnitude of the step input. The Laplace transform of the step input is $\boldsymbol{A/s}$. It therefore adds a pole to the systems transfer function at the origin of the s-plane. Figure 2.8 depicts the step input graphically.

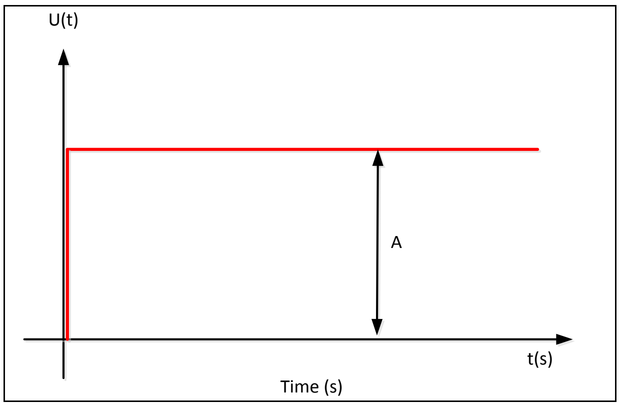


Figure 2.8: A step input of magnitude A

II. Ramp Input:

Mathematically, the ramp input can be defined as:

$$u(t) = \begin{cases} 0 \; for \; t \leq 0 \\ kt \; for \; t > 0 \end{cases} \tag{2.17}$$

Its Laplace transform is given as $\boldsymbol{k/s^2}$, and results in a double pole at the origin of the s-plane. Figure 2.9 depicts a ramp input graphically.

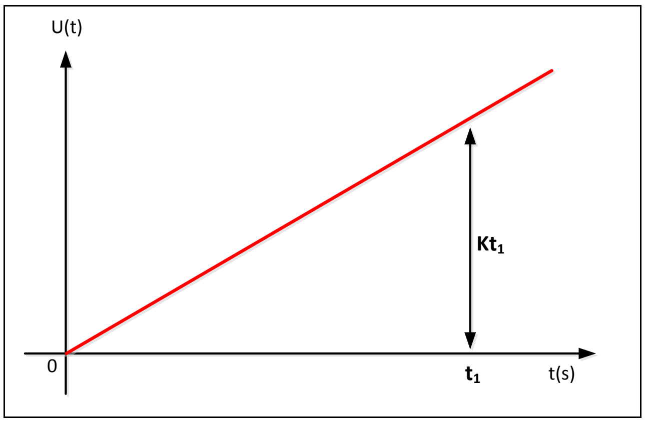


Figure 2.9: Ramp input

III. Pulse input

The pulse input is a rectangular input signal formed by two successive steps of equal magnitude but opposite sign. Pulse inputs are particularly useful when dealing with a functional system.

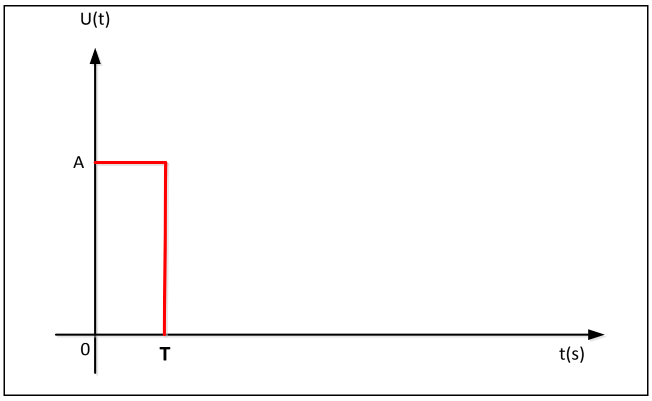


Figure 2.10: Pulse Input signal

IV. Steady state sinusoidal input

The sinusoidal input is probably the most relevant of the forcing inputs. It is applied after all transient effects have disappeared. This input is useful when studying the frequency response of a system. A sine wave, given mathematically in Eq. 2.3, is injected into the system, where A is the amplitude and ω the angular frequency:

$$u(t) = Asin\omega t \qquad (2.18)$$

The Laplace transform of this function is:

$$U(s) = \frac{A\omega}{s^2 + \omega^2} \tag{2.19}$$

This adds two imaginary poles to the s-plane, one at $s = j\omega$ and the other at $s = -j\omega$.

V. Random Input

In real, functional systems, all variables are continually changing. Random inputs are therefore useful in analysing a systems dynamics.

2.3.2.1 Time-domain specifications

In order to design a control system, certain time domain requirements may be stipulated. With reference to Figure 2.11, these are [7]:

1) Rise time (t_r): time taken to reach new set-point
2) Settling time (t_s): Time taken for the system transients to decay sufficiently
3) Overshoot (M_p): The maximum amount the system overshoots its set-point, expressed as a percentage
4) Peak time (t_p): Time taken to reach maximum overshoot point

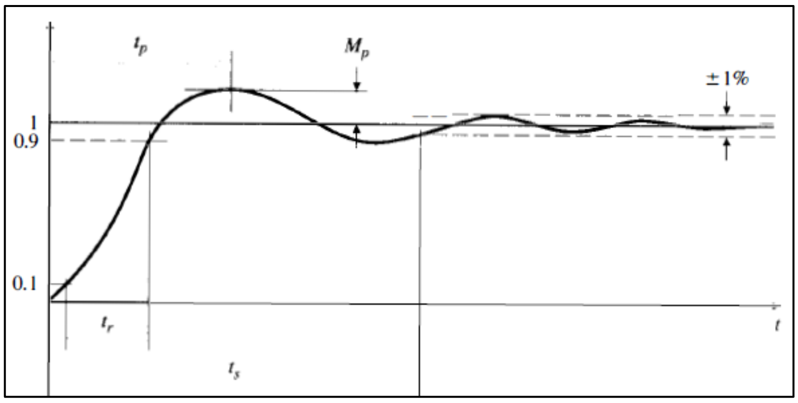


Figure 2.11: Time-Domain Specifications [7]

2.3.3 Frequency response

A linear system’s response to a sinusoidal input is called the system’s frequency response. A system’s frequency response reveals information about its stability. The Bode diagram shown in Figure 2.12 is used to represent a system’s frequency response and typically shows how the magnitude and phase of the system output are affected as the input frequency is varied [7]. The frequency response is discussed further in Chapter 2.4.2.

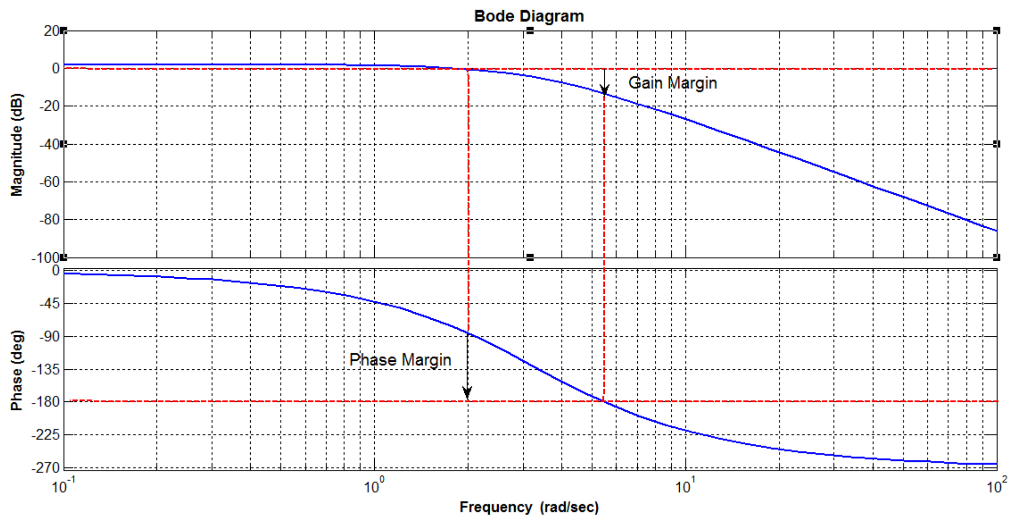


Figure 2.12: Bode plot showing gain and phase margins

2.3.3.1 Frequency-Domain specifications

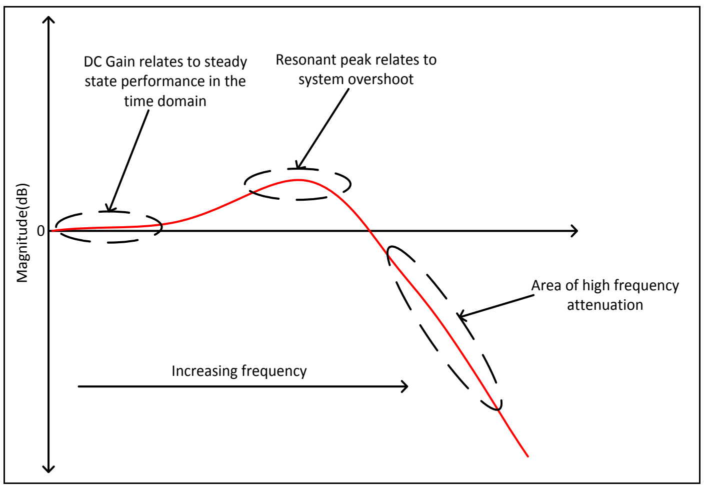

DC Gain relates to steady state performance in the time domain
Resonant peak relates to system overshoot
Magnitude(dB)
0
Area of high frequency attenuation
Increasing frequency

Figure 2.13: Frequency domain specifications

Frequency domain specifications are used to describe a system’s performance when a system is to be designed or analysed in the frequency domain.

They include:

- Maximum Magnitude ratio: This parameter is also referred to as the resonant peak (M_p). It gives an indication of a system's relative stability. A large M_p corresponds to a large peak overshoot in the step response. An optimum value of M_p would be between 1.1 and 1.5 for most design problems.
- Resonant Frequency: This is the frequency at which the system's gain or magnitude ratio is largest. It indicates a systems speed of response.
- Bandwidth: This is defined as the frequency at which the system gain drops to 0.707 ($1/\sqrt{2}$) of its zero frequency level. This value corresponds to a 3 dB fall on the decibel scale. A large bandwidth corresponds to a faster rise time. A large bandwidth will pass higher frequency signals to the output. Low bandwidth systems only allow low frequency signals to pass through the system and are usually slow and sluggish. The bandwidth also indicates the noise filtering characteristics of a system.
- Cut-off rate (or roll-off rate): This is the rate of decrease in the system magnitude outside the system's bandwidth (i.e. after the system gain has fallen by 3 dB from its zero frequency level). A high cut-off rate would indicate a system with good signal-to-noise ratio.
- Gain Margin: The gain margin gives the amount by which the closed-loop gain (magnitude) may be increased before the system becomes unstable (see Figure 2.12). The gain margin must be positive for a stable system (5-10 dB's is a good gain margin for a system to have).
- Phase Margin: The phase margin indicates how much additional phase lag at the gain crossover frequency can be withstood before a system becomes unstable (see Figure 2.12). For stability, the phase margin must be greater than zero. Designers generally try to keep the phase margin above 45 degrees.

When designing a controller, it is important to take the following into consideration:

- A smaller resonant peak and larger phase margin indicate smaller overshoot and hence a more stable system
- A larger gain crossover frequency indicates a faster system response

Design Steps in Frequency domain can be listed as follows:

1) Analyse time response behaviour to determine system deficiencies
2) Plot system's open-loop frequency response (Matlab)
3) Add Controller to change shape of frequency plot

It must be kept in mind during the design phase that magnitude and phase plots are interdependent and hence affect one another.

2.3.4 Design concepts for Linear Systems (LS)

As discussed in section 2.3.1, almost all real systems exhibit some degree of non-linearity. Non-linear systems can be quite complex not only to model but also to analyse and control using standard principles and techniques. It is therefore common practice for such systems to be linearized into equivalent linear systems. This literature study does not detail the different linearization techniques. However, the Jacobian linearization method is used in Chapter 3 to linearize the BOW system. The design concepts discussed in the following sub-sections are ideally used for linear, time-invariant systems.

2.3.4.1 Design concepts in the s-plane

A Laplace-transformed system equation (as shown in Eq. 2.20) is normally presented in the form of a rational polynomial, 's'.

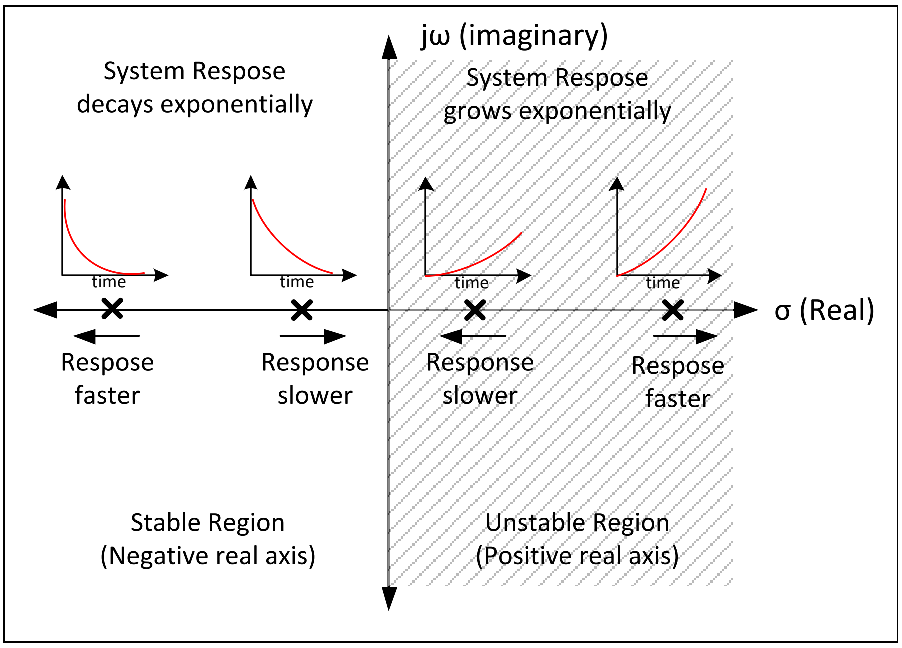
$$F(s) = \frac{N(s)}{D(s)} \tag{2.20}$$

If $D(s) = 0$, the resulting equation is called the system's characteristic equation as it can be said to characterize a system's dynamics. The roots of the characteristic equation are called the system poles. Poles are the values of 's' that make $F(s)$ infinite. The roots of the numerator, $N(s)$, are known as the zeros of system and are the values of 's' that make $F(s)$ zero. The poles and zeros of $F(s)$ could be complex values of 's' that have real and imaginary parts. Any complex root will have the form shown in Eq. 2.21 and can be represented in an Argand diagram or as it's otherwise called, an s-plane plot [1].

$$s = \sigma + j\omega \tag{2.21}$$

2.3.4.2 The Role of poles and zeros

The time response of a linear system depends on its pole-zero locations. Pole positions indicate the stability and speed of a system's response. If all the poles have negative real parts, then the system is said to be stable since it produces an exponential time response which decays to zero. Figure 2.14 below depicts the stable and unstable regions in the s-plane and the effect of pole positions on the speed of the system's response. The further to the left a pole is on the real axis, the faster the system's response will decay to zero. If a pole is positive and lies on the positive real axis, the system becomes unstable since such a pole gives rise to a term which has a positive exponential in time – i.e. the response of the system grows exponentially.



jω (imaginary)
System Respose decays exponentially
System Respose grows exponentially
time
time
time
time
σ (Real)
Respose faster
Response slower
Response slower
Respose faster
Stable Region (Negative real axis)
Unstable Region (Positive real axis)

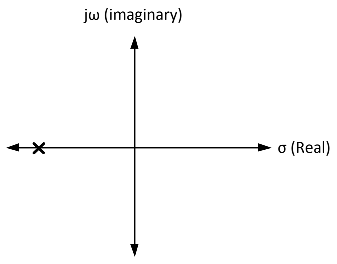
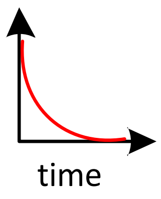
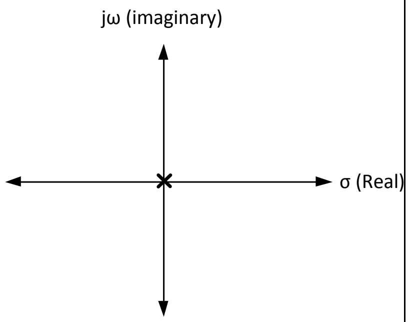
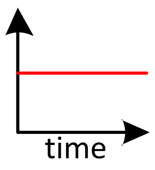
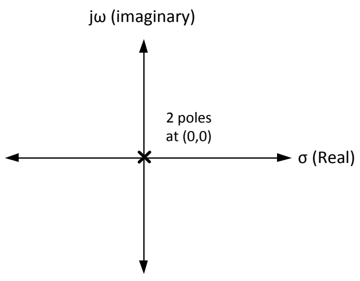
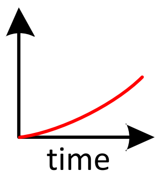
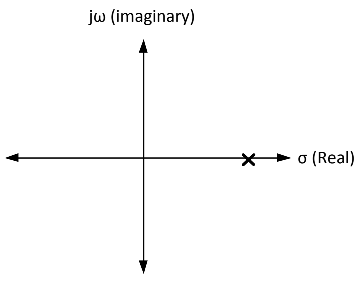
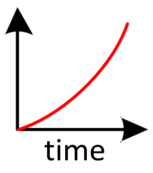
Figure 2.14: Effect of pole positions in the S-plane

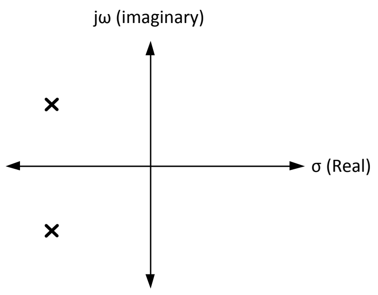
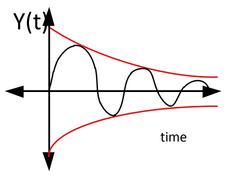
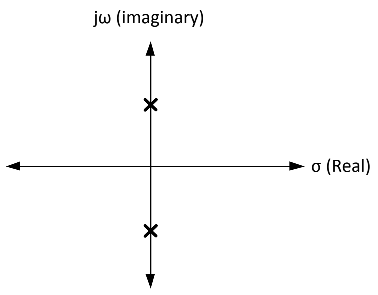
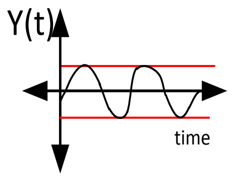
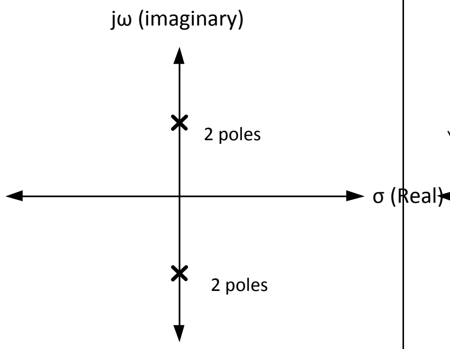
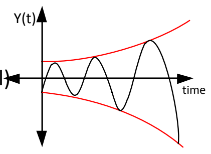
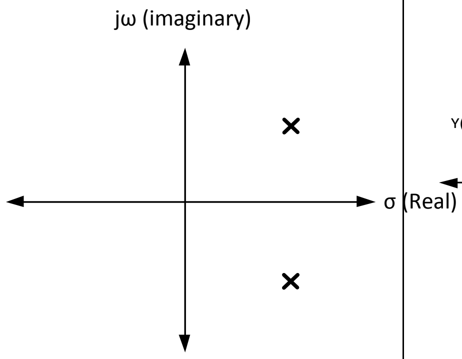
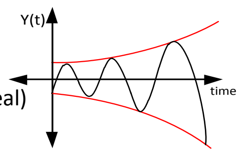
The zeros of a system adjust the performance of the system. The closer a zero is to a pole, the smaller the influence that that particular pole will have on the system's response. Pole-zero cancellation occurs if a pole and a zero coincide. Such a cancellation does not mean that the pole is lost. Rather, it simply means that at the chosen output, the dynamic term associated with a particular pole cannot be observed [1].

2.3.4.3 System Stability

Stability can be defined as a system’s response to an impulse. There are four main definitions of stability, namely: asymptotic stability, marginal stability, conditional stability and instability. Table 2-1 summarizes these definitions and also shows how pole locations influence stability. If all the poles of a system are contained in the left half of the s-plane, then it is asymptotically stable. If all the poles are in the right hand part of the s-plane, then the system is unstable. Poles on the imaginary axis are on the boundary between stability and instability. Such systems are called marginally stable systems [3].

Table 2-1: Effects of pole positions in S-plane on system response

	Pole Positions in S-plane	Impulse response	Comment
1			Asymptotically stable
2			Marginally stable
3			Unstable
4			Unstable

5			Asymptotically stable
6			Marginally stable
7			Unstable
8			Unstable

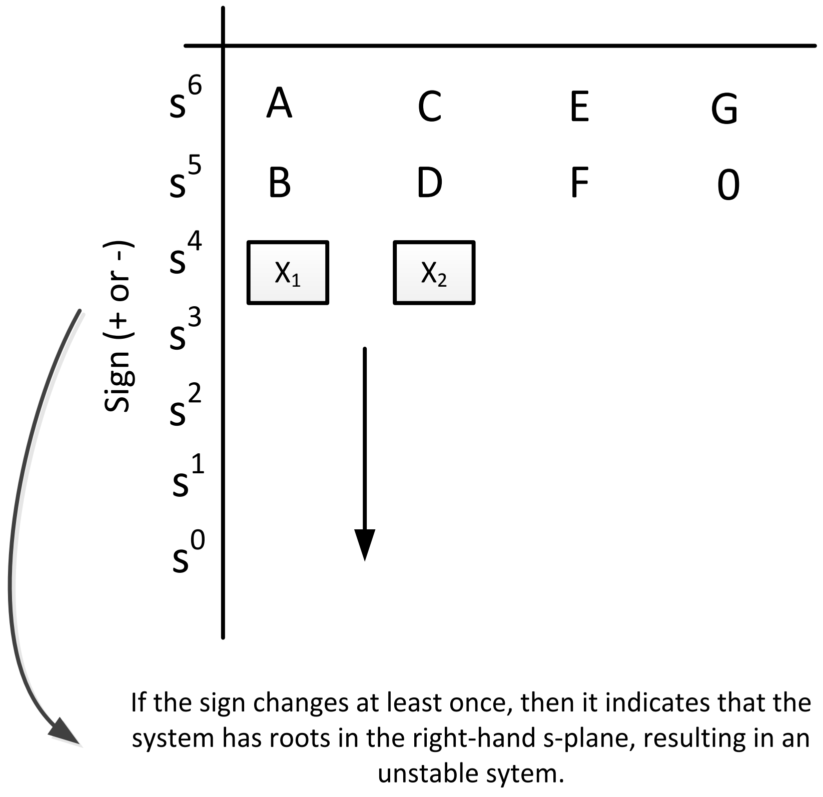
2.3.4.4 Routh-Herwitz stability criterion

The Routh stability criterion provides a quick and easy method of establishing a system's stability. This method of stability analysis does not require the locations of a system's poles to be determined. This makes it particularly useful when dealing with high order polynomials. The Routh stability criterion may also be used to establish the limiting values for the system gain beyond which the system would become unstable. The Zeigler-Nichols PID tuning method discussed in section 2.5.1.1 makes use of the Routh-Herwitz criterion to find the value of the proportional control gain, K_c.

Guidelines for using the Routh-Herwitz criterion

Consider a system's characteristic equation:

$$As^6 + Bs^5 + Cs^4 + Ds^3 + Es^3 + Fs^2 + G = 0 \quad (2.22)$$



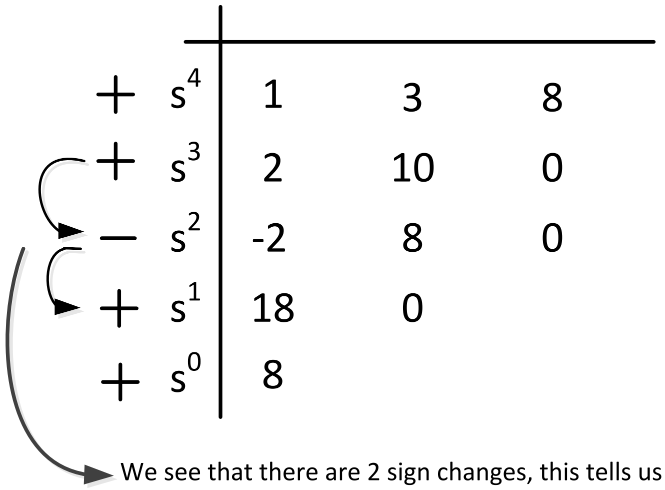
s6 A C E G
s5 B D F 0
s4 X1 X2
s3
s2
s1
s0
Sign (+ or -)
If the sign changes at least once, then it indicates that the system has roots in the right-hand s-plane, resulting in an unstable sytem.

$$X_1 = \frac{B.C - A.D}{B} \quad (2.23)$$

$$X_2 = \frac{B.E - A.F}{B} \tag{2.24}$$

Example: Consider the following characteristic equation,

$$s^4 + 3s^3 - 5s^2 + s + 2 = 0 \tag{2.25}$$



+	s^4	1	3	8
+	s^3	2	10	0
−	s^2	-2	8	0
+	s^1	18	0	
+	s^0	8		

We see that there are 2 sign changes, this tells us that there are 2 roots in the right-hand s-plane. The system is therefore unstable.

2.4 Controller design techniques

Over the years, various control techniques have been utilized in order to stabilize naturally unstable systems. Traditional control techniques have been used for many decades and are ideally suited to linear-type systems. New advanced techniques such as NN control have now surfaced and are showing promising results for their ability to control non-linear processes. The aim of any control algorithm is to alter the location of a system's poles and zeros in such a way that the system becomes stable. This could also involve adding new poles and zeros in strategic locations in the s-plane to either enhance or inhibit the effects of existing poles and zeros. Various controller design techniques have been developed to aid the process of controller design. Sultan and Mirza [16] for instance use some of these methods to analyse and design suitable controllers for a non-linear inverted pendulum system. Their work is used as a guideline for the controller design stage of this research. A few of the common traditional as well as the more advanced methods of control in use today are briefly discussed in the following sections.

2.4.1 Root-locus Controller Design

This control system design technique is used to determine the roots of the closed-loop characteristic equation when the open-loop gain constant, K, is increased from zero to infinity. The closed-loop poles are plotted on the s-plane as K is varied from zero to infinity. A suitable value for K is then selected to produce the necessary transient response as required by performance specification. The loci always commence at open-loop poles and terminate at open-loop zeros when they exist [17]. The disadvantage of using this method is that only one variable can be varied at a time and additional methods must be used to find additional parameters.

An understanding of the general rules of root loci construction are important when designing a controller (using root-locus method) as sometimes it becomes necessary to force the root-locus to bend towards a desired region in order to meet design specifications [3]:

- Starting point of the root loci ($K = 0$): The root loci start at the poles of G(s)H(s). They are considered to start at the points at which the gain K is zero.
- End point of the root loci ($K \rightarrow \infty$): The root loci end at zeros of G(s)H(s).
- Root-loci on the real axis: As a direct result of the angle condition, the root loci may be found on a given section of the real axis only if the total number of poles and zeros of G(s)H(s) on the real axis to the right of a section is odd.
- The number of branches of the root loci: The number of branches of the root loci is equal to the number of poles (or zeros) of the open-loop transfer function. If the finite poles and zeros are equal in number then the whole locus is generally possible on a drawing. Where the number of poles (P) exceed the number of zeros (Z) or the number of zeros exceeds the number of poles, the branches terminate at infinity. There are then P minus Z or Z minus P branches which tend asymptotically to the straight line sections of the loci.
- Symmetry of the root-locus: The root-loci are symmetrical with respect to the real axis, since any of the complex roots always appear in complex conjugate pairs.

- Asymptotes of root-loci: For large values of the roots, P minus Z branches of the root-loci are asymptotic to straight lines with angles to the real axis. The asymptotes do not necessarily pass through the origin but because of symmetry they do not intersect on the real axis.
- Intersection of the loci with the imaginary axis: The intersection of the loci with the imaginary axis marks the stage at which the real parts of the roots change from negative to positive, resulting in an unstable system.
- Break-away points on the real axis: The points in the s-plane where multiple roots of the characteristic equation are found are called the 'breakaway points' of the root-locus diagram. At such a point two or more roots loci branches branch away or meet. Where branches between two poles meet on the real axis the loci then branch away. Where branches between two zeros meet on the real axis, the loci move into the real axis.

2.4.1.1 The effect of adding open-loop poles and zeros

The effect of the addition of a real or complex conjugate pole to the left-hand side of the s-plane is to increase the closed-loop stability of the system. This is shown by the bending of the root-loci towards the imaginary axis (i.e. more to the left of the s-plane). The effect of adding a zero into the left-hand s-plane is to increase the stability of the system. This remains true for the addition of real zeros or conjugate pairs. Moving a pole closer to the origin in a stable system slows down the response of the system [3].

2.4.2 Frequency Domain Controller Design

Frequency domain analysis is a study of the steady state system output in response to constant amplitude yet variable frequency sinusoidal input. Steady state errors, in terms of amplitude and phase, relate directly to the dynamics of a system as expressed in a transfer function.

Consider a sinusoidal input of amplitude A_1 being fed into a system G(s):

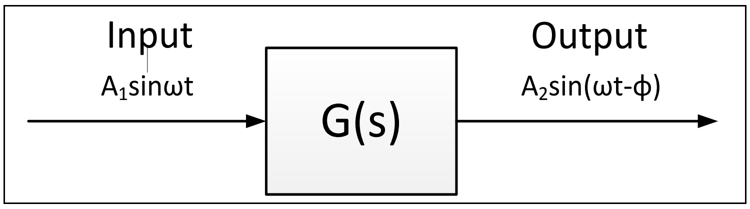


Figure 2.15: Plant with phase and amplitude difference between input and output

As shown in Figure 2.15, the output signal amplitude, given by A_2 may be affected positively or negatively depending on the plant dynamics. The phase relationship is given by ϕ and can be positive or negative again depending on the nature of the plant G(s). The amplitude ratio A_2/A_1 is given by:

$$\frac{A_1}{A_2} = |G| \tag{2.26}$$

Where $|G|$ is the modulus or gain of the system [17].

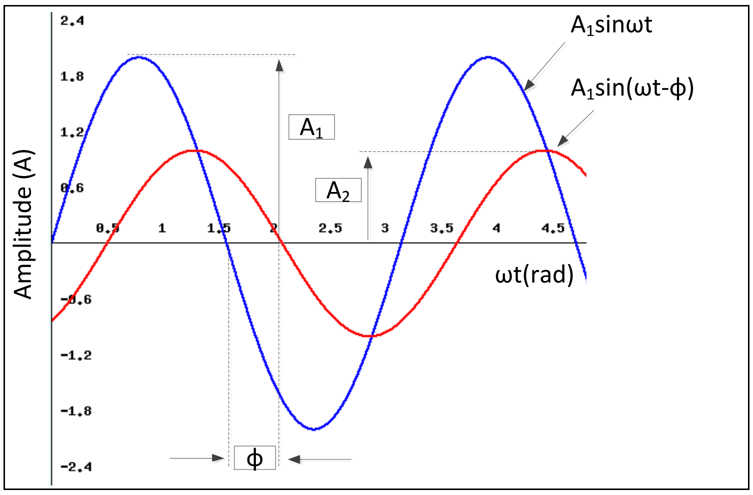


Figure 2.16: Plot of input vs. output of plant G(s) [17]

Figure 2.16 shows graphically how the amplitude and phase differ between input and output of G(s). The frequency of the output is the same as the input. The phase angle and the gain are dependent on the frequency of the input signal. As the frequency of the input is varied, it produces a variation in the gain and phase angle. Using the relationship between frequency, gain and phase, a frequency response plot may be drawn in order to analyse a given system.

The frequency response has the following advantages:

- It provides a rich source of information to the designer about the plant
- Deductions about a system's stability can be made quite easily
- It settles ambiguities found in other analytical methods such as the root-locus
- It shows the effects of individual pole locations (not always possible with the root-locus method)
- It can easily be related to the time response and hence allows the designer to understand how each pole or zero affects the time response

The Bode plot, as discussed briefly in section 2.3.3 comprises a graph of magnitude $G(j\omega)$ against the frequency ω and a graph of the phase angle ϕ as a function of the frequency. The magnitude is plotted on a log scale expressed in decibels. The phase angle on the other hand is plotted on a linear scale [3]. In both cases, the frequency is plotted on a logarithmic x-axis scale.

$$|G(j\omega)|dB = 20log_{10}|G(j\omega)| \qquad (2.27)$$

The Bode plot usually starts with a flat region equal to the DC magnitude or DC gain of the system. The system gain (K) thus has a direct effect on the DC gain of the system as shown in Figure 2.17. The DC gain relates to the steady state performance of a system in the time domain. The system gain does not have an effect on the system's phase.

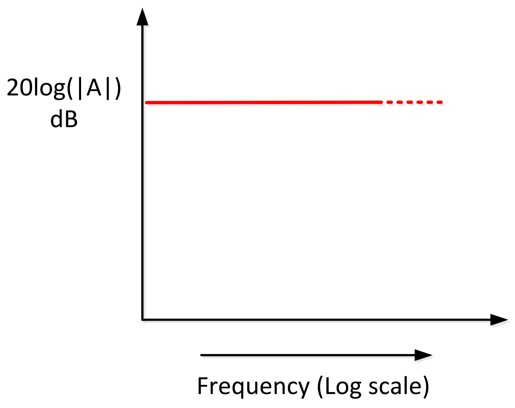


Figure 2.17: DC gain (effect of K only)

For the addition of every pole, the slope of the line decreases by 20dB/ decade at that pole's frequency as highlighted in Figure 2.18. Two poles will thus cause the slope to decrease at a rate of 40dB/ decade.

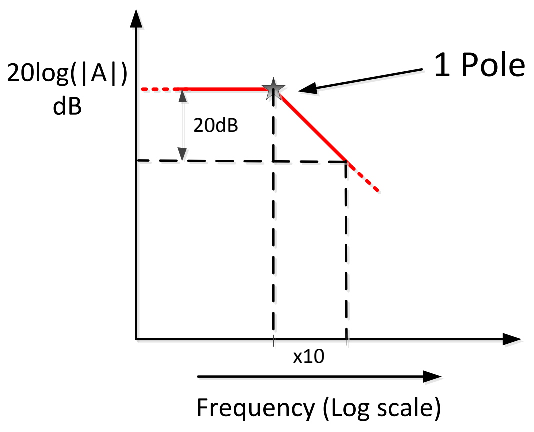


Figure 2.18: Effect of adding a pole

For every zero, the slope of the line increases by 20dB/ decade at that zero's frequency as indicated in Figure 2.19. For 2 zeros, the slope will increase at a rate of 40dB/ decade.

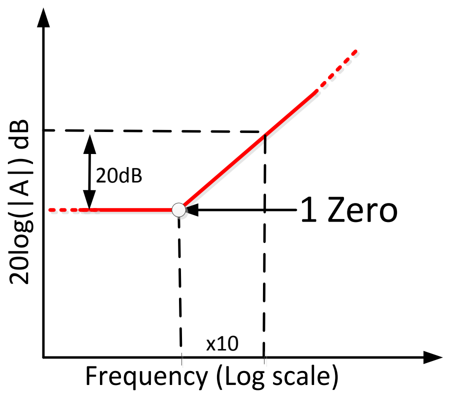


Figure 2.19: Effect of adding a zero

A zero at the origin causes a -90 degree shift in phase between the system input and output as shown in Figure 2.20.

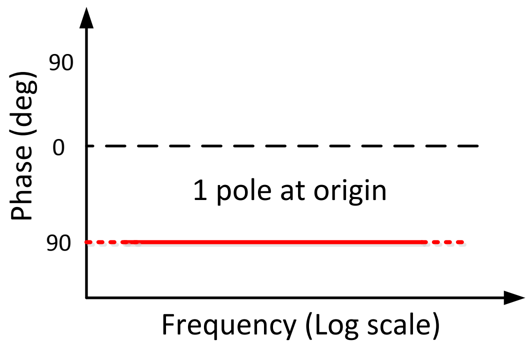


Figure 2.20: Effect of adding a pole on the phase angle

A pole at the origin causes a +90 degree shift in phase between the system input and output as shown in Figure 2.21.

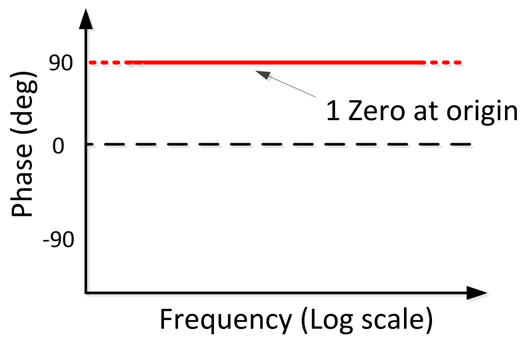


Figure 2.21: Effect of adding a zero on the phase angle

2.4.3 State space representation of dynamical systems

The classical design techniques discussed above such as the root-locus, time domain and bode plot method are generally only applicable to [17]:

a) Single input, single output systems (SISO)
b) Systems that are linear or that can be linearized and are time invariant (i.e. have parameters that do not vary with time)

The state space method is useful in dealing with:

a) Multiple input, multiple output systems (MIMO)
b) Non-linear and time invariant systems
c) Alternative controller approaches

The state of a system may be defined as a set of state variables which at some initial time t_0 together with the input variables completely determines the behaviour of the system for time $t \geq t_0$. The state variables are the smallest number of states that are required to describe the dynamic nature of the system. These variables do not all have to be measurable.

$$\dot{x} = Ax + Bu \tag{2.28}$$

$$y = Cx + Du \tag{2.29}$$

Where y is the system output equation and where x is an n dimensional state vector:

$$\begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_n \end{bmatrix} \tag{2.30}$$

And u is the m dimensional input vector:

$$\begin{bmatrix} u_1 \\ u_2 \\ . \\ . \\ u_n \end{bmatrix} \tag{2.31}$$

A is the n x n system matrix:

$$\begin{bmatrix} a_{11} & a_{12} & .. & a_{1n} \\ a_{21} & a_{22} & .. & a_{2n} \\ . & & & \\ . & & & \\ a_{n1} & a_{n2} & .. & a_{nn} \end{bmatrix} \tag{2.32}$$

B is the n x m control matrix:

$$\begin{bmatrix} b_{11} \dots b_{1m} \\ b_{21} \dots b_{2m} \\ . \\ . \\ b_{n1} \dots b_{nm} \end{bmatrix} \tag{2.33}$$

2.4.4 Observability and controllability

A system is controllable if a control vector $\mathrm{u(t)}$ can be found which will enable us to force the system from an arbitrary initial state $\mathrm{x(0)}$ to some arbitrary finite state $\mathrm{x(t_f)}$ in a finite time $\mathrm{t_f}$. To determine if a system is controllable, the controllability matrix must be examined. It is defined by:

$$C = [B \;\; AB \;\; A^2B \ldots A^{n-1}B] \tag{2.34}$$

The system is controllable only if this matrix (Eq. 2.34) has rank equal to the number of states present in the system. In the same way, it is important for control engineers to ensure that a system of state variables can be determined by a lesser number of outputs, i.e. can the state variables be measured by the measurements of the output that are possible. A system is therefore completely observable if the output, y, over a finite time, contains the information which completely defines the state x [3]. To determine if a system is observable, the observability matrix may be examined. It is defined by:

$$\varphi = \begin{bmatrix} C \\ \mathrm{CA} \\ \mathrm{C}A^2 \\ . \\ . \\ \mathrm{CA^{n-1}} \end{bmatrix} \tag{2.35}$$

The system is observable if and only if this matrix has rank equal to the number of states present in the system. Mathematical tools such as Matlab may easily be used to evaluate the rank of the controllability and observability matrices of a given system [18]. Determining whether or not a system is controllable and observable is important before time is wasted attempting to design a controller.

2.5 Traditional control techniques

Many linear control algorithms and techniques have been devised, studied and tested. The most common of these is the PID controller [19]. Other variants of this controller exist such as PI, PD and PI-PD controllers. Similarly, compensators such as the Lead, Lag and Lead-Lag compensators are alternative control approaches.

Methods utilizing full state feedback control such as the Linear Quadratic Regulator (LQR) are slowly becoming more established as the tools utilized in generating accurate system models become cheaper and more accessible.

2.5.1 PID Controller

As already discussed, the most widely used type of controller in industry today is the PID controller. According to Vlachogiannis and Roy [20], up to 95% of all controlled processes in industry utilize PID controllers. The PID controller is a robust control algorithm that can be tuned by trial and error methods [21]. This inherent simplicity makes it a favourable choice in environments where the specialized skills required in modelling and design of control systems is lacking. However, in more complex applications, these trial and error methods of tuning PID controllers is impractical, time consuming and sometimes dangerous. In order to design the most suitable PID controller for a particular system, simulation is arbitrary and requires the formulation of an accurate system model. Figure 2.22 shows the basic structure of a closed-loop PID control system [22].

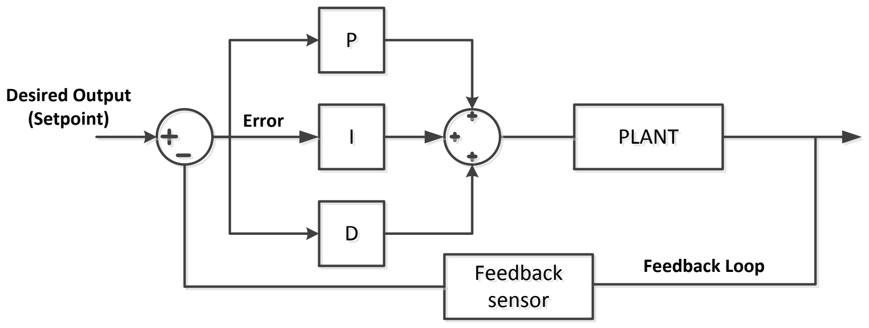


Figure 2.22: Basic closed-loop system with PID controller

$$PID = K_p . e(t) + K_i . \int_0^t e \, . dt + K_d . \frac{de(t)}{dt} \tag{2.36}$$

Equation 2.36 shows the standard PID algorithm in the parallel form. The proportional term makes changes to the output in proportion to the error value. The integral term affects the output in proportion to both the magnitude and duration of the error. Increasing the integral time makes the output respond slower to an error. The derivative term affects the output in proportion to the rate of change of the error. The methods discussed in section 2.4 above may be used to design PID controllers including the: time response method, root-locus method, frequency response method and State Space (SS) method [13]. Depending on the complexity of the system being modelled; the process of modelling can be both time consuming and costly, requiring special equipment to measure system parameters accurately. Furthermore, for systems that have non-linear characteristics, the PID controller will only work on an equivalent linear approximation of the non-linear system and only within a small operating range about the equilibrium point/s [5, 23].

In his work, Yurkevich [24], expounds on these points and particularizes the limitations of the traditional PID controller especially in the presence of plant uncertainty or non-linearity. He expresses the need for a controller that is able to tackle the problem of non-linearity and thus investigates a novel, time-based method for designing a PID controller for a non-linear system. Various other works including Tan et al. [21] propose self-tuning PID controllers that are well suited for the control of non-linear systems.

The transfer function of the PID controller in the ideal or standard form, which is actually the more common form in use in industry, is given as:

$$G_c(s) = K_c(1 + T_d s + 1/T_i s) \tag{2.37}$$

Where,

K = Proportional gain
T_d = Derivative action time
T_i = Integral action time

Sometimes it is necessary to convert the integral gain K_i and derivative gain K_d into the equivalent integral time and derivative time respectively as shown in Eq. 2.38-2.40.

$$K_p = K_c \tag{2.38}$$

$$K_i = \frac{K_c}{T_i} \tag{2.39}$$

$$K_d = K_c . T_d \tag{2.40}$$

PID control, where all three controller elements are used, is not always necessary. Depending on the nature of the system needing control, sometimes P action, PD action or PI action alone will suffice.

If T_d = 0 and T_i = ∞, a P controller will result.
If T_d = 0 and T_i = Finite, a PI controller will result.

2.5.1.1 Zeigler-Nichols rules for controller tuning

In cases where the necessary tools for controller design and tuning are not accessible, PID controllers may be manually tuned using standard procedures. One of the simplest and therefore most common of these procedures is the Zeigler-Nichols method for controller tuning.

Consider the closed-loop system shown in Figure 2.23 below:

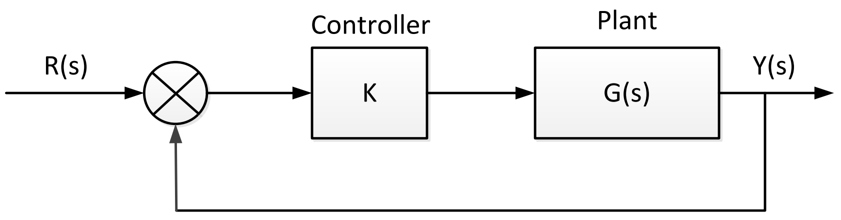


Figure 2.23: Closed-loop system with Proportional (P) control

Assume that this system has the property, that under purely proportional control it is asymptotically stable in the range $0 \leq K \leq K_c$ and goes unstable for $K > K_c$. For such a system, the following practical procedure is followed:

- Turn up the gain K, until continuous oscillations are observed in the system. At this gain, K_c, the closed-loop system is marginally stable, on the boundary between stable and unstable behaviour.
- Note the value of K_c and the period of oscillations, T.
- For P control: $K = 0.5K_c$
- For PI control: $K = 0.45K_c$ and $T_i = 0.83T$
- For PID control: $K = 0.6K_c$, $T_i = 0.5T$ and $T_d = 0.125T$

It must be noted that the determined Zeigler-Nichols' gains often only form a starting point for controller tuning. In most cases, manual fine-tuning will still be required.

If the plant model happens to be available, the Routh-Herwitz criterion (discussed in section 2.3.4.4) may be used to find the value of K_c and the corresponding period of oscillation T.

Procedure:

1) Find the systems closed-loop characteristic equation under pure proportional control.
2) Using the Routh-Herwitz Criterion, find the value of K_c that produces an all zero row.
3) Use the divisor polynomial to find the period of oscillation T and then apply the Zeigler-Nichols tuning method.

Example:

Consider the following closed-loop characteristic equation with only proportional control, K.

$$F(s) = s^3 + 6s^2 + 11s + 6(1 + k) = 0 \qquad (2.41)$$

The resulting Routh array is as follows:

Row			
0	s^3	1	3
1	s^2	6	6(1+k)
2	s^1	11- (1+k)	

For row 2 to be zero, K must be set to 10. Therefore K_c = 10. The divisor polynomial is obtained from row 1 and is:

$$s^2 + 11 = 0 \qquad (2.42)$$

$$s = j\sqrt{11} \qquad (2.43)$$

Since $s = j\omega$, it follows that

$$j\omega = j\sqrt{11} \tag{2.44}$$

$$\therefore \ \omega = \sqrt{11} \tag{2.45}$$

$$T = \frac{2\pi}{\omega} = 1.895s \tag{2.46}$$

Therefore according to the Zeigler-Nichols' tuning method the PID gains are as follows: K = 6, T_i = 0.947 and T_d = 0.237.

Although this tuning method is useful in finding controller gains, this method provides no indication of what controller modifications would be needed if certain performance specifications were stipulated. Other design techniques such as the root-locus or frequency domain method would be required to meet given performance specifications.

2.5.2 Lead compensator

The lead compensator, as the name suggests, adds phase lead to a system as depicted in Figure 2.25. It improves the phase margin and damping and also speeds up the system response. The transfer function of this compensator consists of a single zero and pole. In order for this structure to behave as a lead compensator, the zero must be located before the pole on the Bode plot, i.e.:

The transfer function for the lead compensator is given as:

$$G(s) = K_p\left(\frac{s+z}{s+p}\right), where\ z < p \tag{2.49}$$

This can be rearranged to give:

$$G(s) = K_p.\left(\frac{z}{p}\right)\left(\frac{\frac{1}{z}s+1}{\frac{1}{p}s+1}\right) \tag{2.50}$$

Another arrangement of this controller is given as:

$$C_p = K_p\left(\frac{1+Ts}{1+\alpha Ts}\right) \tag{2.51}$$

This can be rewritten as:

$$C_p = K_p . \frac{1}{\alpha}\left(\frac{1+\alpha Ts}{1+Ts}\right) \quad (2.52)$$

Where the zero is at -1/T and the pole is at -1/αT. If K_p is unity then if α is small (say 0.05 to 0.1), the resulting compensation is that of a lead network [3]. Consider a lead compensator with a pole at 100 rad/ sec and a zero at 10 rad/ sec as shown in the frequency response plot in Figure 2.24.

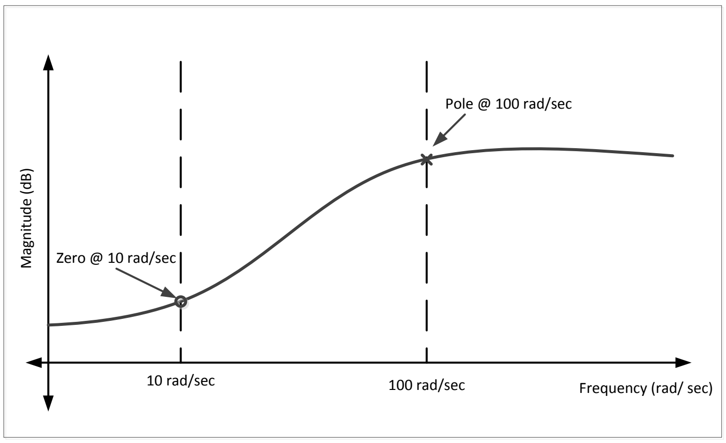


Figure 2.24: Magnitude vs. frequency plot of a lead compensator

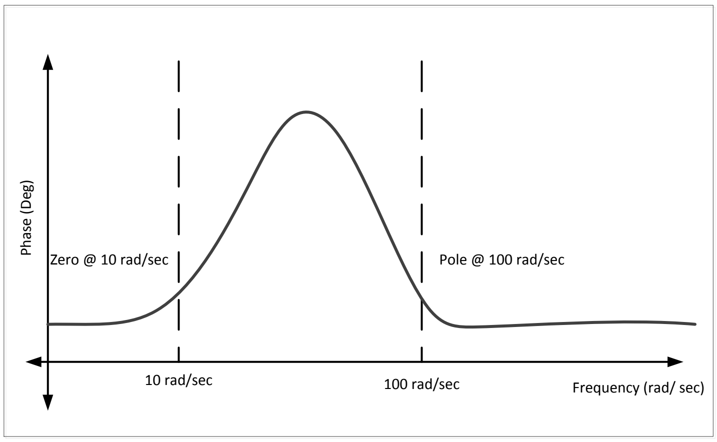


Figure 2.25: Phase vs. frequency plot for lead compensator

From Figure 2.24 it can be seen that the zero causes the magnitude to rise, whereas the pole causes it to fall. The lead controller achieves similar results to a PD controller.

2.5.3 Lag compensator

The Lag compensator introduces phase lag or negative phase to a system as shown in Figure 2.27. This improves the disturbance rejection (due to increased high frequency attenuation) but slows down the system's response. The gain margin is not greatly affected with this strategy. In this compensator the pole is located before the zero, i.e.:

$$G(s) = K_p\left(\frac{s+z}{s+p}\right), where\ z > p \tag{2.53}$$

An alternative arrangement of this compensator is given as:

$$C_p = K_p\left(\frac{1+Ts}{1+\alpha Ts}\right)\ \alpha > 1 \tag{2.54}$$

This can be rewritten as:

$$C_p = K_p.\frac{1}{\alpha}\left(\frac{1+\alpha Ts}{1+Ts}\right) \tag{2.55}$$

If α is large (say 10 to 20) then the resulting compensation is that of a lag network. T must be chosen to be at least 10 times larger than the largest time constant of the system [3]. Consider a lead compensator with a pole at 10 rad/ sec and a zero at 100 rad/ sec. A plot of this compensator's frequency response is given in Figure 2.26 and Figure 2.27.

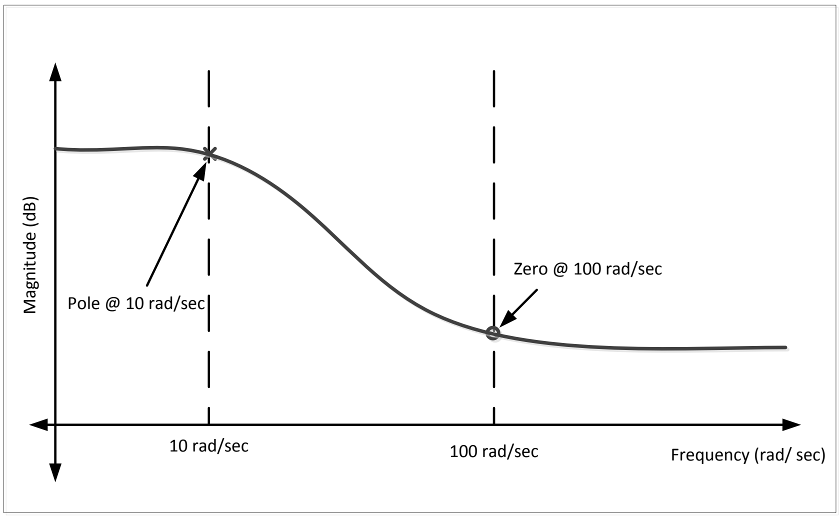


Figure 2.26: Magnitude vs. frequency plot for lag compensator

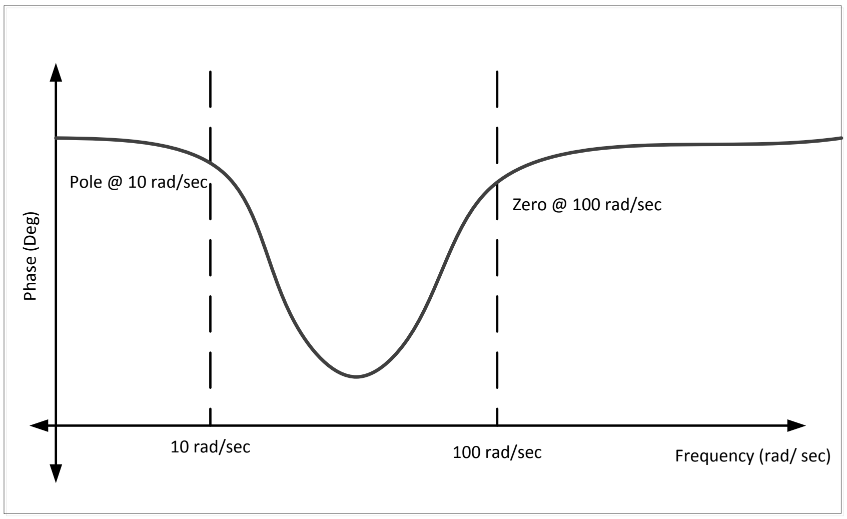


Figure 2.27: Phase vs. frequency plot for lag compensator

2.5.4 Lead-Lag compensator

A Lead-Lag compensator effectively combines the lead and lag compensators discussed above to produce a compensator which at low frequencies acts as lag network and at high frequencies acts as a lead network [3]. The use of lead-lag compensators makes it possible to meet many system specifications without incurring the penalties of excessive bandwidth or an over sluggish response [1].

$$C_p = \left(\frac{1+T_1 s}{1+\alpha T_1 s}\right)\left(\frac{1+T_2 s}{1+T_2 s/\alpha}\right), \alpha > 1, T_1 > T_2 \tag{2.56}$$

Szczudlak and Fasheh [25] in their research implemented a digitized form of the Lead and Lead-Lag compensator on a microcontroller to control an inverted pendulum system. One major point that stands out in their study is that their theoretical model of the system was able to predict the rise and settling times of the system quite accurately. However, this was not true regarding the system overshoot. They attributed the differences between the real world implementation and simulation primarily to the assumptions they made in determining system parameters. In fact these estimates became restrictive factors when implementing their controllers in real time. They also noted that the microcontroller they used took too long to update the compensator values, thus limiting the degree of viable control.

Saha et al. [26] in their research paper outline the design process of a lead-lag controller using the frequency domain approach to control a motor with a cascaded phase shifter in each segment of a multi segment snake robot. They successfully met all their design requirements with this controller and were able to achieve the desired snake-like motion.

2.5.5 LQR Controller state feedback design

The Linear Quadratic Regulator (LQR) provides an optimal control law for a linear system with a quadratic performance index [17]. The LQR is a robust controller that guarantees a phase margin greater than 60 degrees.

A system can be expressed in state variable form as:

$$\dot{x} = Ax + Bu, \qquad y = Cx + Du \tag{2.57}$$

$$x(t) \in R^n, u(t) \in R^m$$

With the assumption that all the states are measurable, we can find a state-variable control law that gives the desired closed-loop properties. The closed-loop system using this control law becomes:

$$\dot{x} = (A - BK)x + Bv = A_c + Bv \tag{2.58}$$

To design a state-variable feedback controller that is optimal, we may define the performance index (PI) as:

$$J = \frac{1}{2}\int_{t_0}^{t_1}(x^T Qx + u^T Ru)dt \tag{2.59}$$

Where: $u = -Kx$

And the feedback gain matrix K is given by:

$$K = R^{-1}B^T P(t) \tag{2.60}$$

P is found by solving the continuous time Riccati differential equation.

$$A^T P(t) + P(t)A - P(t)BR^{-1}B^T P(t) + Q = -P(t) \tag{2.61}$$

The LQR computations are relatively complex but can be simplified by making use of the LQR tool in Matlab.

Balaševičius et al. [27] in their research discuss the implementation of state feedback control in the form of an LQR controller on a PLC to control a chemical reaction vessel. The reaction vessel model is that of a second order system. It is noted that in this type of system, not all state variables can be measured or controlled; in fact only the control input and the output of the system can be measured. To solve this problem, they first obtain a system model and then reconstruct the non-measured state variables from the measured control input and output of the system. LQR control is implemented on the modelled system firstly in Matlab and then finally it is effected on an actual PLC. It is observed that this type of control does not perform so well in an actual implementation because of the PLCs limited execution speed.

For real-world implementation on a digital controller, some of the major disadvantages of optimal control strategies such as LQR are that:

1) Certain states that need to be observed may not be directly observable. State observers may have to be designed in order to observe various states.
2) A good model of the system is needed. If the model is incomplete, perhaps due to un-modelled dynamics, it may be difficult to get a controller that meets expectations.
3) Non-linear models must be linearized or else the system may become unstable.

2.6 Advanced control techniques

Artificial Neural Networks (ANNs) are based on the operation of biological neurons in the brain. They are composed of interconnected neurons that act as processing units interconnected between single or multiple inputs and outputs. Each connection between neurons has an adjustable weighting factor that determines the strength of the connection. This interconnected and adaptive structure gives NNs a non-linear and parallel problem-solving ability that is not found in conventional processing structures.

In the control environment, NNs have thus received widespread attention, especially for their ability to learn non-linear characteristics through experimental data, without prior knowledge of the plant [22]. Research has proven that NNs can estimate every non-linear function with at least one hidden layer. NNs are therefore extensively used in simulation and control of non-linear processes [10, 28, 29]. The cumbersome process of system modelling found with conventional controllers is thus eliminated provided that suitable operational data can be obtained from the plant for the purpose of training the network [30]. Hagan and Demuth [31] in their publication show that NNs have been successfully applied in the identification and control of dynamical systems. They expound further on the universal approximation capabilities of the multi-layer perceptron that makes it a popular choice for modelling non-linear systems and for implementing general purpose non-linear controllers. Their research suggests a number of non-linear controller configurations.

According to Han et al. [12], Feed-forward Multilayer Neural Networks (MNNs) are the most prevalent NN architectures for identification and control applications. A widely used training method for feed-forward MNNs is Back Propagation (BP). The Levenberg-Marquardt algorithm is very efficient for training small to medium sized networks and it also uses BP [12]. The disadvantage of BP learning is the long and unpredictable training process with the rate of convergence being seriously affected by the initial weights.

Unlike traditional controllers, intelligent NN controllers are able to adapt to parameter changes in the plant. Thus the need for regular retuning is eliminated [13]. It has also been shown that sensor noise or other mild disturbances have little effect on NNs [4]. Special care, however, must be taken when training MNNs to ensure that they do not over fit the training data and then fail to generalize well in new situations [31]. Since NNs can have several inputs and outputs they may also be used for multiple input and multiple output systems (MIMO) [4]. The drawbacks of using a NN as a controller are that:

- The control system is not operational or performs poorly during the training process
- The training can take a long time
- Unpredictable disturbances cannot be eliminated

- Training data may be hard to attain and the training process does not always guarantee the best results

2.6.1 Standard NN Controllers for dynamical systems

Many standard intelligent NN based control techniques have been devised over the years. The sections that follow investigate some of the more prominent techniques used in the control environment today [32, 33].

2.6.1.1 NN Model Reference Control

In the work of Rahmat et al. [22], an intelligent controller is applied to a non-linear and unstable system namely the “Ball on Beam” system. Specifically, a NN Model Reference Control (MRC) scheme is applied to control the plant. This scheme makes use of Levenberg-Marquardt BP for the training process in which a NN is trained to follow a reference model. The results from the research show that this intelligent scheme, although slower, produces similar results to a PID controller implemented on the same plant.

Jain and Nigam [34] in their research paper discuss how the limitations found with conventional feedback controllers due to variations in process dynamics may be overcome using Model Reference Adaptive Control (MRAC). Their results are based on simulations carried out in Matlab and Simulink.

Straussberger et al. [35] in simulation and in an actual implementation use MRAC to adaptively control a 2-wheel self-balancing laboratory plant called the “Mono Chair” that experiences parameter variation as it navigates over diverse terrain.

2.6.1.2 NN Predictive controller

With the aid of NNs, it is possible to predict the future behaviour of a plant based on historical plant data. In the first step of operation, a NN is trained to represent the forward dynamics of the plant. The prediction error between the system output and the NN output is used as the training signal for the NN. The NN plant model uses previous inputs and previous plant outputs to predict future values of the plant output [36].

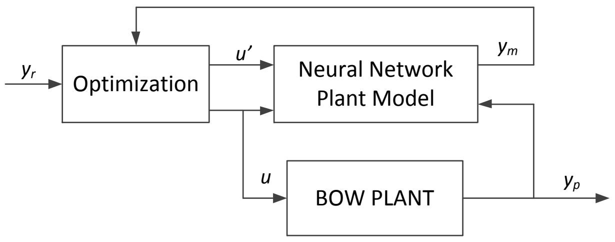


Figure 2.28: NN-Predictive Controller [31]

The NN Predictive controller predicts the plant response over a specified time horizon. The optimization block in Figure 2.28 determines the values of *u'* that minimize cost function *J* according to (Eq. 2.62) below. The optimal control signal *u* is then fed into the plant to establish control [32, 33].

$$J = \sum_{j=N_1}^{N_2}(y_r(t+j) - y_m(t+j))^2 + \rho\sum_{j=1}^{N_u}(u'(t+j-1) - u'(t+j-2))^2 \qquad (2.62)$$

Where: N_1, N_2 and N_u define the horizons over which the tracking error and control increments are evaluated, u' is the tentative control signal, y_r is the desired response and y_m is the network model response, ρ determines the contribution that the sum of the squares of the control increments has on the performance index [36].

NN based predictive controllers have been used extensively to solve non-linear dynamical problems, particularly those experiencing time delays. Trajanoski and Wach [37], for instance, use a NN predictive controller in a closed-loop insulin delivery system. The proposed control strategy is based on offline system identification and is carried out in simulation only. The system to be controlled is noted to be non-linear in nature with many unknowns. According to their simulation results, stable control is achievable even in the presence of large noise levels or for unknown or variable time delays. Jin-quan and Lewis [38] discuss a new recurrent NN predictive feedback control structure for a class of non-linear dynamic time-delayed systems. The proposed control structure consists of a linearized subsystem local to the controlled plant and a remote predictive controller located at a master command station. In the local linearized subsystem, a recurrent NN with an online weight tuning algorithm is employed to approximate the dynamics of the time-delay-free non-linear plant. The result is an adaptive NN compensation scheme for non-linear systems with time delays.

2.6.2 NN PID Controllers

Traditional control strategies have proven to be quite robust as long as they remain within their operational range. NNs have the ability to learn and adapt to non-linear plant variation. By combining aspects of traditional and intelligent control strategies, superior controllers are born that maintain the best of both worlds. Hagan and Demuth [31] on this note, discuss an adaptive PID controller based on the Error Recurrent (ER) NN. Because of its fast tracking capabilities, it is possible to design a real time controller based on NNs. They propose a NN-PID controller that has the robust features of traditional controllers as well as the adaptive nature of NNs. In the proposed architecture, the hidden layer neurons simply work as PID controller terms as shown in Figure 2.29. Adaptive control is performed through an online learning process. Results from their research showed that the NN-PID controller is robust but is generally slower than the standard PID controller.

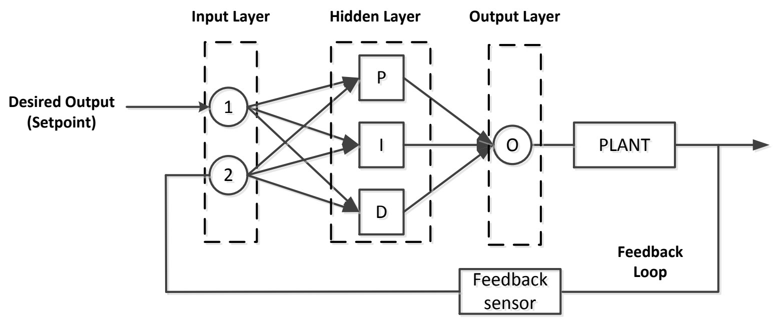


Figure 2.29: PID-NN architecture [13]

Cho and Kim [4] in their research examine the precise control of an AC servo motor using a NN-PID controller. In their research an online-type NN-PID controller using past data as well as current inputs and outputs in order to control the AC servo motor is implemented. This type of controller has proven to be a robust controller especially in dealing with load disturbances and/ or sensor noise. Their findings show that the proposed controller (see Figure 2.30) can tune the conventional PID controller using an indirect NN which can be controlled by only inputs and outputs even in a Jacobian of unknowns. The indirect NNs is composed of an emulator supervising control object and the NN controlling object. They conclude that the NN-PID controller is superior to most other control strategies.

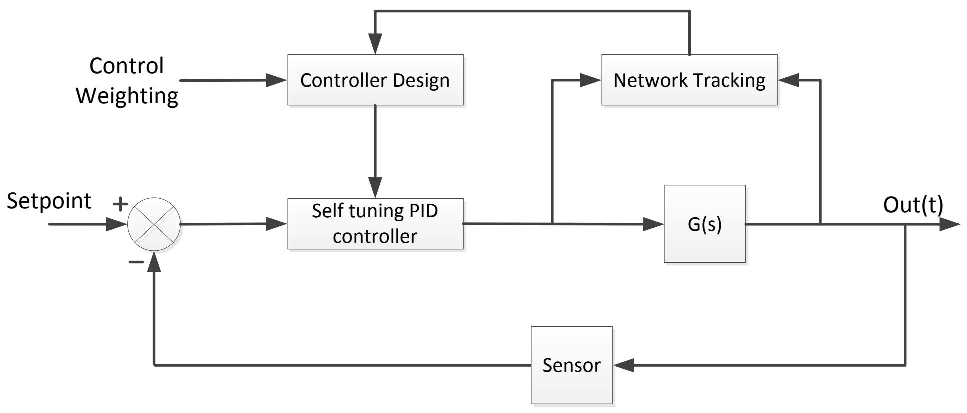


Figure 2.30: NN-PID control system architecture

Lee and Park [39] discuss the drawbacks of using only a NN as a controller. They conclude from their experiments that:

- The control system is mostly non-operational during the training process,
- Unpredictable disturbances cannot be eliminated

Hsu et al. [40] highlight how a conventional multi-loop PID controller is combined in parallel with a multi-layer feed forward NN [39]. Such a system has the following advantages over one in which only a NN is used:

- The controlled plant remains fully operational and flexible even when the NN is inoperable.
- The traditional PID controller is robust and guarantees a zero offset at steady state whereas even well trained NNs are unable to guarantee a zero offset at steady state.

Lee and Park [39] make use of the error back-propagation algorithm to train their NN. They illustrate how the connection weights are adjusted mathematically in the steepest decent manner. Abood et al. [41] in a similar work suggest an offline, 3 layer Neuro-PID controller with 4 input neurons, 5 hidden layer neurons and 3 output layer neurons that form the PID controller gains. The resultant PID gains are then fed into a standard PID controller and used to control a dynamic power system. In comparison with the traditional PID controller, their results show the Neuro-PID controller’s performance to be superior particularly, in instances where the load changed in the power system. Despite their preference for fuzzy-PID controllers, another study by Yongquan et al. [42] shows how an NN-PID controller may be used to stabilize linear or non-linear plants alike.

2.6.3 Model Predictive Control (MPC)

MPC is an advanced, optimization based control strategy applicable to a wide range of industrial applications such as chemical plants and internal combustion engines. It is truly a model based control strategy that falls into the class of receding horizon control algorithms [43]. There are several restrictions on applying new control methods in industrial applications for example: (i) new methods are usually not available in a 'ready-to-use' industrial format, (ii) the hardware requirements are relatively high due to the complexity of implementation and computational demands; (iii) the complexity of implementation and maintenance makes the methods unattractive to non-specialized engineers. However, there are a number of instances where "new" control methods have been successfully implemented on standard industrial computer platforms such as the PLC. Valencia-Palomo and Rossiter [44] highlight one such instance in their study. They demonstrate how a MPC can be coded into a PLC using a standard industrial programming language to make MPC an accessible alternative for low level control loops. Most MPC applications include plants having multiple inputs and outputs.

2.7 Programmable Logic Controllers

Early electrical control was based on relays. Modern control systems still include relays, but these are rarely used for logic. Most modern controllers use a computer to achieve control. The dominating industrial computer platform is the PLC. PLCs offer numerous advantages [45]:

- Cost-effective solution for controlling complex systems
- Computational abilities allow more sophisticated control
- Troubleshooting aids make programming easier and reduce downtime
- Reliable components make these likely to operate for years before reaching failure

2.7.1 Selecting a PLC

After the planning phase of a process, the necessary automation equipment can be easily selected. This decision is usually based upon the following requirements [45]:

- Number of logical inputs and outputs required
- Memory requirement (1Kb and up) depending on the size of the user program
- Number of special I/O modules (high speed modules, communication modules etc.)
- Scan time: High speed processes will require shorter scan times
- Communication to remote stations or to other devices
- Programming language to be used

2.7.2 Communication with PLCs

Many control systems use networks to communicate with other controllers and computers. Typical applications include [45]:

- Data acquisition tasks
- Remote monitoring and control applications

A wide variety of networks are commercially available, and each has particular strengths and weaknesses. Certain field networks such as Actuator Sensor Interface (ASI), Devicenet, Interbus, Profibus and Industrial Ethernet have become industrial standards. Industrial Ethernet has become by far the most widely used networking medium in industry because of its open protocol. Profibus is another popular choice of network in industry and may utilize RS-485, Ethernet and fibre optics.

2.7.3 Advanced Control using PLCs

Most control algorithms such as PID, LQR, NN etc. are, in modern times, implemented on digital platforms. In digital systems (discussed in detail in section 2.8), sampling is arbitrary. When a digital computer is used to implement controllers, the ideal sequence of operation is as follows:

- Wait for clock interrupt
- Read analogue input
- Compute control signal
- Set analogue output
- Update controller variables

A cyclic interrupt is thus required to maintain a fixed sampling rate and thus allow the control algorithms to function correctly. It is imperative that the processing delays are determined and added up for each step given above. Failure to do so may result in severe 'lagging'. This information is also useful when it comes to choosing the specific hardware that will work for a particular application. PLCs are often used to implement control sequences in open-loop or closed-loop control. The most common industrial programming languages include [44]:

- Ladder Diagram: graphical language that uses a standard set of symbols to represent relay logic. The basic elements are coils and contacts that are connected by links. Ladder programming is highly visual and easy to understand and diagnose.
- Function Block Diagram (FBD): is a graphical language that corresponds to digital circuits including OR gates, AND gates and so on.
- Structured Text Language (STL): is a general purpose, high level programming language similar to PASCAL or C. STL is particularly useful for complex arithmetic calculations but also allows the use of conditional statements such as IF, THEN, ELSE, WHILE and CASE structures.

According to Abdi et al. [46], NNs have not been used extensively on PLCs. PLCs are predominantly used in industrial applications for interlocking and supervisory control; most of which is based on simple logic or mathematical operations. Abdi et al. [46] further propose and implement a three layer perceptron NN in a Function Block (FB) on a Siemens S7-300 PLC. The network has a manual and automatic mode; in the manual mode, the network parameters such as the weights are selected by the user. In automatic mode, network parameters are calculated automatically by the back propagation training method. The network is completely trained within about 10 minutes after which the PLC is placed into run mode. Historical input and output data is stored in the memory of the PLC and used for training purposes. The designed NN controller is based on target data obtained from an ordinary PID controller. Similarly, El Monsef and Areed [47] implemented a NN on a PLC to control the oil level in a tank. Topalova [48] goes further in his study and discusses the implementation of a Modular multi-Applicable NN (MANN) classification structure for industrial implementation on PLCs. He makes use of an S7-317 PLC from Siemens to implement his strategy. The resulting algorithms are ideally used for visual recognition tasks but are not limited to these. In the first step of implementation training data is acquired. Next, input variables are selected based on their level of influence. In the final stage, the NN is optimized by reducing the input parameter set without losing important information. He also makes use of a PC running NeuroSystems (Siemens 2006) software developed for defining NN topology and parameters. The network is thus trained offline and then downloaded to the PLC in the form of FBs. The system is then tested with 20 exemplars for 4 different classes. The obtained accuracy is found to be between 87-95 %. He concludes that the MLPNN structure yields good results concerning the recognition accuracy. The reduced number of inputs is a good precondition for also minimizing the number of neurons in the hidden layer. In this way the total number of weights is also reduced which affects the computational resources when implementing the trained MLP structure in the PLC for real time work.

2.7.4 Connectivity of PLC to Matlab/ Simulink

Although SCADA (Supervisory Control and Data Acquisition) systems do exist to allow users to access PLC data areas for purposes of monitoring, trending and control, these software packages are either very expensive or limited in terms of experimental scope. It would be a great advantage, if PLC data could be accessed and assessed in a software environment that is more suited to intensive mathematical computation and data manipulation such as Matlab or Simulink. Persin et al. [49] discuss some of these exact limitations with existing SCADA systems and suggest the use of OPC (OLE (Object Linking and Embedding) for Process Control) server and client concepts as a possible cost effective solution to create a real time connection between Matlab/ Simulink and the PLC as shown in Figure 2.31.

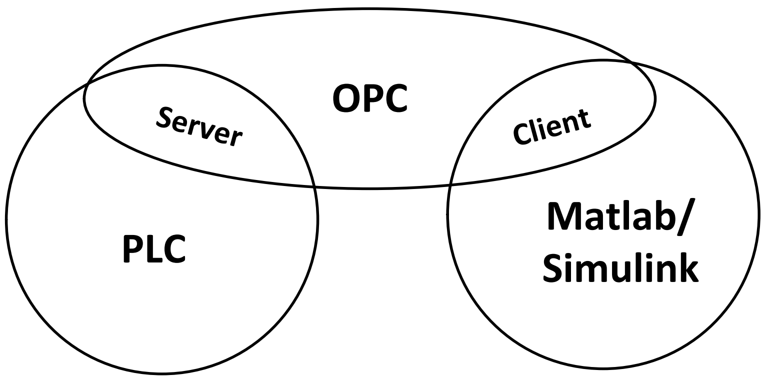


Figure 2.31: Matlab and PLC Connection using OPC

In the same work by Persin et al. [49], the data refresh rate is determined to be between 60 ms to 700 ms over a Multi-Point Interface (MPI) connection depending on the amount of data being processed at that time. With this capability, it is suggested that advanced control strategies that are not directly implementable on a PLC be executed directly in the Matlab/ Simulink environment. Process variables in the PLC can then be altered via the established OPC connection [50].

2.7.5 SCADA and remote control

SCADA systems are used for monitoring and control of various industrial processes. Such systems are also finding their way into other notable areas such as buildings, homes, ships and even more recently in experimental laboratories. SCADA systems have made tremendous progress over recent years in terms of functionality, scalability, performance and openness. SCADA can be divided into two categories: the ‘client layer’ which caters for the Human Machine Interface (HMI) and the ‘data server layer’ which handles most of the process data control activities. The data servers communicate with devices in the field though process controllers such as PLCs over fieldbuses or networks. A network that is commonly used is Ethernet which utilizes the globally adopted TCP/ IP protocol [51]. SCADA software can be deployed on most normal PC’s or on HMIs. Remote, wirelessly controlled SCADA systems are becoming more popular in setups that require distributed operation and interfacing such as in mobile robots and rotating equipment where physical cabling would be a hindrance to normal operation.

There are a number of publications in which wireless SCADA systems are utilized. Bai et al. [52] for instance, implement a Wireless Sensor Network (WSN) on a wind power plant which has the merits of distributed information processing, remote monitoring and control. A wireless SCADA system is then used to pool together and display all information from the WSN. Bayindir and Cetinceviz [53] in their study propose and develop a laboratory based, wirelessly controlled pumping station that provides a convenient solution for process plants where cabling is not possible. The pumping station makes use of a rugged wireless pressure transmitter and water pump that are designed to withstand harsh environments. These in turn are linked up to a PLC and SCADA system over a Wireless LAN connection. Their research concludes that unlike wired systems, their wirelessly controlled systems allow for enhanced mobility, scalability and flexibility.

The implementation of wireless technology could potentially lower installation and maintenance costs. Limitations of wireless technology include, for instance, security issues, reliability, coverage area and fault tolerance. For this reason, it is strongly advised to use wireless products that are specifically designed for the area in which they will be used. The pump station makes use of an ET200s distributed I/O module connected to a SCALANCE W744-1 PRO client module that wirelessly links to a SCALANCE W788-1 PRO wireless access point. A Siemens S7-300 PLC with a CP343-1 Lean Communications Processor (CP) is then connected to this wireless network through the access point for remote control capability. Notably, Siemens provides a complete industrial solution for this application.

2.8 Implementation of controllers on to Digital platforms

Control algorithms are usually implemented in machine code (binary commands) or higher level languages such as assembler, basic or C/ C++. Most other programmable devices use variations of these higher level languages. PLCs for instance use SCL (Structured Control Language) which is a language based on C that easily allows the use of mathematical functions, comparison functions, loops and control structures. The advantages of using microprocessors for control are:

- Programs may be easily modified
- Advanced control laws can be implemented on such systems

The major disadvantage of using a microprocessor is that:

- They work in discrete time only
- Additional hardware may be required for analogue to digital conversion (ADC)
- The sampling rate must be suited to the application

The z-transform is the discrete time counter-part of the Laplace transform. Like the Laplace transform, the z-transform is essential when it comes to implementation of Laplace domain algorithms on to digital, sampled systems. Standard procedures exist for conversion between the ‘s’ domain and the ‘z’ domain. To save the effort of having to make numerous calculations, these conversions are handled relatively easily in mathematical programs such as Matlab.

A digital control system may be represented by the block diagram shown in Figure 2.32 [17, 54].

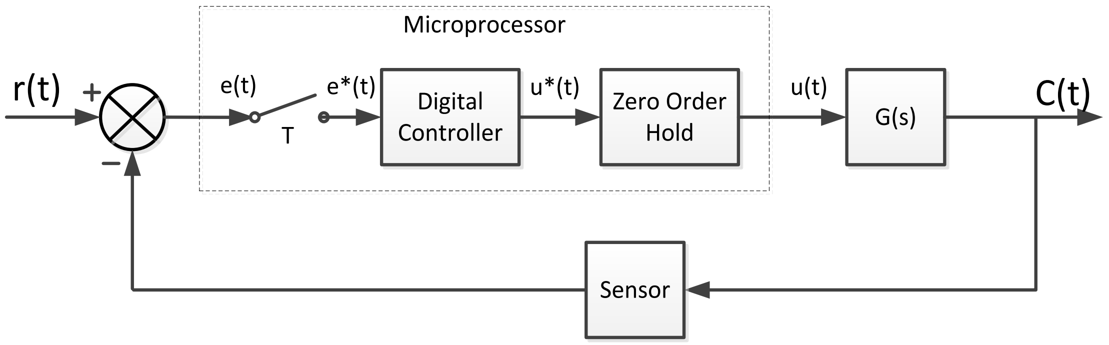


Figure 2.32: Digital Control System

It is possible to map from the s-plane to the z-plane using the relationship:

$$z = e^{sT} \tag{2.63}$$

And

$$s = \sigma \pm j\omega \tag{2.64}$$

Therefore,

$$z = e^{(\sigma \pm j\omega)T} = e^{\sigma T} e^{j\omega T} \tag{2.65}$$

Where

$$T = \frac{2\pi}{\omega_s} \tag{2.66}$$

Now, in order to prepare the continuous controller arrangements discussed thus far for implementation onto digital systems, let the differentiation of an error signal be represented as:

$$u(t) = \frac{de}{dt} \tag{2.67}$$

In a discrete system, this differentiation can be approximated to:

$$u(kT) = \frac{e(kT) - e(k-1)T}{T} \tag{2.68}$$

Considering the general continuous PID controller discussed in the above sections, the transfer function may be given as [7, 17]:

$$\frac{U}{E}(s) = \frac{K_1(T_i T_d s^2 + T_i s + 1)}{T_i s} \tag{2.69}$$

This can be simplified by using Tustin's rule which gives a better approximation to integration:

$$s = \frac{2(z-1)}{T(z+1)} \tag{2.70}$$

Yielding:

$$\frac{U}{E}(z) = K_1 \frac{(b_2 z^2 + b_1 z + b_0)}{z(z-1)} \tag{2.71}$$

Where:

$$b_0 = \frac{T_d}{T} \tag{2.72}$$

$$b_1 = \left(\frac{T}{2T_i} - \frac{2T_d}{T} - 1\right) \tag{2.73}$$

$$b_2 = \left(\frac{T}{2T_i} + \frac{T_d}{T} + 1\right) \tag{2.74}$$

In order to implement this digital controller on to a digital system the difference equation must be obtained by dividing Eq. 2.71 throughout by the highest power of z, giving:

$$u(kT) = \left(\frac{1}{b_2}\right) e(kT) - \left(\frac{b_1}{b_2}\right) u(k-1)T - \left(\frac{b_0}{b_2}\right) u(k-2)T \qquad (2.75)$$

Many industrial processes use PLCs as already outlined in section 2.7. Most PLCs come with pre-programmed control blocks that can be found in their program libraries. These are usually easier to implement than self-coded controllers. For example, Van Dessel [55], describes how a PID control block from the function library is simply imported and used in a Phoenix Contact PLC to control the level in a tank.

2.9 Summary

In the first part of this chapter, standard control principles and methods are reviewed. These include: principles for modelling of mechanical, electrical and electro-mechanical systems; methods for system analysis and controller design techniques. Following this, the most dominant traditional control techniques in use in industry today are examined including PID, Lead, Lead-Lag and LQR. The Chapter goes on to explore modern control approaches that include pure NN control, adaptive PID control and MPC. Finally, light is shed on PLCs and their applicability in the manufacturing and process industries as well as the possibility of real time interconnection of PLCs to scientific software tools such as Matlab/ Simulink for the purpose of research.

From the literature review, it is clear that traditional control strategies presently overshadow modern control strategies, especially where implementation on PLCs is concerned. A major reason for this is because traditional controllers require much less processing power and are generally easier to implement and tune than modern controllers. Furthermore, modern control strategies often require historical plant data and additional sensory information which in the real world may be too difficult or costly to obtain. On the other hand, modern, NN based control is a growing field of study that is slowly becoming established in areas where traditional strategies fall short. This is especially true when dealing with systems that are non-linear in nature or are otherwise too complex to model. The benefit of combining traditional and modern NN based control strategies is also highlighted. This is particularly seen in the case of the NN-PID controller where the PID gains of a traditional PID controller are determined by an artificial NN. These ‘combined’ control strategies are often simpler to implement than the pure NN strategies.

Chapter 3.
MATHEMATICAL TREATMENT AND SIMULATION OF THE BALL-ON-WHEEL (BOW) SYSTEM

3.1 Modelling of the BOW system

In this chapter, the BOW system is mathematically modelled. Its non-linear system equations are presented and then converted into the state space form to allow for linearization. Thereafter, the BOW system is analysed using Matlab to investigate the position of its poles and zeros in the s-plane and to observe its frequency domain characteristics. Various methods for deriving the dynamic equations of a system have been formulated over the last century. One method commonly adopted in analysing complex systems with multiple degrees of freedom is the Lagrangian dynamics technique. This technique is based on the concept of generalized coordinates and generalized forces.

The Lagrangian, L, is defined as the difference between the kinetic and potential energies of all of the particles of the system expressed in generalized coordinates as shown in Eq. 3.1.

$$L = E_K - E_P \tag{3.1}$$

Lagrange's equation for a system with both conservative and non-conservative forces is given as:

$$F_i = \frac{\delta}{\delta t}\left(\frac{\delta L}{\delta \dot{q}_i}\right) - \frac{\delta L}{\delta q_i} \tag{3.2}$$

Where q_i is the generalized coordinate associated with the force F_i. In the case of rotary joints, the generalized forces become torques, τ, and the generalized coordinates become angular displacements, θ [56, 57].

Consider the free-body diagram of the BOW system as shown in Figure 3.1:

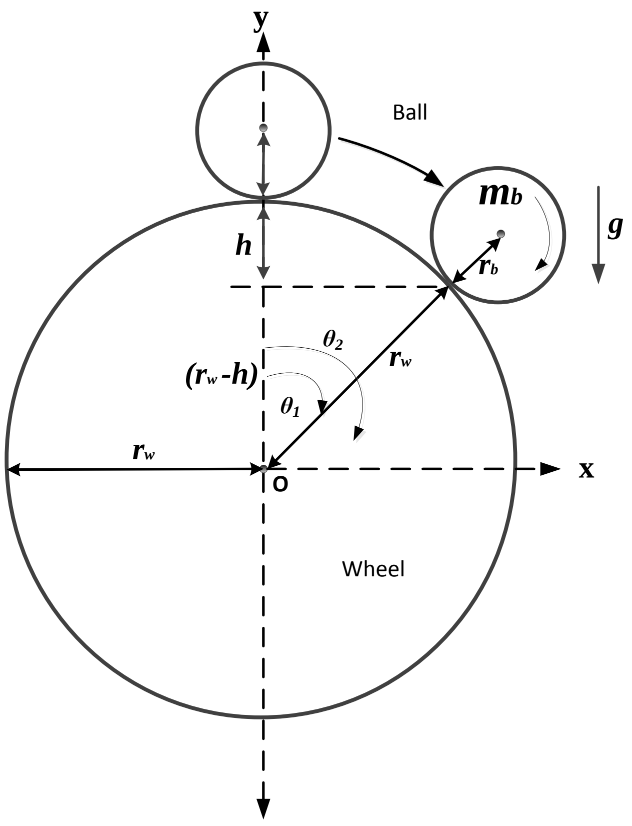


Figure 3.1: Free-body diagram of the BOW System

A spherical ball of mass, m_b, rolls through a small angle θ_1 on the periphery of the wheel under the assumption that the ball rolls without slipping. From Eq. 3.2, it follows that,

$$q = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} \tag{3.3}$$

Where q is the generalized coordinates of the system; θ_1 is the angle between the y-axis and the centre of the ball and θ_2 is the angle of rotation of the wheel (see Figure 3.1 above).

$$F_i = \begin{bmatrix} 0 \\ \tau \end{bmatrix} \tag{3.4}$$

Where τ is the torque exerted on the wheel.

The mechanical energy of this system is comprised of both potential and kinetic energy of the rolling ball. As the ball rolls down the incline of the wheel, its initial gravitational potential energy is being converted into the two types of kinetic energy; translational and rotational. The ball rolls depending on the distribution of its mass. The moment of inertia is given by the distribution of mass in the ball away from the axis of rotation. A ball with a lot of mass concentrated at the centre is easier to roll than one with less. The larger the moment of inertia is, the smaller the translational velocity that the ball will experience [58].

The principle that can unite a rolling object's rotational and translational kinetic energies is called the parallel axis theorem (see Figure 3.2), given by:

$$I_p = I_{cm} + mh^2 \tag{3.5}$$

$$KE_{Tot} = \frac{1}{2} I_p \omega^2 \tag{3.6}$$

$$KE_{Tot} = \frac{1}{2}(I_{cm} + mh^2)\omega^2 \tag{3.7}$$

$$KE_{Tot} = \frac{1}{2} I_{cm}\omega^2 + \frac{1}{2} mh^2\omega^2 \tag{3.8}$$

If: $h = r, then: \; KE_{Tot} = \frac{1}{2} I_{cm}\omega^2 + \frac{1}{2} mr^2\omega^2$ (3.9)

$$r\omega = v, it\ follows\ that\ r^2\omega^2 = v^2 \tag{3.10}$$

Hence: $KE_{Tot} = \frac{1}{2} I_{cm}\omega^2 + \frac{1}{2} mv^2$ (3.11)

Where,

I_p Balls moment of inertia from any point p.

I_{cm} Balls moment of inertia about the centre of mass

h Perpendicular distance from p to the centre of mass

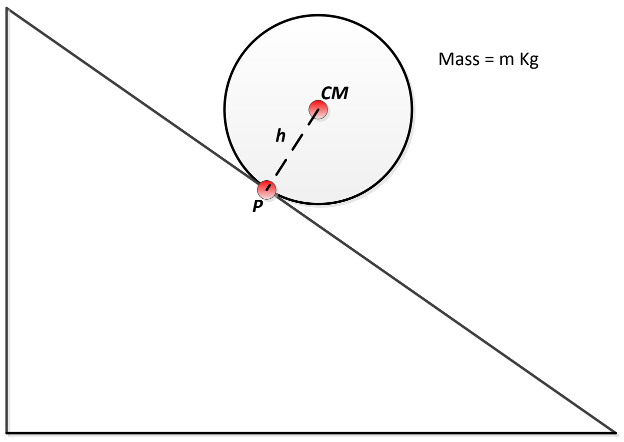


Figure 3.2: Parallel axis theorem

Considering again the BOW system as depicted in Figure 3.1, the kinetic energy, due to motion, possessed by the ball as it rolls on the periphery of the wheel is given as [11]:

$$K_{E\_ball} = K_{E\_Trans} + K_{E\_Rot} \tag{3.12}$$

Where, K_{E_ball} = Total kinetic energy of the ball, K_{E_Trans} = kinetic energy due to translational motion of the ball and K_{E_Rot} = kinetic energy due to rotation of the ball and:

$$K_{E\_Trans} = \frac{1}{2}mv^2 = \frac{1}{2}m_b(r_w + r_b)^2\dot{\theta}_1{}^2 \tag{3.13}$$

Where, m is the mass of the ball and v is its velocity and:

$$K_{E\_Rot} = \frac{1}{2}I_b\omega^2 = \frac{1}{2}I_b\dot{\theta}_3{}^2 \tag{3.14}$$

The moment of inertia of the ball is given by:

$$I_b = \frac{2}{5}m_b{r_b}^2 \tag{3.15}$$

$$K_{E\_ball} = K_{E\_Rot} + K_{E\_Trans} = \frac{1}{2}m_b(r_w + r_b)^2\dot{\theta}_1{}^2 + \frac{1}{2}I_b\dot{\theta}_3{}^2 \tag{3.16}$$

Then according to the Lagrangian equation:

$$L = T - V \tag{3.17}$$

Where T is the kinetic energy of the system and V is the potential energy of the system. The kinetic energy possessed by the wheel due to rotation is given by:

$$K_{E\_wheel} = \frac{1}{2} I_{\,w} \dot{\theta}_2{}^2 \tag{3.18}$$

Where the wheels moment of inertia is given by:

$$I_w = \frac{1}{2} m_w r_w{}^2 \tag{3.19}$$

Therefore the total kinetic energy possessed by the system is given by:

$$T_t = \frac{1}{2} m_b (r_w + r_b)^2 \dot{\theta}_1{}^2 + \frac{1}{5} m_b r_b{}^2 \dot{\theta}_3{}^2 + \frac{1}{4} m_w r_w{}^2 \dot{\theta}_2{}^2 \tag{3.20}$$

Now, since θ_3 (the rolling angle of the ball) is not measurable, it must be expressed in terms of θ_1 and θ_2, giving:

$$T_t = \frac{1}{2} m_b (r_w + r_b)^2 \dot{\theta}_1{}^2 + \frac{1}{5} m_b (r_w \dot{\theta}_2 - r_w \dot{\theta}_1 - r_b \dot{\theta}_1)^2 + \frac{1}{4} m_w r_w{}^2 \dot{\theta}_2{}^2 \tag{3.21}$$

The potential energy possessed by the system is given by

$$V = m_b g (r_w - r_b) cos\theta_1 \tag{3.22}$$

Therefore, according to Eq. 3.17,

$$L = \frac{1}{2} m_b (r_w + r_b)^2 \dot{\theta}_1{}^2 + \frac{1}{5} m_b (r_w \dot{\theta}_2 - r_w \dot{\theta}_1 - r_b \dot{\theta}_1)^2 + \frac{1}{4} m_w r_w{}^2 \dot{\theta}_2{}^2 - g (r_w - r_b) cos\theta_1 \tag{3.23}$$

And according to Eq. 3.2, the non-linear system dynamic equations are then given as:

$$(7r_b + 7r_w)\ddot{\theta}_1 - 2r_w \ddot{\theta}_2 - 5g sin\theta_1 = 0 \tag{3.24}$$

$$(-\frac{2}{5} r_w{}^2 m_b - \frac{2}{5} r_w r_b m_b)\ddot{\theta}_1 + (\frac{1}{2} m_w r_w{}^2 + \frac{2}{5} r_w{}^2 m_b)\, \ddot{\theta}_2 = \tau \tag{3.25}$$

These two equations (Eq. 3.24 & Eq. 3.25) are only true as long as the centripetal force is large enough to maintain circular motion of the ball on the wheel. The system must be defined and modelled in the state space form if the Jacobian method of linearization is to be used; hence the state variables are declared as follows:

$$x_1 = \theta_1 \qquad \dot{x_1} = x_2 = f_1$$

$$x_2 = \dot{\theta}_1 \qquad \dot{x_2} = \ddot{\theta}_1 = f_2$$

$$x_3 = \theta_2 \qquad \dot{x_3} = x_4 = f_3$$

$$x_4 = \dot{\theta}_2 \qquad \dot{x_4} = \ddot{\theta}_2 = f_4$$

$$\dot{X} = AX + BU \tag{3.26}$$

$$Y = CX + DU \tag{3.27}$$

Where:

A = system matrix

B = input Matrix

C = output Matrix

D = feed-forward matrix

The linearized A and B system matrices (now called Jacobian matrices) are given as follows:

$$JA = \begin{bmatrix} \frac{df_1}{dx_1} & \frac{df_1}{dx_2} & \frac{df_1}{dx_3} & \frac{df_1}{dx_4} \\ \frac{df_2}{dx_1} & \frac{df_2}{dx_2} & \frac{df_2}{dx_3} & \frac{df_2}{dx_4} \\ \frac{df_3}{dx_1} & \frac{df_3}{dx_2} & \frac{df_3}{dx_3} & \frac{df_3}{dx_4} \\ \frac{df_4}{dx_1} & \frac{df_4}{dx_2} & \frac{df_4}{dx_3} & \frac{df_4}{dx_4} \end{bmatrix} \tag{3.28}$$

Application of Eq. 3.28 yields Eq. 3.29. To simplify the presentation of these matrices, the notations N, M, R and P are used. These are defined in Eq. 3.36 through to Eq. 3.44.

$$JA = \begin{bmatrix} 0 & 1 & 0 & 1 \\ Ncos\theta_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ Mcos\theta_1 & 0 & 0 & 0 \end{bmatrix} \tag{3.29}$$

$$JB = \begin{bmatrix} \frac{df_1}{d\tau} \\ \frac{df_2}{d\tau} \\ \frac{df_3}{d\tau} \\ \frac{df_4}{d\tau} \end{bmatrix} \tag{3.30}$$

Application of (3.30) yields (3.31):

$$JB = \begin{bmatrix} 0 \\ R \\ 0 \\ P \end{bmatrix} \tag{3.31}$$

Now, in order to linearize the system, make $x_1 = \theta_1 = 0$. Since cos(0) = 1, the system matrices are given as follows:

$$JA = \begin{bmatrix} 0 & 1 & 0 & 1 \\ N & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ M & 0 & 0 & 0 \end{bmatrix} \tag{3.32}$$

$$JB = \begin{bmatrix} 0 \\ R \\ 0 \\ P \end{bmatrix} \tag{3.33}$$

$$JC = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \tag{3.34}$$

$$JD = 0 \tag{3.35}$$

Where:

$$R = \frac{H}{EH - FG} \tag{3.36}$$

$$N = \frac{IF}{EH - FG} \tag{3.37}$$

$$P = \frac{G}{FG - EH} \tag{3.38}$$

$$M = \frac{IE}{FG - EH} \tag{3.39}$$

$$E = -\frac{2}{5} m_b {r_w}^2 - \frac{2}{5} m_b r_w r_b \tag{3.40}$$

$$F = \frac{1}{2} m_w {r_w}^2 + \frac{2}{5} m_b {r_w}^2 \tag{3.41}$$

$$G = -7r_b - 7r_w \tag{3.42}$$

$$H = 2r_w \tag{3.43}$$

$$I = 5g \tag{3.44}$$

3.2 Simulation of the BOW system

Based on the system model determined above, the BOW system is then simulated in Matlab using the initial system parameters given in Table 3-1. The implemented Matlab code can be found in Appendix B.

Table 3-1: Initial System Parameters

Parameter	Value	Comment
m_w	2.5 Kg	Mass of wheel
r_w	0.195 m	Radius of wheel
m_b	0.1 Kg	Mass of ball (hard rubber ball)
r_b	0.025 m	Radius of ball
g	9.81 m/s^2	Acceleration due to gravity

From the simulations, an examination of the controllability and observability matrices (discussed in section 2.4.4) reveals that the BOW system is fully controllable but only partially observable. In the case of the controllability matrix, the rank, as computed in Matlab by the following command: 'rank (ctrb(sys_ss))' is equal to the number of states present in the system, i.e. four. However, in the case of the observability matrix, the rank, as computed by 'rank(obsv(sys_ss))' in Matlab, is equal to two. This vital information reveals that a standard feedback controller (such as PID) may be designed and implemented in order to adequately control the BOW plant. However, in the case of full state feedback (such as LQR), for which all the states must be observable, special care must be taken because control cannot be achieved using this method without the addition of state observers. It is apparent from the rank of the observability matrix that only the system's input and output are directly observable.

The state space system model given in section 3.1 above is converted from the state-space format into the transfer function form for ease of use. This conversion is done in Matlab using the 'state space' to 'transfer function' command (ss2tf). The 'mineral' command is then used to eliminate poles and zeros that have no effect on the response of the system thus reducing the overall complexity of the system.

The resulting transfer function for the BOW system is given as:

$$G(s) = (4.888e^{-15}s + 28.31)/(s^2 - 5.329e^{-15}s - 32.74) \qquad (3.45)$$

A step command to the input of the system yields an infinitely increasing response as the ball rolls off the surface of the wheel (see Figure 3.3.) away from the zero degree equilibrium point.

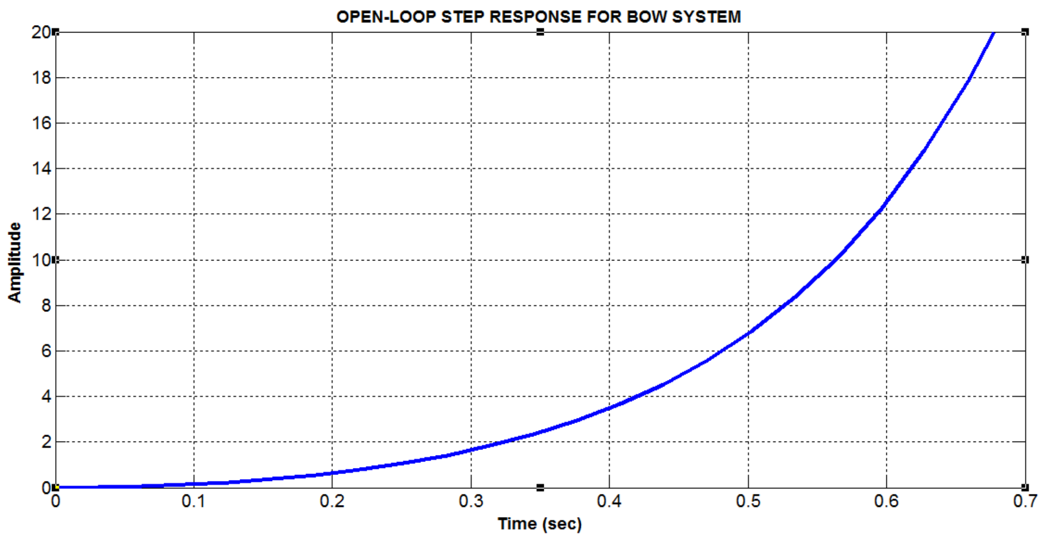


Figure 3.3: Step response of BOW System

The BOW plant is therefore shown to be a naturally unstable system. Control effort is necessary if the system is to be stabilized about the system’s equilibrium point.

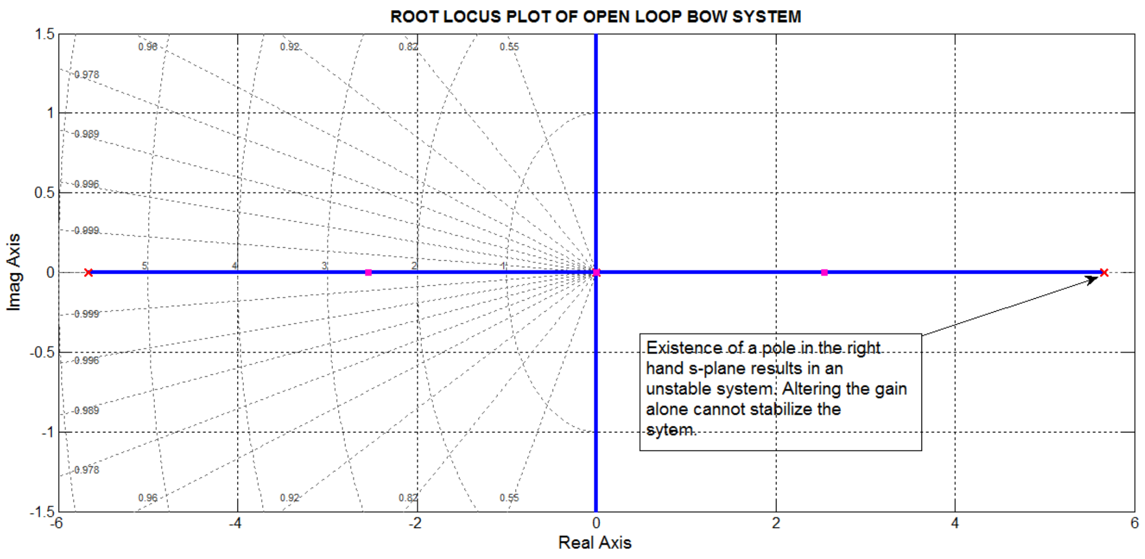


Figure 3.4: Root-locus plot for BOW System

A closer examination of the system dynamics reveals a pole in the right hand s-plane of the root-locus plot (see Figure 3.4). Consequently, this pole yields an unstable system.

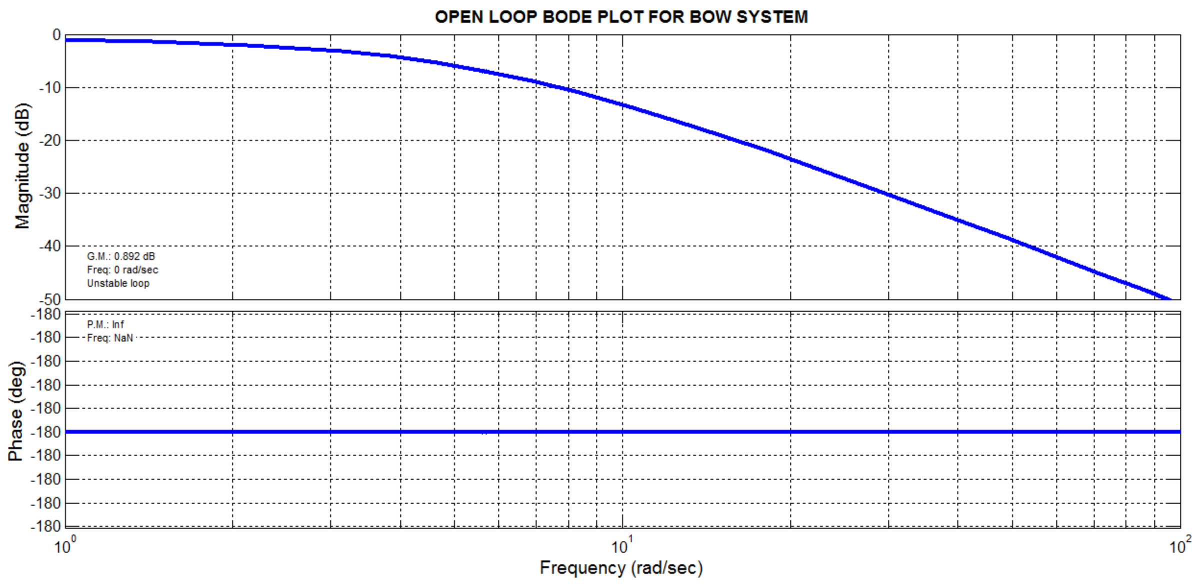


Figure 3.5: Open-loop Bode plot for BOW System

The Bode plots for the uncompensated open-loop BOW system (shown in Figure 3.5) reveal a very small gain margin of 0.892 dB and an infinite phase margin due to the phase never crossing -180 degrees. This again exposes the BOW system to be a naturally unstable system.

Now that the nature of the system is known, the following sections proceed to look into the design of appropriate modes of control that will stabilize the BOW system.

3.3 Controller design requirements

Design requirements for the control of the BOW System are as follows:

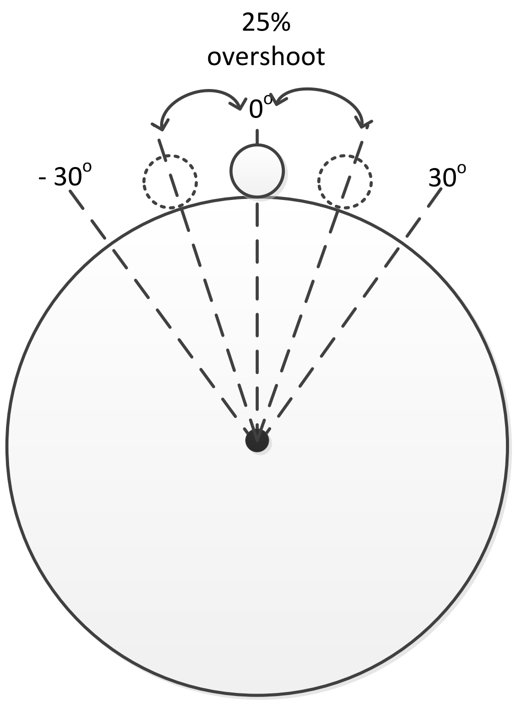


Figure 3.6: Controller design requirements

Referring to Figure 3.6, the equilibrium point of the system is located at the top centre of the wheel (zero degrees) and is therefore the system set-point. An over-damped system is undesirable especially for systems that use tooth belt couplings; hence an overshoot of about 25% (7.5 degrees to either side of the equilibrium point) is acceptable for the BOW system. The rise time must be kept within 500 ms and the settling time also within 500 ms. For feedback control, it may be considered that the laser distance sensor has a unity gain. A steady state error of less than +/- 5 degrees is acceptable.

3.4 Controller design and simulation

In order to stabilize the BOW system about its equilibrium point, various control strategies are designed and simulated on the linearized model of the plant before actual implementation. The following traditional (linear) control algorithms are considered in this section: a PD and PID controller, a Lead and Lead-Lag compensator and a Linear Quadratic Regulator (LQR).

As highlighted in the literature review, the most common of the traditional controllers is the PID controller for its simplicity. (Eq. 3.46) shows the PID algorithm in the parallel form.

$$PID = K_p . e(t) + K_i . \int_0^t e \, . dt + K_d . \frac{de(t)}{dt} \qquad (3.46)$$

The transfer function of the PID controller in the ideal or standard form is given by (3.47):

$$G_c(s) = K_c(1 + T_d s + 1/T_i \, s) \qquad (3.47)$$

PID control, where all three controller elements are used is not always necessary. Depending on the nature of the system needing control, sometimes P action, PD action or PI action will suffice, i.e., not all controller elements need to be used. In the case of the BOW system, a P or PI controller are unable to stabilize the system.

In regard to intelligent control, a NN-Predictive controller and a NN-PID controller are simulated in Matlab/ Simulink.

3.4.1 PD controller design:

Using the SISO design tool in Matlab, a PD controller is designed for the BOW system based on the linearized system model.

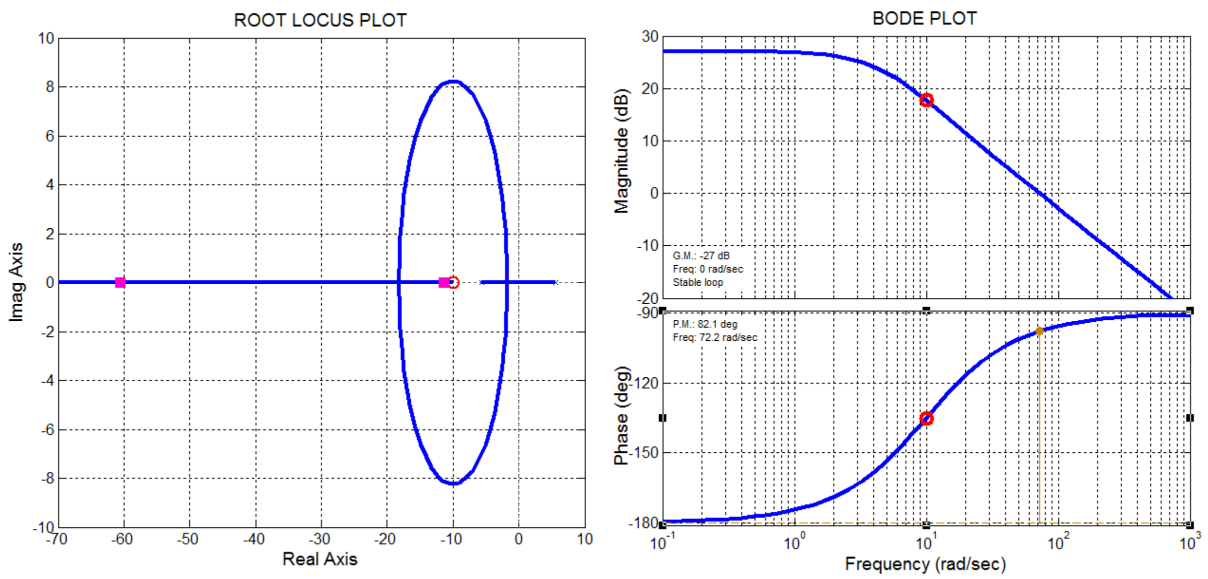


Figure 3.7: Root-locus and Bode plot for BOW System with PD controller

A PD controller is created by placing a real zero at -10 on the real axis of the root-locus plot or at 10 rad/ s on the Bode diagram as shown in Figure 3.7. The resulting PD compensator has a gain margin of -27 dB and a phase margin of 82.1 degrees and can be represented as:

$$C(s) = 24.946 \times \frac{(1+0.1s)}{1} \tag{3.48}$$

The resulting closed-loop step response (see Figure 3.8) shows that the system is stable and will operate well within the required specifications. However, in keeping with the research goals, the PD controller, because of its similarity to the PID and Lead compensators (discussed in section 3.4.2 and 3.4.3 respectively) is deemed unnecessary for actual implementation on the PLC in real time. See Appendix B for the Matlab code.

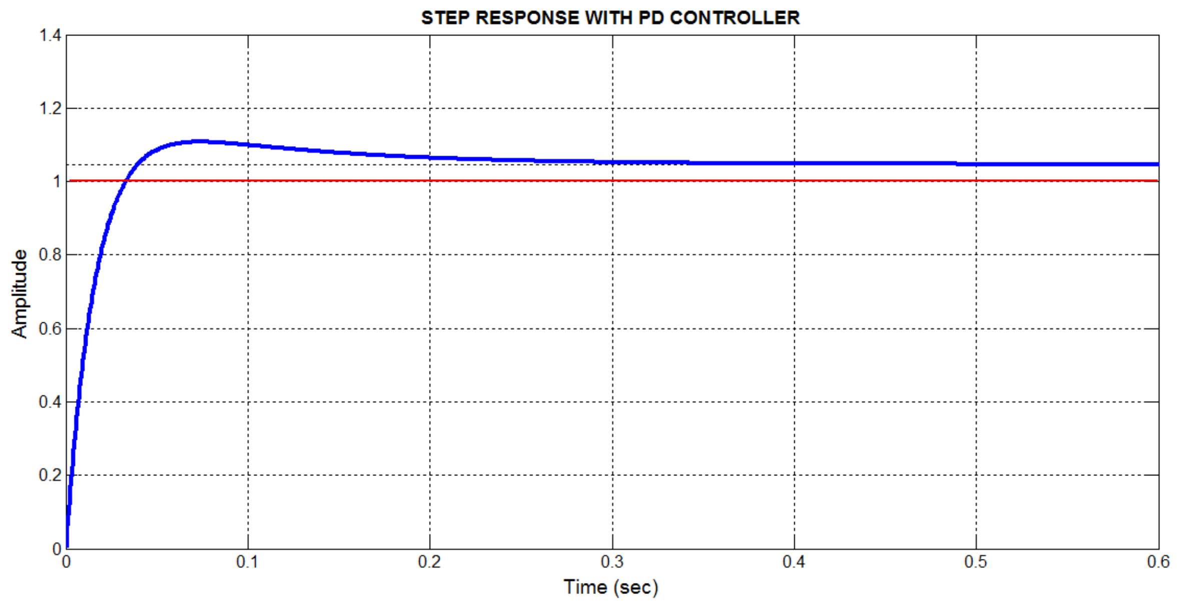


Figure 3.8: Step response of BOW System with PD compensator applied

3.4.2 PID Controller design

Referring to Figure 3.9, a real zero is placed at -5.66 and -0.9 on the real axis of the root-locus. On the Bode diagram a real zero is placed at 5.66 and 0.9 rad/ s. An integrator is placed at the origin of the s-plane. The resulting PID controller takes the following form:

$$C(s) = 112 \times \frac{(1+0.18s)(1+1.1s)}{s} = \frac{22.176s^2+134.4s+112}{s} = \frac{K_d s^2+K_i s+K_p}{s} \qquad (3.49)$$

Equation 3.49 encapsulates the following PID gains: K_p = 134.4, K_i = 112 and K_d =22.176. For even better performance, a first order derivative filter may be added to the basic PID controller, yielding the following transfer function:

$$C(s) = 22.243 \times \frac{(1+0.14s)(1+1.2s)}{s(1+0.0049s)} \qquad (3.50)$$

The derivative filter ensures that the noise created in the system by the derivative gain is significantly reduced. It also generally improves the performance of the controller. The compensated system (even without the filter) is entirely stable and has a gain margin of -40.5 dB and a phase margin of 89.4 degrees.

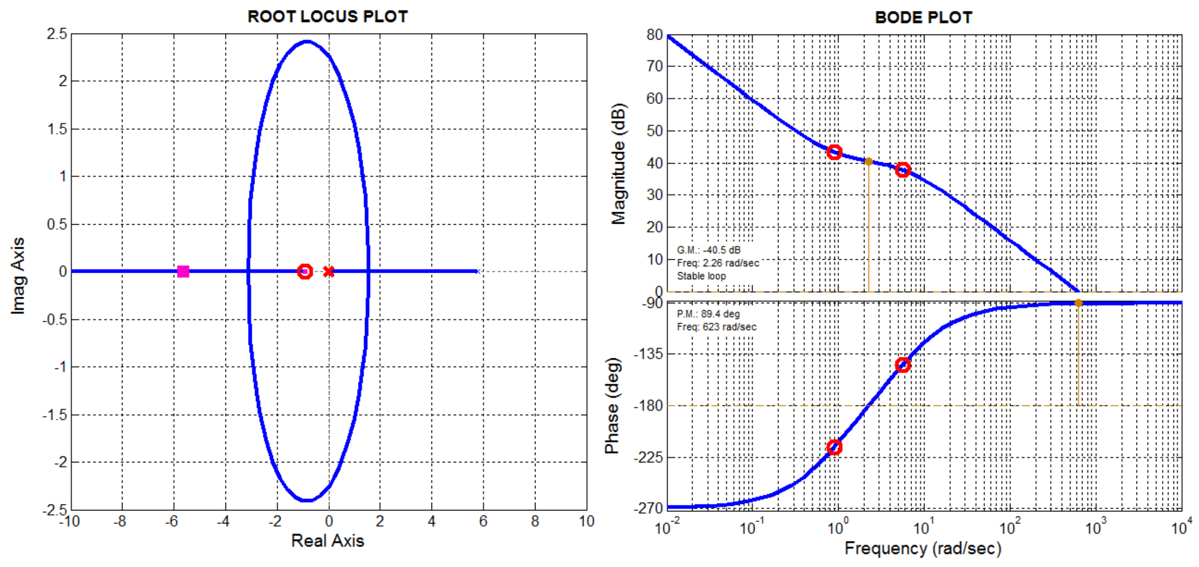


Figure 3.9: Root-locus and Bode plots for compensated BOW System (without filter)

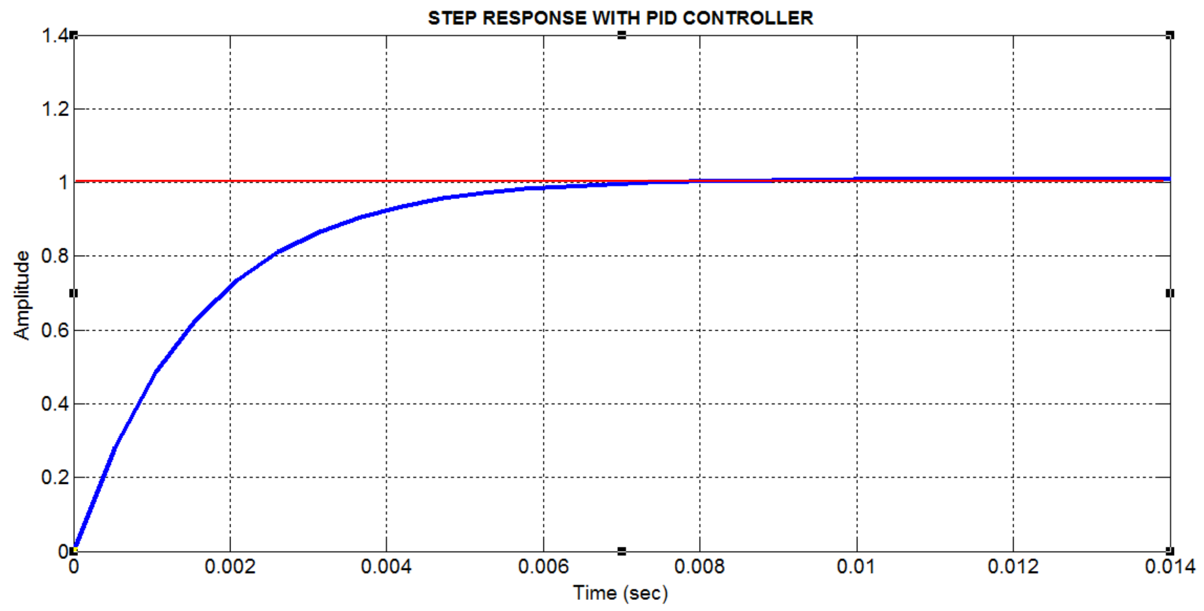


Figure 3.10: Step response simulation with PID controller (without filter)

Figure 3.10 shows the closed-loop step response produced as a result of using the PID controller to control the plant. According to the simulation, the system is able to reach the set-point within 6 ms with no overshoot at all. In reality this would not be possible on the actual BOW system because of mechanical and computational limitations that the model is unable to cater for.

The PID controller shown in Eq. 3.49 is then digitized using Tustin's bilinear transformation and implemented in real time on the PLC (see Matlab code in Appendix B and PLC code in Appendix C).

The designed PID controller was also tested in the Simulink environment. This proved useful for testing its disturbance rejection capability with the inclusion of a saturation limit (of approximately +/- 3 NM of torque based on actual hardware parameters) on the output of the controller to account for the fact that the actual BOW system is only capable of finite (limited) controller action. Figure 3.11 shows the Simulink model that was used to carry out the simulation. It incorporates a specially constructed disturbance signal injected on the plant output as can be seen in Figure 3.12.

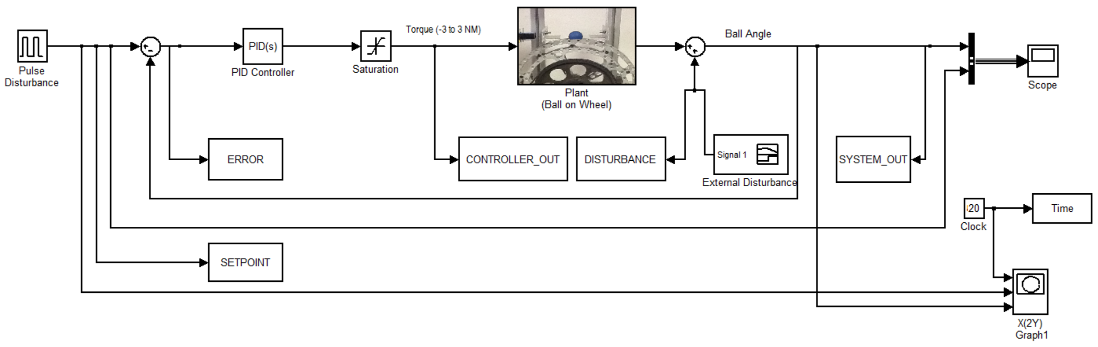


Figure 3.11: Simulink model of PID Controller with disturbance injection

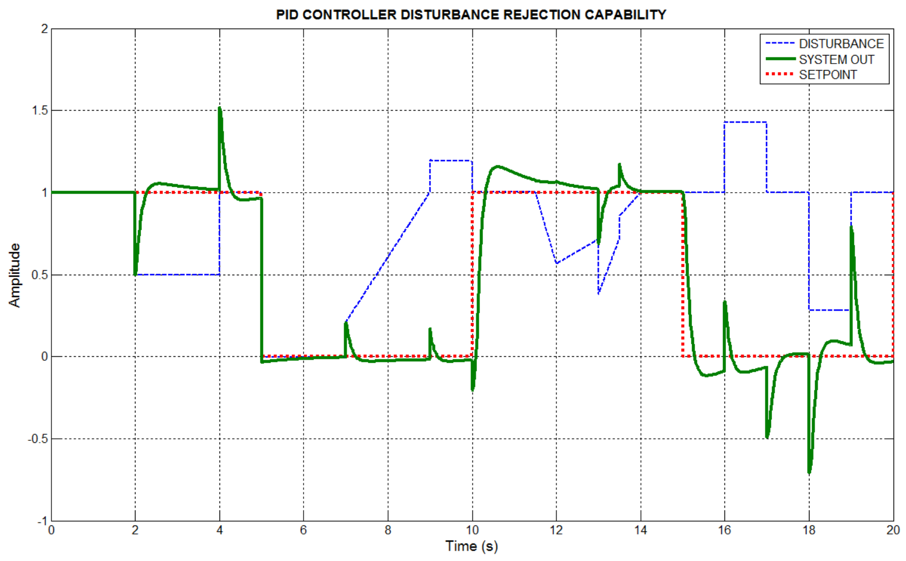


Figure 3.12: Disturbance rejection capability of PID controller

The PID controller, as seen in Figure 3.12, succeeds in rapidly rejecting all external disturbances injected into the system with minimal negative influence on the plant stability.

3.4.3 Lead compensator design

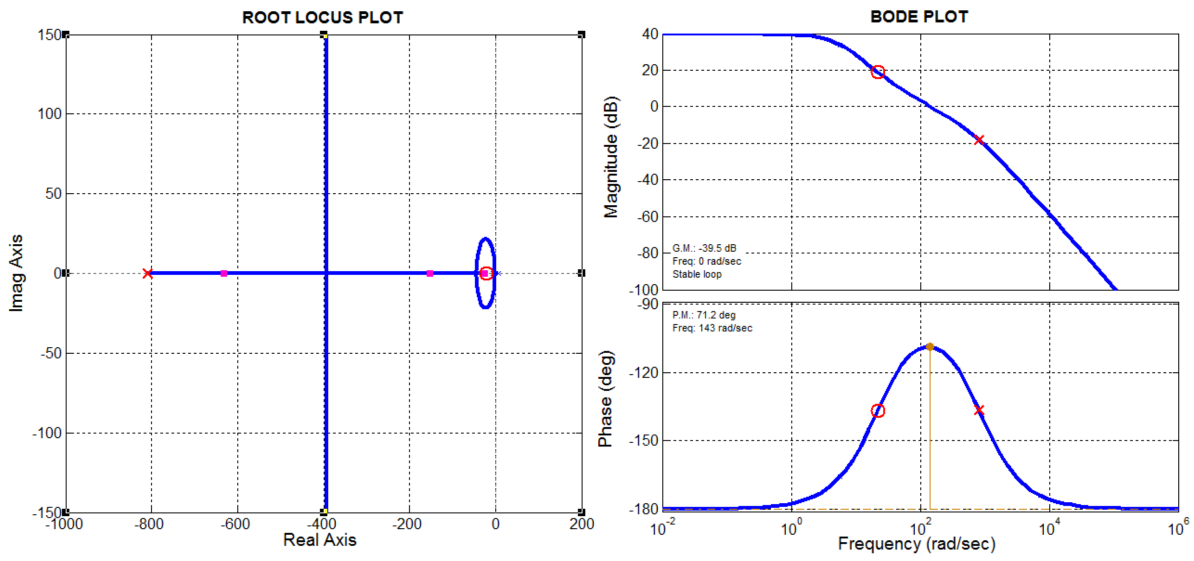


Figure 3.13: Root-Locus and Bode plot for Lead compensator

To create a Lead compensator, a real zero is placed at -1 and a real pole at -810 in the s-plane and at 1 and 810 rad/ sec respectively on the frequency plot as shown in Figure 3.13. The compensator introduces positive phase to the system, effectually making the system more responsive with a gain margin of -39.5 dB and a phase margin of 71.2 degrees. The lead compensator is similar to the PD controller in that they both essentially form a low pass filter. However, the lead controller produces a much faster response.

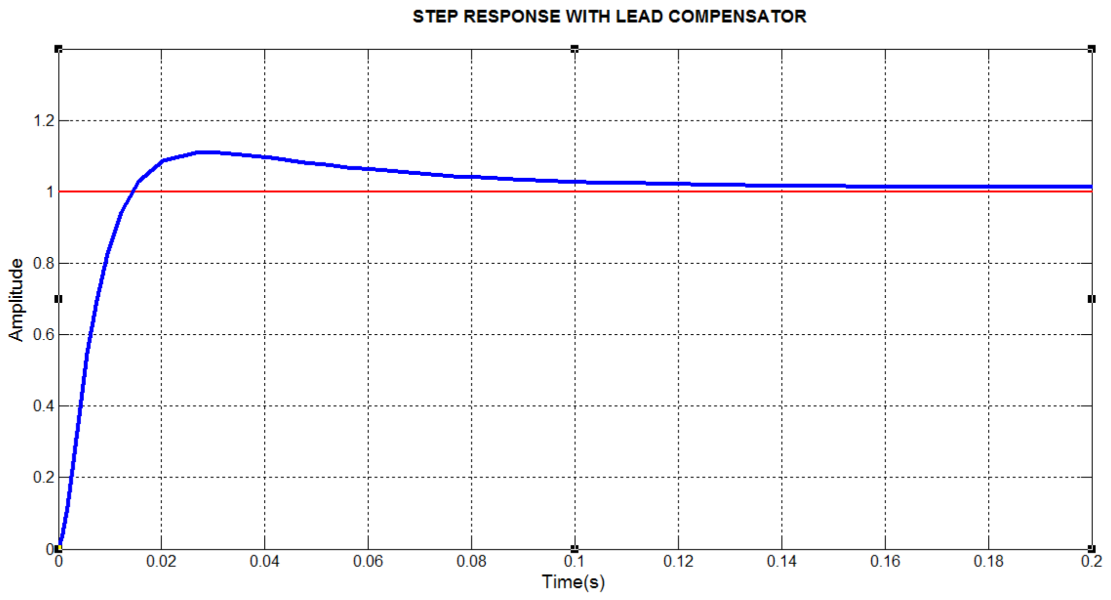


Figure 3.14: Step response for Lead compensator

The Lead compensator transfer function is given as:

$$C(s) = 112 \times \frac{(1+0.045s)}{(1+0.0012s)} \tag{3.51}$$

The result of using this controller is a stable system as shown by the step response in Figure 3.14. Like the previous controller, the Lead compensator is able to withstand and reject external disturbances without losing stability as highlighted in Figure 3.16. Figure 3.15 shows the Simulink model that was used to test the compensator's disturbance rejection capability.

The Lead compensator is digitized using Tustin's bilinear transformation and implemented in real time on the PLC (see Matlab code in Appendix B and PLC code in Appendix C).

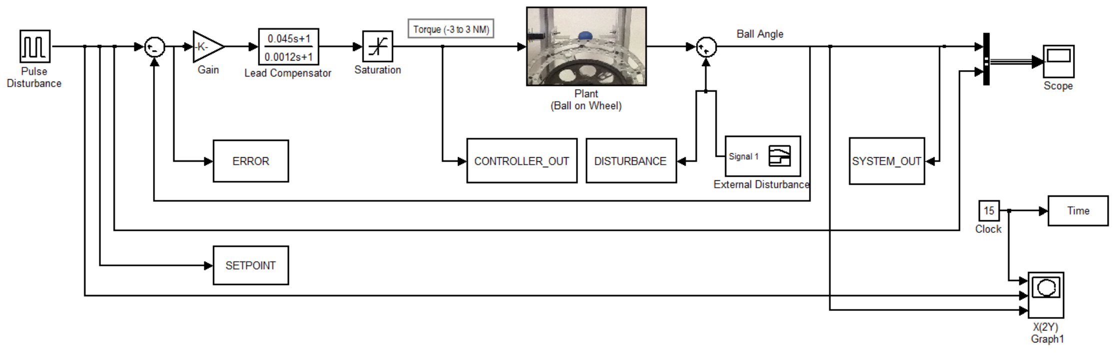


Figure 3.15: Closed-loop Simulink model with Lead compensator

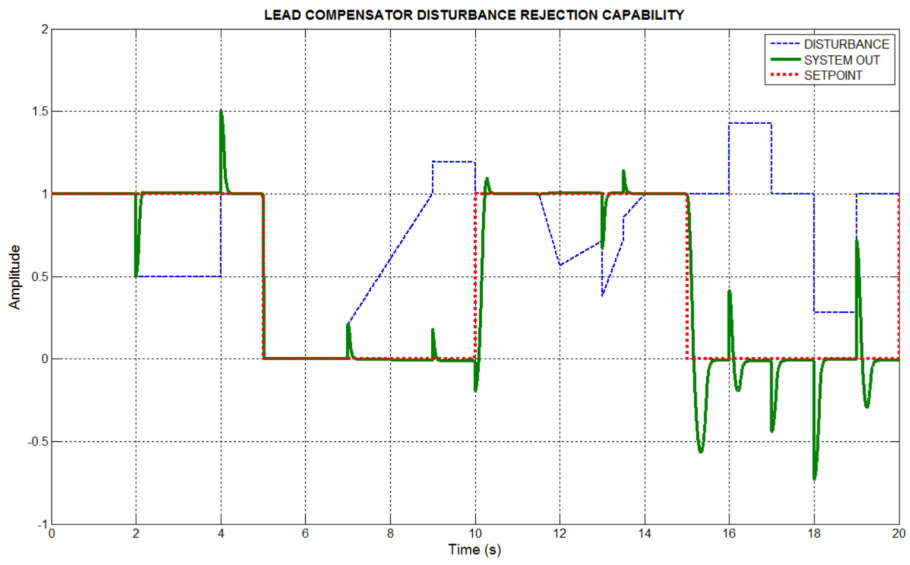


Figure 3.16: Disturbance rejection capability of Lead compensator

3.4.4 Lead-lag compensator design

The Lead-Lag compensator transfer function is given as:

$$C(s) = 49.08 \times \frac{(1+0.16s)(1+1.4\text{e}-6s)}{(1+0.00039s)(1+0.0002s)} \tag{3.52}$$

When the Lead-lag compensator is implemented to control the BOW plant, it can be seen from Figure 3.17 that at low frequencies the closed-loop system exhibits ‘phase lead’ owing to a real zero placed at -6.4 and a real pole at –2544.7 in the s-plane (i.e. 6.4 and 2544.7 rad/ sec respectively on Bode plot). The system then exhibits a phase lag at higher frequencies because of a real zero placed at –7.27e5 and a real pole at -5.06e3 in the s-plane (i.e. 7.27e5 and 5.06e3 rad/ sec respectively on Bode plot). Lower frequencies are amplified, while higher frequencies are attenuated. Furthermore, the compensator produces a gain and phase margin of 30.9 dB and 81 degrees respectively.

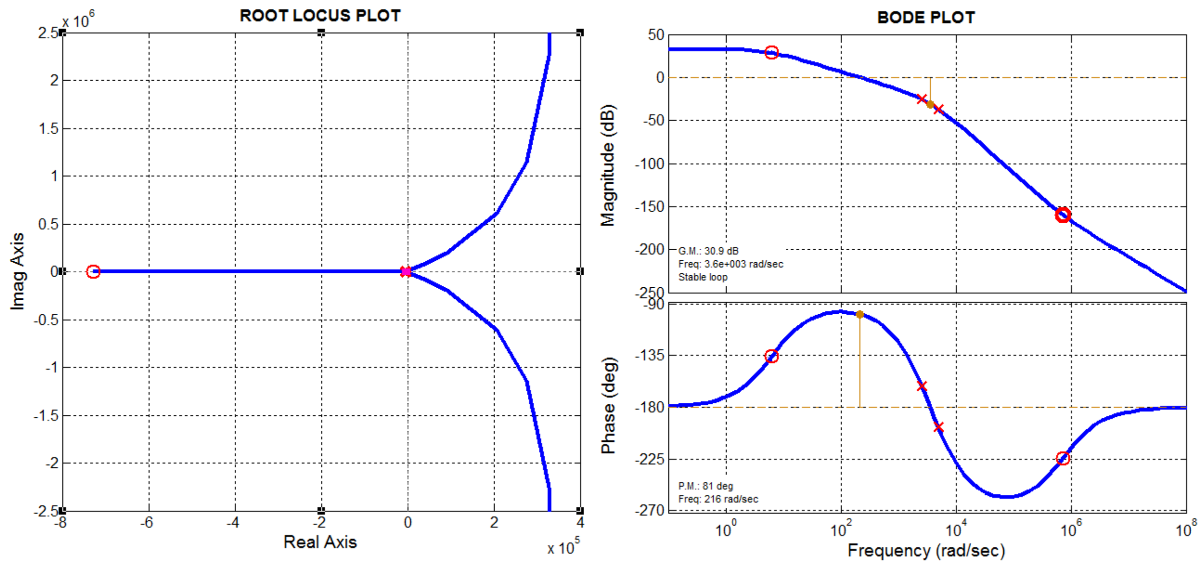


Figure 3.17: Root-Locus and Bode plot for Lead-Lag compensator

A step response (see Figure 3.18) reveals a stable system similar to that produced by the PID controller. The Lead-lag controller is able to reject all external disturbances subjected to the plant with little effect on the stability. This is clearly seen in Figure 3.20. Figure 3.19 shows the Simulink model that was used to test the capability of the compensator with the influence of an external disturbance.

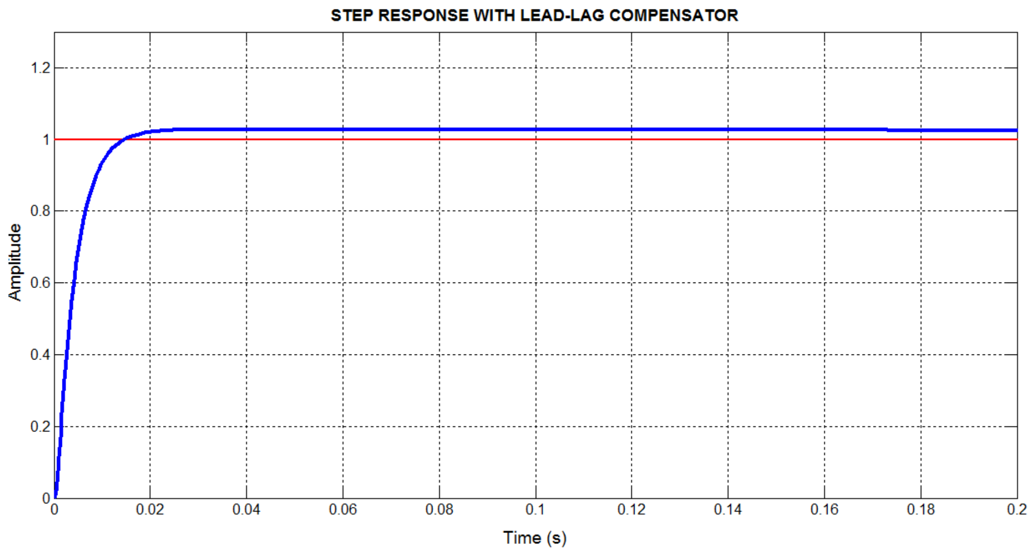


Figure 3.18: Step response for Lead-Lag compensator

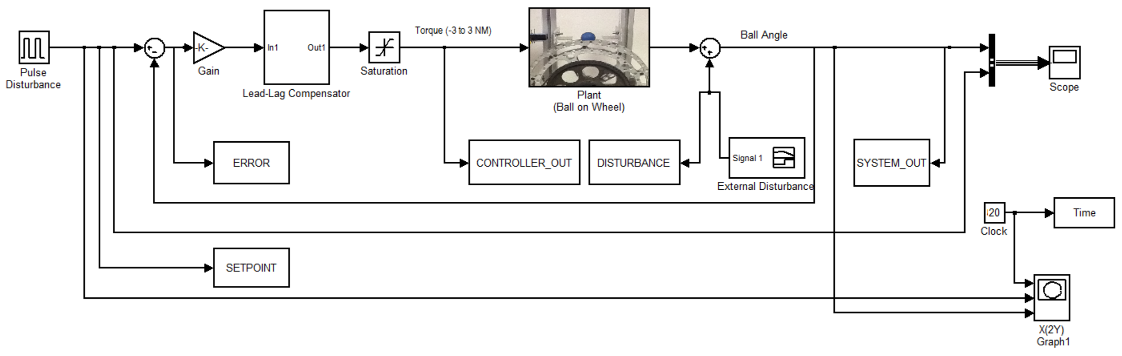


Figure 3.19: Simulink model of Lead-Lag compensator with disturbance injection

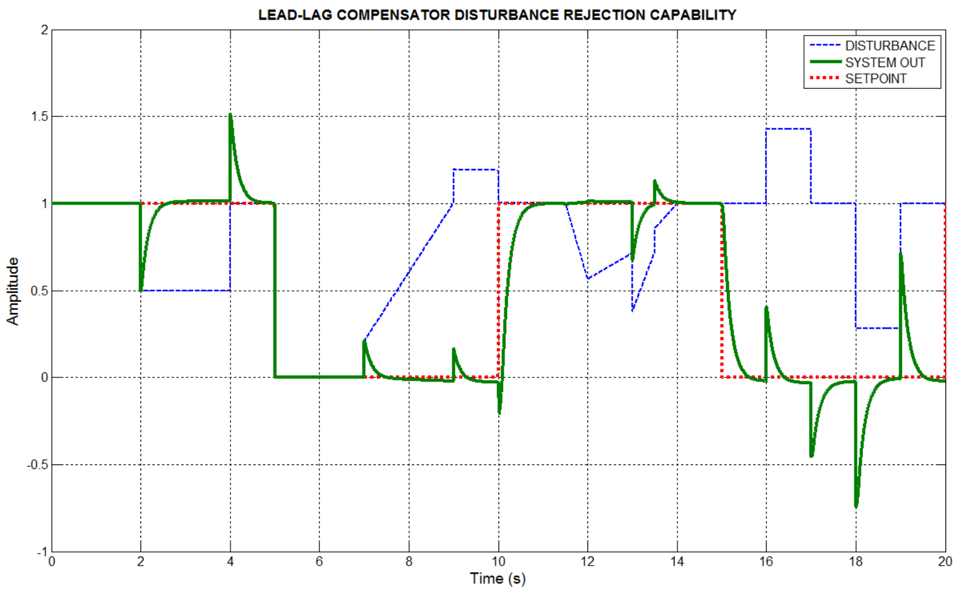


Figure 3.20: Disturbance rejection capability of Lead-Lag compensator

The Lead-Lag compensator is digitized using Tustin's bilinear transformation and implemented in real time on the PLC (see Matlab code in Appendix B and PLC code in Appendix C).

3.4.5 LQR – Linear Quadratic Regulator design

Assuming that all state variables are measurable, the vector of state-feedback control gains (K) must be found in order to control the BOW system. The first step in designing this controller is to determine that the system is in fact controllable. Satisfaction of this property means the state of the system can be driven anywhere in a finite time. Since the controllability matrix is a 4x4 matrix (determined in section 3.2) the rank of the matrix must be 4 meaning that the system is controllable and the LQR method may be used to stabilize the BOW plant. The Matlab command 'ctrb' is used on the system model of the BOW plant to generate the controllability matrix while the command 'rank' is used to test the rank of the system. The resulting controllability matrix is given as:

$$C = \begin{bmatrix} 0 & 25.5744 & 0 & 819.1687 \\ 5.1679 & 0 & 815.4935 & 0 \\ 0 & 20.4066 & 0 & 3.6752 \\ 20.4066 & 0 & 3.6752 & 0 \end{bmatrix} \qquad (3.53)$$

The Matlab function 'lqr' allows the parameters, R and Q to be chosen which will balance the relative importance of the control effort (u) and deviation from 0 (error) respectively in the cost function. The simplest case is to assume that R = 1 and Q = C'C.

The response after the LQR controller is implemented on the BOW system is depicted in Figure 3.21. By altering variables in the Q matrix, the response can be improved even further. The resulting system transfer function is given as:

$$LQR_{TF} = \frac{25.57s^2 - 1.819e^{-13} + 3.609e^{-14}}{s^4 + 11.32s^3 + 32.03s^2 + 1.396e^{-5} + 2.501e^{-12}} \qquad (3.55)$$

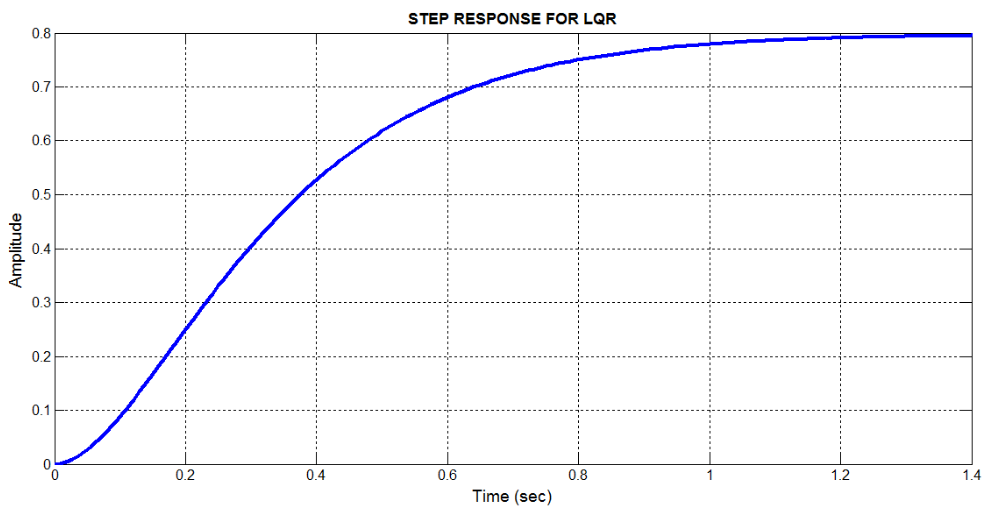


Figure 3.21: Step response of LQR controller

The complete Matlab code for the above LQR computations can be found in Appendix B. Because of hardware and software limitations, the state feedback LQR controller could not be implemented in real-time as initially hoped.

3.4.6 NN-Predictive controller design

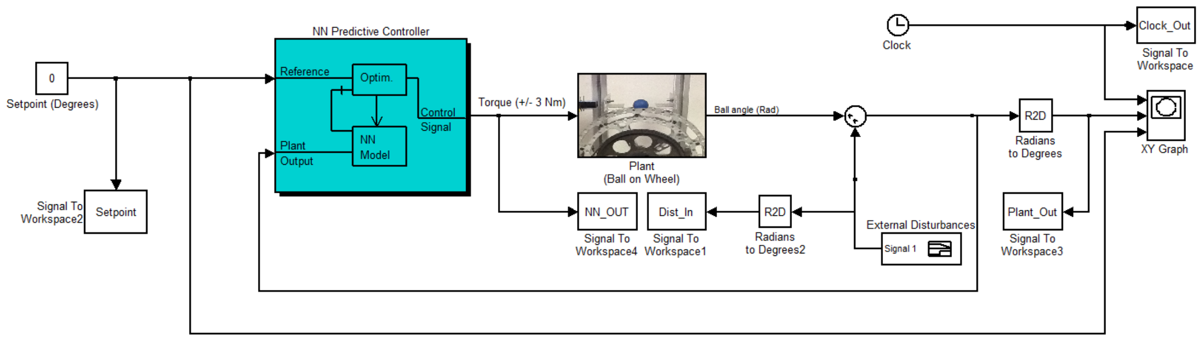


Figure 3.22: NN-Predictive controller applied to BOW System in Simulink

Figure 3.22 shows the Simulink model incorporating a NN-predictive controller to stabilize the BOW plant. The predictive controller could not be implemented in real-time due to software and hardware constraints. The controller, when executed in real-time was not able to alter the drive torque within an acceptable time period in order to control the BOW plant over an OPC network connection. However, simulation results were obtained for the predictive controller when executed on the model of the BOW system. Detailed simulation results are given in the results section.

3.4.7 NN-PID controller design

An NN-PID controller was designed and simulated in Matlab before actual implementation onto the PLC in SCL (see Appendix B for Matlab code and Appendix C for actual implemented code). The structure of the implemented NN-PID controller with all its connective weights is shown in Figure 3.23.

The controller is an 'online' controller, meaning that it does not require historical data for purposes of training.

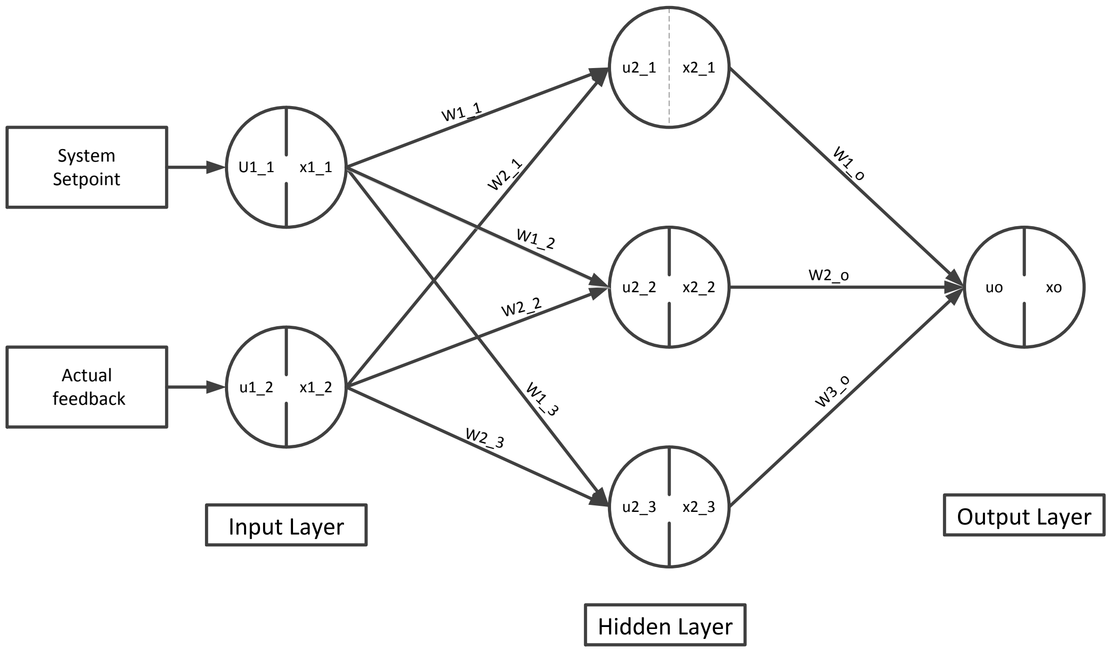


Figure 3.23: NN-PID structure

The hidden to output layer weights w_{1_o}, w_{2_o} and w_{3_o} essentially form the PID controller gains K_p, K_d and K_i respectively. The network uses Back Propagation (BP) and utilizes the gradient descent learning algorithm to update its weights and thus minimize the system error. The sigmoid activation function is utilized in the BP computations. The reader is referred to [59] for a detailed derivation and layout of the BP algorithm.

For simplicity only the input to output properties of each neuron of the network shown in Figure 3.23 are given below (Eq.3.56 to Eq. 3.65):

Input Layer neurons:

$$u1_1 = system\ setpoint = x1_1 \tag{3.56}$$

$$u1_2 = actual\ feedback = x1_2 \tag{3.57}$$

P-action neuron transfer function:

$$u2_1 = x1_1 w1_3 + x1_2 w2_2 \tag{3.58}$$

$$x2_1(t) = \begin{cases} -1 & u2_1(t) < -1 \\ u2_1(t) & -1 \le u2_1(t) \le 1 \\ 1 & u2_{1(t)} > -1 \end{cases} \tag{3.59}$$

I-action neuron transfer function:

$$u2_1 = x1_1 w1_1 + x1_2 w2_1 \tag{3.60}$$

$$x2_2(t) = \begin{cases} -1 & u2_2(t) < -1 \\ u2_2(t-1) + u2_2(t) & -1 \le u2_2(t) \le 1 \\ 1 & u2_2(t)\ > -1 \end{cases} \tag{3.61}$$

D-action neuron transfer function:

$$u2_1 = x1_1 w1_1 + x1_2 w2_3 \tag{3.62}$$

$$x2_3(t) = \begin{cases} -1 & u2_3(t) < -1 \\ u2_3(t) - u2_3(k-1) & -1 \le u2_3(t) \le 1 \\ 1 & u2_3(t) > -1 \end{cases} \tag{3.63}$$

Output layer neuron:

$$uo = x2_1 w1_o + x2_2 w2_o + x2_3 w3_o \tag{3.64}$$

$$xo = uo \tag{3.65}$$

The designed NN-PID controller is executed on the model of the BOW system in simulation in order to control it. The simulation results, when the system is subjected to a step input are given in Figure 3.24, Figure 3.25, Figure 3.26 and Figure 3.27 respectively.

In comparison to the traditional controllers examined above, the NN-PID controller is significantly slower in its response especially while the network is undergoing training. It is observed that training occurs relatively quickly (within 1.5 - 2 s) and is able to produce an accurate result with an error of less than 0.01. However, this remains largely dependent on the selected learning rate and initial weights. Similar results can be expected in an actual implementation with the exception of a longer training period owing to system uncertainties and delays.

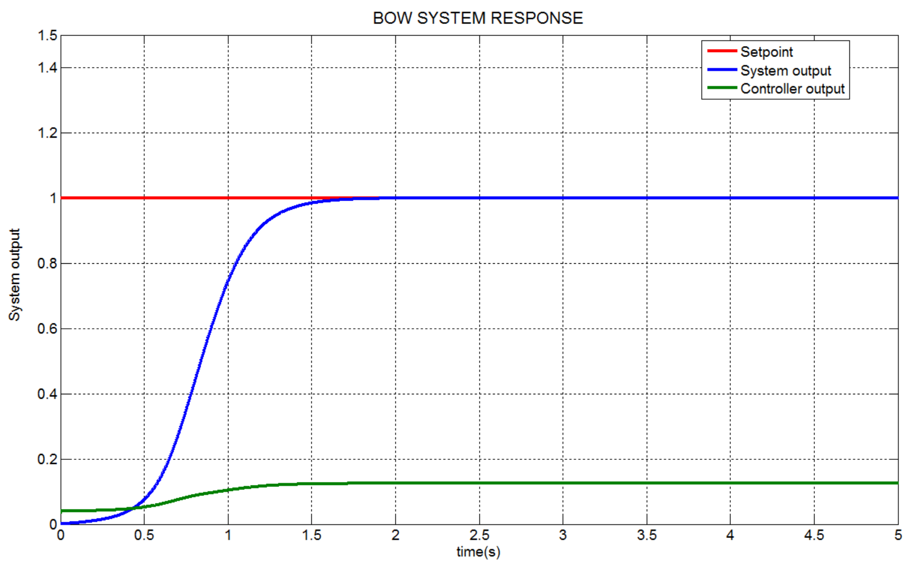


Figure 3.24: BOW response & controller action in response to step input

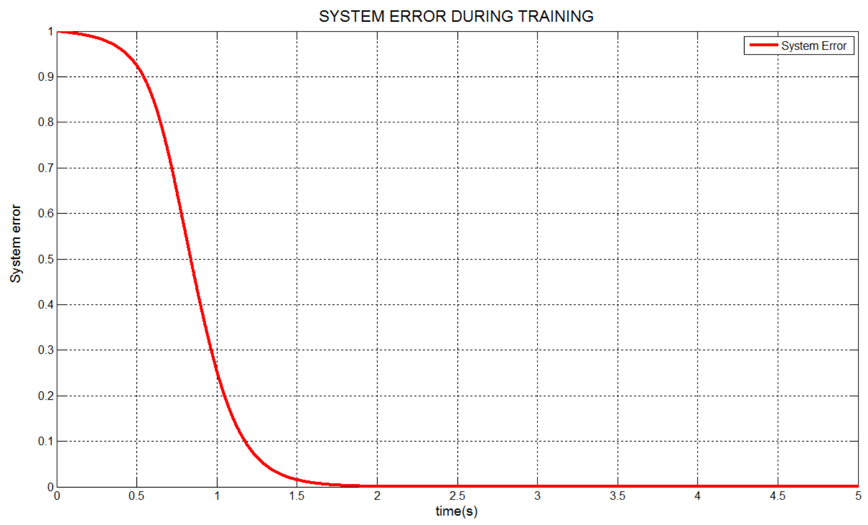


Figure 3.25: Simulation of system error during training

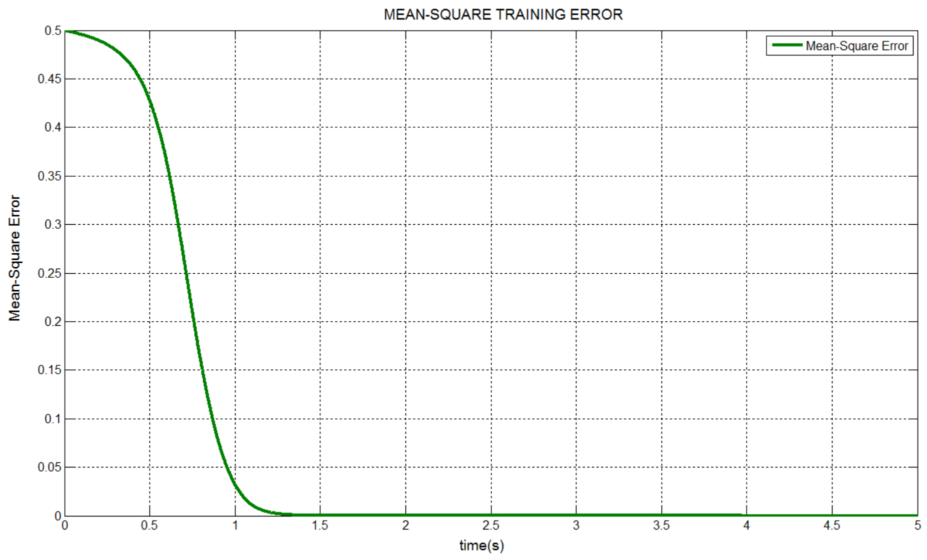


Figure 3.26: Simulation of root-mean-square training error

The PID gains are automatically tuned during the training period and stabilize when the system error is sufficiently small (see Figure 3.27). The controller is thus able to continually adjust itself in order to maintain the system set-point even when an external disturbance is introduced or when system parameters are altered. If the stability of the BOW plant is lost perhaps due to the NN over training itself or failing to locate the point of minimum error, the training need only be re-initialized.

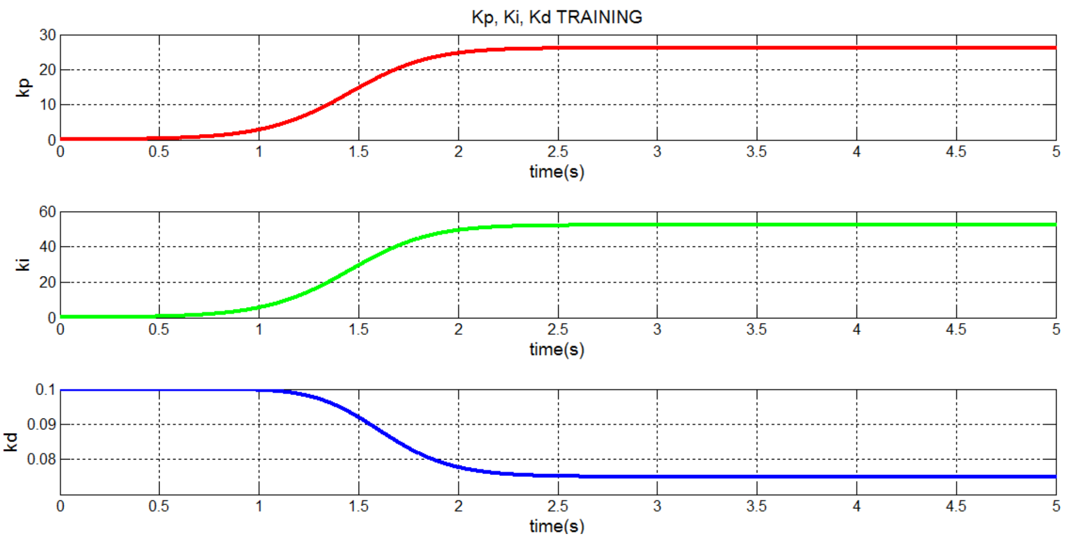


Figure 3.27: Training of PID gains in simulation

3.5 Summary

In this chapter, a mathematical model of the BOW platform is proposed using the Lagrangian energies approach. The non-linear system is then linearized about the system's zero degree equilibrium point. Various traditional control strategies including PD, PID, Lead, Lead-Lag LQR controllers are then designed and simulated based on the determined model. Standard design procedures are used including the root-locus, frequency domain and time domain methods as outlined in the literature review section. The designed controllers are then digitized using Tustin's bilinear transformation. Thereafter, the digitized controllers are implemented on the PLC in Structured Code Language (SCL) and called within in a cyclic interrupt set to a period of 1 ms. The process of implementation is detailed further in the following section. Two NN control strategies are also designed and simulated including a NN-Predictive controller and a NN-PID controller.

Because of hardware limitations, in particular a very poor update rate between Matlab and the PLC over an OPC connection, the NN-Predictive controller and the LQR could not be implemented in real-time as originally intended.

Chapter 4.
EXPERIMENTAL SETUP OF THE BALL-ON-WHEEL (BOW) SYSTEM

4.1 Introduction to the BOW System

The BOW system is a mechatronic-type control system that was selected for its strong non-linearity (shown in the system equations in section 3.1) and inherent instability [56]. It may therefore represent any existing non-linear and unstable SISO system found in industry. The aim of the control system is to balance various balls of different size, weight and surface texture on the top-centre of the wheel by controlling the torque applied to the wheel. The apparatus consists of two aluminium wheel coupled to a servo motor via a tooth belt (see Figure 4.1 and Figure 4.2). The servo motor is controlled by a Siemens servo drive which acts as a Profibus slave to an S7-300 PLC. A laser distance sensor is used for actual feedback of the ball position. The wheel angle and applied torque are calculated by the drive and retrieved cyclically by the PLC over the existing network.

4.2 Mechanical design

Mechanical drawings of the BOW system are presented in Appendix A. The frame of the system was constructed using locally available aluminium extrusion. The aluminium wheel was laser cut to specification. Excess material was removed from the wheel, without affecting its structural integrity, to reduce its overall weight and hence also reduce the magnitude of the torque required by the motor to change the direction of rotation or the speed. A CAD assembly drawing as well as an image of the actual construction of the BOW system can be seen in Figure 4.1 and Figure 4.2 respectively.

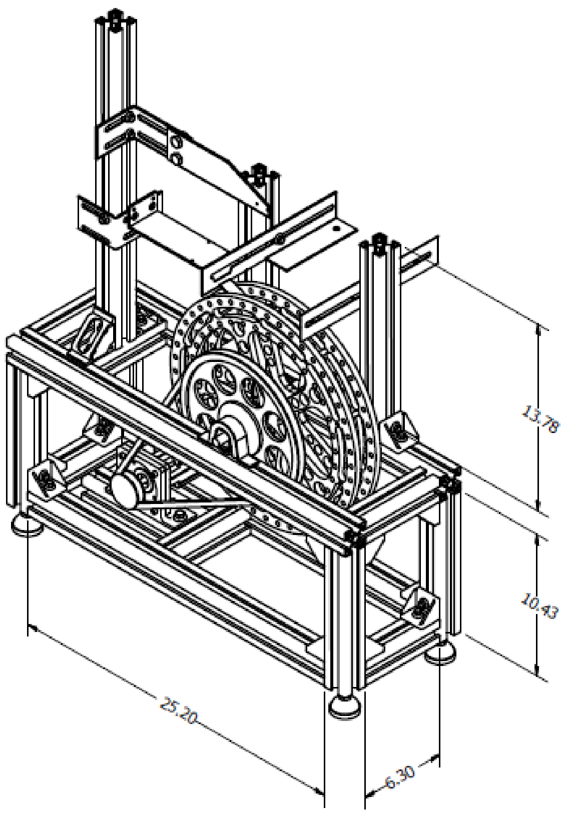


Figure 4.1: CAD drawing of BOW System

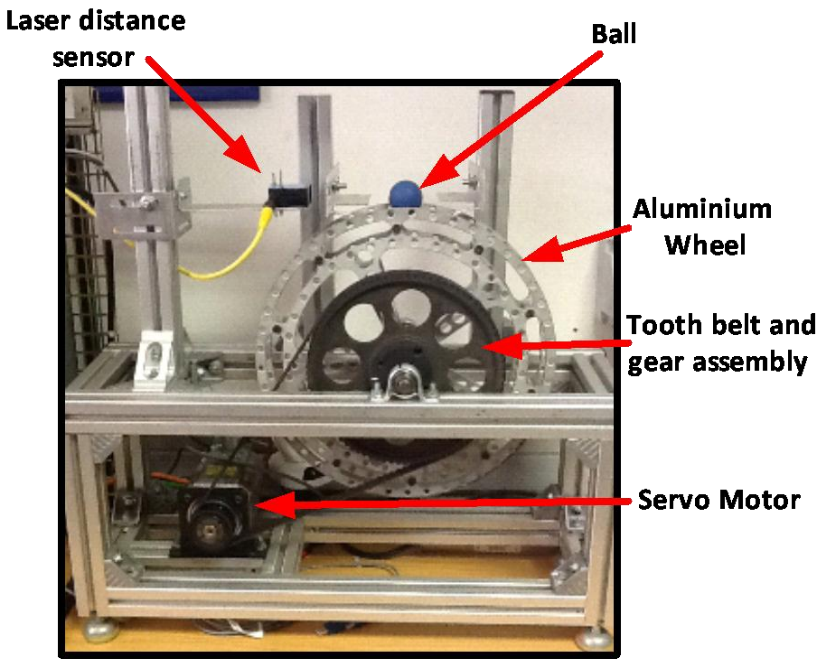


Figure 4.2: BOW System

Referring to the coupling between the servo motor and the wheel (Figure 4.2), the system is geared in the ratio 1:4.42 via a tooth belt in order to improve the general response of the wheel as the ball rolls along its periphery. The gearing also increases the torque produced by the motor by a factor of 4.42. This essentially serves to reduce the load on the motor especially when changing rotational direction at high speed.

The groove in which the ball rests is adjustable and may be set to 3 defined distances using specially designed spacers (30mm, 35mm and 40mm) to accommodate balls of different sizes. The wheel periphery is also lined with rubber to improve surface contact and thus improve the system response. More images of the completed system can be found in Appendix D.

4.3 System overview

The block diagram shown in Figure 4.3 depicts the BOW system with all its components and their respective interconnections. A more detailed overview can be seen in Figure 4.20 (section 4.3.5).

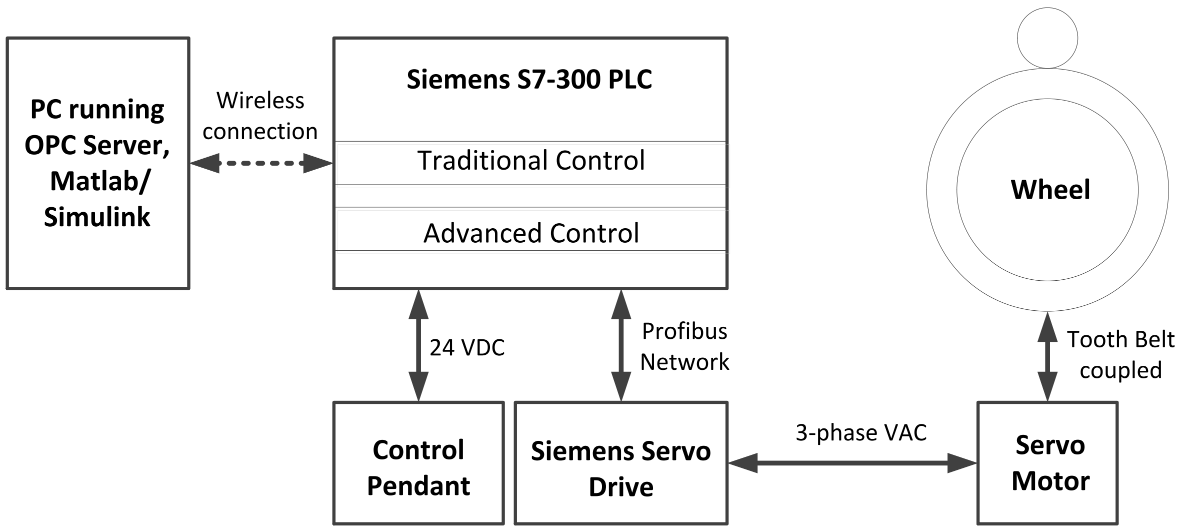


Figure 4.3: BOW System overview

The sections that follow discuss the role of each part of the BOW system and the reasons behind their selection.

4.3.1 Drive and motor

Before the actual drive and motor selection process begins, a consideration of the drive system in its entirety is needed. The following factors were considered during the selection process of the servo motor and drive system used in the BOW system [60, 61]:

1) Is the motion linear or rotary? What drive system is to be used (spindle, toothed belt, etc.)?
2) What variables are to be controlled (current, speed, position)? With what accuracy is control required? Is an open-loop control system sufficient? How is the controlled variable measured? Where do commands come from?
3) Is there sufficient power available to drive the load under all operating conditions and to compensate for the expected losses in the drive train?
4) What are the maximum voltage and current available?
5) What is the cost involved?

4.3.1.1 Siemens servo drive and servo motor

The Masterdrive Motion Control (MC) from Siemens was chosen as the main motor control unit for the BOW system. It is an AC-AC frequency converter designed for industrial servo drive applications. It is ideally used in applications where [60, 61]:

- a very high level-dynamic response is required
- angular synchronism between drives is necessary

It thus satisfies the most stringent demands paced on servo technology. The drive is also Profibus enabled for cyclical user data exchange via a communication board. It can therefore be connected into a Profibus network as a slave to some master system such as a PLC. The drive also offers a flexible (freely assignable) configuration of cyclic messages of up to 16 words [60].

The drive is powered on a 3 phase power supply (400-480 VAC) and outputs a controlled voltage in the same range. It also comes with the following features:

- An R232 port for communication with a PC (for parameter changes and data capture)

- An Encoder port
- PROFIBUS RS485 port for communication with a master system such as a PLC
- Analogue I/O (X101)
- Emergency input

Figure 4.4 gives a description of the ports and terminals included on the drive:

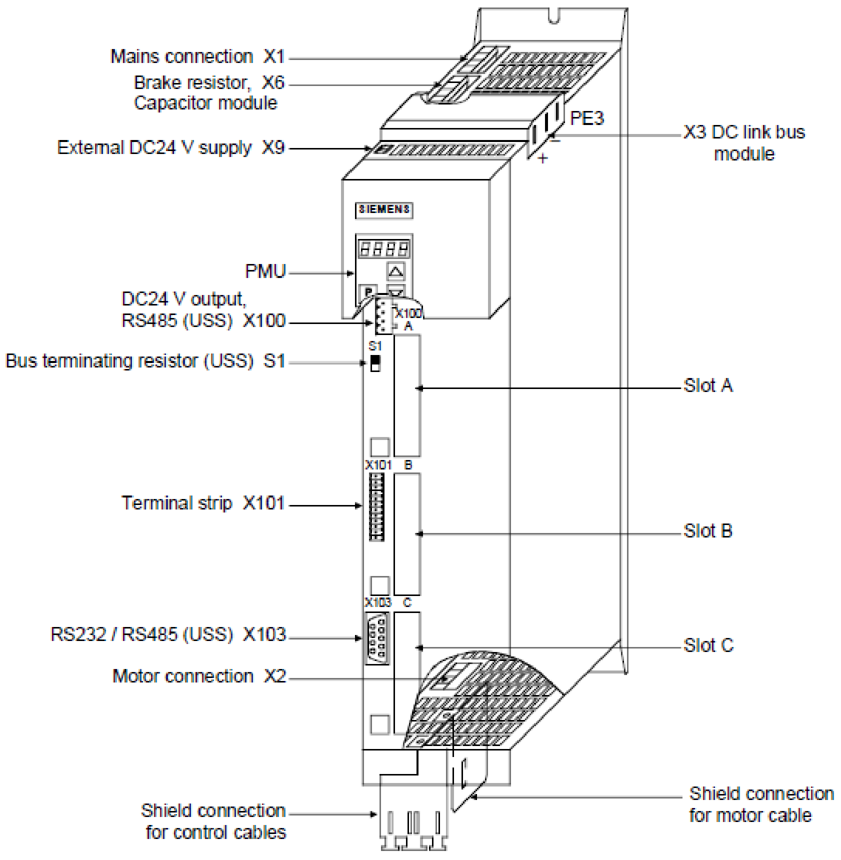


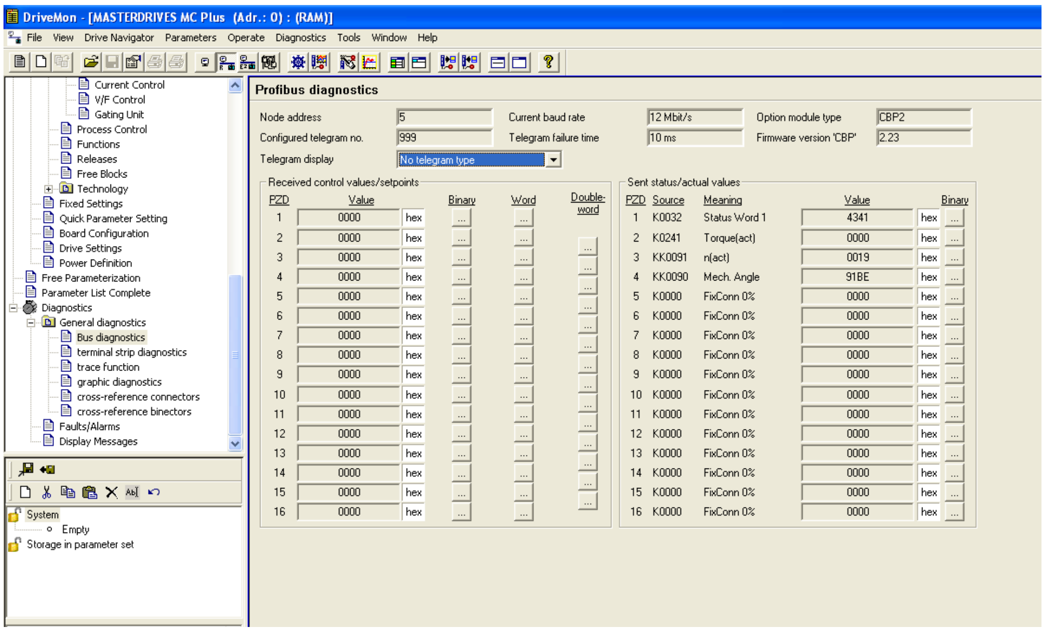
Figure 4.4: Master Drive MC unit [60, 61]

In the BOW system, the drive is controlled by the Siemens S7-300 PLC via the Profibus communication board (CB) that is standard with the drive (x103). A data telegram that contains a control word, a set-point channel, a status word and a feedback channel is exchanged cyclically between the PLC and drive. The data is divided into two areas:

- The process data area (PZD) which contains the control words and set-points or status information and actual values.

- The parameter area (PKW) for reading/ writing of parameters, e.g. reading out of faults, min max limits etc.

Each bit in the control word has a specific function, e.g. start, run, stop etc. The reader may refer to [61] for a complete breakdown of the control word. The drive speed, torque or position are controlled via the set-point channel depending on which control action is selected [61]. Figure 4.5 below shows how the data telegram is selected in the drive software tool (Drivemonitor). Using freely assignable function blocks in the drive, it is possible to link specific functions to the process data area or parameter channel.



Profibus diagnostics

Node address	5	Current baud rate	12 Mbit/s	Option module type	CBP2
Configured telegram no.	999	Telegram failure time	10 ms	Firmware version 'CBP'	2.23
Telegram display	No telegram type				

Received control values/setpoints

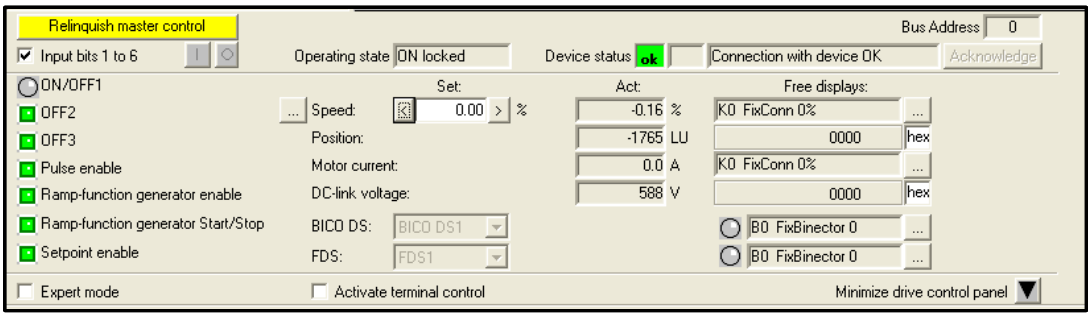
PZD	Value		Binary	Word	Double-word
1	0000	hex	...	...	
2	0000	hex	...	...	...
3	0000	hex	...	...	...
4	0000	hex	...	...	...
5	0000	hex	...	...	...
6	0000	hex	...	...	...
7	0000	hex	...	...	...
8	0000	hex	...	...	...
9	0000	hex	...	...	...
10	0000	hex	...	...	...
11	0000	hex	...	...	...
12	0000	hex	...	...	...
13	0000	hex	...	...	...
14	0000	hex	...	...	...
15	0000	hex	...	...	...
16	0000	hex	...	...	

Sent status/actual values

PZD	Source	Meaning	Value		Binary
1	K0032	Status Word 1	4341	hex	...
2	K0241	Torque(act)	0000	hex	...
3	KK0091	n(act)	0019	hex	...
4	KK0090	Mech. Angle	91BE	hex	...
5	K0000	FixConn 0%	0000	hex	...
6	K0000	FixConn 0%	0000	hex	...
7	K0000	FixConn 0%	0000	hex	...
8	K0000	FixConn 0%	0000	hex	...
9	K0000	FixConn 0%	0000	hex	...
10	K0000	FixConn 0%	0000	hex	...
11	K0000	FixConn 0%	0000	hex	...
12	K0000	FixConn 0%	0000	hex	...
13	K0000	FixConn 0%	0000	hex	...
14	K0000	FixConn 0%	0000	hex	...
15	K0000	FixConn 0%	0000	hex	...
16	K0000	FixConn 0%	0000	hex	...

Figure 4.5: Selecting data telegram in drive software tool

Basic test functionality is possible directly from the PC using Drivemonitor (PC drive software). Data exchange in this case is via RS232. PLC control is disabled when PC control is active and vice versa. Figure 4.6 shows the basic drive control panel in Drivemonitor.



Relinquish master control | Bus Address 0

☑ Input bits 1 to 6 | Operating state: ON locked | Device status: ok | Connection with device OK | Acknowledge

	Set:	Act:	Free displays:
ON/OFF1			
🟩 OFF2	Speed: 0.00 %	-0.16 %	K0 FixConn 0%
🟩 OFF3	Position:	-1765 LU	0000 hex
🟩 Pulse enable	Motor current:	0.0 A	K0 FixConn 0%
🟩 Ramp-function generator enable	DC-link voltage:	588 V	0000 hex
🟩 Ramp-function generator Start/Stop	BICO DS: BICO DS1		B0 FixBinector 0
🟩 Setpoint enable	FDS: FDS1		B0 FixBinector 0

☐ Expert mode | ☐ Activate terminal control | Minimize drive control panel

Figure 4.6: Drive test panel in Drive Monitor

Various controls and/ or set-points may be changed using the above panel, however functionality is limited. In the BOW system, the drive is connected as a Profibus slave and receives control commands and set-points directly from the PLC which acts as the master system in the configuration. The drive may be configured and operated in any one of the following control modes:

- Speed control
- Torque or current control
- Position control mode

Only one mode may be active at a time. For ideal operation, the BOW system is operated in Torque control mode. This satisfies the requirements of the system model as discussed in the previous section, where the designed controller must manipulate the torque being fed to the wheel.

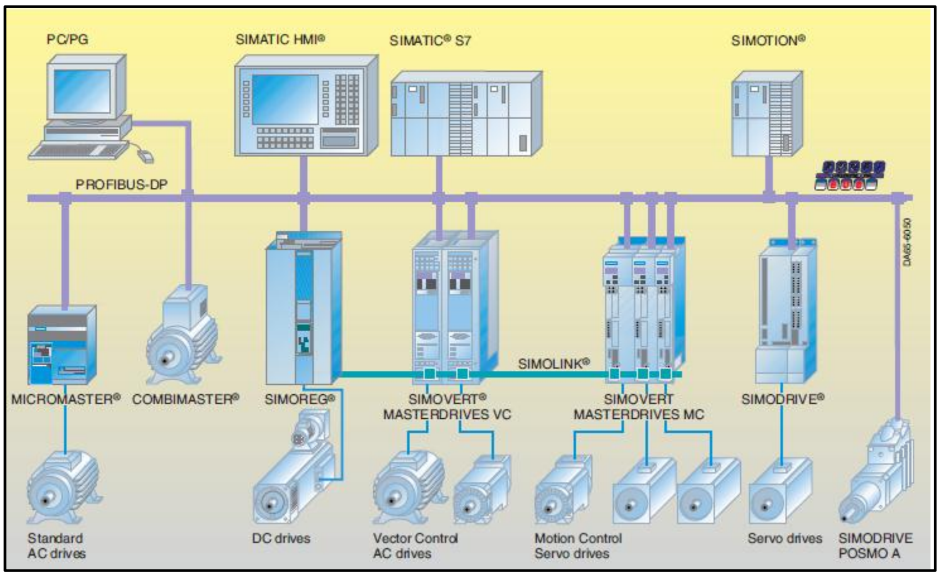


Figure 4.7: Masterdrive on Profibus network

Figure 4.7 depicts a typical industrial Profibus network architecture including multiple drives for multiple axis control and servo synchronization. Similar setups can be found in in CNC milling applications.

Servo motors allow for precise control of angular position. They are used in many applications where high speed as well as precision is a requirement. One such area of application is industrial robotics. To achieve this sort of precision, the motor is usually coupled with an encoder to provide position feedback in a closed-loop system. There is huge variety of servo motors available on the market.

The Siemens 1FK7042-5A71-1TGO servo motor was selected and used in the BOW system. It is a synchronous AC motor. Table 4-1 shows the key motor specifications.

Table 4-1: Table of motor specifications

Maximum Rated	M_o=3 Nm	I_o=2.2 A	$\mathrm{n_{max}} = 9000$ rpm
Nominal	M_N=2.6 Nm	I_N=1.95 A	n_N =3000 rpm

As previously mentioned, a tooth belt drive-train is used as the means of coupling between the wheel and servo motor. This type of drive-train was chosen because it works well to prevent undesirable slippage. This is a common problem with most other types of drive-train. The tooth belt also allows a certain degree of flexibility in the system, effectively reducing the system’s rigidity. This is ideal for the BOW system because of the high torques involved especially when the direction of rotation is rapidly changed. The motor and drive can be seen in Figure 4.8.

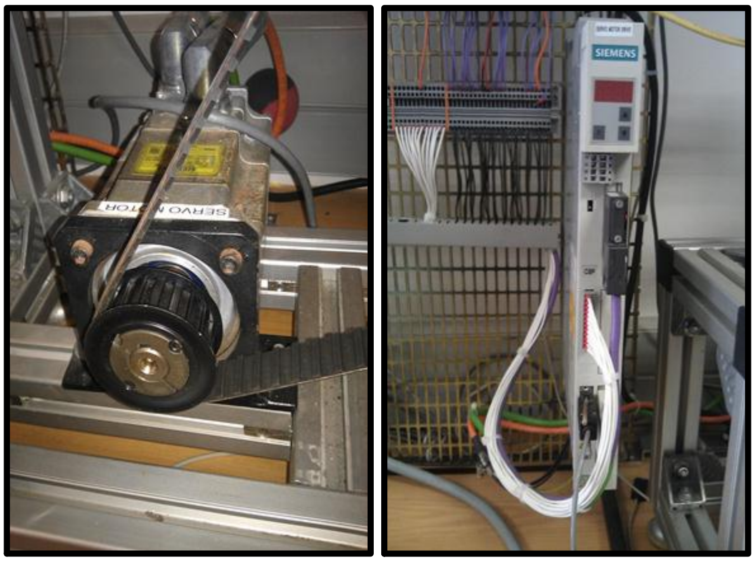
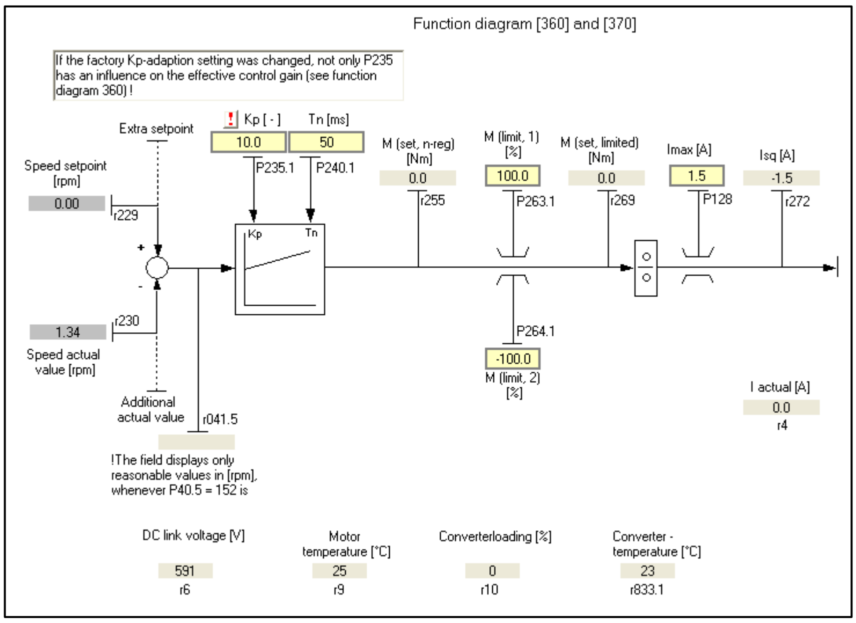


Figure 4.8: Drive (Masterdrive MC) and motor

The servo motor, in speed control mode, is controlled via a speed controller algorithm within the drive. In most situations, the controller gains are tuned automatically, however, Figure 4.9 shows how the controller gains can be fine-tuned, in the drive software by the user, to obtain the best performance.



Function diagram [360] and [370]

If the factory Kp-adaption setting was changed, not only P235 has an influence on the effective control gain (see function diagram 360) !

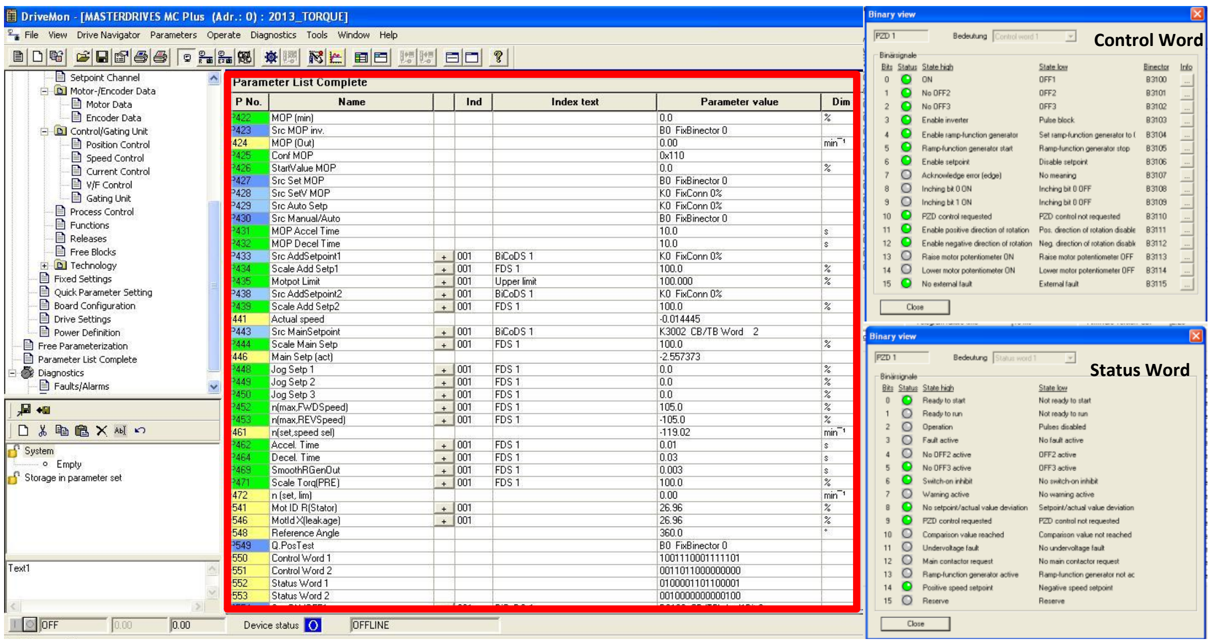
!The field displays only reasonable values in [rpm], whenever P40.5 = 152 is

Figure 4.9: Speed controller in servo drive

In torque control mode, the drive is able to measure (as a means of feedback) the current being drawn by the motor.

The drive is thus able to directly control the torque by controlling the current being supplied to the motor.

In position control mode, the drive measures the angular position of the shaft via an inbuilt shaft encoder on the rear of the motor. This feedback value is then used in the position control loop to move the motor to the desired set-point.



Parameter List Complete

P No.	Name		Ind	Index text	Parameter value	Dim
P422	MOP (min)				0.0	%
P423	Src MOP inv.				B0 FixBinector 0	
424	MOP (Out)				0.00	min⁻¹
P425	Conf MOP				0x110	
P426	StartValue MOP				0.0	%
P427	Src Set MOP				B0 FixBinector 0	
P428	Src SetV MOP				K0 FixConn 0%	
P429	Src Auto Setp				K0 FixConn 0%	
P430	Src Manual/Auto				B0 FixBinector 0	
P431	MOP Accel Time				10.0	s
P432	MOP Decel Time				10.0	s
P433	Src AddSetpoint1	+	001	BiCoDS 1	K0 FixConn 0%	
P434	Scale Add Setp1	+	001	FDS 1	100.0	%
P435	Motpot Limit	+	001	Upper limit	100.000	%
P438	Src AddSetpoint2	+	001	BiCoDS 1	K0 FixConn 0%	
P439	Scale Add Setp2	+	001	FDS 1	100.0	%
441	Actual speed				-0.014445	
P443	Src MainSetpoint	+	001	BiCoDS 1	K3002 CB/TB Word 2	
P444	Scale Main Setp	+	001	FDS 1	100.0	%
446	Main Setp (act)				-2.557373	
P448	Jog Setp 1	+	001	FDS 1	0.0	%
P449	Jog Setp 2	+	001	FDS 1	0.0	%
P450	Jog Setp 3	+	001	FDS 1	0.0	%
P452	n(max,FWDSpeed)	+	001	FDS 1	105.0	%
P453	n(max,REVSpeed)	+	001	FDS 1	-105.0	%
461	n(set,speed sel)				-119.02	min⁻¹
P462	Accel. Time	+	001	FDS 1	0.01	s
P464	Decel. Time	+	001	FDS 1	0.03	s
P469	SmoothRGenOut	+	001	FDS 1	0.003	s
P471	Scale Torq(PRE)	+	001	FDS 1	100.0	%
472	n (set, lim)				0.00	min⁻¹
541	Mot ID R(Stator)	+	001		26.96	%
546	MotId X(leakage)	+	001		26.96	%
548	Reference Angle				360.0	°
P549	Q.PosTest				B0 FixBinector 0	
550	Control Word 1				1001110001111101	
551	Control Word 2				0011011000000000	
552	Status Word 1				0100001101100001	
553	Status Word 2				0010000000000100	

Binary view

PZD 1 Bedeutung Control word 1

Control Word

Binärsignale

Bits	Status	State high	State low	Binector	Info
0	●	ON	OFF1	B3100	...
1	●	No OFF2	OFF2	B3101	...
2	●	No OFF3	OFF3	B3102	...
3	●	Enable inverter	Pulse block	B3103	...
4	●	Enable ramp-function generator	Set ramp-function generator to (	B3104	...
5	●	Ramp-function generator start	Ramp-function generator stop	B3105	...
6	●	Enable setpoint	Disable setpoint	B3106	...
7	○	Acknowledge error (edge)	No meaning	B3107	...
8	○	Inching bit 0 ON	Inching bit 0 OFF	B3108	...
9	○	Inching bit 1 ON	Inching bit 0 OFF	B3109	...
10	●	PZD control requested	PZD control not requested	B3110	...
11	●	Enable positive direction of rotation	Pos. direction of rotation disable	B3111	...
12	●	Enable negative direction of rotation	Neg. direction of rotation disabl	B3112	...
13	○	Raise motor potentiometer ON	Raise motor potentiometer OFF	B3113	...
14	○	Lower motor potentiometer ON	Lower motor potentiometer OFF	B3114	...
15	●	No external fault	External fault	B3115	...

Close

Binary view

PZD 1 Bedeutung Status word 1

Status Word

Binärsignale

Bits	Status	State high	State low
0	●	Ready to start	Not ready to start
1	○	Ready to run	Not ready to run
2	○	Operation	Pulses disabled
3	○	Fault active	No fault active
4	○	No OFF2 active	OFF2 active
5	●	No OFF3 active	OFF3 active
6	●	Switch-on inhibit	No switch-on inhibit
7	○	Warning active	No warning active
8	●	No setpoint/actual value deviation	Setpoint/actual value deviation
9	●	PZD control requested	PZD control not requested
10	○	Comparison value reached	Comparison value not reached
11	○	Undervoltage fault	No undervoltage fault
12	○	Main contactor request	No main contactor request
13	○	Ramp-function generator active	Ramp-function generator not ac
14	●	Positive speed setpoint	Negative speed setpoint
15	○	Reserve	Reserve

Close

Figure 4.10: Parameter selection in Drivemonitor

Drive parameters may be monitored or altered directly from Drivemonitor as shown surrounded by the red rectangle in Figure 4.10. For maximum flexibility and control, specific drive parameters may be linked to the control words (see upper right side of Figure 4.10), status words (see lower right side of Figure 4.10), or set-point channels for direct manipulation in the PLC.

4.3.2 OPC Server and client

In order to capture experimental data for analysis and comparison and also execute controllers directly from Matlab/ Simulink the S7-300 PLC is linked wirelessly using Siemens wireless technology to a laptop running Matlab/ Simulink (see the architecture described in Figure 4.3 above). OPC Server software running on this computer then give Matlab/ Simulink (i.e. the client) exclusive access to all PLC I/ O's and memory areas within a 1 – 1000 ms time frame depending on the amount of data acquisition and processing required. This feature provides the flexibility needed to design, implement and execute complex controllers directly from the Matlab/ Simulink environment. Thoroughly tested algorithms may also later be generated and implemented directly on to the PLC for real time execution. The major benefit of running the controllers directly from the Matlab/ Simulink environment is that relatively complex operations in the Laplace (s) domain can be performed with ease and system data can easily be captured and stored for further analysis [50].

Simatic Net OPC Server software from Siemens is installed and used on a PC. The server is set up in the hardware configuration as shown in Figure 4.11 below.

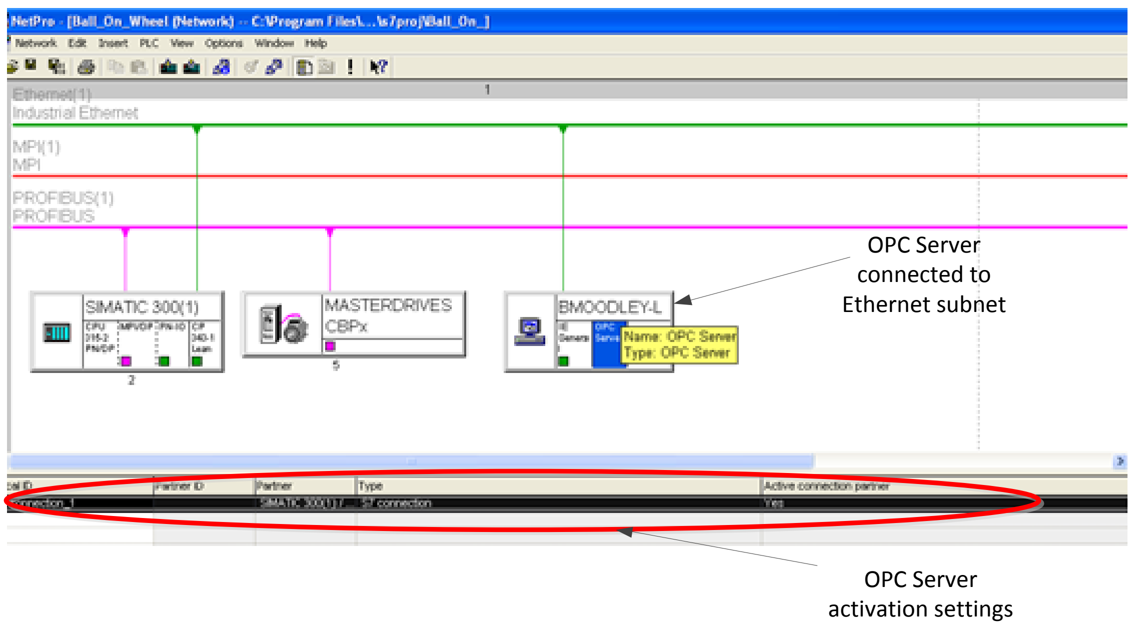
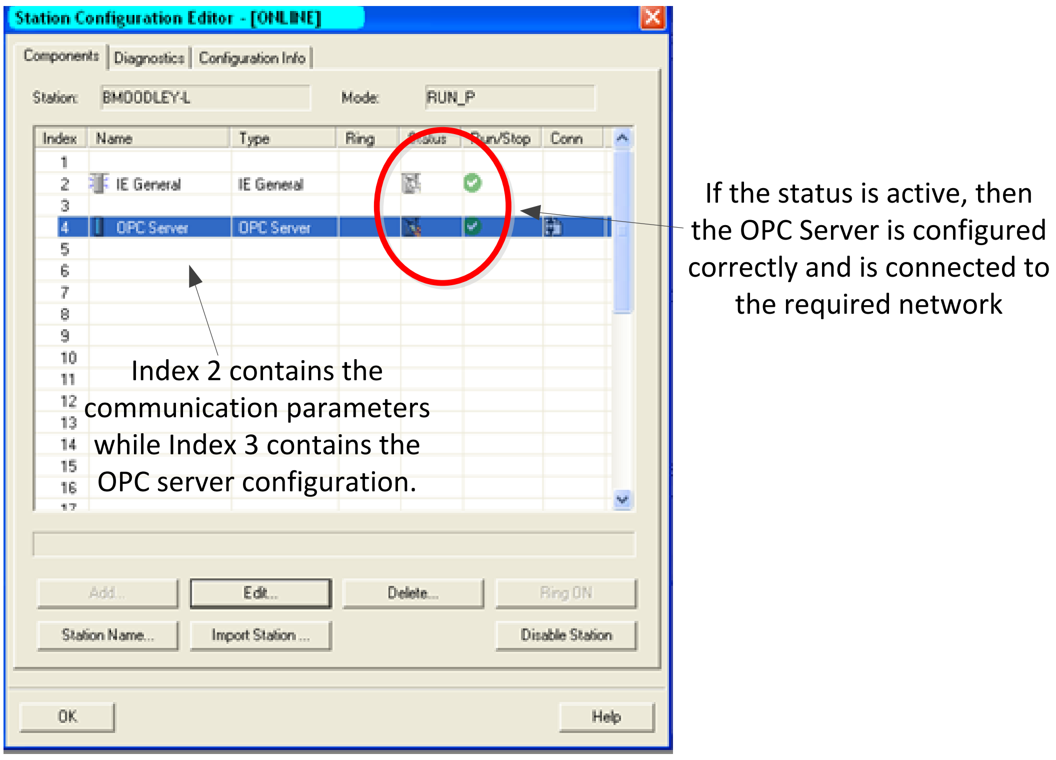

NetPro - [Ball_On_Wheel (Network) -- C:\Program Files\...\s7proj\Ball_On_]
Network Edit Insert PLC View Options Window Help
Ethernet(1)
Industrial Ethernet
MPI(1)
MPI
PROFIBUS(1)
PROFIBUS
SIMATIC 300(1)
MASTERDRIVES CBPx
BMOODLEY-L
Name: OPC Server
Type: OPC Server
OPC Server connected to Ethernet subnet
Partner ID
Partner
Type
Active connection partner
Connection_1
SIMATIC 300(1)
S7 connection
Yes
OPC Server activation settings

Figure 4.11: OPC Server setup in Step 7 software

Thereafter the server is activated using another tool called the station configuration editor shown in Figure 4.12.


Station Configuration Editor - [ONLINE]

Components | Diagnostics | Configuration Info

Station: BMODDLEY-L Mode: RUN_P

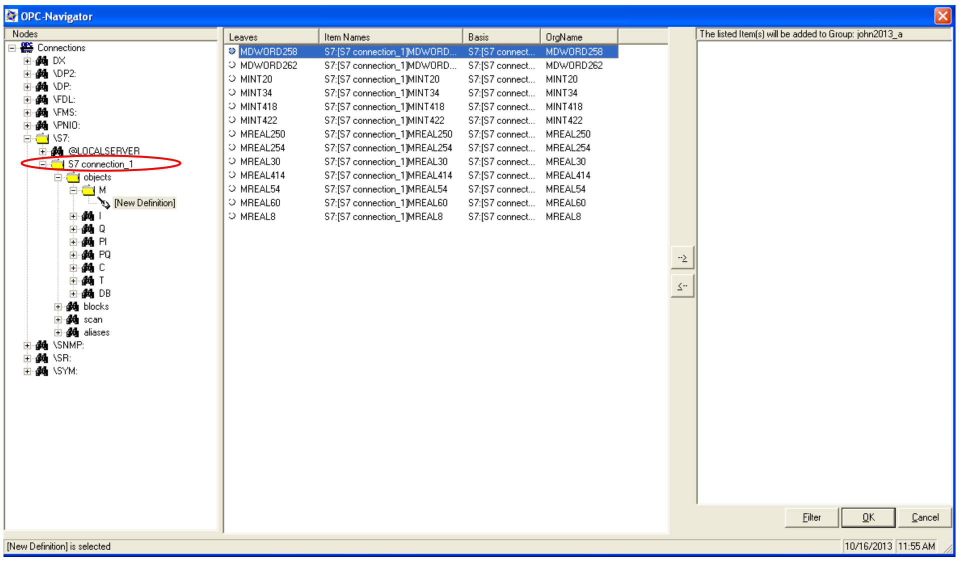
Index	Name	Type	Ring	Status	Run/Stop	Conn
1						
2	IE General	IE General			✓	
3						
4	OPC Server	OPC Server			✓	
5						
6						
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						

If the status is active, then the OPC Server is configured correctly and is connected to the required network

Index 2 contains the communication parameters while Index 3 contains the OPC server configuration.

Figure 4.12: OPC Server settings

Using another software tool called OPC Scout, the active server is selected and all memory areas and I/O areas of the PLC become available over the configured connection, i.e. S7 Connection as shown in the Figure 4.13 below.


OPC-Navigator

Nodes
- Connections
  - DX
  - \DP2:
  - \DP:
  - \FDL:
  - \FMS:
  - \PNIO:
  - \S7:
    - @LOCALSERVER
    - S7 connection_1
      - objects
        - M
          - [New Definition]
        - I
        - Q
        - PI
        - PQ
        - C
        - T
        - DB
      - blocks
      - scan
      - aliases
  - \SNMP:
  - \SR:
  - \SYM:

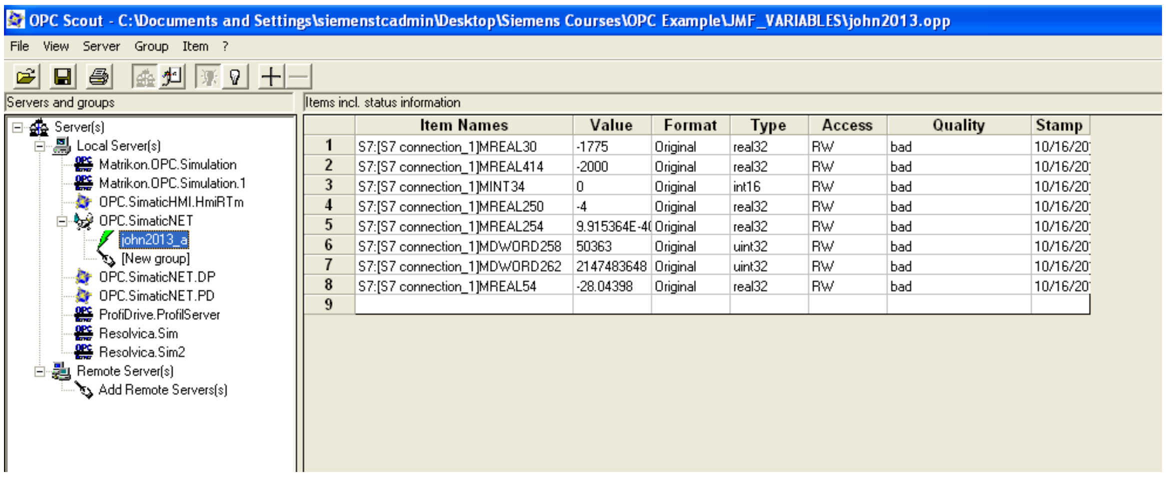
Leaves	Item Names	Basis	OrgName
MDWORD258	S7:[S7 connection_1]MDWORD...	S7:[S7 connect...	MDWORD258
MDWORD262	S7:[S7 connection_1]MDWORD...	S7:[S7 connect...	MDWORD262
MINT20	S7:[S7 connection_1]MINT20	S7:[S7 connect...	MINT20
MINT34	S7:[S7 connection_1]MINT34	S7:[S7 connect...	MINT34
MINT418	S7:[S7 connection_1]MINT418	S7:[S7 connect...	MINT418
MINT422	S7:[S7 connection_1]MINT422	S7:[S7 connect...	MINT422
MREAL250	S7:[S7 connection_1]MREAL250	S7:[S7 connect...	MREAL250
MREAL254	S7:[S7 connection_1]MREAL254	S7:[S7 connect...	MREAL254
MREAL30	S7:[S7 connection_1]MREAL30	S7:[S7 connect...	MREAL30
MREAL414	S7:[S7 connection_1]MREAL414	S7:[S7 connect...	MREAL414
MREAL54	S7:[S7 connection_1]MREAL54	S7:[S7 connect...	MREAL54
MREAL60	S7:[S7 connection_1]MREAL60	S7:[S7 connect...	MREAL60
MREAL8	S7:[S7 connection_1]MREAL8	S7:[S7 connect...	MREAL8

The listed Item(s) will be added to Group: john2013_a

[New Definition] is selected 10/16/2013 11:55 AM

Figure 4.13: OPC Scout variable selection

Many different variable groups may be created using OPC scout. Selected variables are added to the created group by highlighting them and shifting them into the group area. Once the variables have been moved into the created group, they may be monitored or updated in real time. Any OPC client (including Matlab and Simulink) can now access these variables. The refresh rate of OPC Server depends on the specifications of the system that is being accessed. Since, in this case, it is an S7-300 PLC, the minimum possible sampling rate is 1 ms.


OPC Scout - C:\Documents and Settings\siemenstcadmin\Desktop\Siemens Courses\OPC Example\JMF_VARIABLES\john2013.opp

Servers and groups

- Server(s)
  - Local Server(s)
    - Matrikon.OPC.Simulation
    - Matrikon.OPC.Simulation.1
    - OPC.SimaticHMI.HmiRTm
    - OPC.SimaticNET
      - john2013_a
      - [New group]
    - OPC.SimaticNET.DP
    - OPC.SimaticNET.PD
    - ProfiDrive.ProfilServer
    - Resolvica.Sim
    - Resolvica.Sim2
  - Remote Server(s)
    - Add Remote Servers(s)

Items incl. status information

	Item Names	Value	Format	Type	Access	Quality	Stamp
1	S7:[S7 connection_1]MREAL30	-1775	Original	real32	RW	bad	10/16/20
2	S7:[S7 connection_1]MREAL414	-2000	Original	real32	RW	bad	10/16/20
3	S7:[S7 connection_1]MINT34	0	Original	int16	RW	bad	10/16/20
4	S7:[S7 connection_1]MREAL250	-4	Original	real32	RW	bad	10/16/20
5	S7:[S7 connection_1]MREAL254	9.915364E-4	Original	real32	RW	bad	10/16/20
6	S7:[S7 connection_1]MDWORD258	50363	Original	uint32	RW	bad	10/16/20
7	S7:[S7 connection_1]MDWORD262	2147483648	Original	uint32	RW	bad	10/16/20
8	S7:[S7 connection_1]MREAL54	-28.04398	Original	real32	RW	bad	10/16/20
9							

Figure 4.14: Monitoring of PLC variables in OPC server

In Simulink, an OPC toolbox is provided. Using this toolbox, an OPC Client is set up to access the variables that were configured in the OPC Server. As previously discussed, Matlab or Simulink may then be used as the control platform to remotely execute complex algorithms on the BOW system.

Figure 4.15 shows how the OPC Client toolbox is utilized to read and write information directly to and from the PLC via Simatic net OPC Server. Figure 4.16 shows a screen shot of a customized actuation signal that was created using the Signal Builder function in Simulink for experimental actuation of the servo motor. The feedback signal from the distance sensor is read-in into Matlab/ Simulink in a similar manner using the ‘OPC Read’ block. The captured data can then easily be plotted on a graph that can be used for analytical purposes.

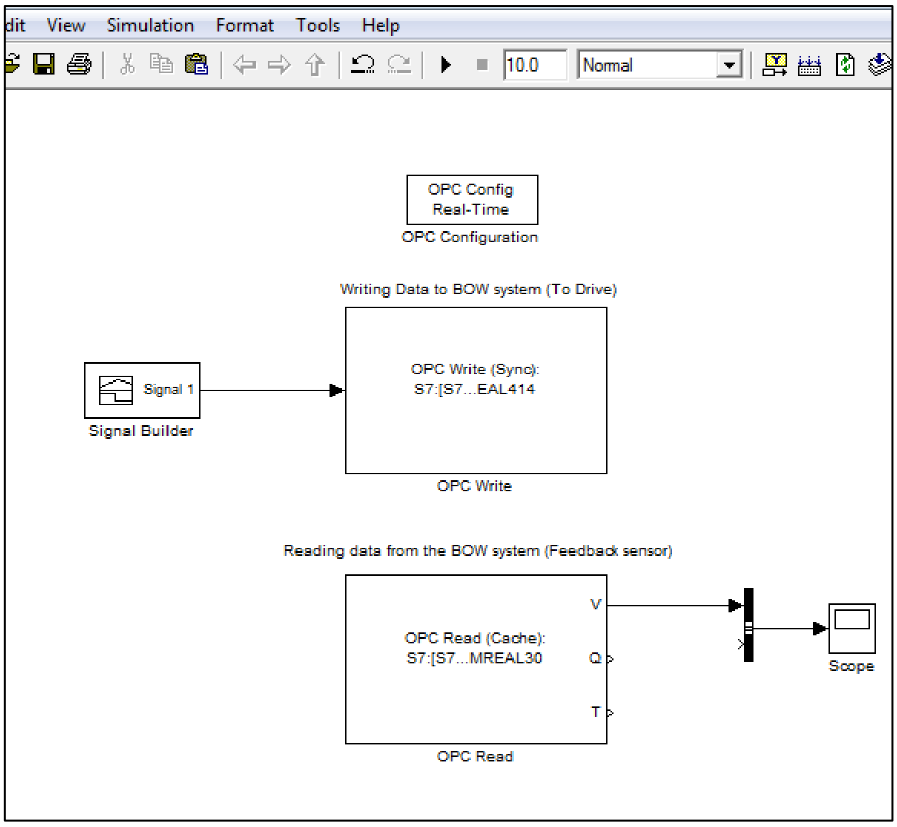

dit View Simulation Format Tools Help
10.0
Normal
OPC Config Real-Time
OPC Configuration
Writing Data to BOW system (To Drive)
Signal 1
Signal Builder
OPC Write (Sync): S7:[S7...EAL414
OPC Write
Reading data from the BOW system (Feedback sensor)
OPC Read (Cache): S7:[S7...MREAL30
V
Q
T
OPC Read
Scope

Figure 4.15: OPC read and write operations in Simulink

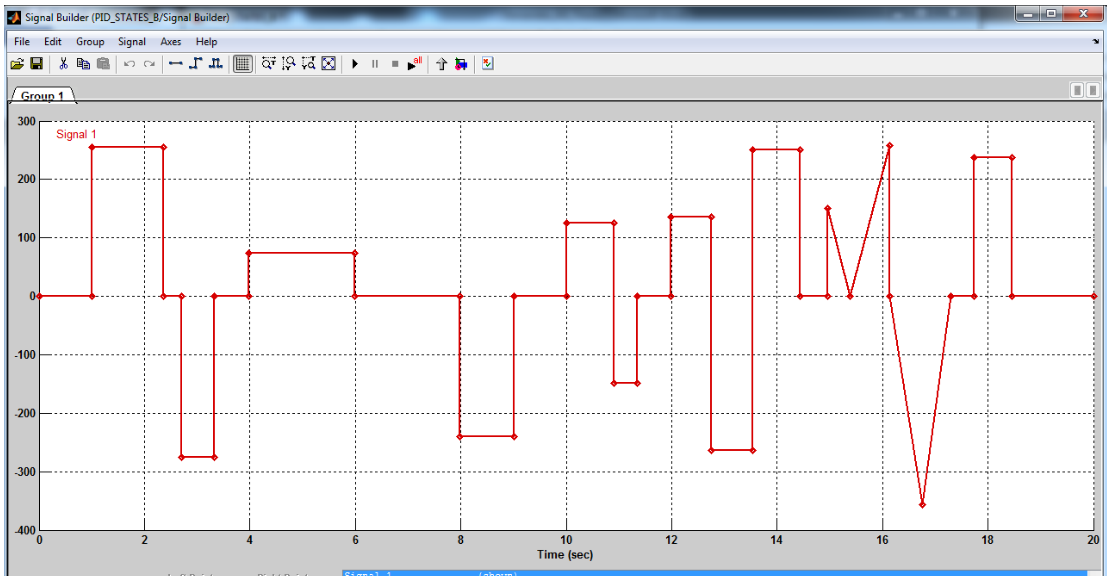


Figure 4.16: Custom signal created in the Signal Builder tool

4.3.3 Siemens S7-300 PLC

4.3.3.1 Area of application of S7-300 PLC

The S7-300 PLC is used almost anywhere where the manufacturing process can be automated. This includes the automobile industry, plastic processing, the packaging industry, chemical plants and also the food and beverage industry just to name a few. The S7-300 is an all-purpose automation system used for applications which require robust control.

The S7-300 PLC has a modular design, meaning that the user may add on modules at any time as they are required. Modules include [62]:

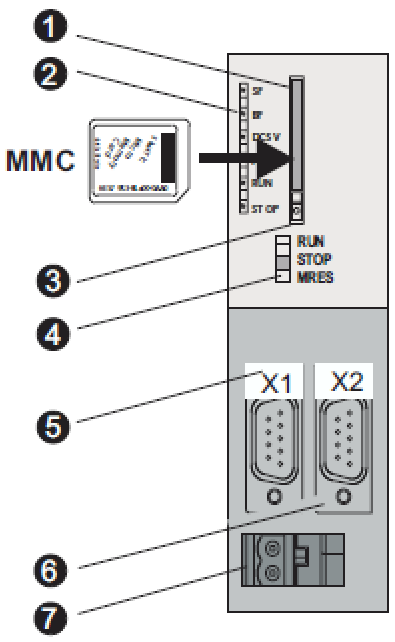
- CPU’s for various performance ranges
- Signal Modules (Digital Input, Digital output and analogue modules)
- Function Modules (FM)
- Communication processors (CP)
- Power supplies (PS)
- Interface modules

4.3.3.2 Important specifications of the S7-315-DP-PN PLC

Table 4-2: Important CPU specifications [62]

CPU 315-2PN/DP	
Programming package	Step 7, from version 5.1
Memory	
Integrated work memory	128 KB
Load Memory	Plugged in MMC (max. 8 MB)
Processing Times	
Bit operations	Min. 0.1 µs
Word instructions	Min. 0.2 µs
Fixed point arithmetic	Min. 0.2 µs
Floating-point maths	min. 6 µs
Timers / Counters	
S7 counters	256
Counting range	0-999
S7 timers	256
Timing range	10 ms to 9990s
Data Area size	
Flag bits (Markers)	2048 bytes
Clock frequency bits	8
Data blocks	1023 (DB 1 to DB 1023)
Local data	Max 1024 bytes per task
Programming blocks	
Total	1024 (FC's and FBs); Also depends on MMC
Nesting depth per priority class	8
Address areas (I/O's)	
Total I/O address area	Max 2048 bytes
Distributed I/O	Max. 2000
Digital channels	Max. 16384
Analogue channels	Max 1024
Signalling Functions	
Number of stations that can log in for signalling functions	16
S7 Communications	
Profinet interface	• Open communication via Profinet and Simaticnet OPC server. • S7 Communication for data exchange between PLCs • Programming, commissioning and diagnostics with Step 7 Connection to HMI and SCADA
DP interface (Profibus)	
• Constant cycle time	yes
• Activate/ deactivate DP slaves	yes
• Transmission rates	12 MBaud
• No. DP slaves per station	124
Voltages and currents	
Power supply	24 VDC
• Permitted range	20.4 V to 28.8V
Current consumption (No-load operation)	Normally 60 mA
Power Consumption	2.5 W

Table 4-2 above shows relevant specifications of the S7-315-DP-PN PLC that was used as the computational platform in the BOW system [62].



1) **Memory card slot (MMC)**
2) **Status and error displays**
3) **MMC ejector**
4) **Mode selector switch**
5) **Interface 1 (x1 MPI – Multipoint Interface)**
6) **Interface 2 (x2 DP peripheral)**
7) **Power supply**

Figure 4.17: Siemens S7-300 CPU

Figure 4.17 depicts the external features of the S7-315 2PN/DP CPU.

4.3.3.3 Interrupts

Most control algorithms are time based and therefore require a fixed sampling rate. In the S7-300 PLC, this is achieved using the cyclic interrupt. Specifically, Organization Block 35 (OB35) is used to generate the cyclic interrupt. The interrupt may be set up to occur periodically between 1 ms - 60000 ms (1 minute). For the BOW system, the cyclic time is set to the smallest available time of 1 ms to allow for rapid execution and update of the implemented control algorithms [62].

4.3.4 Control pendant

A control pendant (shown in Figure 4.18) is wired to the digital I/O card of the PLC. Among other functions, the E-stop is used to provide emergency response in the case of a fault or hazard. Other functions include: start (green push-button), stop (red push-button) and selection of controller modes (PID, Lead, Lead-Lag, NNPID etc.) using the toggle switches.

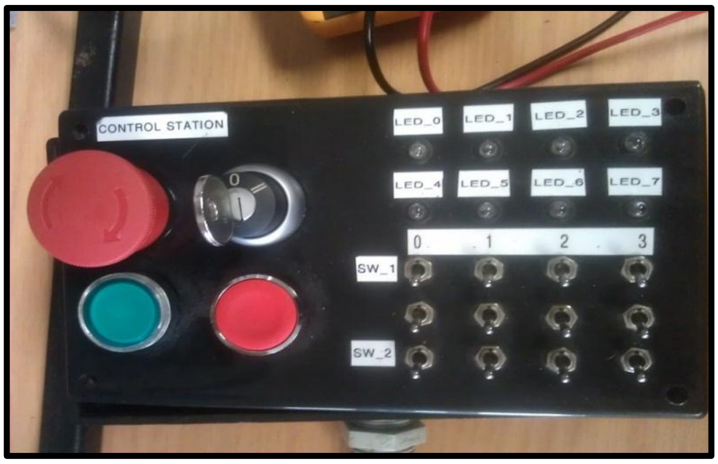


Figure 4.18: Control Box for BOW system

4.3.5 Wireless connectivity

Wireless (remote) connectivity to the BOW system (i.e. to the PLC) is obtained from the control PC using a Siemens Scalance Wi-Fi router shown in Figure 4.19 below. The PLC is given a unique IP address of 192.168.0.1 and subnet a mask of 255.255.255.0. The router and the laptop are also given their own IP addresses within the same subnet. The entire system architecture can be seen in Figure 4.20.

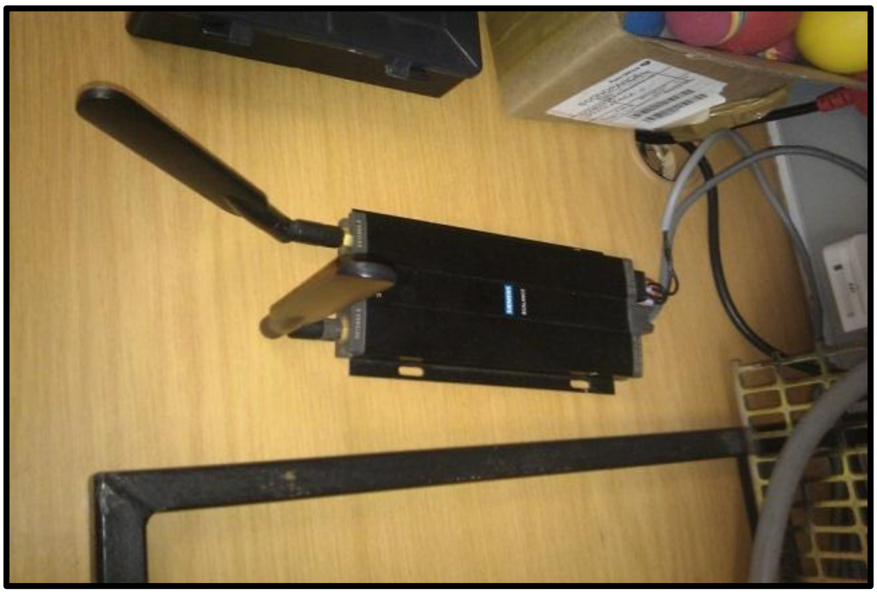


Figure 4.19: Scalance wireless switch (2 port)

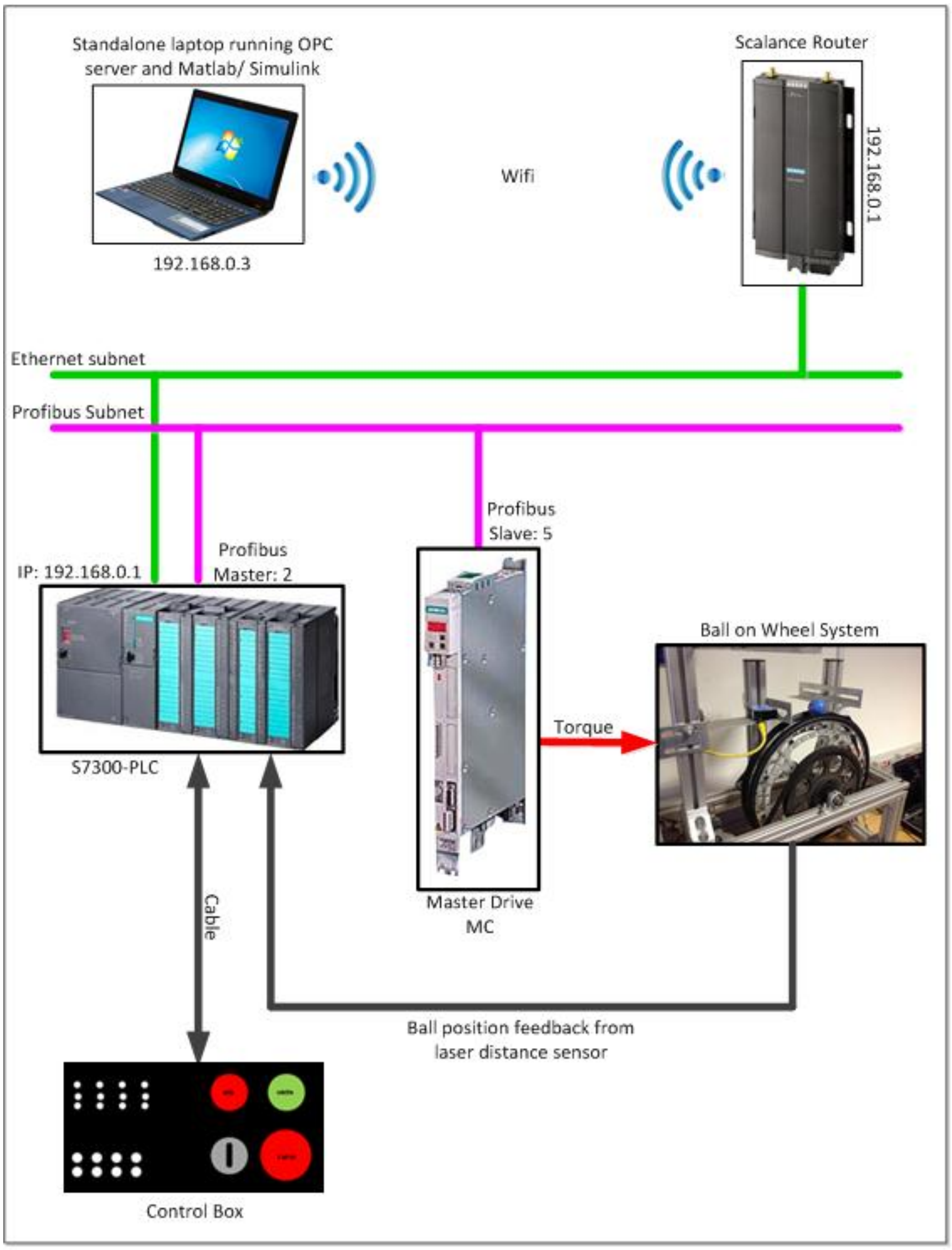


Figure 4.20: Complete system overview

4.3.6 Feedback sensing

The balls angular position (see Figure 4.20 above), is captured using a Sick DT20 laser distance sensor that has a 150 mm measuring range. This distance is read in as voltage on a high specification analogue PLC module that has analogue to digital conversion times in the micro-second range.

The read-in voltage is then calibrated into an angle in degrees. This way, the angle of the ball with respect to the top centre of the wheel is always known in real time and can be used for feedback purposes in the control loop.

4.4 Controller implementation

The controllers designed in section 3.4 which include a PID controller, a Lead and Lead-lag compensator and an NN-PID controller are implemented in their digitized form onto the PLC in SCL. Each controller is created in the form of an FB and called on demand in the cyclic interrupt 0B35. The algorithm to be used is selected by the user from the control pendant. Programming interlocks prevent multiple controllers from being called at the same time. See Appendix C for the implemented PLC code.

4.5 Parameter variation for experimental work

The following system parameters are varied by changing the type of ball that is used in the experiment:

1) Ball radius: the ball radius affects the spherical size of the ball and hence it's contact area with the wheel. Generally, the smaller the ball, the greater the surface contact will be with the wheel periphery. The greater the surface contact, the greater the frictional force between the ball and wheel will be.
   The radius of the ball also affects the ability of the feedback sensor to accurately determine the position of the ball as it rolls on the wheel. Unpredictable non-linear uncertainties are thus created.
2) Surface Texture: balls with smoother surfaces possess lower frictional coefficients when in contact with other surfaces than balls with rougher or more rubbery surfaces. The surface texture also affects the laser distance sensor's ability to accurately read the ball's position on the wheel. Smoother surfaces are more reflective while rough or uneven surfaces are less reflective.
3) Bounce: rubber balls with a hollow interior exhibit more 'bounce' than rubber balls with a solid interior and plastic balls in general.
4) Ball weight distribution and mass-moment of inertia: the more mass a ball has concentrated at its centre, the easier it is for it to get rolling. Conversely, balls that have their mass distributed away from their core (i.e. hollow balls) are

generally more difficult to get rolling. Heavier balls are more difficult to slow down if rolling too quickly. Lighter balls with less mass concentrated at the centre are generally easier to control but tend to be over-responsive, reacting even to slight wheel movements.

Table 4-3 in section 4.6 describes the features and specifications of each ball used in the experiment.

4.6 Experimental strategy

The BOW system is a non-linear dynamical system that is naturally unstable. For this reason, a controller is required in order to maintain stability and thus prevent the ball from rolling off the wheel surface. The system's equilibrium point is found at 0 degrees (i.e. the top centre of the wheel). The implemented controllers try to restore the system to this point whenever a disturbance is introduced. In the following sections, the response of the BOW system while traditional control strategies are implemented is compared to its response while NN control is implemented. To clearly observe the system's response, 2 pulses which act as disturbance inputs are injected into the system at 2 second intervals.

Each implemented controller, with the exception of the NN-PID controller is tuned to balance a specific ball (Ball E is picked because it is an average specification ball that is well suited to the BOW system - see Table 4-3). As the balls are altered, plant parameters such as the ball weight, radius and moment of inertia are also changed. According to the system model, this changes the plant dynamics and has a direct influence on the extent of control possible. Unpredictable non-linear disturbances are also introduced (such as feedback sensor distortion or noise, surface contact friction, uneven contact surfaces that disrupt the motion of rolling balls and ball bounce etc.). These uncertainties could not be accounted for by the system model because of there being no way to measure them. The balls listed in Table 4-3 increase in radius and mass from Ball A to F. Ball G also increases in radius but has a mass of only 52.3g.

Table 4-3: Ball specifications

	Ball	Radius	Mass	Description
A		20.75 mm	7.2 g	• Light weight, uneven texture, hollow ball • Little or no bounce
B		20.25 mm	3.6 g	• Small, very light ball with hardly any bounce • Multi-colour surface to create sensory disturbance
C		21.15mm	23 g	• Soft rubber squash ball; fits well in wheel groove, hence more surface contact with wheel
D		25.5 mm	99.5 g	• Smooth billiard ball painted blue • Hardly any 'bounce'
E		24.65 mm	100 g	• Medium sized, hard rubber ball • Reflects laser beam well • Exhibits little bounce
F		27.2 mm	181 g	• Uneven surface, solid core ball • Odd shape affects rolling ability and sensory ability
G		30 mm	52.3 g	• Medium Size Rubber ball exhibiting tremendous 'bounce' • Dual colour – light and dark to create a sensory disturbance

Mechanical limits exist on the BOW system at +/- 30 degrees from the top centre of the wheel. The ball may sometimes roll on top of the limit, giving the false impression that the ball is rolling beyond 30 degrees. For this reason (let the reader take note), should the ball go beyond +/- 30 degrees, it may be considered that stability has been lost.

4.7 Summary

In this chapter the BOW research platform is introduced in its entirety. Its mechanical and electrical construction is reviewed in light of the research goals. The function and interplay of all system components including the PLC, the processing software, the wireless access and control, the drive and the mechanical structure are discussed. The chapter details the methods used to create deliberate parameter variation in order to test and compare the implemented controllers. It also highlights the experimental approach followed. The BOW system's non-linear nature makes it ideal for investigating the shortfalls found with traditional control algorithms and exposes the need for more modern or intelligent methods of control.

Chapter 5.
RESULTS AND DISCUSSIONS

5.1 Introduction

In this chapter the response of the BOW system is reviewed as various control algorithms are implemented in real-time. Balls of different weight, radius and surface texture are used to test the implemented controller’s capability as system parameters are deliberately varied as discussed in sections 4.5 and 4.6.

One of the main goals of this research is to compare the ability of traditional and modern control techniques in controlling a non-linear plant when implemented on a standard PLC. In this regard, the following digitized control algorithms were implemented:

1) PID controller (traditional controller)
2) Lead compensator (traditional controller)
3) Lead-Lag compensator (traditional controller)
4) NN-PID controller (modern/ intelligent controller)

The implemented PLC code can be found in Appendix C.

Other aspects of the research include wireless (remote) visualization and control of the BOW plant as well as wireless logging of plant data in Matlab and Simulink for purposes of further analysis.

5.2 Traditional vs. Intelligent control on the BOW System

The investigation interrogates the ability of each implemented controller to maintain system stability and remain within the control system design specifications (as laid out in section 3.3) even while plant parameters are varied or while the plant is subjected to unpredictable non-linear disturbances. The performance of the traditional controllers is compared to that of the intelligent controller in the time-domain (specifically looking at the system settling time and the steady state error). The results obtained in the following sections were obtained in Matlab in real-time over a wireless connection.

5.2.1 Training the NN-PID controller

In order to optimally control the BOW system, the online NN controller must undergo a short training period each time plant parameters are changed or a non-linear disturbance is introduced (i.e. a different ball is used). For simplicity, only two randomly selected sets of training plots (for Ball A and Ball D) are showcased in the results section. The remaining controller training plots may be found in Appendix E.

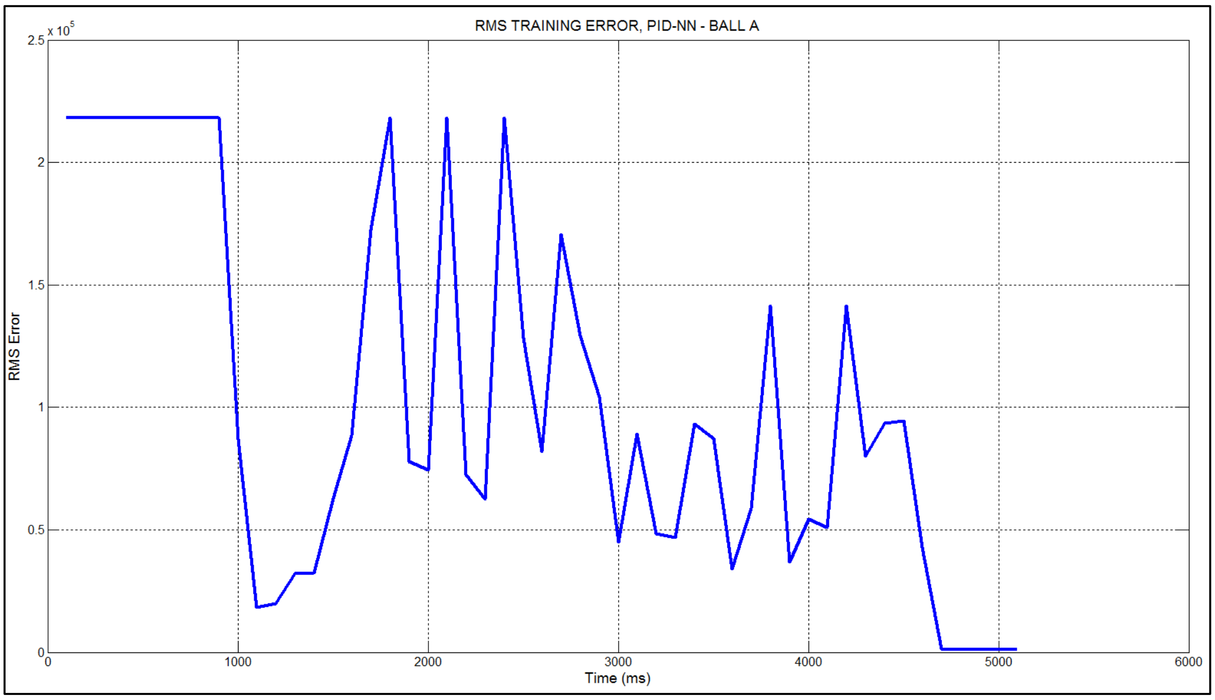


Figure 5.1: RMS training error for Ball A

The NN updates its connective weights according to the back propagation (steepest descent) learning algorithm. The training process occurs online, meaning that during the training process the plant may temporarily display erratic or unstable behavior. The goal of the training algorithm is to update the networks connective weights such that the system’s Root-Mean-Square (RMS) error is reduced to an acceptable minimum value that is specified by the user. Figure 5.1 shows graphically how the RMS error is gradually reduced for Ball A over a 6 second period. During the training process, the plant was manually disturbed a few times to enhance the training and thus make the controller more robust. Figure 5.2 to Figure 5.4 show how the NNs hidden-to-output layer weights, which are essentially taken to be PID gains, are updated for Ball A during training.

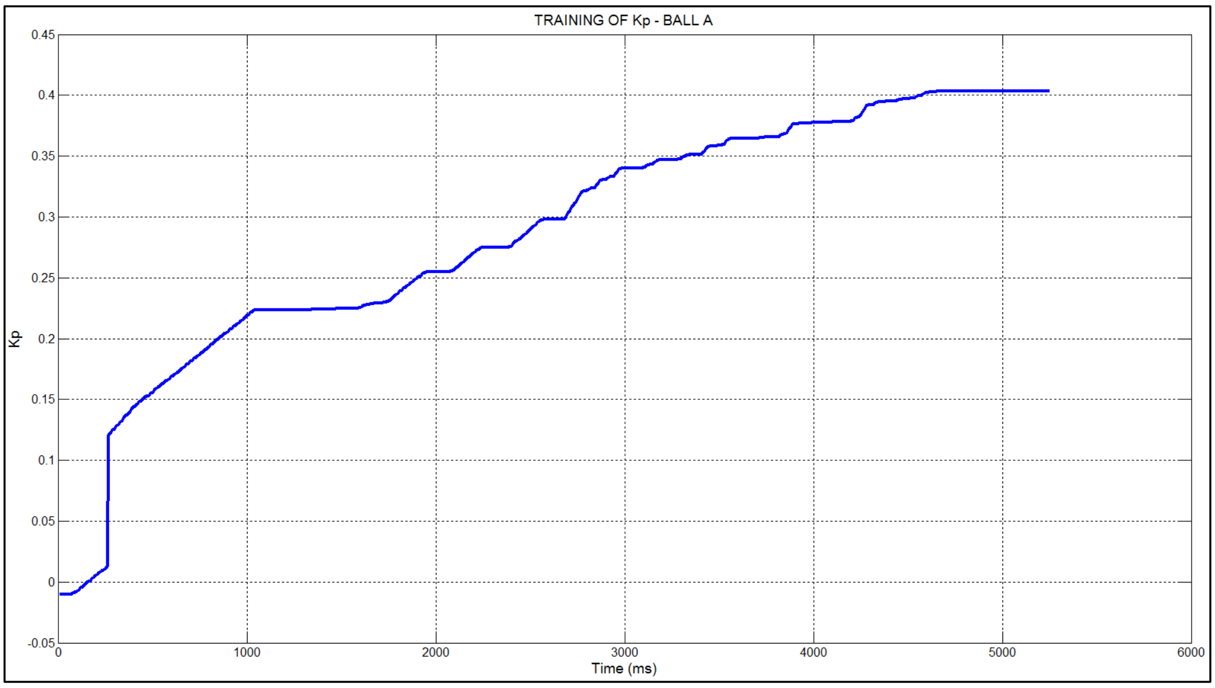


Figure 5.2: Training of Kp for Ball A

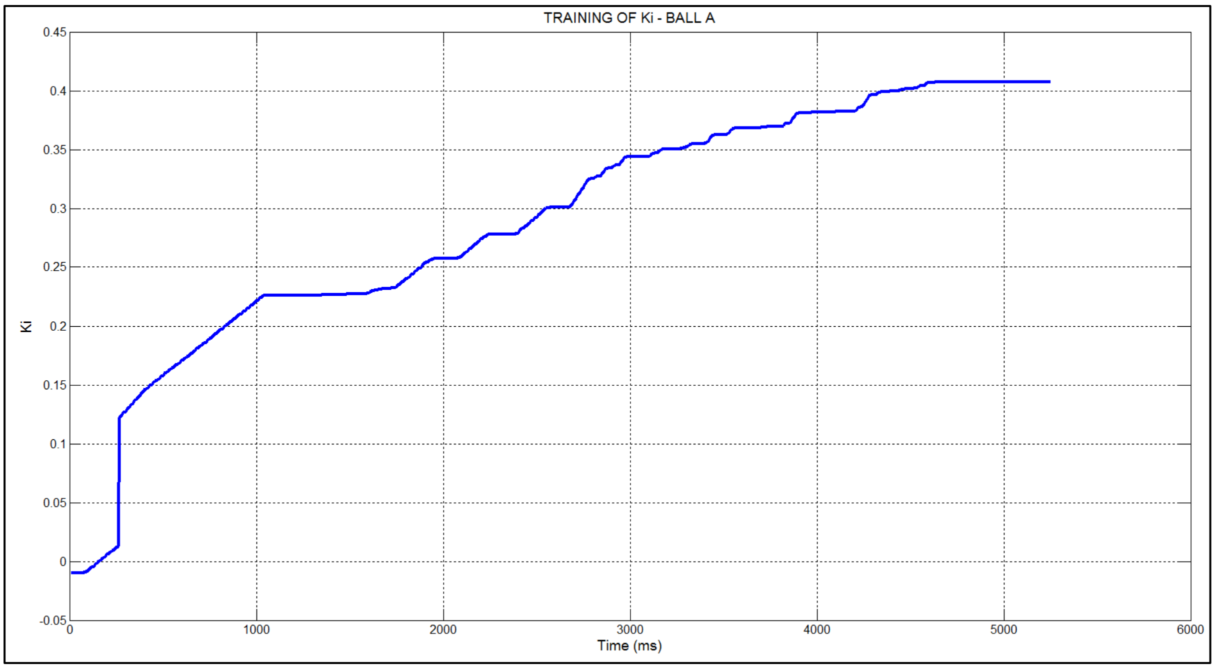


Figure 5.3: Training of Ki for Ball A

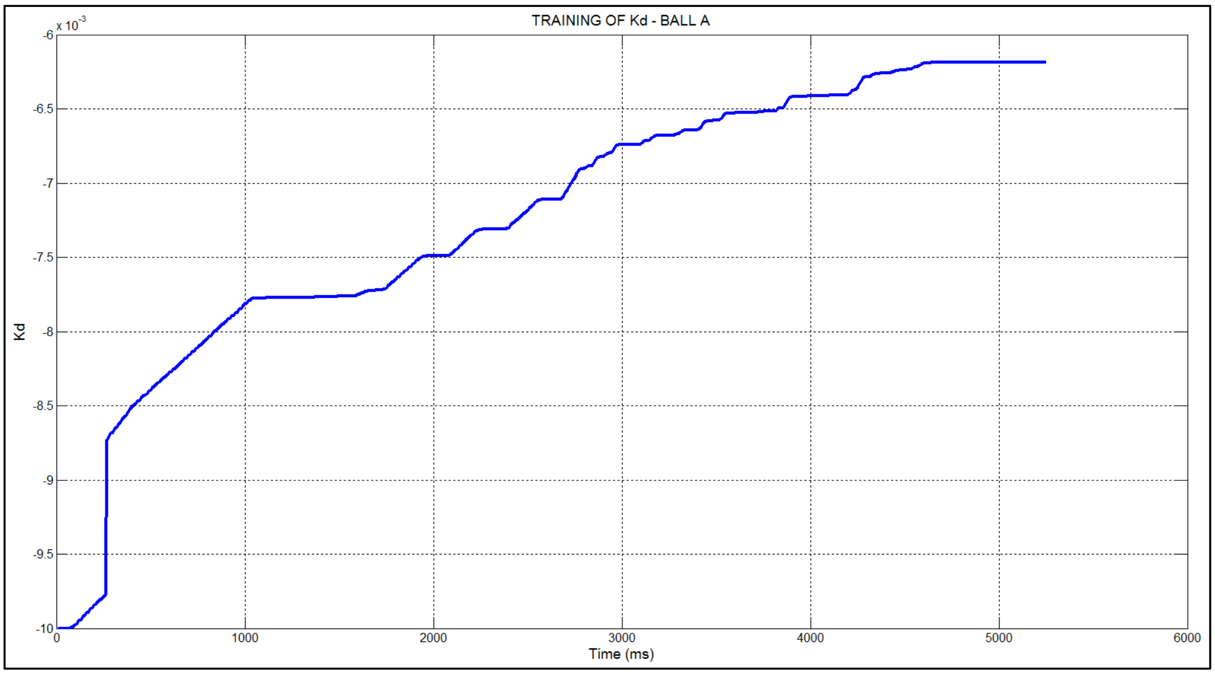


Figure 5.4: Training of Kd for Ball A

Depending on the NNs initial weights as well as the chosen learning rate. The system’s training time may vary. For Ball A, training took about 5 seconds. Figure 5.5 to Figure 5.8 show how the NN learns to balance Ball D. Training for Ball D took about 8 seconds. For all balls used, the final gains occur when the RMS error has reached its minimum possible value. At this point the training is discontinued.

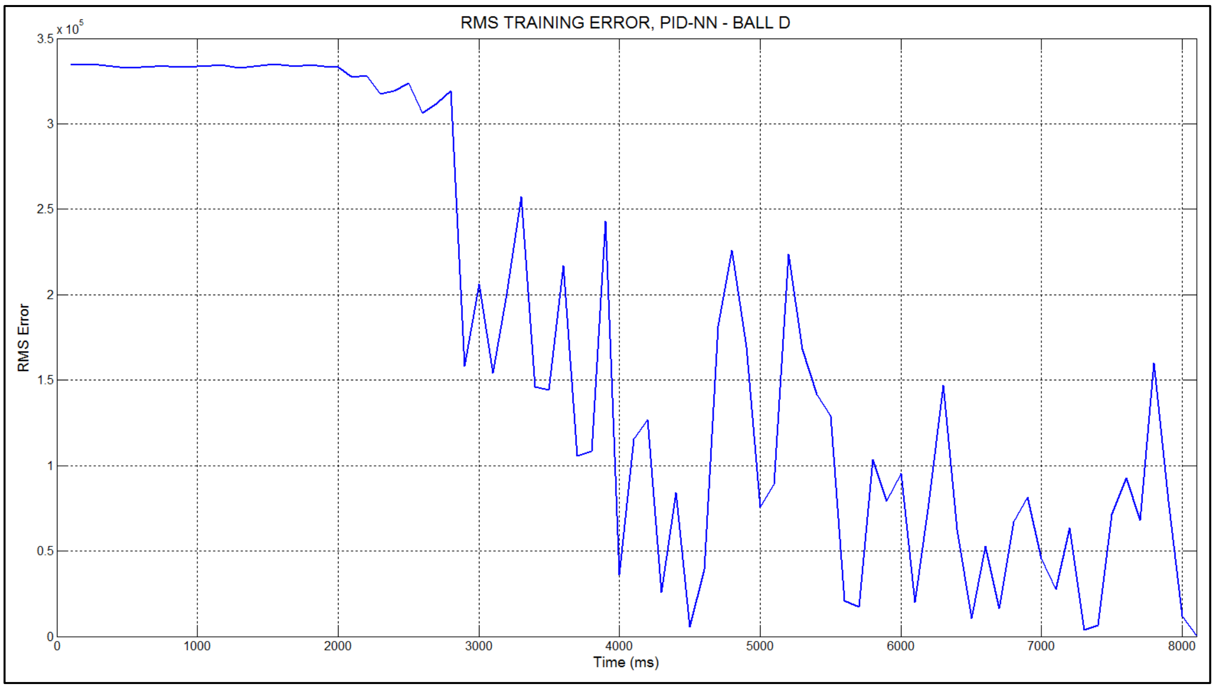


Figure 5.5: RMS training error for Ball D

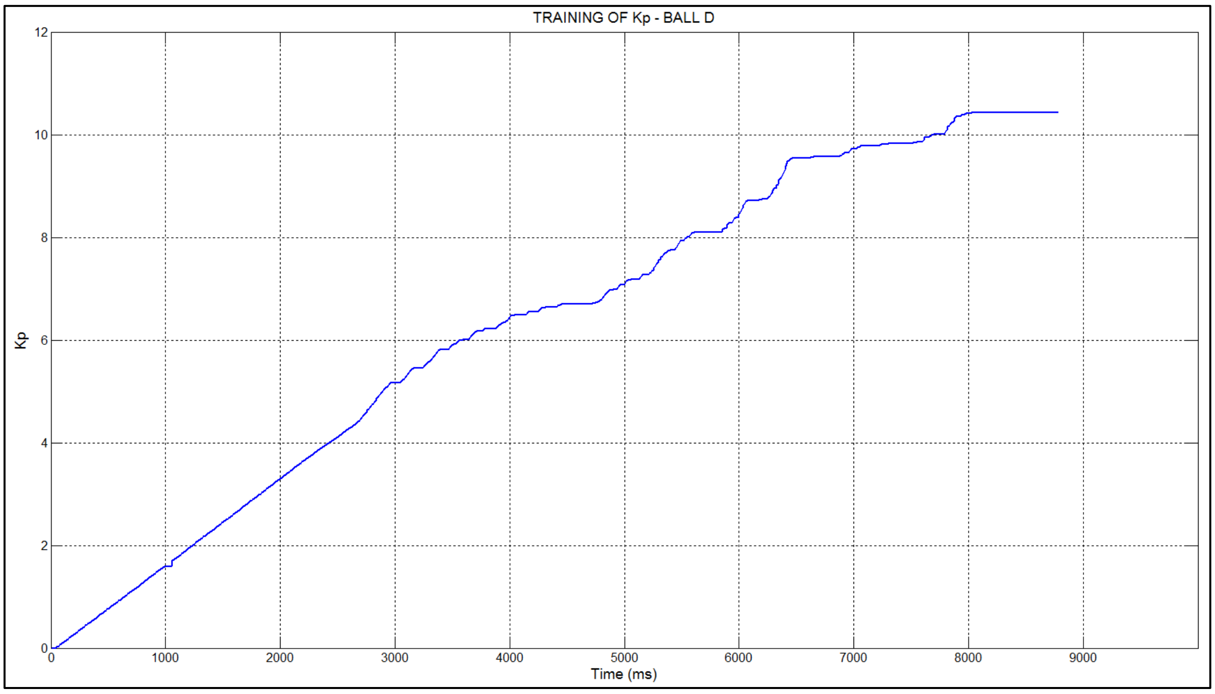


Figure 5.6: Training of Kp for Ball D

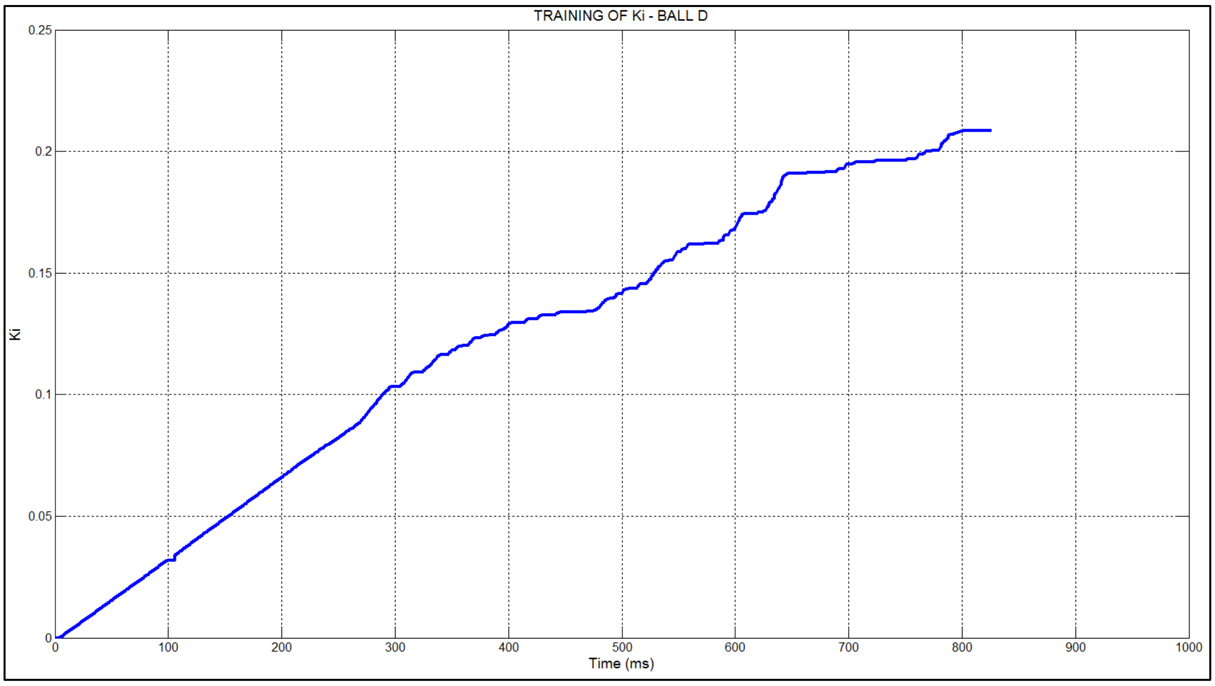


Figure 5.7: Training of Ki for Ball D

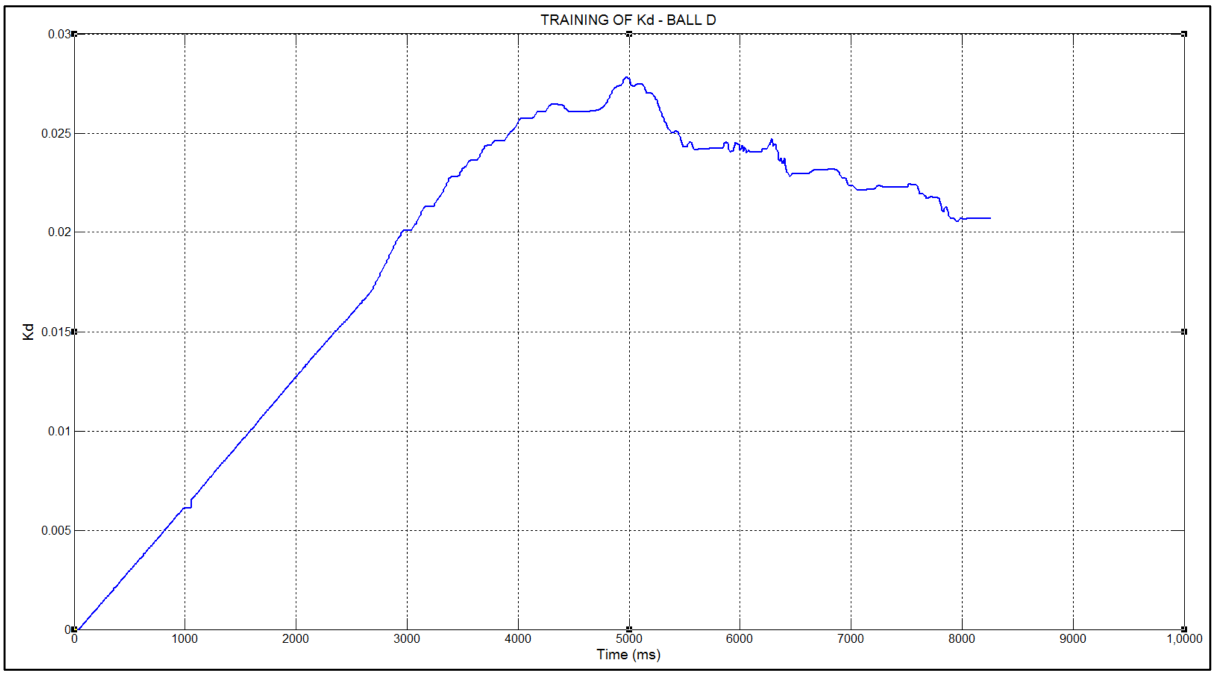


Figure 5.8: Training of Kd for Ball D

5.2.2 PID vs. NN-PID control

In this section the BOW system’s response curves produced by the PID controller are superimposed on the response curves produced by the NN-PID controller. The response curves were produced as the system was subjected to a variety of balls as shown in Table 4-3. Each controller was tested separately, yet under exactly the same conditions. In the case of the NN-PID controller, the trials were carried out only once the NN-PID controller had been trained for each ball. The same pulse disturbance was applied to every ball at 2 second intervals. Figure 5.9 through to Figure 5.15 depict the obtained results. Based on these results, the controllers are then compared in detail with each other in sections 5.26, 5.27 and 5.28.

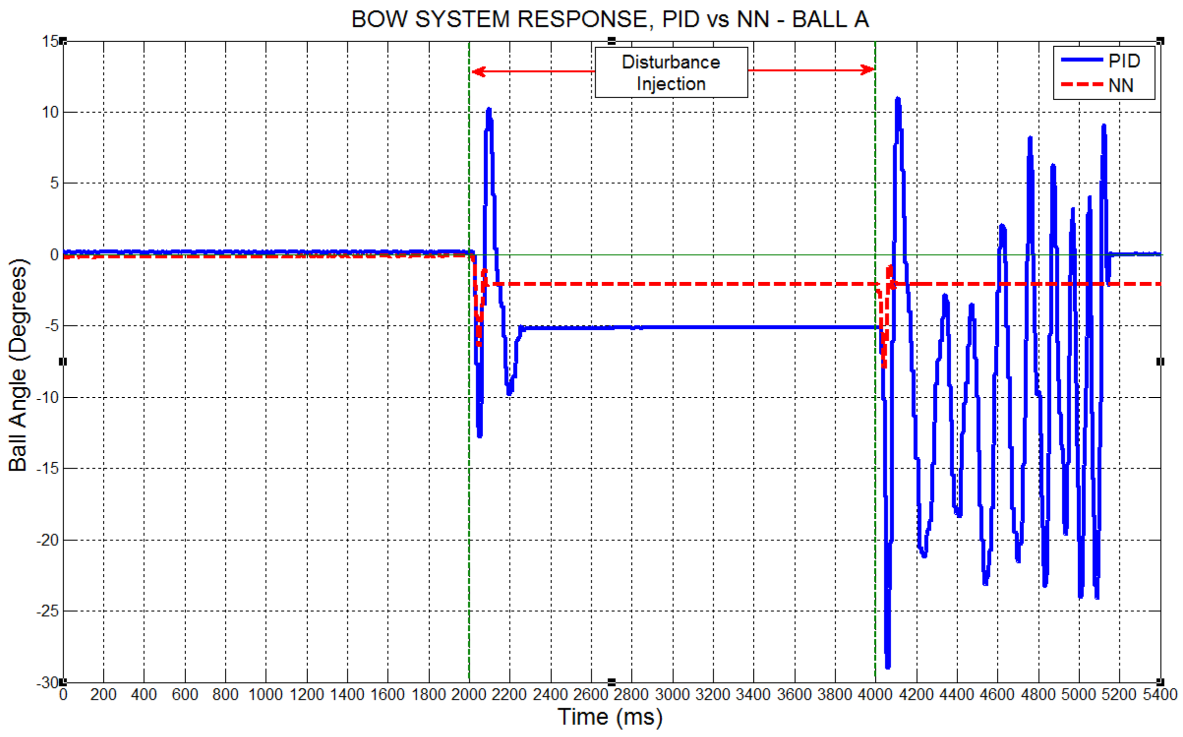


Figure 5.9: PID controller vs. NN-PID, Ball A

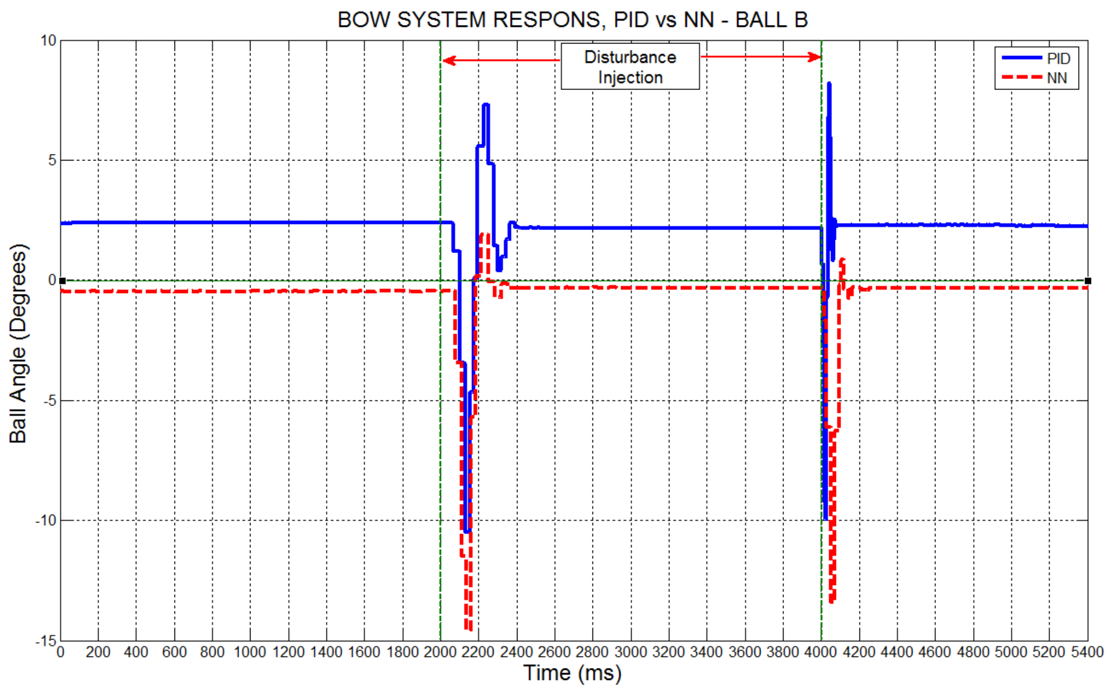


Figure 5.10: PID controller vs. NN-PID, Ball B

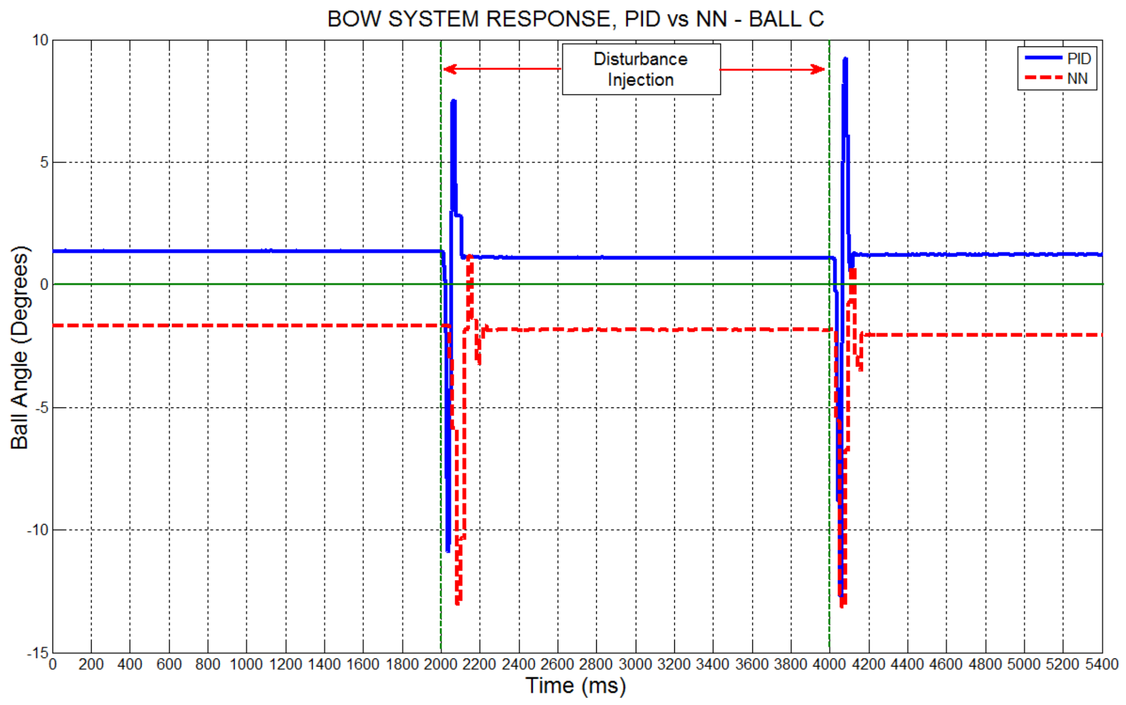


Figure 5.11: PID controller vs. NN-PID, Ball C

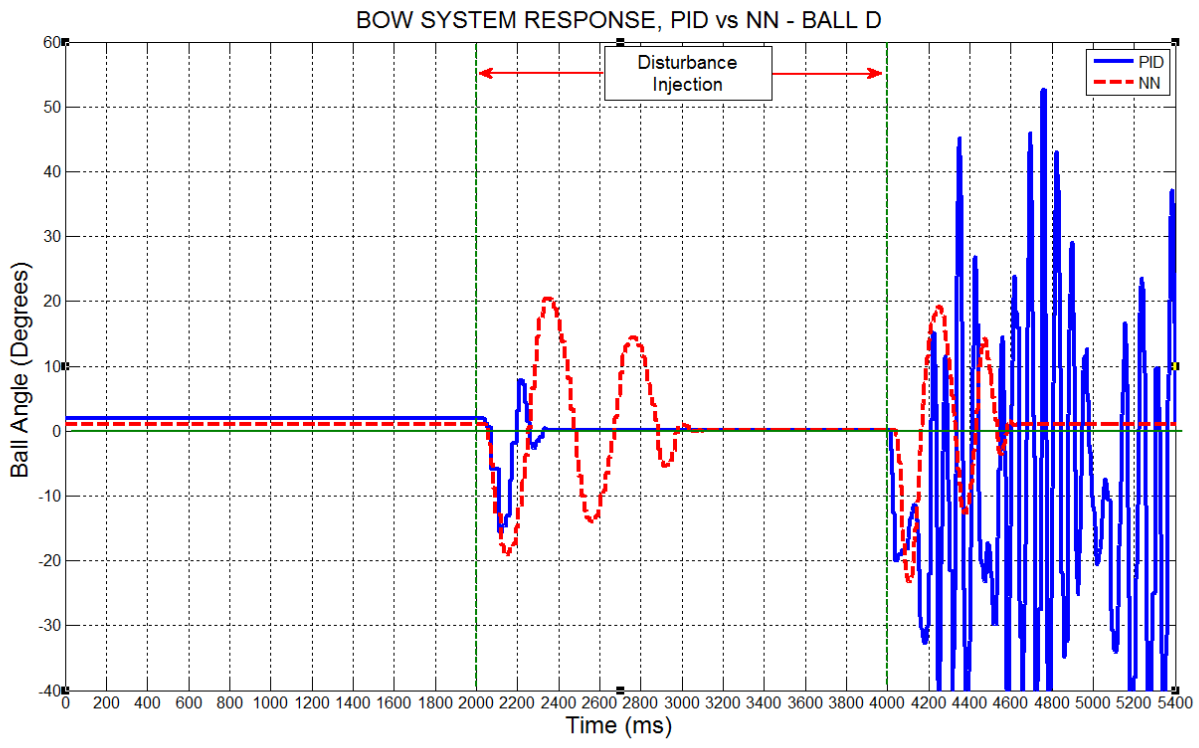


Figure 5.12: PID controller vs. NN-PID, Ball D

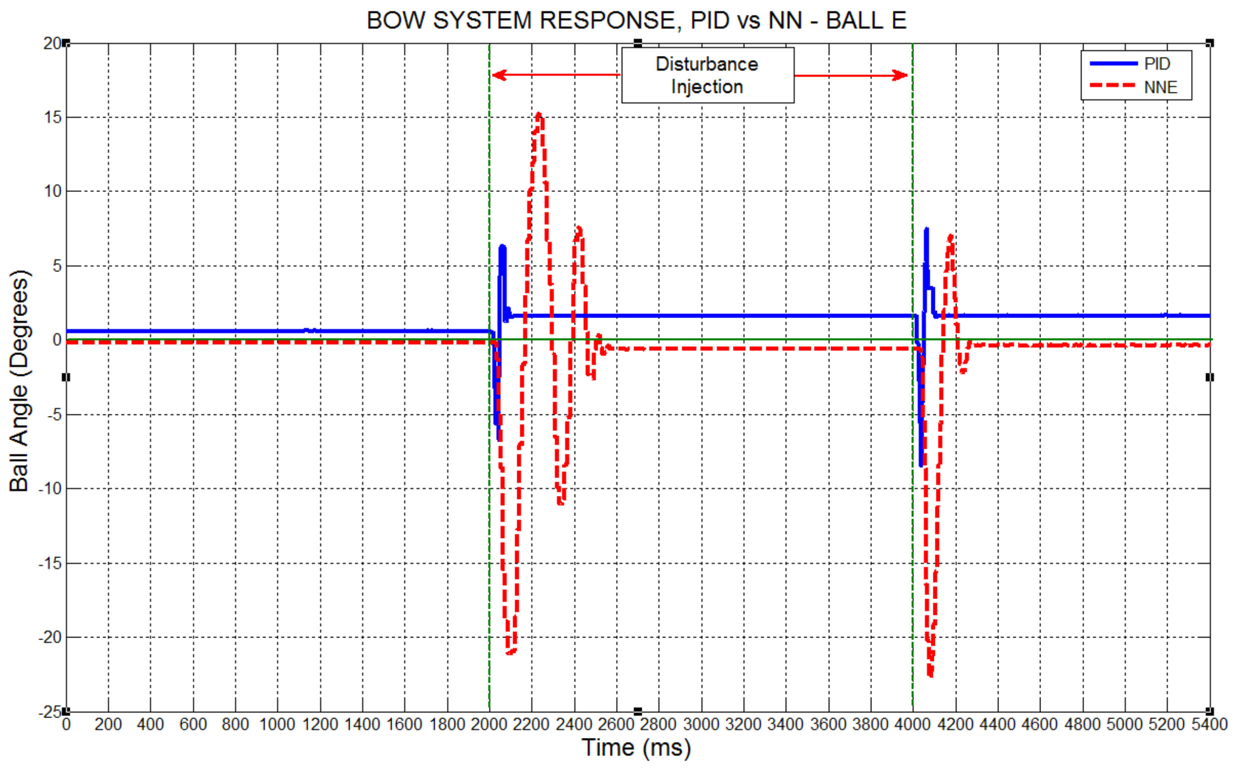


Figure 5.13: PID controller vs. NN-PID, Ball E

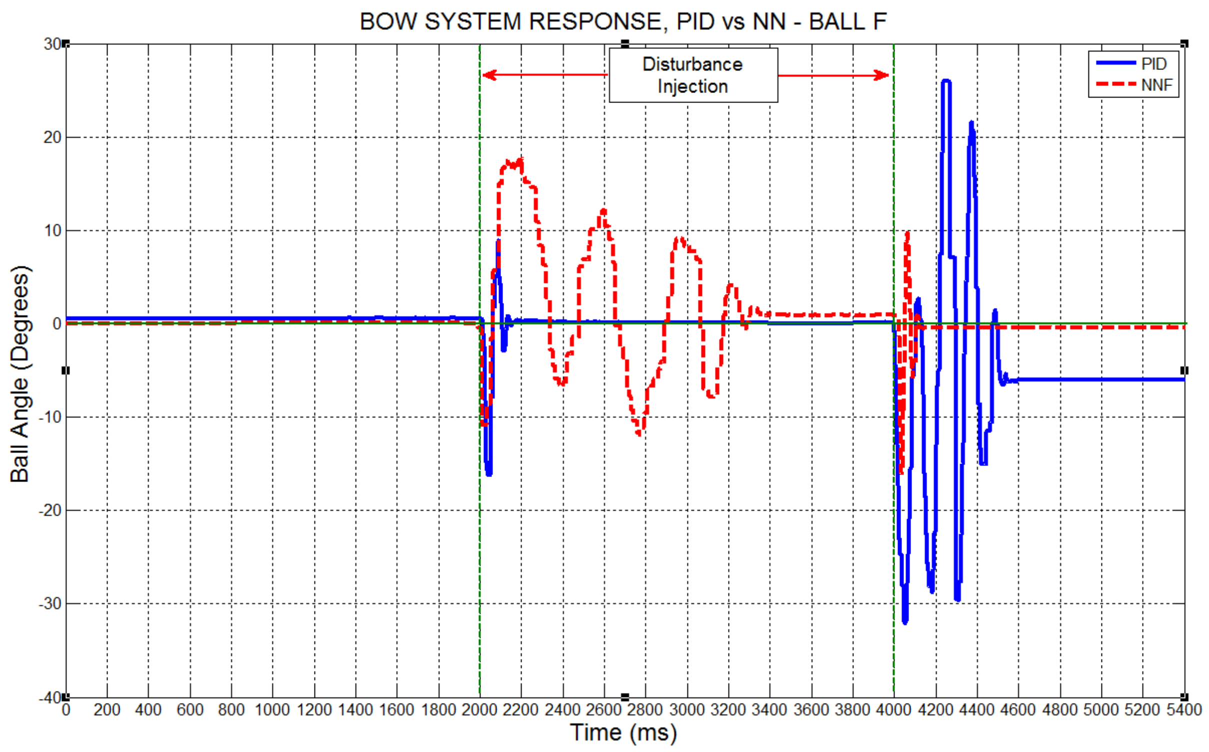


Figure 5.14: PID controller vs. NN-PID, Ball F

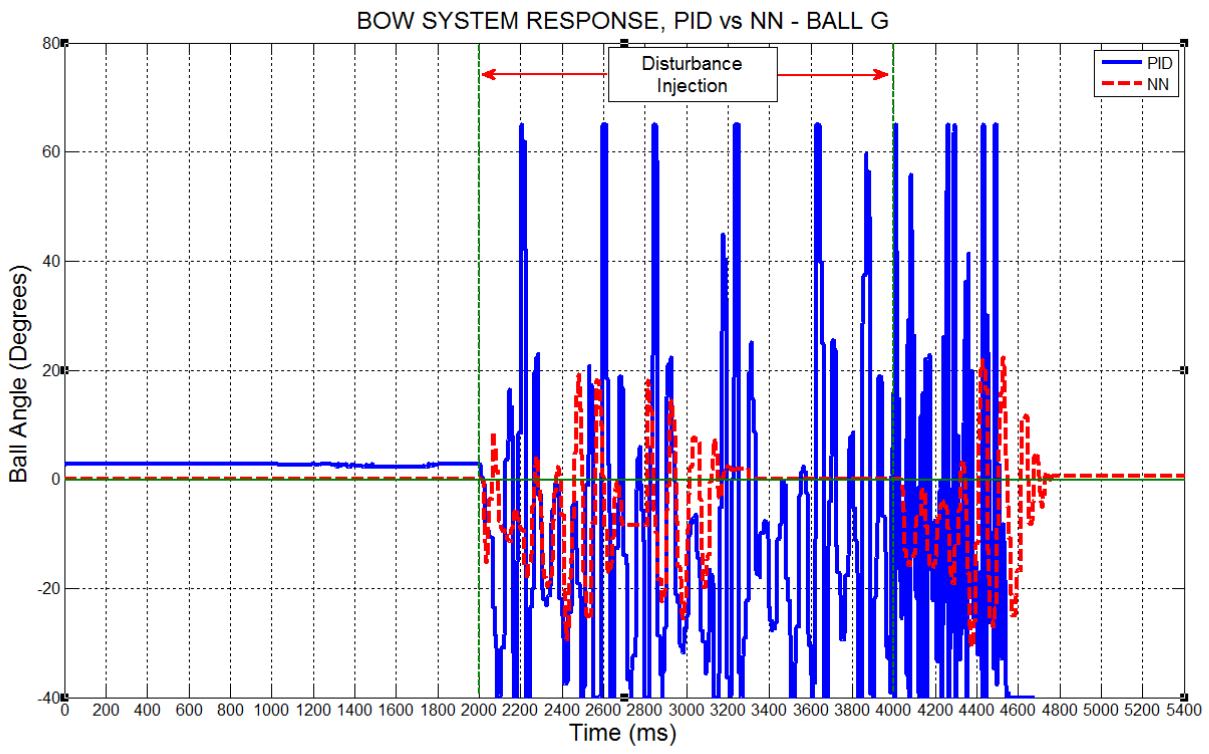


Figure 5.15: PID controller vs. NN-PID, Ball G

5.2.3 Lead compensator vs NN-PID control

In this section the system response curves produced by the Lead compensator are superimposed on the response curves produced by the NN-PID. The response curves were generated as the system was subjected to a variety of balls as shown in Table 4-3. These curves are shown in Figure 5.16 through to Figure 5.22. Based on these results, the controllers are then compared in detail with each other in sections 5.26, 5.27 and 5.28.

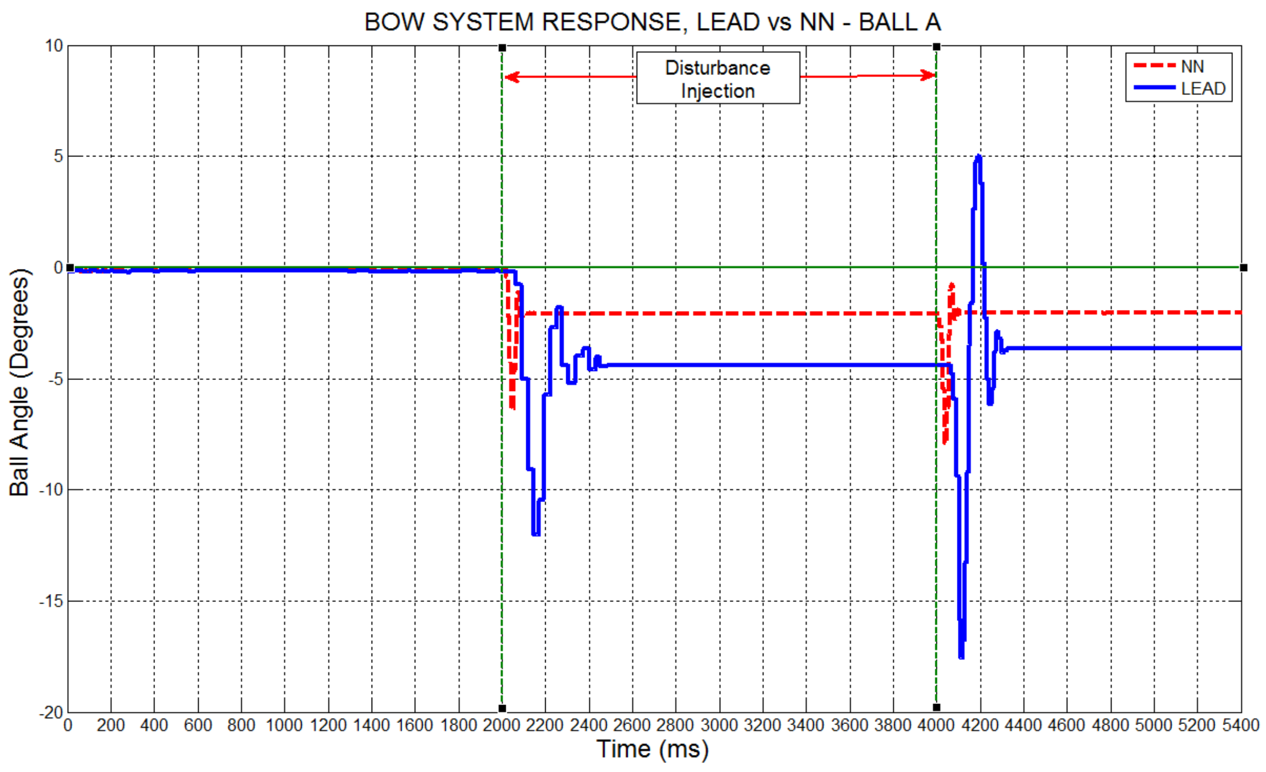


Figure 5.16: Lead compensator vs. NN-PID, Ball A

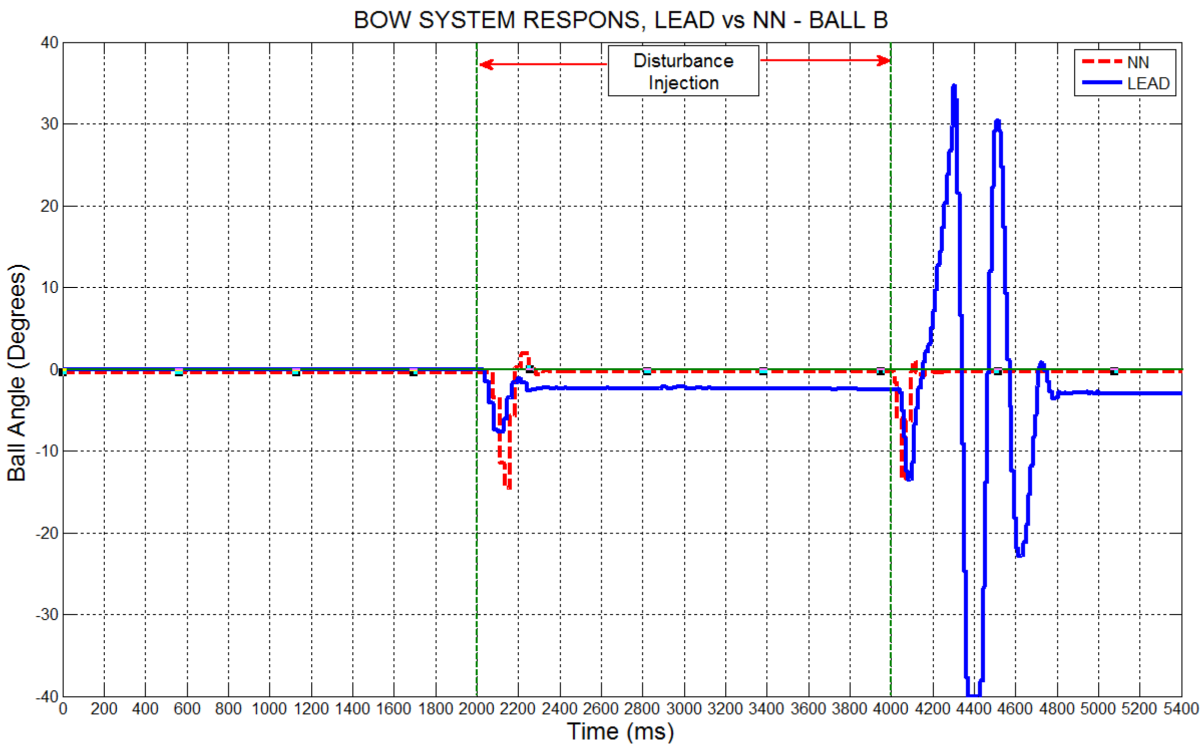


Figure 5.17: Lead compensator vs. NN-PID, Ball B

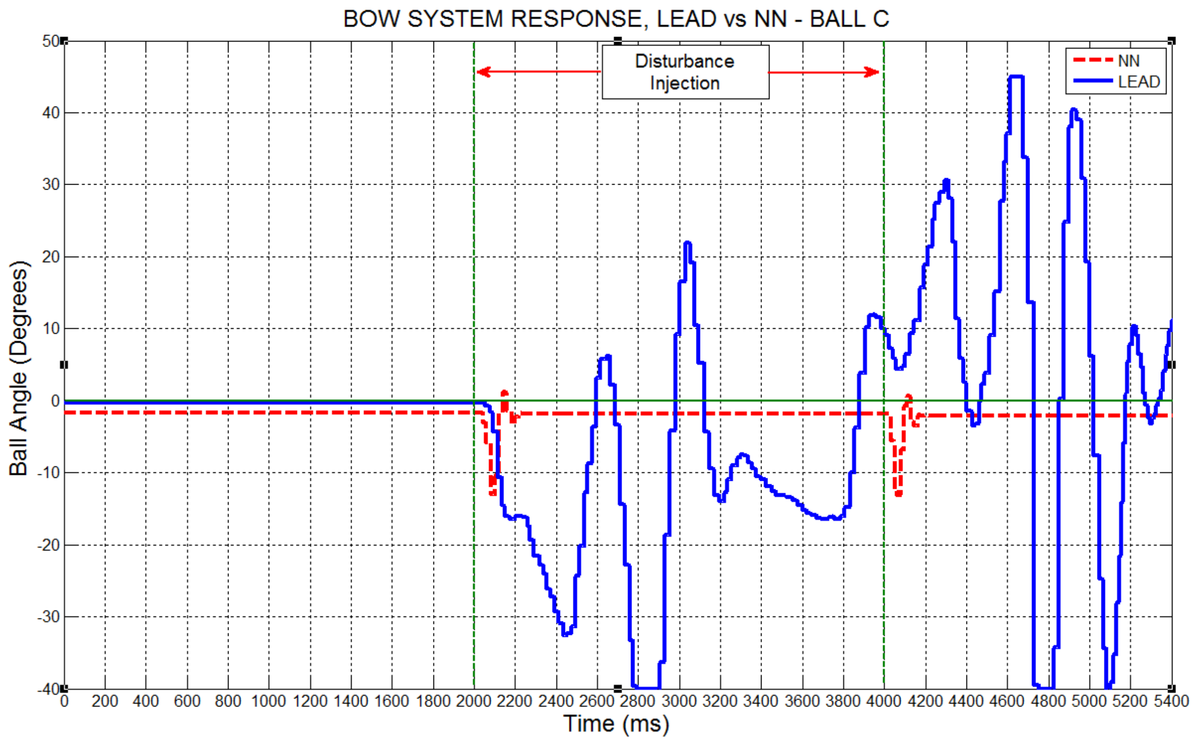


Figure 5.18: Lead compensator vs. NN-PID, Ball C

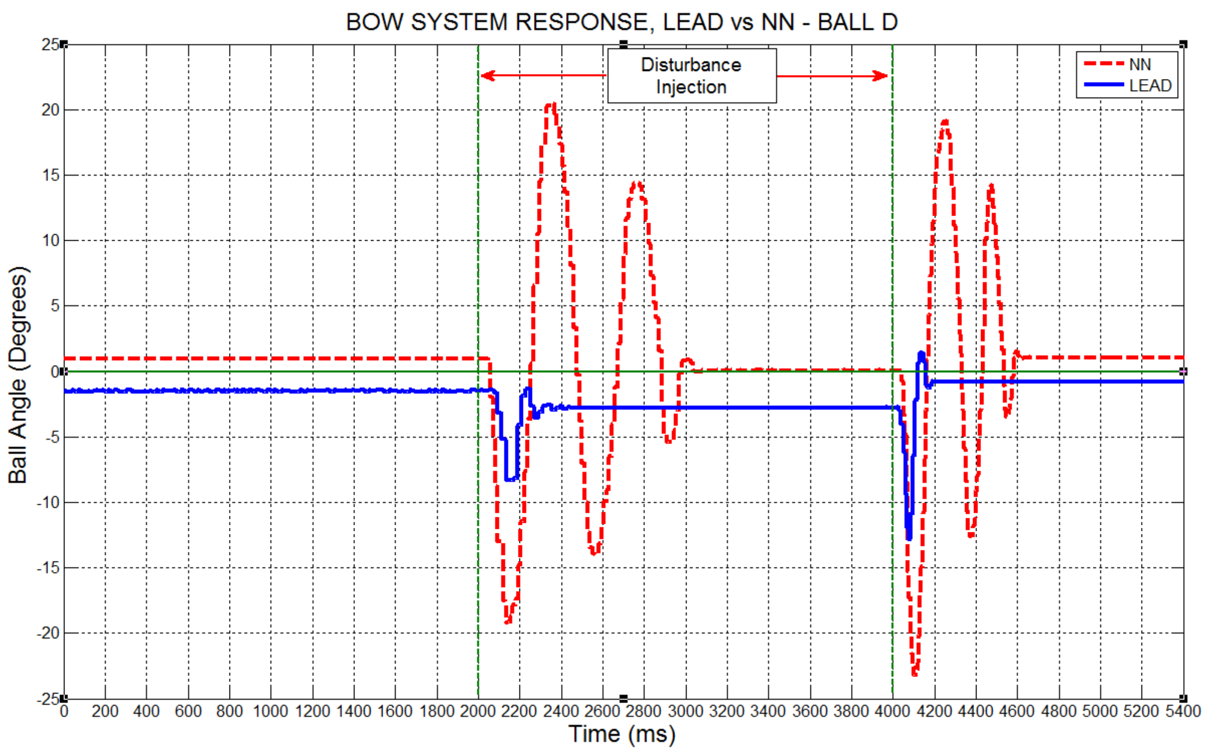


Figure 5.19: Lead compensator vs. NN-PID, Ball D

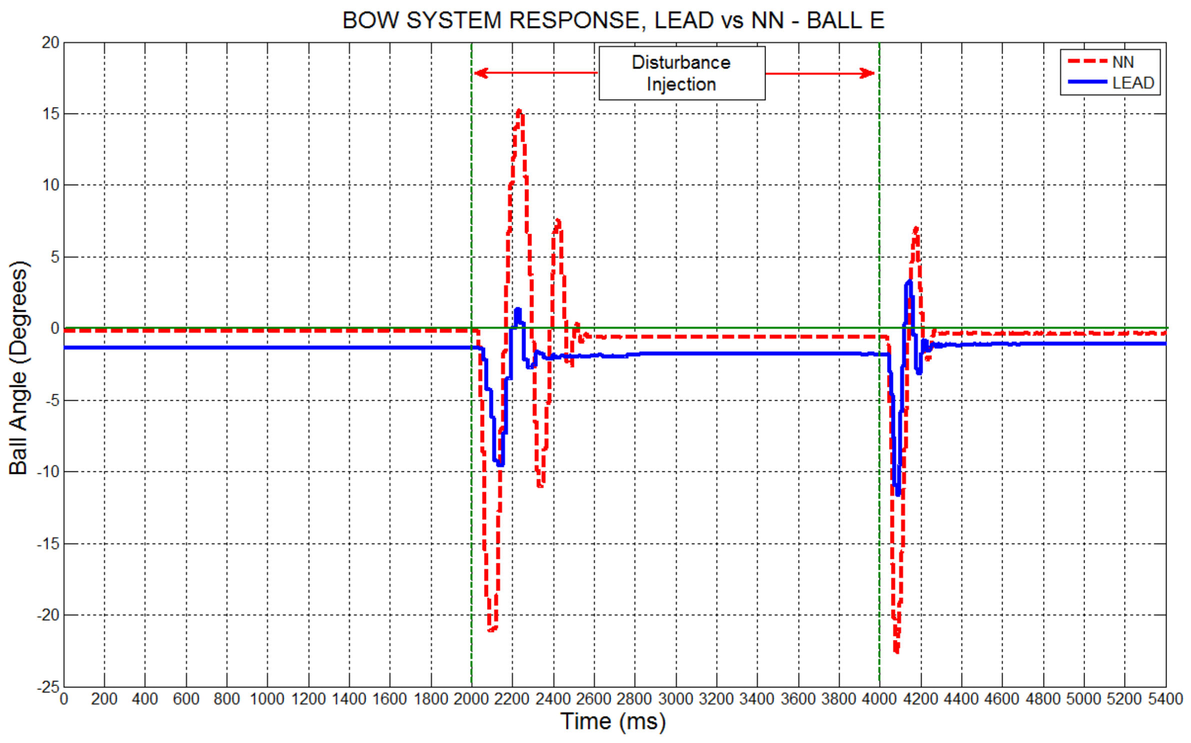


Figure 5.20: Lead compensator vs. NN-PID, Ball E

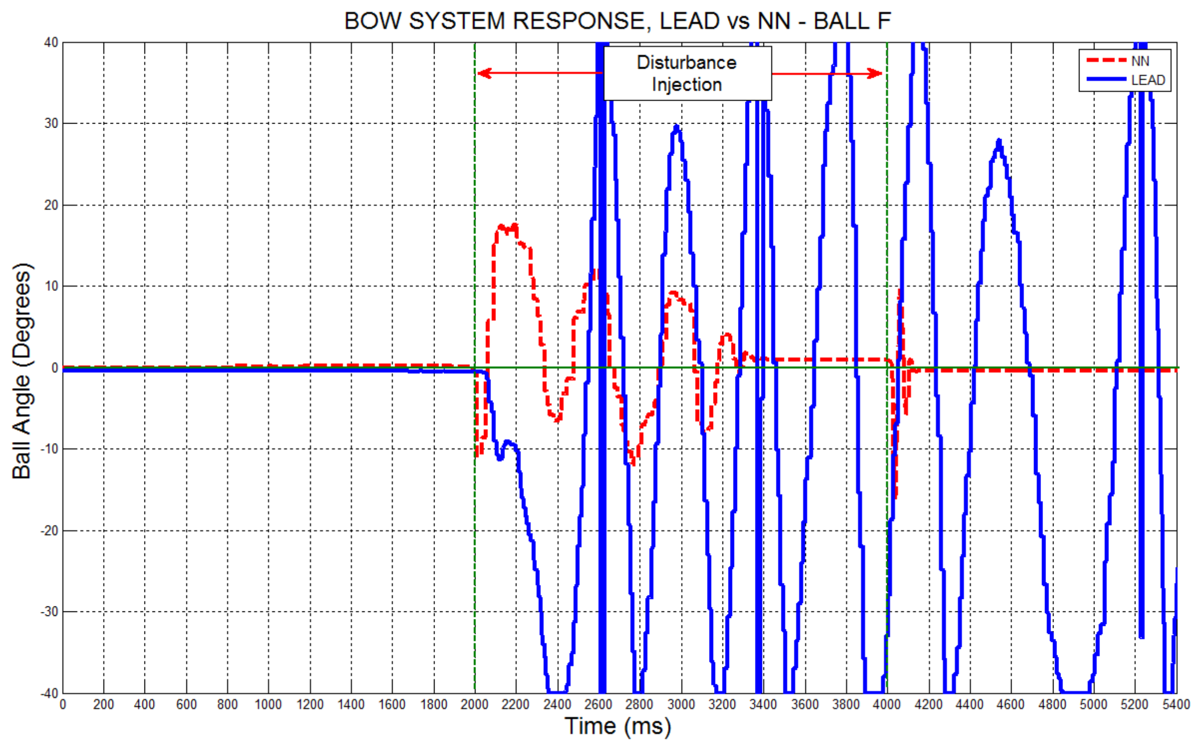


Figure 5.21: Lead compensator vs. NN-PID, Ball F

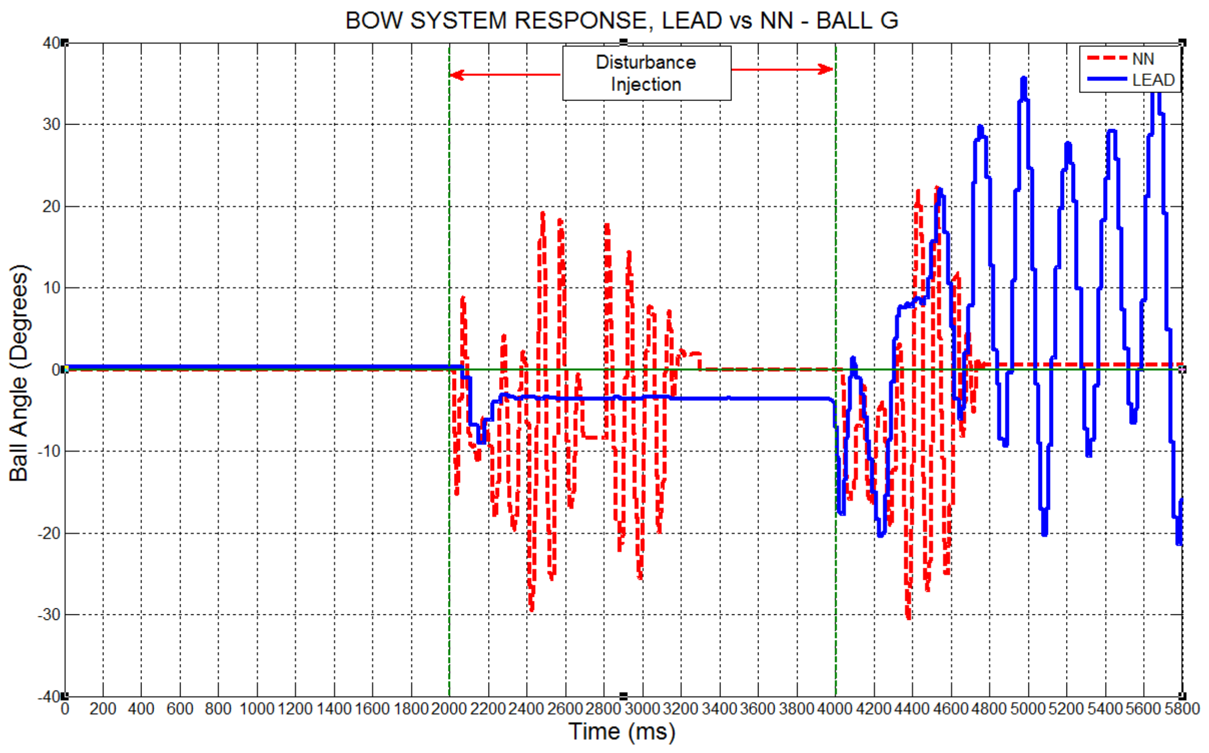


Figure 5.22: Lead compensator vs. NN-PID, Ball G

5.2.4 Lead-Lag compensator vs. NN-PID control

In this section the system response curves produced by the Lead-lag compensator are superimposed on the response curves produced by the NN-PID controller as the system was subjected to the balls shown in Table 4-3. These curves are shown in Figure 5.23 through to Figure 5.29. The results are then examined in detail in sections 5.26, 5.27 and 5.28.

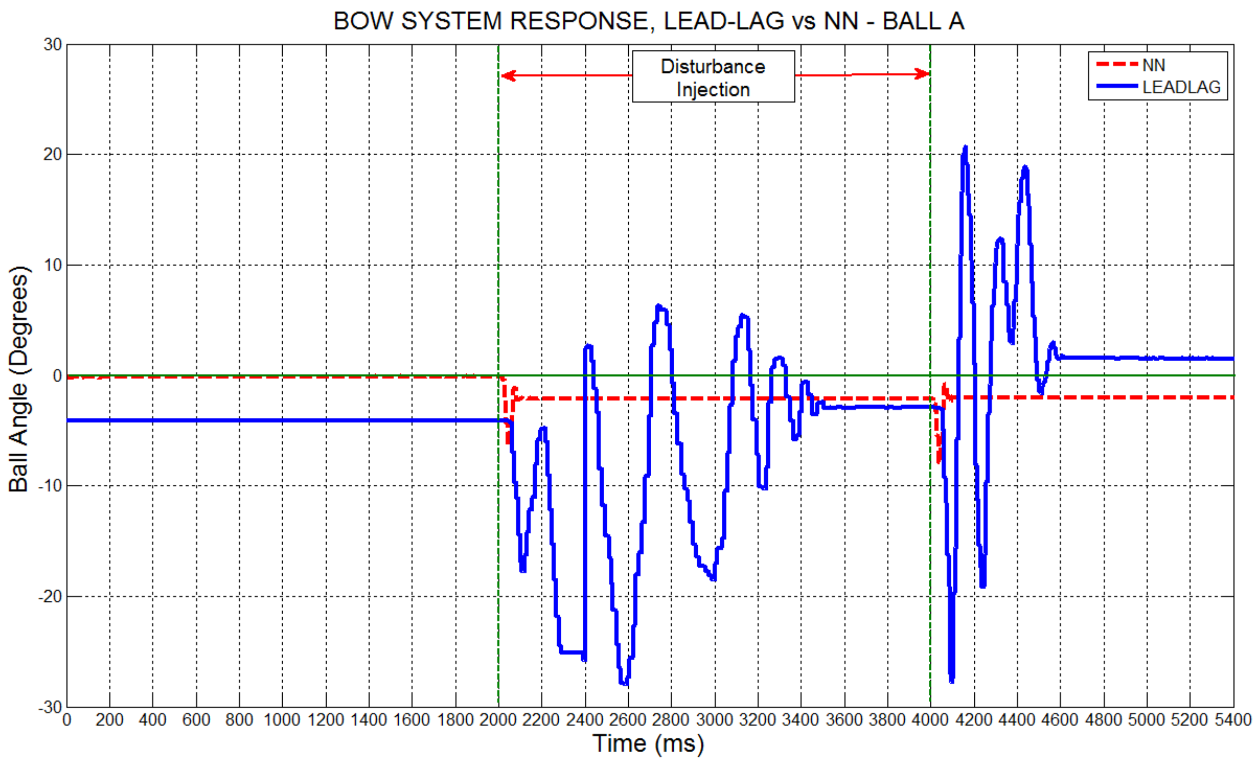


Figure 5.23: Lead-Lag compensator vs. NN-PID, Ball A

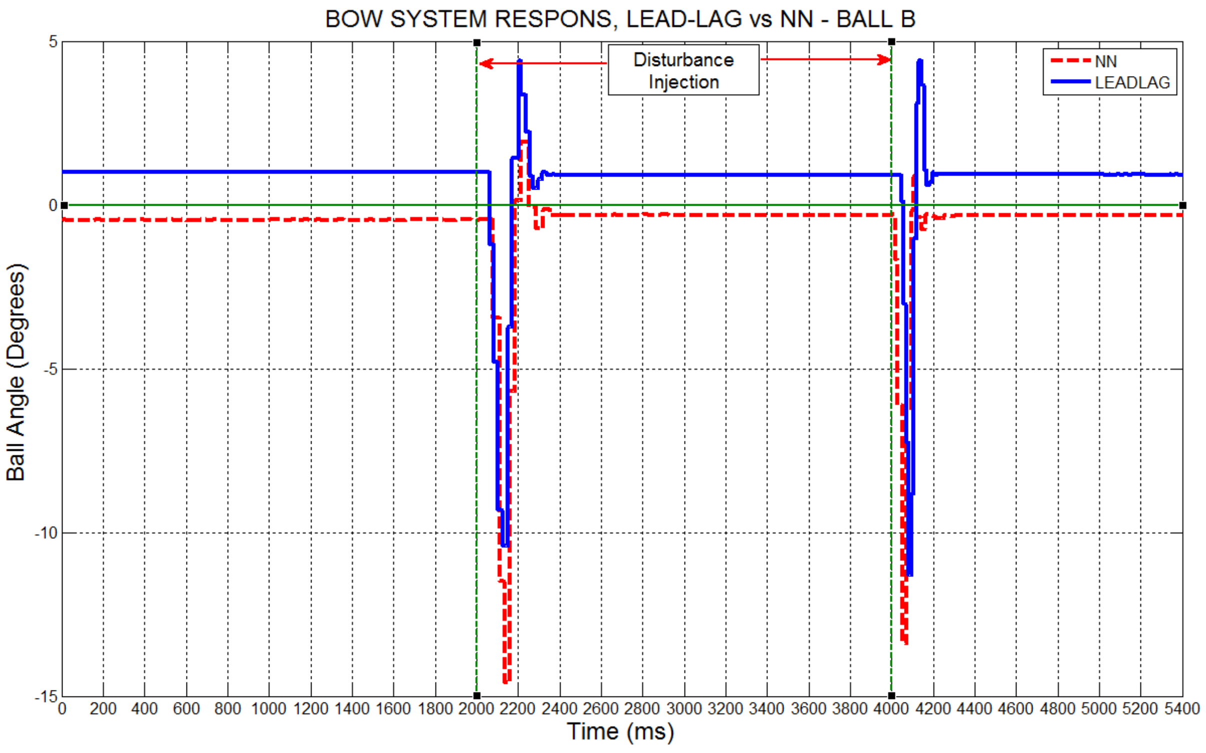


Figure 5.24: Lead-Lag compensator vs. NN-PID, Ball B

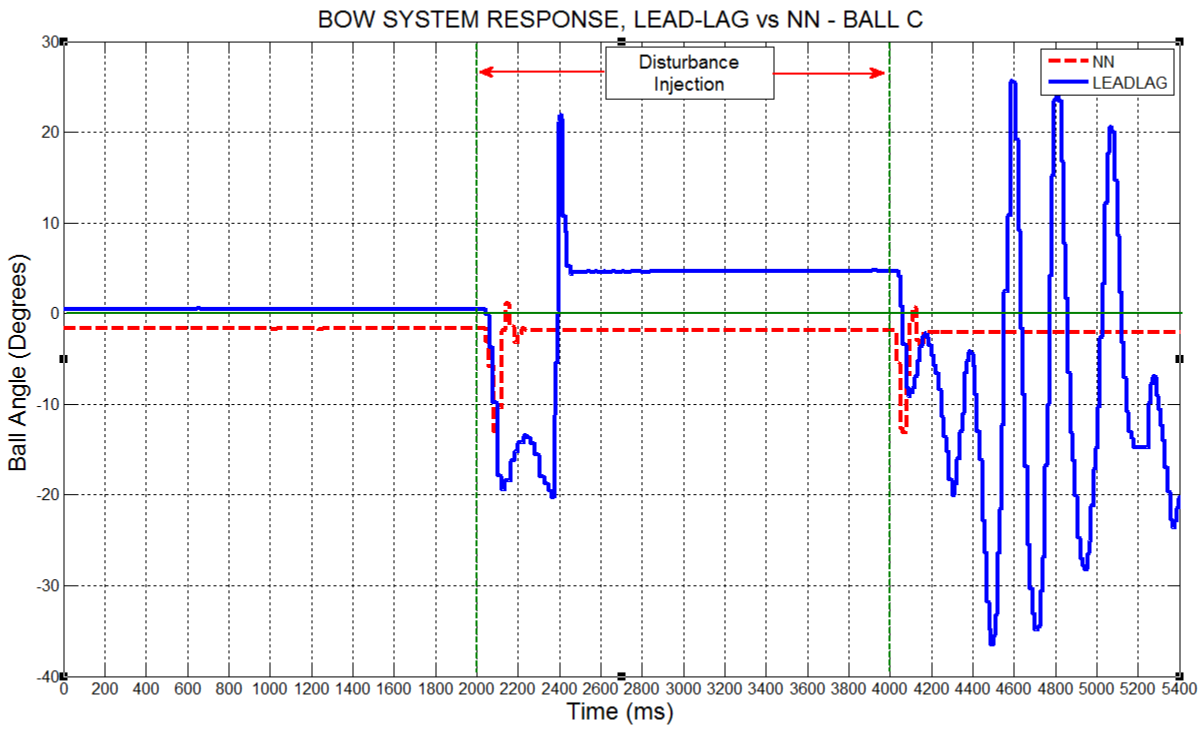


Figure 5.25: Lead-Lag compensator vs. NN-PID, Ball C

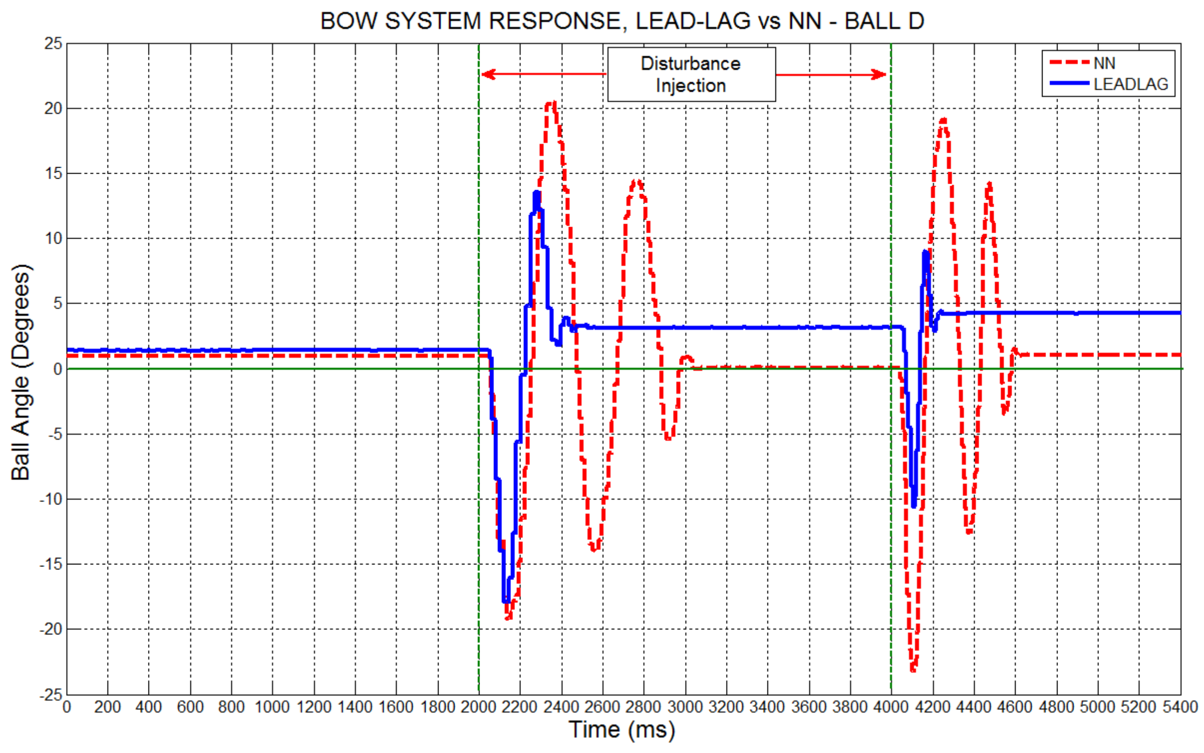


Figure 5.26: Lead-Lag compensator vs. NN-PID, Ball D

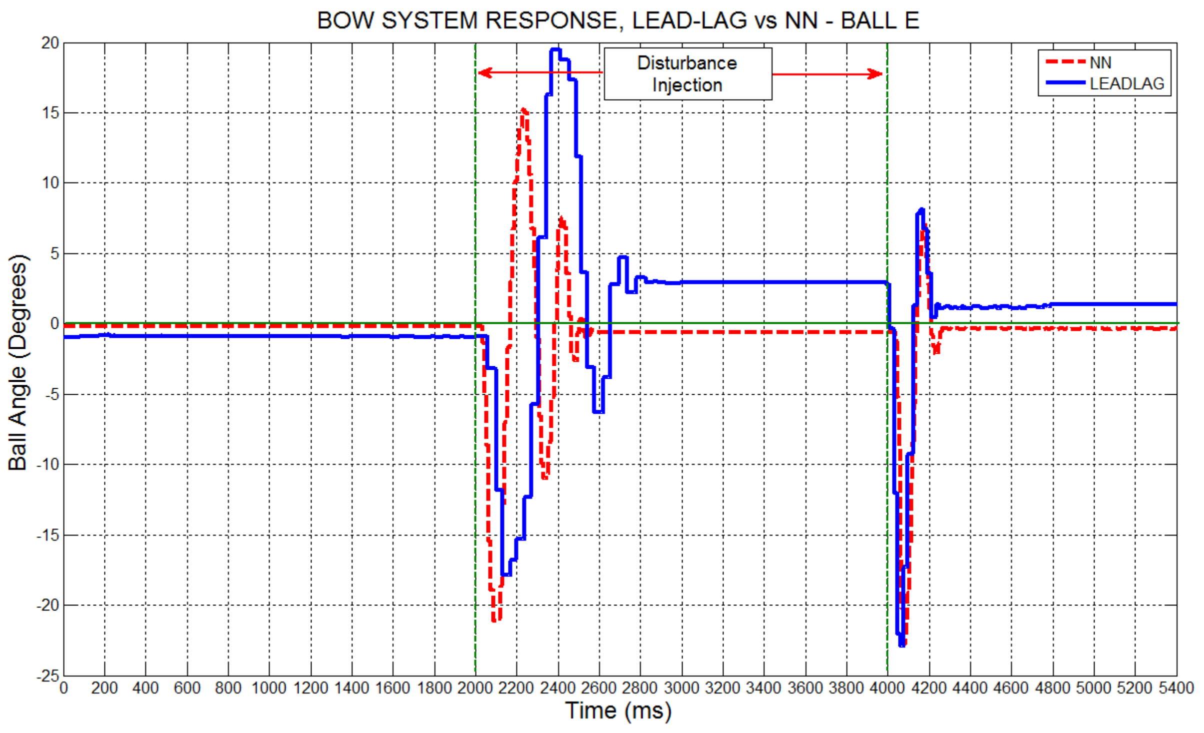


Figure 5.27: Lead-Lag compensator vs. NN-PID, Ball E

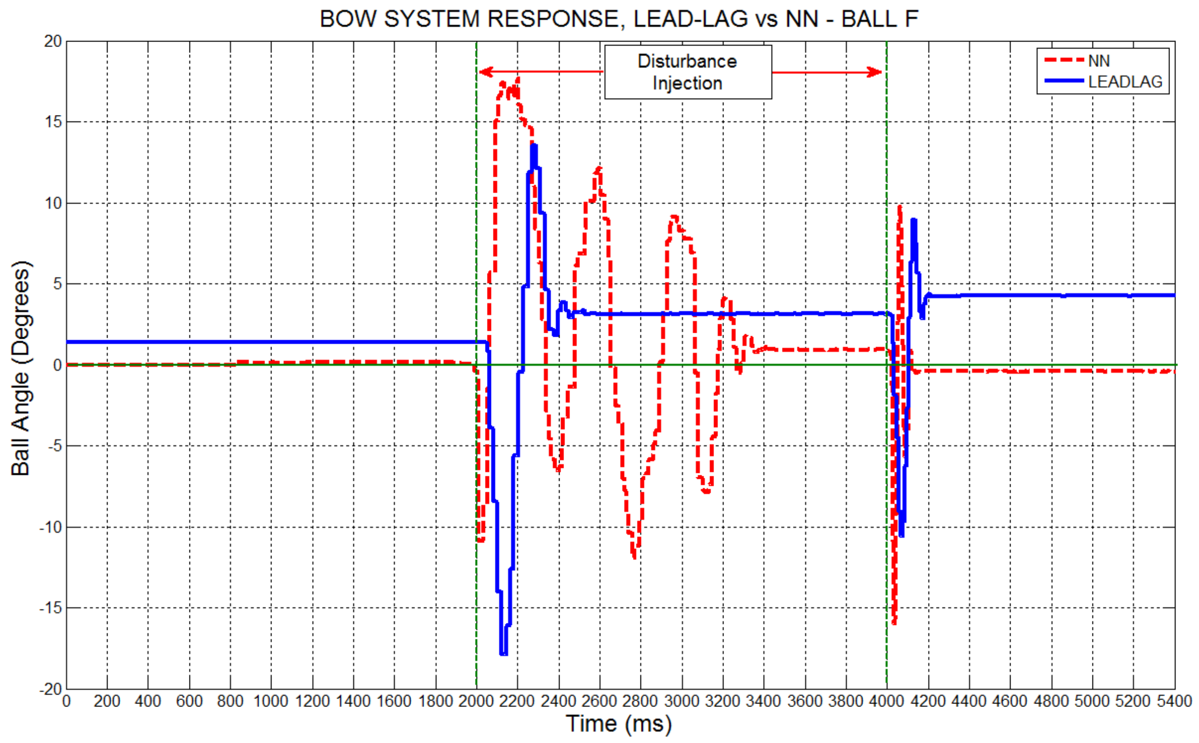


Figure 5.28: Lead-Lag compensator vs. NN-PID, Ball F

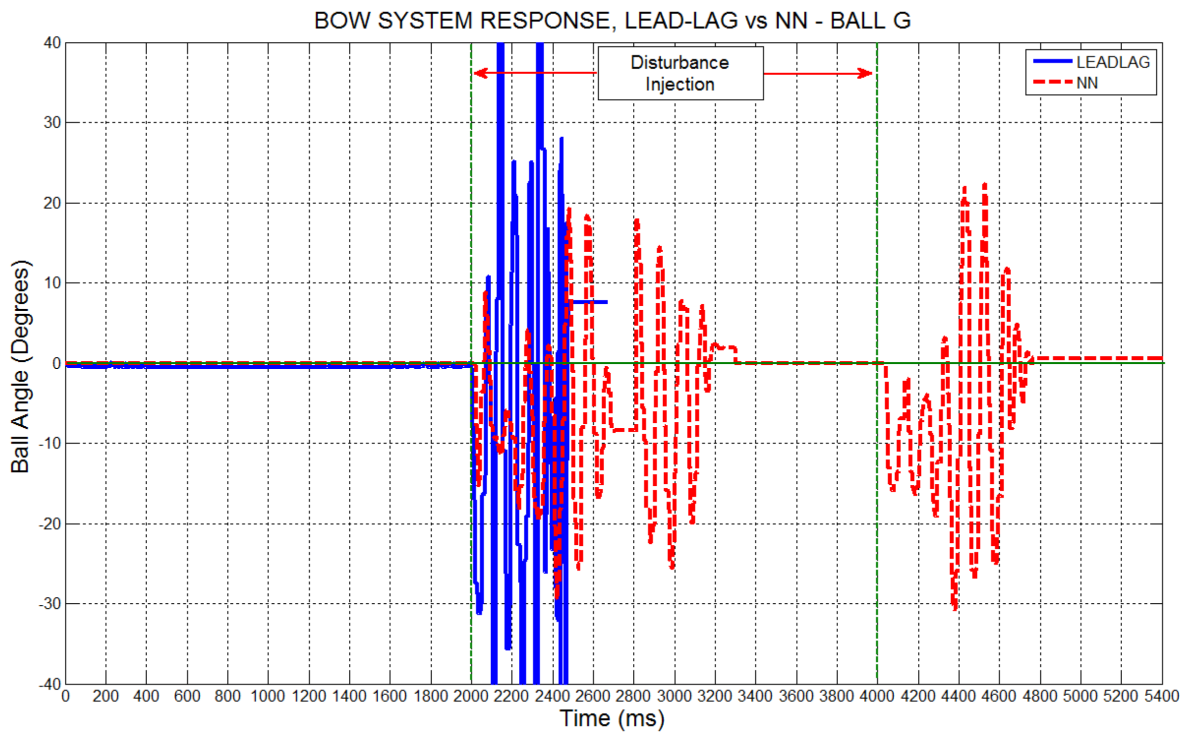


Figure 5.29: Lead-Lag compensator vs. NN-PID, Ball G

5.2.5 Analysis of system stability

From Figure 5.9 through to Figure 5.29, it can be seen that only the NN-PID controller was able to produce a stable response for each ball that was used in the experiment. This is shown clearly in Table 5-1. The score column in Table 5-1 shows the number of balls out of 7 that each controller was able to balance and is an indication of controller robustness.

Table 5-1: Controller stability

	STABLE OR UNSTABLE							Score (out of 7)
CONTROLLER	A	B	C	D	E	F	G	
PID	√	√	√	√	√	√	X	6
LEAD	√	√	X	√	√	X	√	5
LEAD-LAG	√	√	√	√	√	√	X	6
NN-PID	√	√	√	√	√	√	√	7

√	Stable
X	Unstable

The question of whether or not the system (with each controller implemented separately) meets its initial design requirements in terms of the steady state error (less than +/- 5 degrees) and the settling time (500 ms or less) as detailed in section 3.3 is addressed in Table 5-2. The score column in this table again reflects the robustness of each controller.

Table 5-2: Controller performace verification

	Does system meet design requirements for steady state error and settling time?							Score (out of 7)
	BALL							
CONTROLLER	A	B	C	D	E	F	G	
PID	*no*	*yes*	*yes*	*yes*	*yes*	*yes*	no	5
LEAD	*yes*	*no*	no	*no*	*yes*	no	*yes*	3
LEAD-LAG	*no*	*yes*	*no*	*yes*	*no*	*yes*	no	3
NN-PID	*yes*	*yes*	*yes*	*no*	*yes*	*no*	*yes*	5

The PID and NN-PID controllers show outstanding performance in their ability to remain within the design criteria while the balls are changed. Both controllers only fail to meet the system requirements with 2 of the 7 balls. The Lead and Lead-Lag compensators fail to meet the system requirements with 4 balls each.

5.2.6 Analysis of steady state error (Ess)

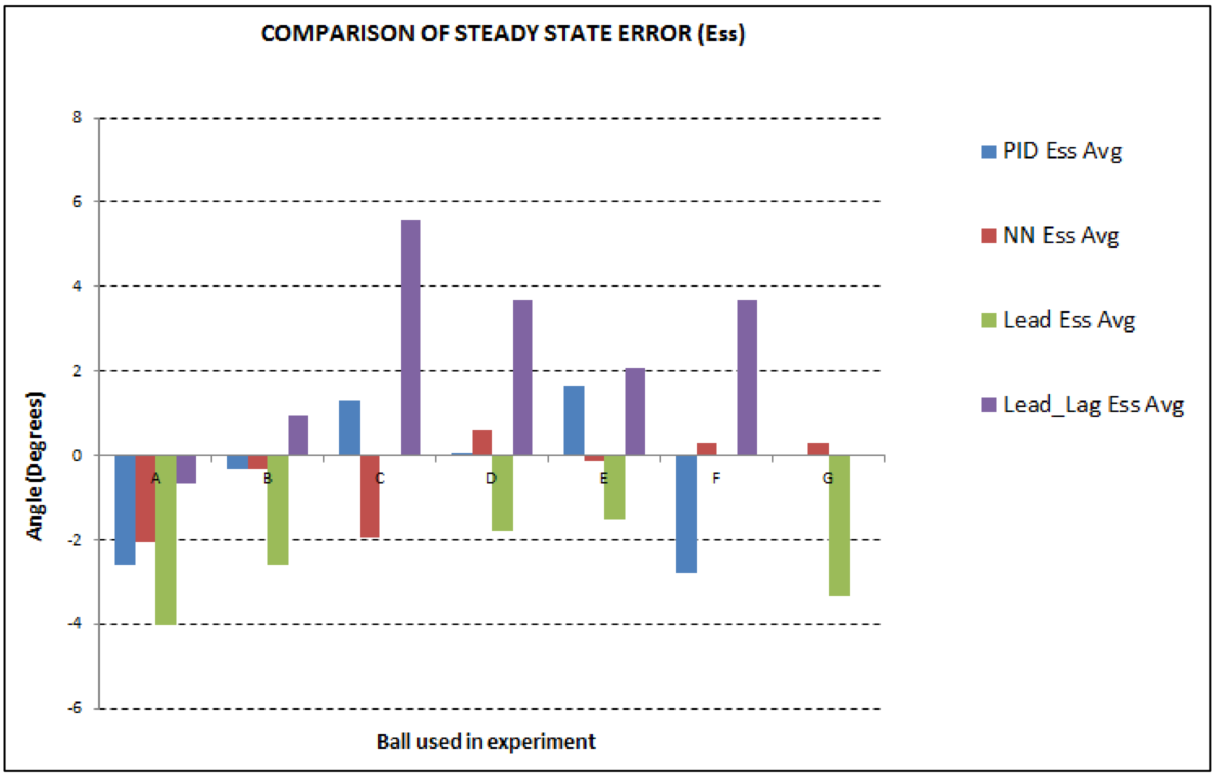


Figure 5.30: Comparison of BOW steady state error (Ess)

Based on the plots given above (from Figure 5.9 to Figure 5.29) the average steady state error is calculated for each controller and presented in Figure 5.30 to give a clear perspective of controller performance. Where no result is shown for a particular ball (i.e. no bar exists), it shows that the system was unstable. It can be seen that the NN-PID controller was the only controller able to produce a steady state error of less than +/- 5 degrees for each ball used in the experiment. This was because the traditional controllers produced an unstable result for at least one ball.

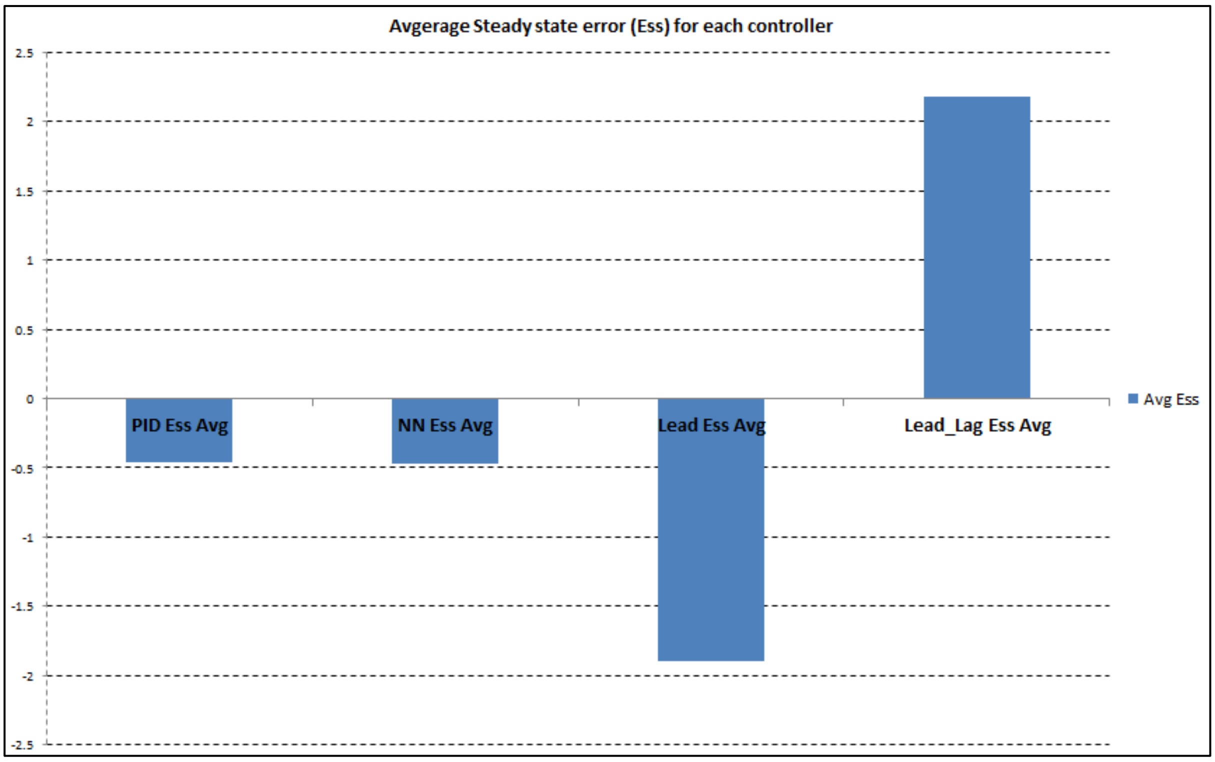


Figure 5.31: Average steady state error (Ess) for each controller

By comparing the average steady state error produced by each controller for each ball used in the experiment (as shown in Figure 5.31), it can be seen that the PID controller as well as the NN-PID controller produce an average steady state error that is almost identical (-0.463 and -0.475 degrees respectively). The Lead and Lead-Lag compensators produce a much higher average steady state error of -1.897 and 2.18 degrees respectively.

5.2.7 Settling time (Ts) comparison

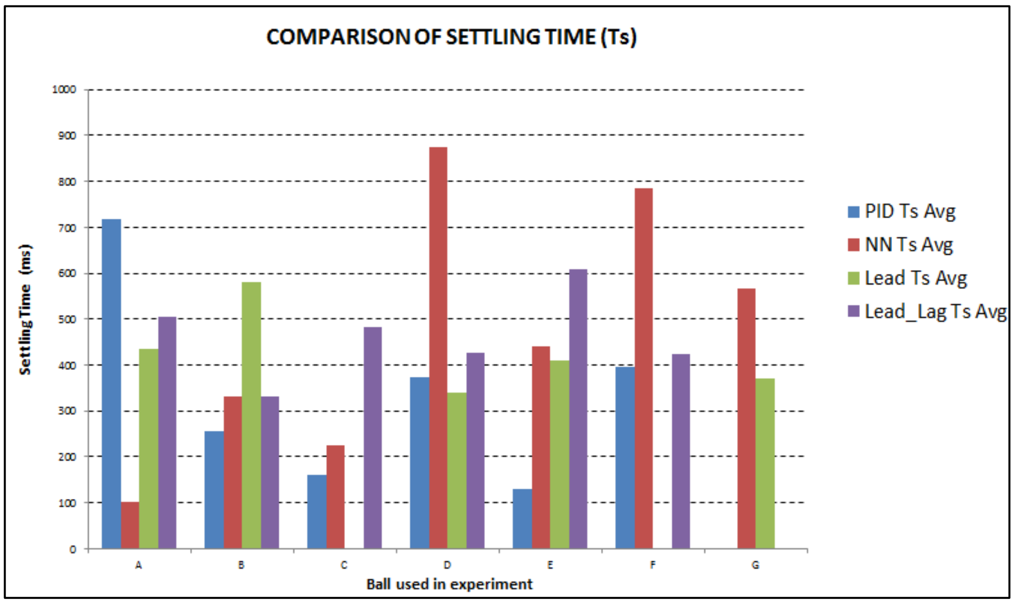


Figure 5.32: Comparison of BOW settling times (Ts)

The average controller settling times vary for each controller as the balls are changed as shown in Figure 5.32. Where no result is shown for a particular ball (i.e. no bar exists), it shows that the system was unstable in that particular instance.

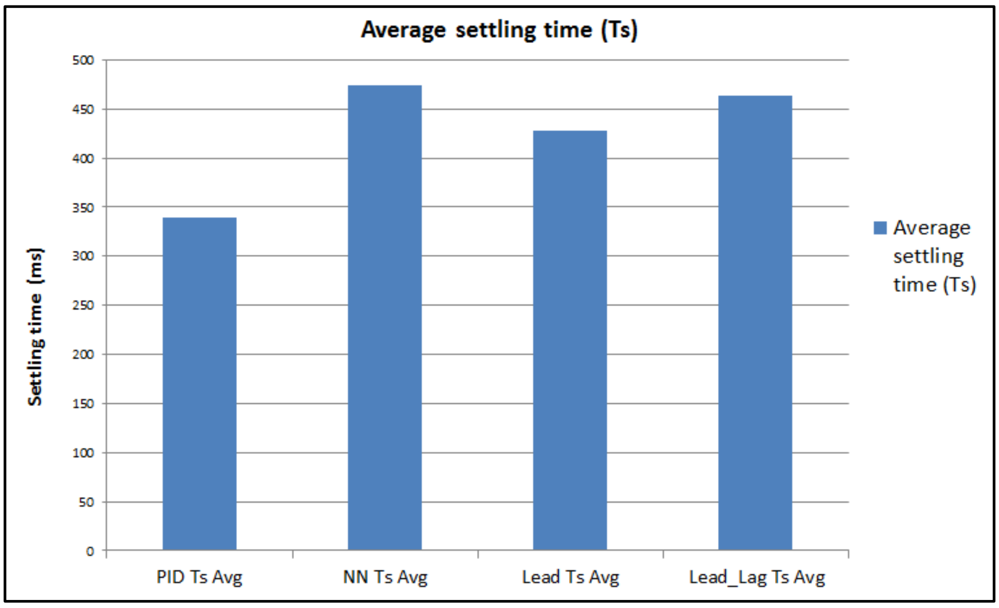


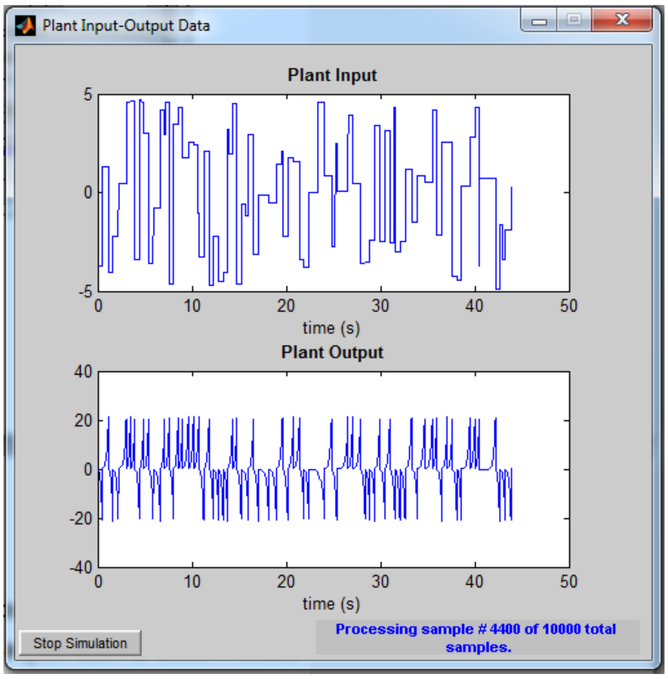
Figure 5.33: Comparison of average settling times

From Figure 5.33, it can be seen that the NN-PID controller has the highest average settling time. This however falls within the 500 ms settling time design constraint laid out in section 3.3. The traditional PID controller has the lowest average settling time of all implemented controllers.

The traditional controllers examined in this thesis were designed in Matlab/ Simulink before actual implementation on the PLC. On this note, the calculated controller gains did not yield ideal performance results on the actual system. In fact, in some instances, an unstable system was produced. This was not unexpected as aspects of the system model were based on assumptions. For example, the model assumes that the ball never slips but rolls along the surface of the wheel. The model also fails to take into consideration other influencing variables such as ball-bounce, ball contact friction and other irregularities. As such, the traditional controllers had to be optimally tuned using manual rule-of-thumb techniques. This process was often unyielding, time consuming and uncertain. The NN-PID controller, once implemented on the BOW system was able to tune itself without significant user interference. Only the initial weights, learning rate and gradient decent momentum factor had to be pre-selected. However, favourable results were achieved when the simulation factors were used in the actual implementation. Exact coding parameters used in each of the developed controllers can be found in the PLC code attached in Appendix C.

5.3 NN-Predictive controller simulation results

The NN-Predictive controller uses a NN model of the BOW plant to predict future plant performance. The controller calculates the control input that will optimize plant performance over a specified future time horizon. The NN is trained using plant input and output data obtained from the determined plant model that has Ball A's parameters of mass and radius incorporated. Once the NN model is trained, the ball parameters in the plant model are changed according to Table 4-3. The predictive controller then adjusts itself to control the plant despite the changes in plant parameters.



Processing sample # 4400 of 10000 total samples.

Figure 5.34: Generating training data for the NN

Figure 5.34 shows how plant data is generated based on the system model. Random inputs are injected into the plant and the plant output is recorded. Using this data, the NN is trained to behave exactly like the plant. Training occurs over 500 epochs using the Levenberg-Marquardt back-propagation algorithm. The NN itself is designed with 9 neurons in the hidden layer. This number of hidden layer neurons produced the most favourable results. From Figure 5.35 and Figure 5.36, it can be seen that the NN is able to mimic the BOW plant very closely with a very small error in the range of 10^{-5}.

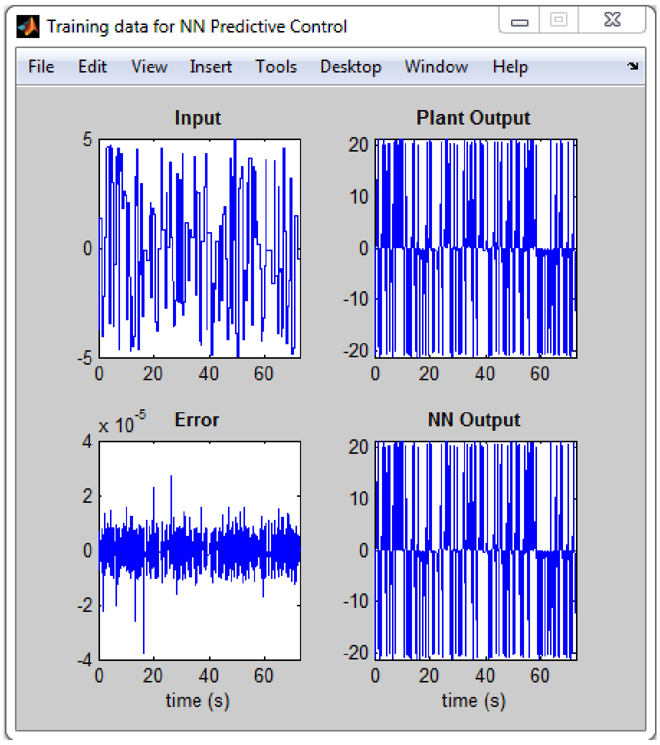


Figure 5.35: NN-Predictive controller training results

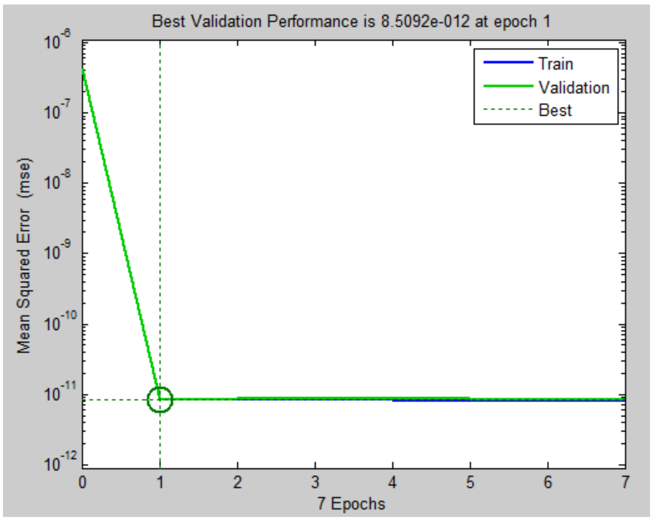


Figure 5.36: NN performance

The error gradient at the end of the training period is 1.64e-05, which for this application is satisfactory.

Figure 5.37 shows the system's responses (superimposed) with the NN controller applied as the ball is changed. The controller is able to adjust itself to accommodate and therefore balance all the balls used.

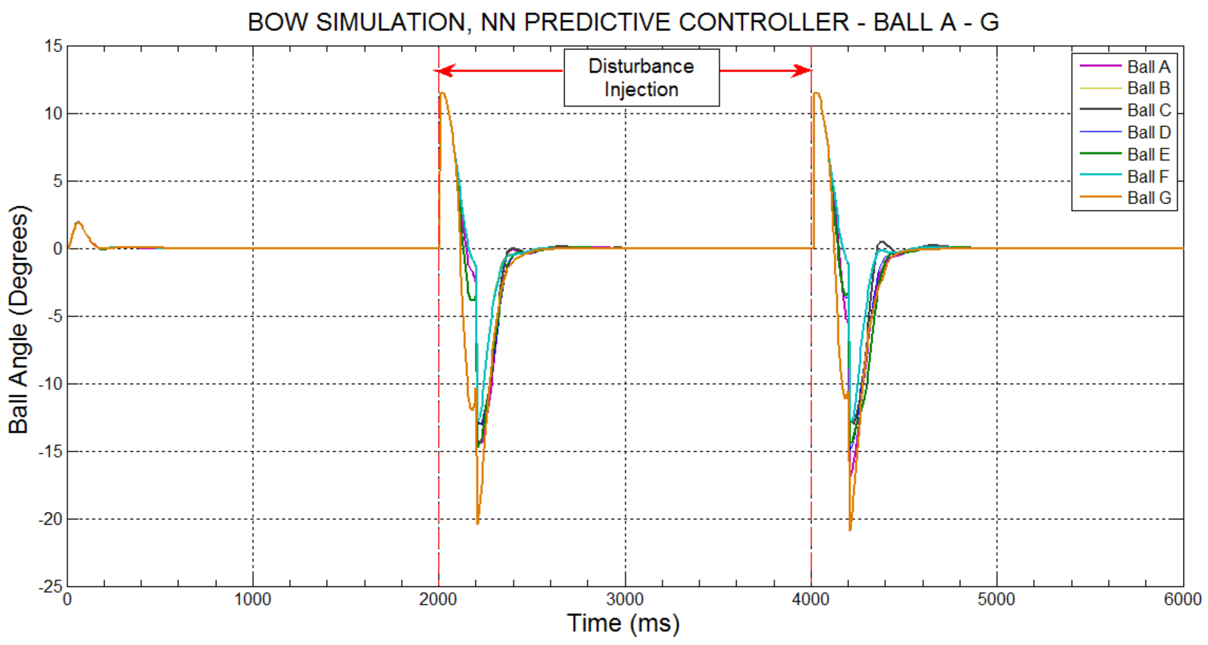


Figure 5.37: Simulation of NN-Predictive controller response

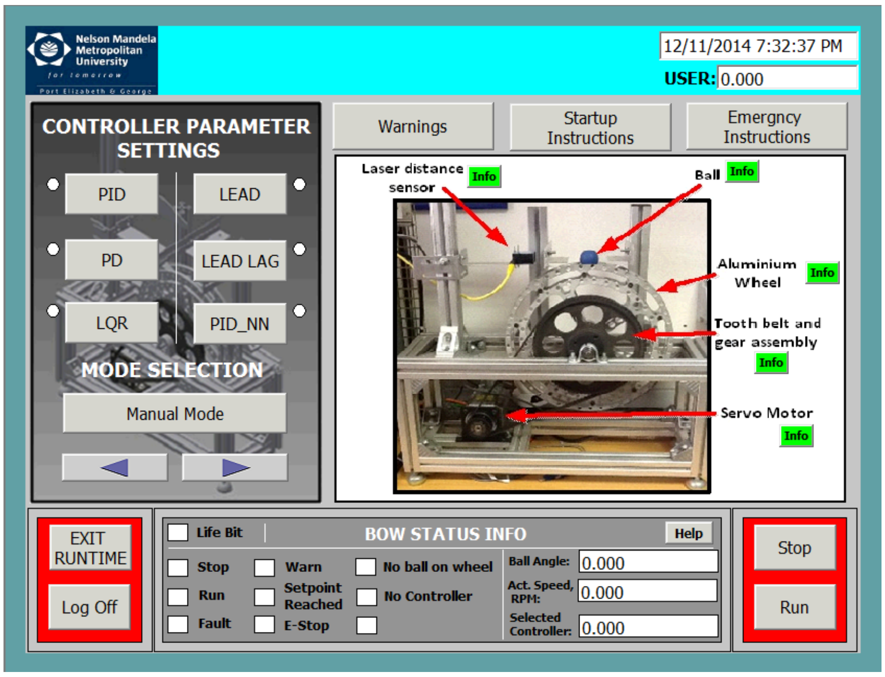
From Figure 5.37, it can also be seen that the predictive controller does not fare so well in rejecting disturbances quickly and also produces a significantly large undershoot. It is also relatively slow in response with an average settling time of about 550 ms. However, the NN controller handles changes in plant parameters very well with no signs of instability as the balls are changed. If necessary, the controller can always be re-trained to improve performance.

Although the mathematical plant model may differ from the actual plant in that it does not take into account frictional forces due to ball surface texture, ball-to-wheel contact area and the balls ability to bounce, results from Figure 5.37 show that in a real world application, the NN-Predictive controller can be used to stabilize non-linear plants without necessarily requiring the plant model. All that is needed is historical plant data that may be collected manually from the plant.

As mentioned in section 3.4.6, it was not possible to execute the NN-Predictive controller on the actual plant in real-time because of poor refresh rates between Simulink and the actual PLC via OPC. Although the controller was indeed able to respond to disturbances in the plant (i.e. the ball being moved from its equilibrium point), it was unable to carry this out quickly enough to stabilize the system. Because of further hardware and software limitations with the existing setup, it was not possible to code the NN-Predictive controller directly on to the PLC.

5.4 Remote Visualization and data capture

Wireless access to the BOW plant was successfully achieved for the purpose of monitoring, variable alteration and data analysis. This was realized firstly in the Matlab/ Simulink environment via OPC Server as disclosed in sections 4.3.2 and 4.3.5. The intention of this particular connection was to analyse plant data directly within a scientific/ mathematical environment and also to execute complex control algorithms to control the plant in real time. Although the latter intention is in fact a possibility (as shown in section 2.7.4), it was not possible for BOW system in particular which demanded a much higher refresh rate to maintain stability than the OPC Server and Client configuration could provide. In the second instance, over an independent wireless connection, controller gains and system variables were successfully monitored and/ or altered in real time using a SCADA solution from Siemens called WinCC Flexible Runtime Advanced running on a remote PC. Sample screen shots of the developed SCADA solution are shown in Figure 5.38, Figure 5.39 and Figure 5.40. The SCADA solution (see Figure 5.40 in particular) was designed with the intention that it would be used as a teaching aid in Control Systems courses.


12/11/2014 7:32:37 PM

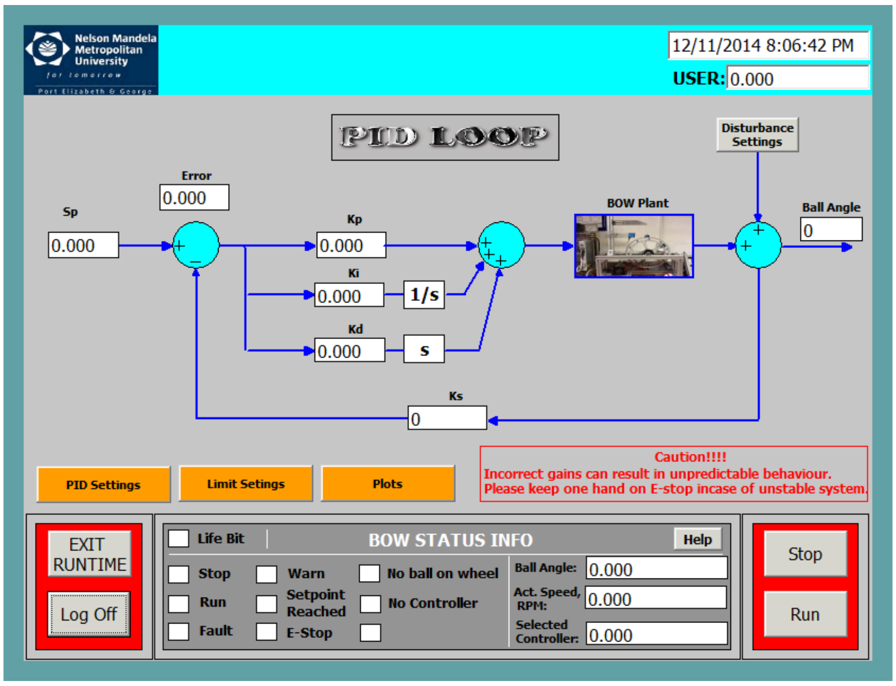
USER: 0.000

Ball Angle: 0.000

Act. Speed, RPM: 0.000

Selected Controller: 0.000

Figure 5.38: SCADA solution – Home Screen



12/11/2014 8:06:42 PM

USER: 0.000

Error: 0.000

Sp: 0.000

Kp: 0.000

Ki: 0.000

Kd: 0.000

Ball Angle: 0

Ks: 0

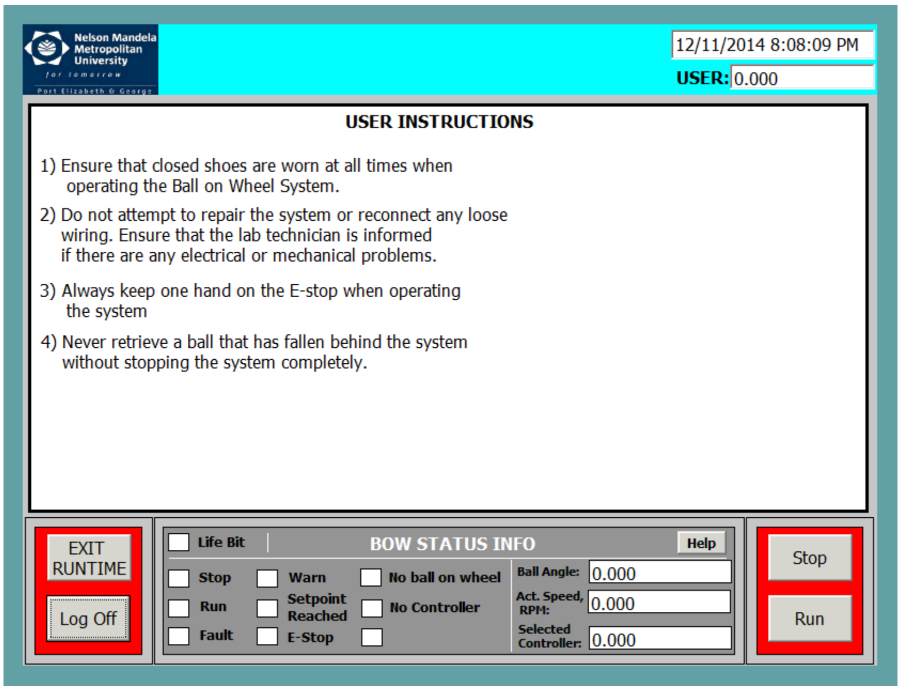
Caution!!!!
Incorrect gains can result in unpredictable behaviour.
Please keep one hand on E-stop incase of unstable system.

Ball Angle: 0.000

Act. Speed, RPM: 0.000

Selected Controller: 0.000

Figure 5.39: SCADA solution PID controller setup



12/11/2014 8:08:09 PM

USER: 0.000

USER INSTRUCTIONS

1) Ensure that closed shoes are worn at all times when operating the Ball on Wheel System.

2) Do not attempt to repair the system or reconnect any loose wiring. Ensure that the lab technician is informed if there are any electrical or mechanical problems.

3) Always keep one hand on the E-stop when operating the system

4) Never retrieve a ball that has fallen behind the system without stopping the system completely.

Ball Angle: 0.000

Act. Speed, RPM: 0.000

Selected Controller: 0.000

Figure 5.40: SCADA solution – user instructions

5.5 Summary

In this chapter, three traditional control algorithms were implemented on the BOW system in their digitized form. These included a PID controller, a Lead compensator and a Lead-Lag compensator. An online NN based PID controller was also implemented as a modern control approach. Using a variety of balls of varying weight and radii (as listed in Table 4-3) to deliberately alter the BOW plant parameters, each controller was tested in terms of its ability to continually maintain stability while remaining within the design specifications. The results from the traditional controllers were then compared in detail to the results from the NN-PID controller. Astoundingly, only the NN-PID controller had a 100% success rate in stabilizing the system every time parameters were changed. The traditional PID controller was also shown to be quite a robust controller with a stabilizing success rate of 86%. A review of whether or not the controllers met the design requirements in each experiment revealed that the traditional PID and NN-PID controllers succeeded in meeting the design requirements with a 71% success rate while the remaining traditional controllers had a much poorer performance with only a 43% success rate. The NN-PID controller maintains all the robust features of the traditional PID controller but comes with the added benefit of ‘learning’ that allows it to continually adapt to its environment.

The possibility of wireless remote control of the BOW system was also investigated. The results showed that for control algorithm execution from a remote location, a much higher refresh rate is required, particularly for the highly responsive BOW system. However, as far as data access, system monitoring and variable manipulation were concerned, the wireless connection worked perfectly.

Chapter 6.
CONCLUSION AND RECOMMENDATIONS

6.1 Conclusion

The main aim of this research was to implement and then compare the performance of traditional control strategies with modern NN based control strategies on a strongly non-linear system. The control strategies were to be implemented on a standard, medium specification industrial PLC with the option of wireless monitoring and control from a remote PC.

In order to carry out a meaningful investigation with the goal of finding a viable industrial solution, a non-linear mechatronic BOW balancing system was designed and constructed using typical industrial equipment including a Siemens S7-300 PLC as its computational platform. The designed system was then mathematically modelled and linearized. In procession, various traditional control strategies including a PID, Lead and Lead-Lag controller were designed, simulated and then implemented in real-time. A NN-Predictive controller and a NN-PID controller were also designed and simulated on the determined plant model, however only the NN-PID controller was implemented in real-time.

Using a variety of balls of varying material, weight, size and surface texture (as listed in Table 4-3) to deliberately alter the BOW plant parameters and also create non-linear disturbances, each implemented controller's performance was evaluated in terms of its ability to withstand the imposed variations and still function within the design criteria. The results from the traditional controllers were then compared in detail to the results from the NN-PID controller.

The results gathered from this research have shown that although the implemented traditional controllers were able to remain functional within a certain limited range of parameter variation, they eventually failed when pushed outside the linear region for which they were initially tuned. The results from this research indicate that NN based control schemes may be utilized to overcome many of the limitations found with traditional controllers since they are able to control non-linear processes that experience broad parametric variation over time and can also overcome process uncertainties. This is possible without the involvement of cumbersome mathematical modelling, linearization or controller design. Although the computational overhead required for the execution of certain NN architectures (such as the NN-Predictive controller) is higher than what standard PLCs can offer, simpler, less demanding NN structures do exist. Even with a limited number of neurons, NN's are still able to solve non-linear problems with remarkable accuracy. This was seen first-hand with the online-learning NN-PID controller that was investigated in this research. This controller was able to adequately control the BOW system in real-time (and without any prior knowledge of the plant) on every account of parameter variation subjected to it. In this way, it was shown to be superior to its conventional counterparts. It maintained all the robust features of the standard PID controller but had the benefit of being able to continually adapt itself to a changing environment through a process of learning.

Apart from showing that NN control may be applied easily on standard PLCs, the research also showed that all PLC data areas could be accessed for purposes of analysis or even manipulated over a wireless connection sustained by OPC Server. Complex control algorithms could, in essence, be implemented to control dynamical systems from a remote location provided that the controller's sampling rate was high enough to match the dynamic systems response. However, this was not the case with the BOW system which proved to be too over-responsive for the described OPC connection.

6.2 Recommendations

The following recommendations can be made in order to broaden the angle of research brought forward in this thesis:

- A sensor that measures ball radius could be attached to the system and used as an additional input to a NN controller. This would improve the generalization ability of the network and effectively improve the control performance.
- A much faster interface between the control computer and the PLC would solve the problem of poor refresh rates and allow multiple other modern control strategies to be executed on the BOW system directly from Matlab/

Simulink in real time. A directly wired network connection (e.g. Profibus or LAN), as opposed to wireless access could be used as a means of speeding up data exchange.

- For less responsive processes, for example temperature control or fluid level control, the current OPC server connection speed would suffice. The research could thus be applied or taken further on such processes.
- The BOW system is an entertaining system to work with and could therefore be used as an educational aid in Control systems classes in order to demonstrate various principles of traditional and modern control.

REFERENCES

1. Thompson S. Control Systems Engineering & Design. New York, USA: Longman Scientific & Technical; 1989.

2. Dorf RC, Bishop HR. Modern Control Systems. USA: Pearson Education International; 2005.

3. Richards RJ. An Introduction to Dynamics and Control. New York, United Kingdom: Longman Inc.; 1979.

4. Cho G, Kim P. A Precise Control of AC Servo Motor using Neural Network PID. Current Science. 2005 July;89:23-9.

5. Nikolaou M, Misra P. Linear and Nonlinear Processes: Recent Developments and Future Direction. 2001.

6. Ren XM, Rad AB. Adaptive non-linear compensation control based on neural networks for non-linear systems with time delay. International Journal of Systems Science. 2009 December;40(12):1283-92.

7. Franklin GF, Powel JD, Emami-Naeini A. Feedback Control of Dynamic Systems. Little C, editor. New Jersey, USA: Pearson Prentice Hall; 2006.

8. Jack H. Dynamic System Modelling. USA2001.

9. Wellstead PE. Introduction to Physical System Modelling. London, Great Britain: Academic Press LTD; 2000.

10. Nikolaou M, Hanagandi V. Control of nonlinear dynamical systems modeled by recurrent neural networks. AIChE journal. 1993;39(11):1890-4.

11. Piche R. Lagrangian DAE Modelling Examples. Tampre1997.

12. Han W-y, Han J-w, Lee C-g, editors. Development of a Self-tuning PID Controller based on neural network for nonlinear Systems. Proceeding of the 7th Mediterranean Conference on Control and Automation (MED99), Haifa, Israel; 1999.

13. Shahraki F, Fanaei M, Arjomandzadeh A. Adaptive system control with PID neural networks. Chemical Engineering transaction. 2009;17:1395-401.

14. Deng SY. Nonlinear & Linear MIMO Control of an Industrial Mixing Process [Thesis]. 2002 [updated December].

15. Chen B-s, Wong C-c. Robust linear controller design: Time domain approach. IEEE transactions on automatic control. 1987;32(2):161-4.

16. Sultan K, Mirza A. Inverted Pendulum, Analysis, Design and Implementation. IIEE Visionaries. 2003.

17. Burns RS. Advanced Control Engineering. Plymouth, UK: Butterworth-Heinemann; 2001.

18. Hespanha JP. Undergraduate Lecture Notes on LQG/ LQR Controller Design. Apuntes para Licenciatura UCBS (University of California, Santa Barbara). 2007 April:1-37.

19. Zhao B, Hu H. A new inverse controller for servo-system based on neural network model reference adaptive control COMPEL. 2009;28(6):1503-15.

20. Vlachogiannis JG, Roy RK. Robust PID controllers by Taguchi's method. The TQM Magazine. 2005;17(5):456-66.

21. Tan K, Huang S, Ferdous R. Robust self-tuning PID controller for nonlinear systems. Journal of process control. 2002;12(7):753-61.

22. Rahmat MF, Wahid H, Abdul-Wahab N. Application of Intelligent Controller in Ball and Beam Controller. Smart Sensing and Intelligent Systems. 2010 March 3.

23. Olver JP. Nonlinear Systems. University of Minnesota - College of Science and Engineering2012.

24. Yurkevich VD, editor PI and PID controller design for nonlinear systems in the presence of a time delay via singular perturbation technique. Electronic Instrument Engineering, 2008 APEIE 2008 9th International Conference on Actual Problems of; 2008: IEEE.

25. Szczudlak J, Fasheh F. Pendulum Project. Project Report. 2012.

26. Saha A, Chatterjee R, Dutta U, Chen S, editors. Motion control of snake robot by lead-lag compensator designed with frequency domain approach. WSEAS International Conference Proceedings Mathematics and Computers in Science and Engineering; 2009: World Scientific and Engineering Academy and Society.

27. Balaševičius L, Januševičius V, Zakaraitė S. State Controller Design in Programmable Logic Controllers. Electronics and Electrical Engineering–Kaunas: Technologija. 2008 (1):81.

28. Mohamed RB, Nasr HB, M'Sahli F. A multimodel approach for a nonlinear system based on neural network validity. Intelligent Computing and Cybernetics. 2011 February;4(3):331-52.

29. Liao F, Lum KY, Wang JL, Benosman M. Adaptive control allocation for non-linear systems with internal dynamics. IET Control Theory Application. 2009 February 4(6):909-22.

30. Levin UA, Narendra SK. Control of Nonlinear Dynamical Systems Using Neural Networks: Controllability and Stabilization. Transactions on Neural Networks. 1993 March;4(2):192-206.

31. Hagan MT, Demuth BH. Neural Neteorks for Control 2002.

32. Abraham A. Artificial Neural Networks. Sydenham PH, & Thorn, R., editor. Stillwater, Oklahoma, USA: John Wiley & Sons; 2005.

33. Garces F, Becerra VM, Kambhampati C, Warwick K. Strategies for Feedback Linearization - A Dynamic Neural Network Approach. Grimble M, Johnson M, editors. London, United Kingdom: Springer; 2003.

34. Jain P, Nigam M. Design of a Model Reference Adaptive Controller Using Modified MIT Rule for a Second Order System. Advance in Electronic and Electric Engineering. 2013;3(4):477-84.

35. Straussberger F, Schwab M, Huber M, Baumann C, Fink M, Michaels A, et al., editors. A model reference adaptive control strategy for a self-balancing chair. Methods and Models in Automation and Robotics (MMAR), 2013 18th International Conference on; 2013: IEEE.

36. Demuth H, Beale M. Neural Network Toolbox. 2004.

37. Trajanoski Z, Wach P. Neural predictive controller for insulin delivery using the subcutaneous route. Biomedical Engineering, IEEE Transactions on. 1998;45(9):1122-34.

38. Jin-quan H, Lewis FL. Neural-network predictive control for nonlinear dynamic systems with time-delay. Neural Networks, IEEE Transactions on. 2003;14(2):377-89.

39. Lee M, Park S. Process Control Using a Neural Network Combined with the Conventional PID Controller. The Institute of Control, Automation and Systems Engineers. 2000 September 3;2(3):196-200.

40. Hsu C, Chiu C, Tsai J. Auto-tuning PID Controller Design using a Sliding-mode approach for DC Servomotors International Journal of Intelligent Computing and Cybernetics. 2010 November;4(1):93-110.

41. Abood AA, Tuaimah FM, Maktoof AH. Modeling of SVC Controller based on Adaptive PID Controller using Neural Networks. International Journal of Computer Applications. 2012;59(6):9-16.

42. Yongquan Y, Ying H, Bi Z, editors. A PID neural network controller. Neural Networks, 2003 Proceedings of the International Joint Conference; 2003: IEEE.

43. International H. Model Predictive Controller. Honeywell OnRamp2014.

44. Valencia-Palomo G, Rossiter J. Novel Programmable Logic Controller implementation of a predictive controller based on Laguerre Functions and multiparametric solutions. Control Theory & Applications, IET. 2012 July;6(8):1003-14.

45. Jack H. Automating Manufacturing Systems with PLCs [E-book]. 2007 [updated May].

46. Abdi H, Salami A, Ahmadi A, editors. Implementation of a New Neural Network Function Block to PLC Library Function. World Academy of Science, Engineering and; 2007.

47. El Monsef WA, Areed F, editors. Design of Neural-PLC Controller For Industrial Plant. MLMTA; 2006.

48. Topalova I. Modular Multi-Applicable Neural Network Classification Structure with PLC Industrial ImplementationUnknown. Available from: https://www.iconceptpress.com/download/paper.

49. Persin S, Tovornik B, Muskinja N. OPC-driven data exchange between Matlab and PLC - controlled system. International Journal of Engineering Education. 2003;19(4):586-92.

50. Martins J, Lima C. A Matlab/ Simulink Framework for PLC controlled Processes. 2010.

51. Boyer SA. SCADA: supervisory control and data acquisition: International Society of Automation; 2009.

52. Bai X, Meng X, Du Z, Gong M, Hu Z, editors. Design of Wireless Sensor Network in SCADA system for wind power plant. Automation and Logistics, 2008 ICAL 2008 IEEE International Conference on; 2008: IEEE.

53. Bayindir R, Cetinceviz Y. A water pumping control system with a programmable logic controller (PLC) and industrial wireless modules for industrial plants—An experimental setup. ISA transactions. 2011;50(2):321-8.

54. Ogata K. Discrete-time control systems: Prentice Hall Englewood Cliffs, NJ; 1995.

55. Van Dessel M. Control of an industrial process using PID control blocks in automation controller. Tagungsband AALE 2012. 2012:137-45.

56. Ho M-T, Tu Y-W, Lin H-S. Controlling a ball and wheel system using full-state-feedback linearization [Focus on Education]. Control Systems, IEEE. 2009;29(5):93-101.

57. Awtar S, Bernard C, Boklund N, Master A, Ueda D, Craig K. Mechatronic Design of a ball-on-Plate Blancing Syatem. Mechatronics. 2002;12(2):217-28.

58. Colwell C, H. Physics Lab 1997 [updated Feb]. Available from: http://dev.physicslab.org/Document.aspx?doctype=3&filename=RotaryMotion_RotationalDynamicsRollingSpheres.xml.

59. Rojas R. The backpropagation algorithm. Neural Networks: Springer; 1996. p. 149-82.

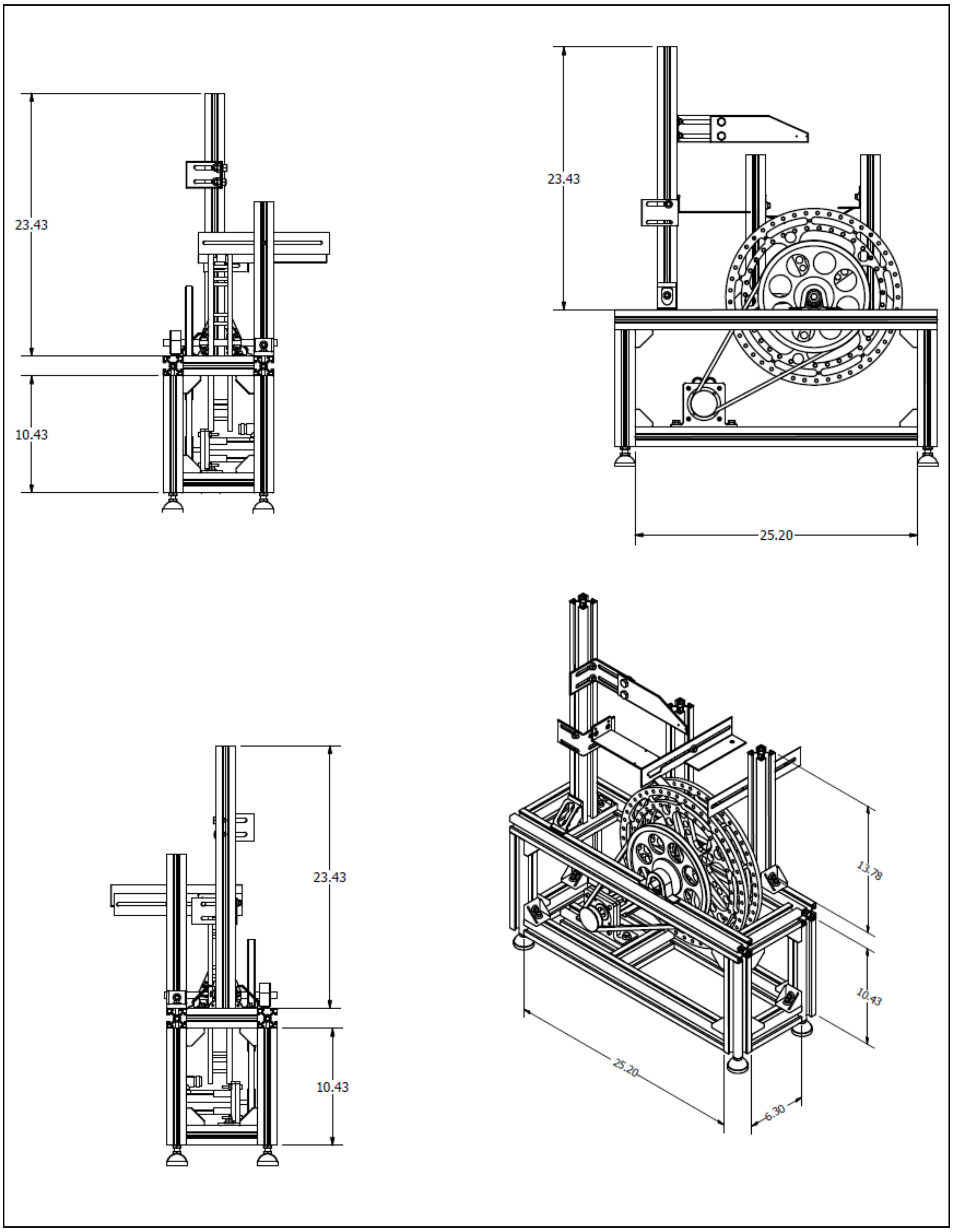
60. Siemens. Simovert Masterdrives MC. 2004.

61. Siemens. Simovert Masterdrives Motion Control Compendium. 2006.

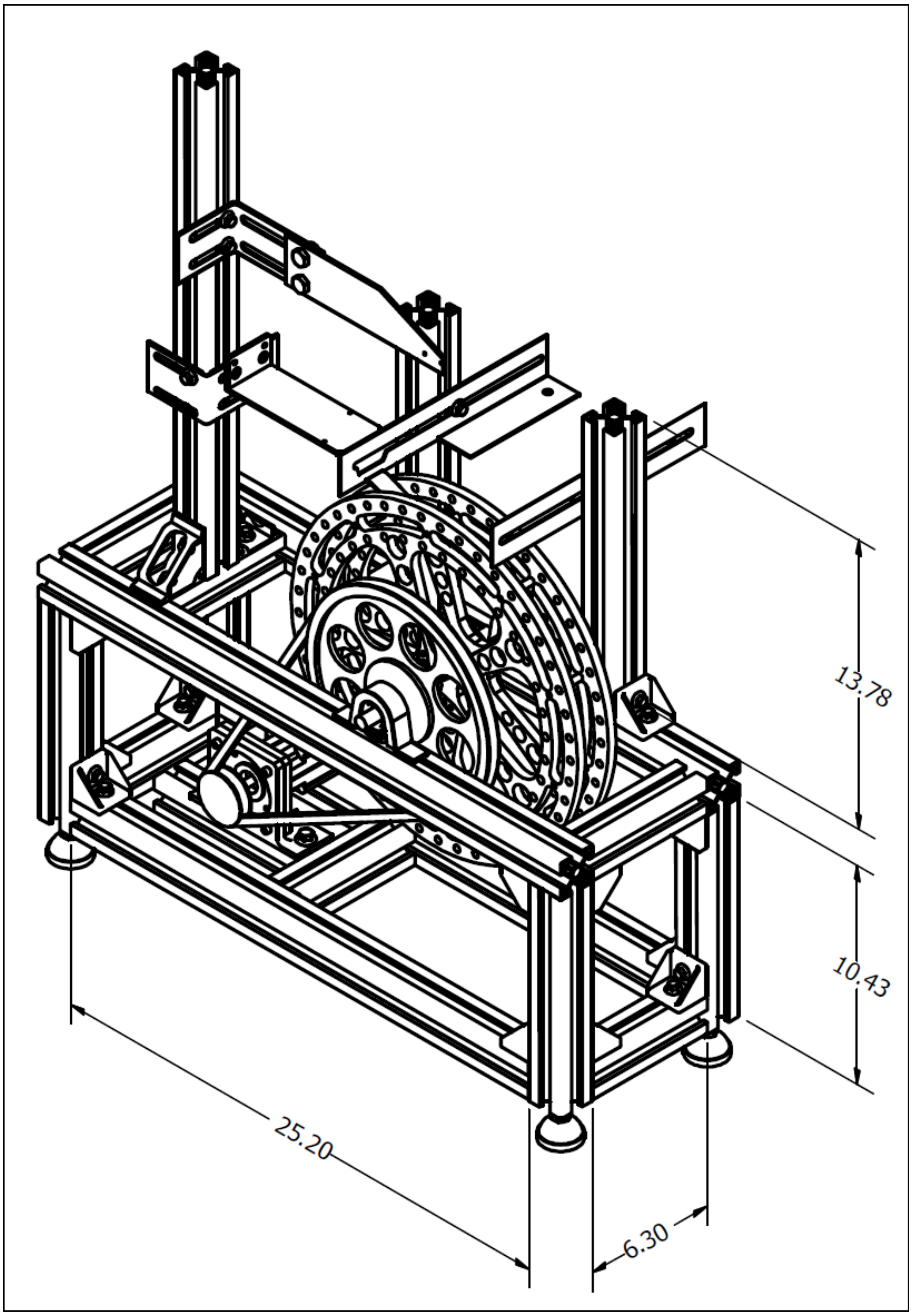
62. Siemens A. S7-300 Programmable Controller Hardware and Installation. 2013.

APPENDIX A: MECHANICAL DESIGN

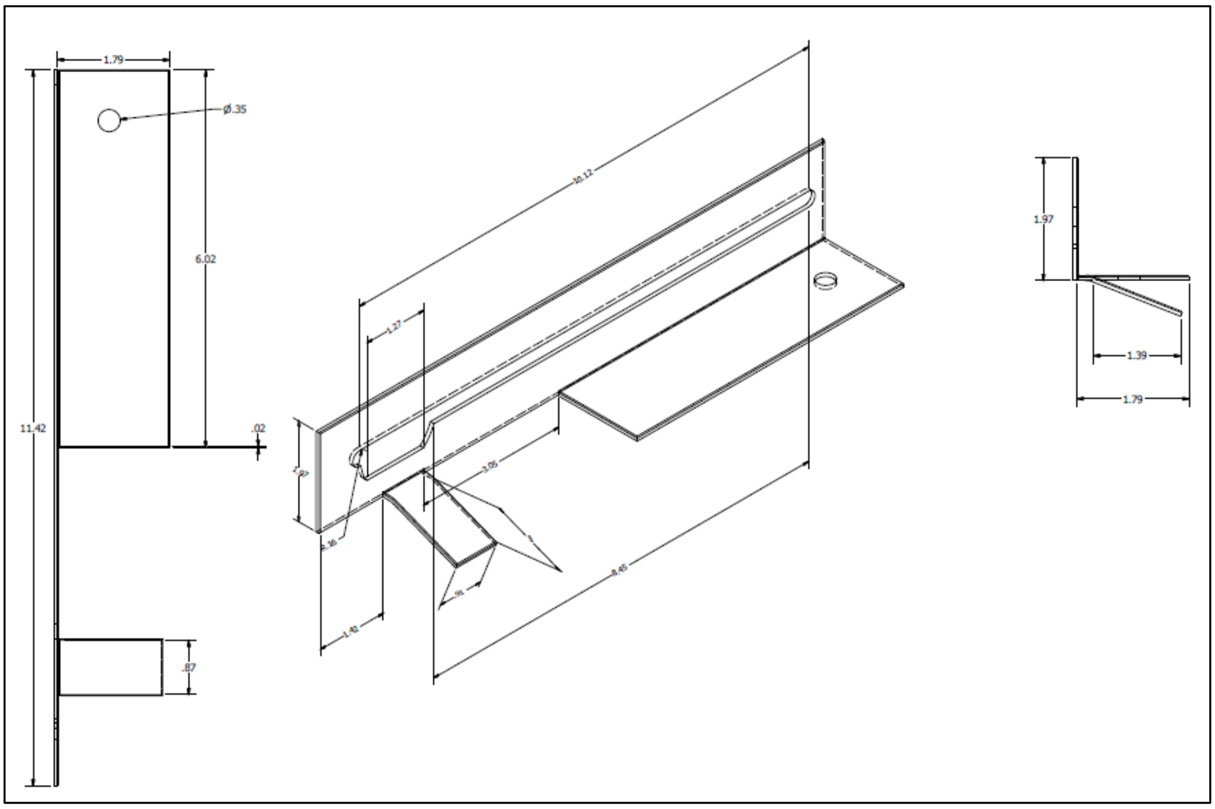
Ball on Wheel balancing system overview



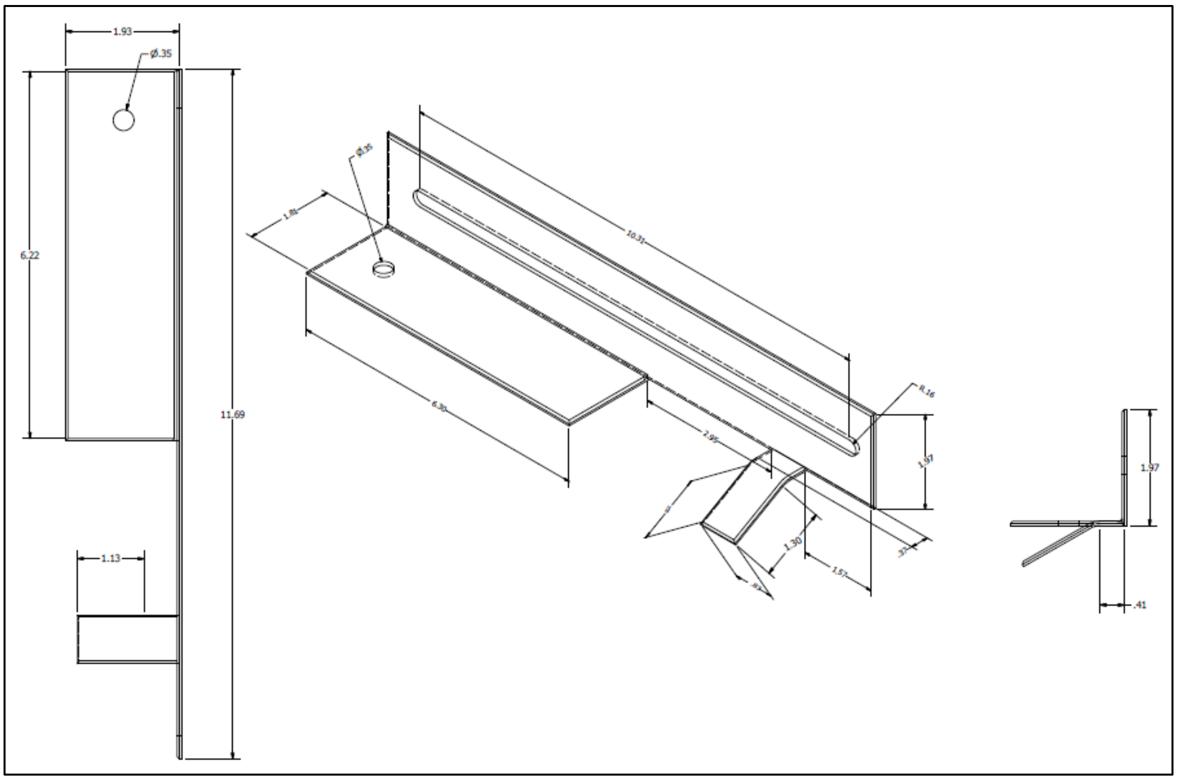
Assembled view of Ball on Wheel balancing system



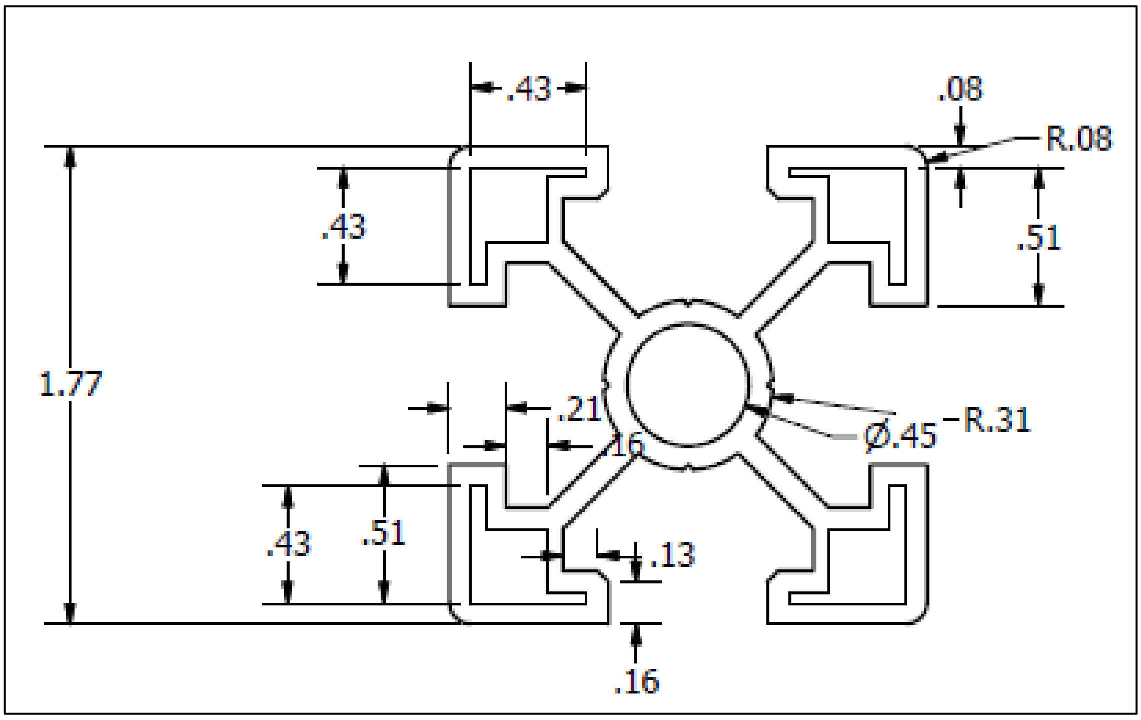
Left ball limit



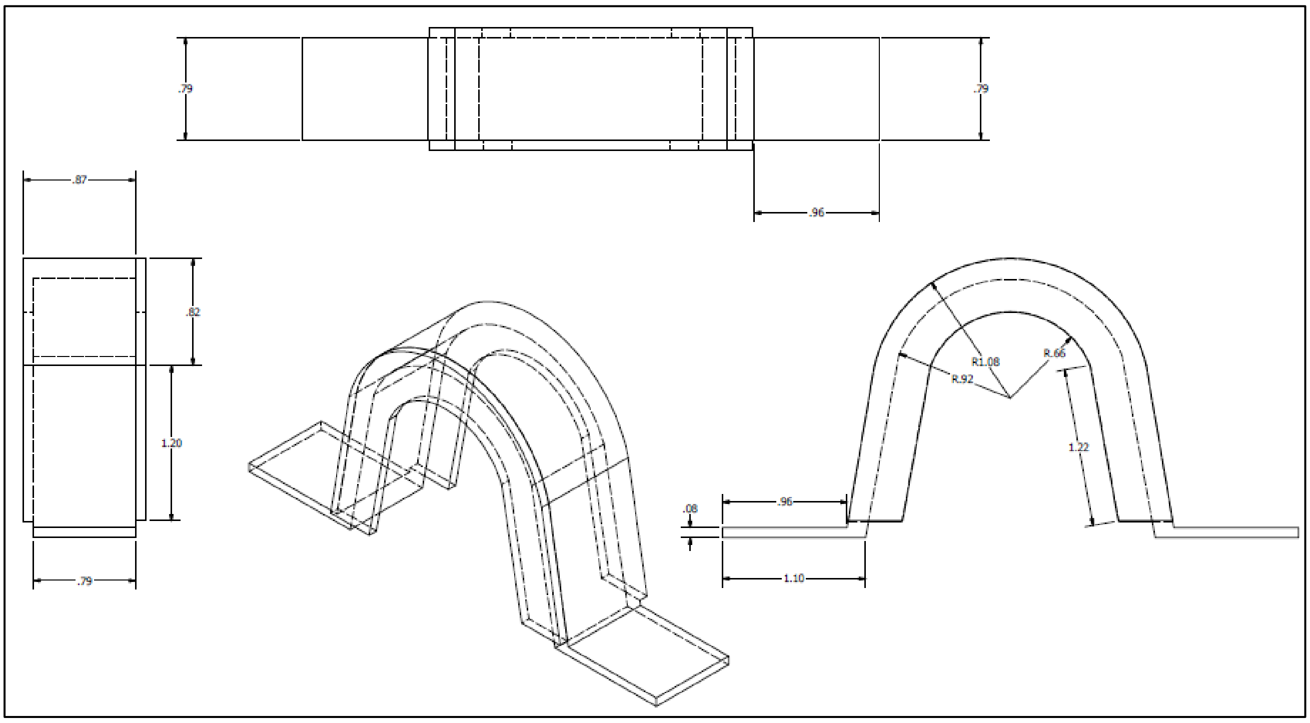
Right ball limit



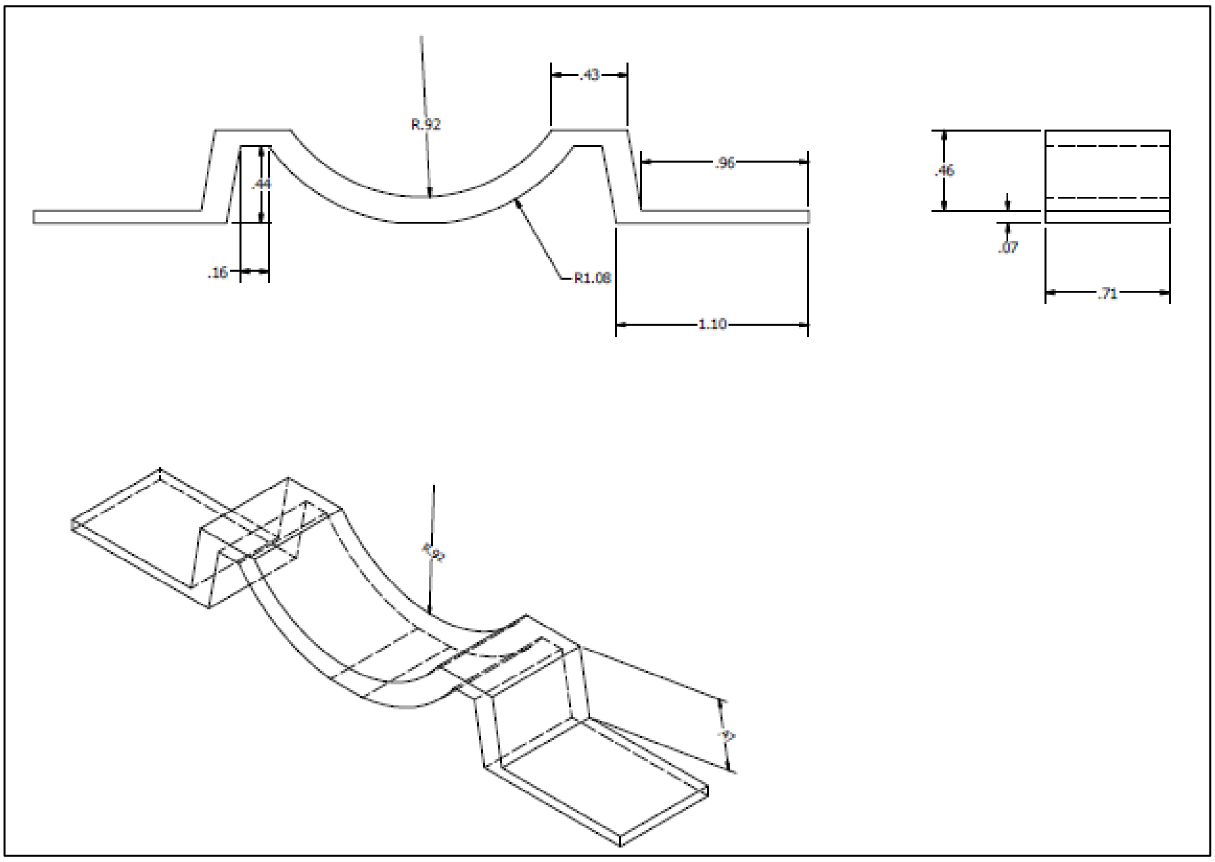
Aluminium extrusion cross-sectional view



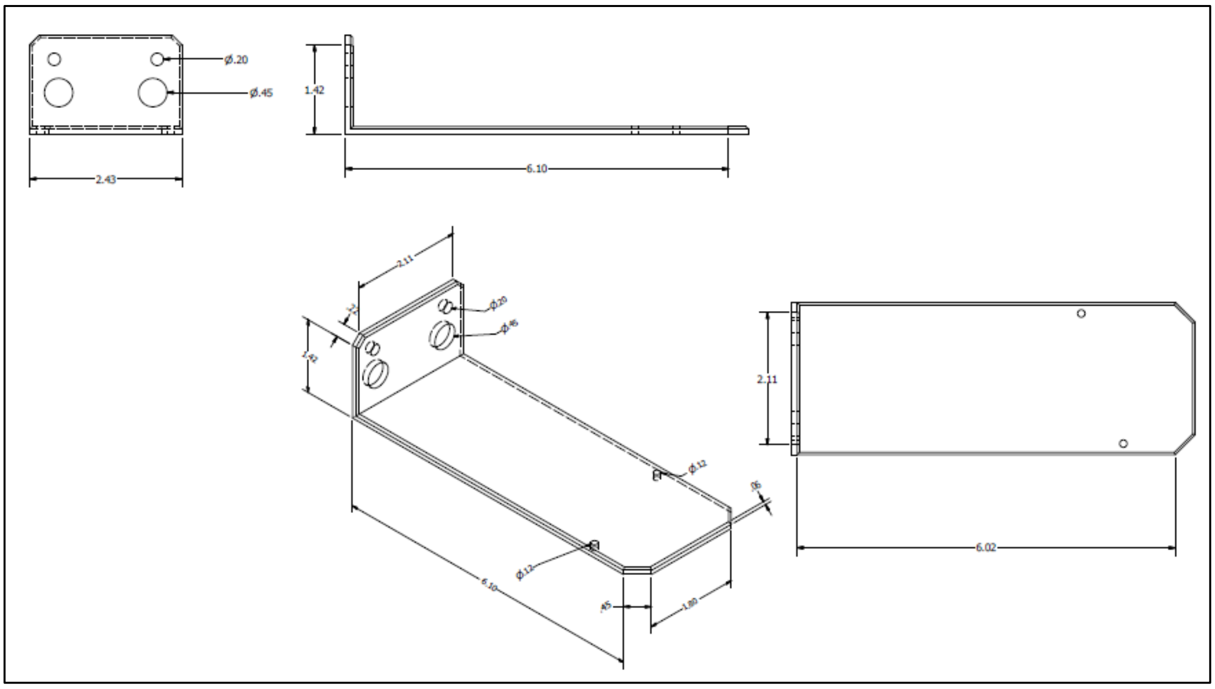
Bearing holder (Top)



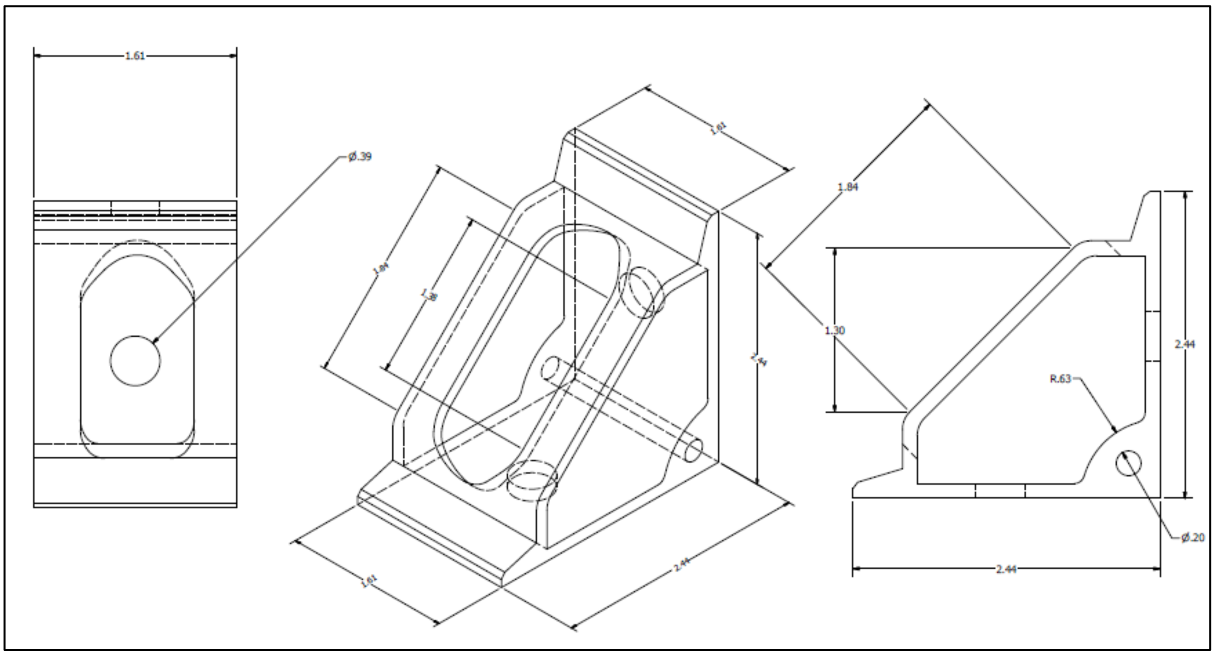
Bearing holder (Bottom)



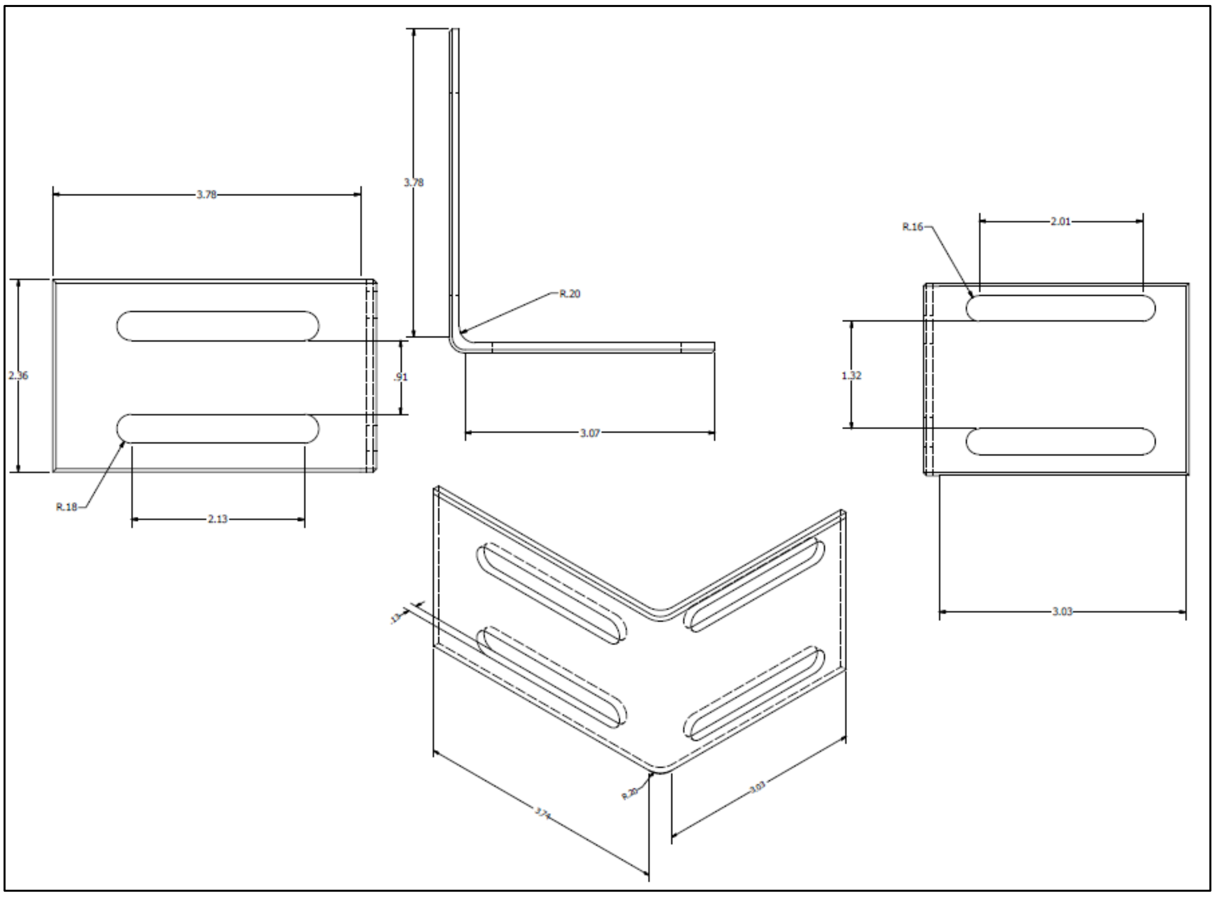
Distance sensor holder



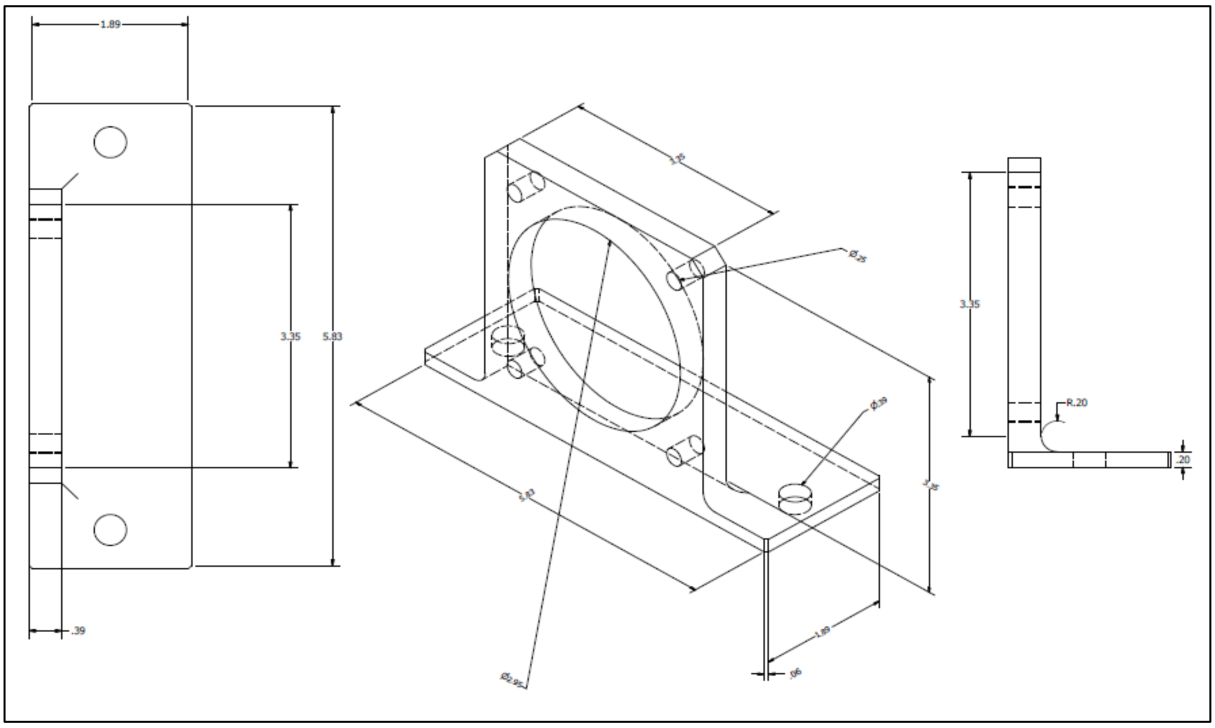
Extrusion angle/ corner fastener



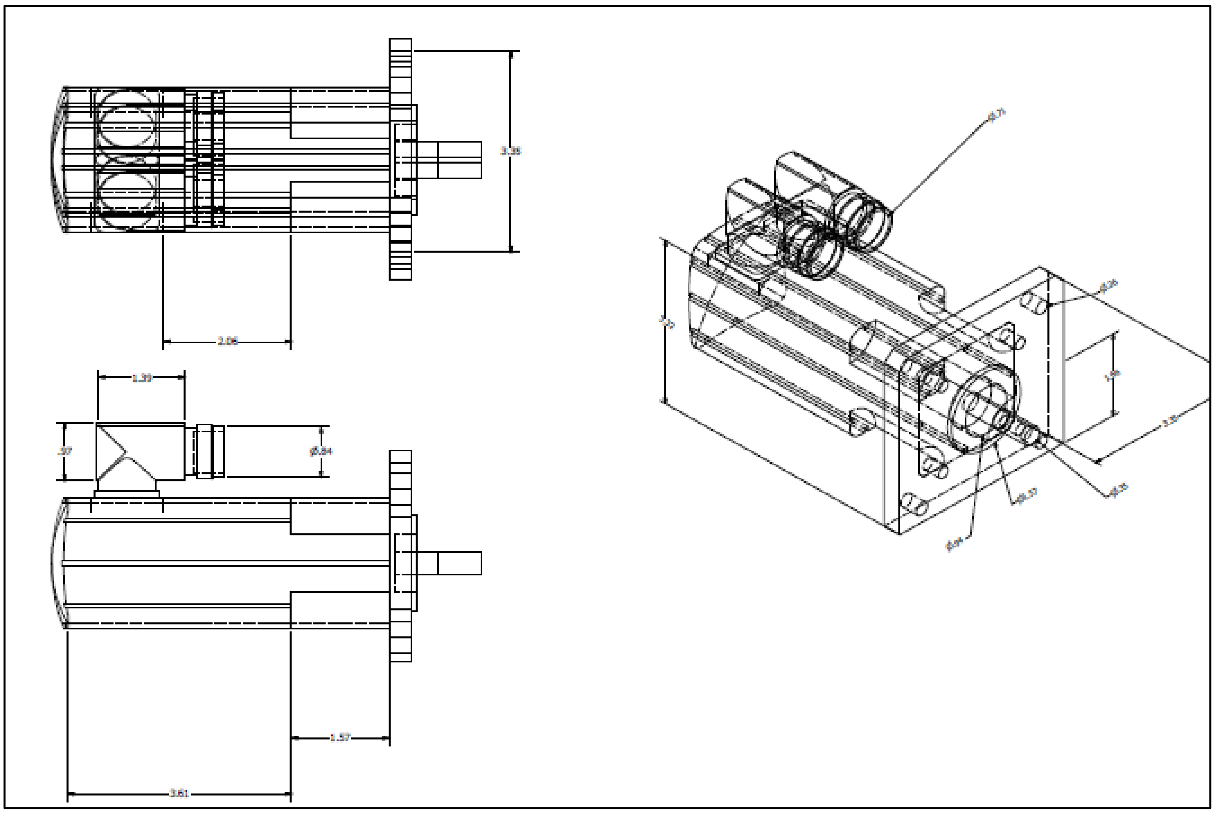
Distance sensor position adjuster



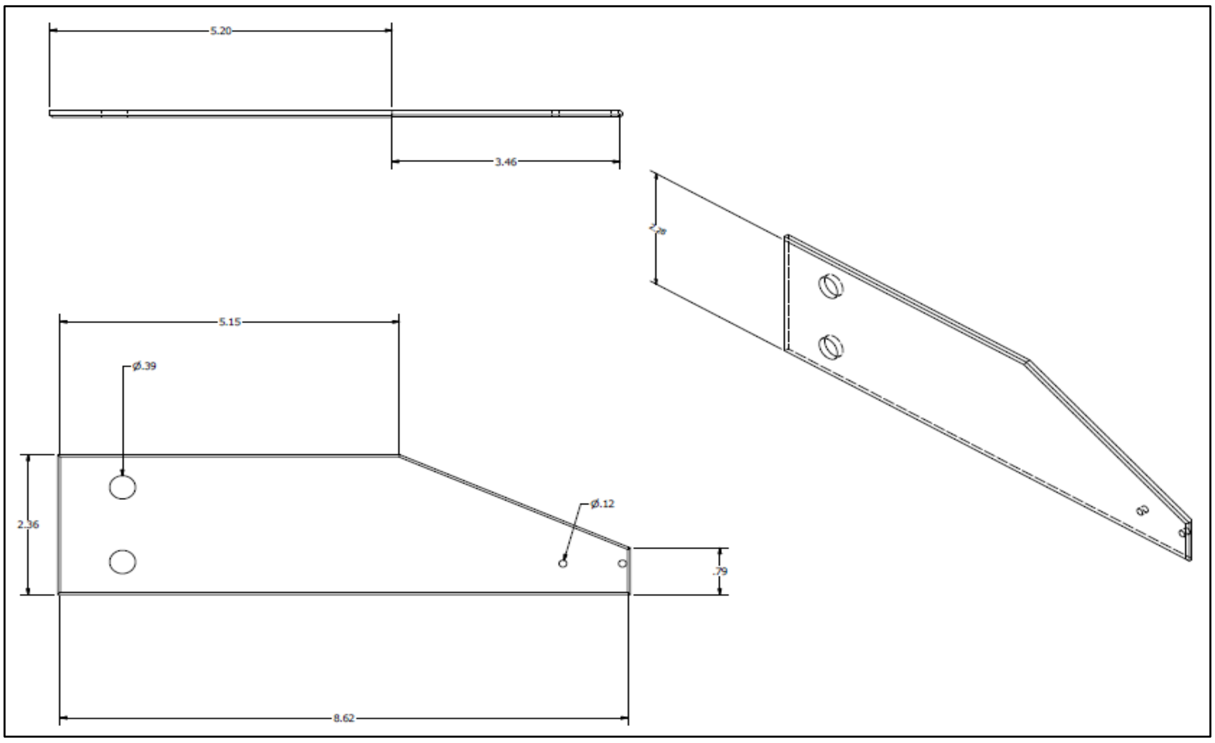
Motor fastening bracket



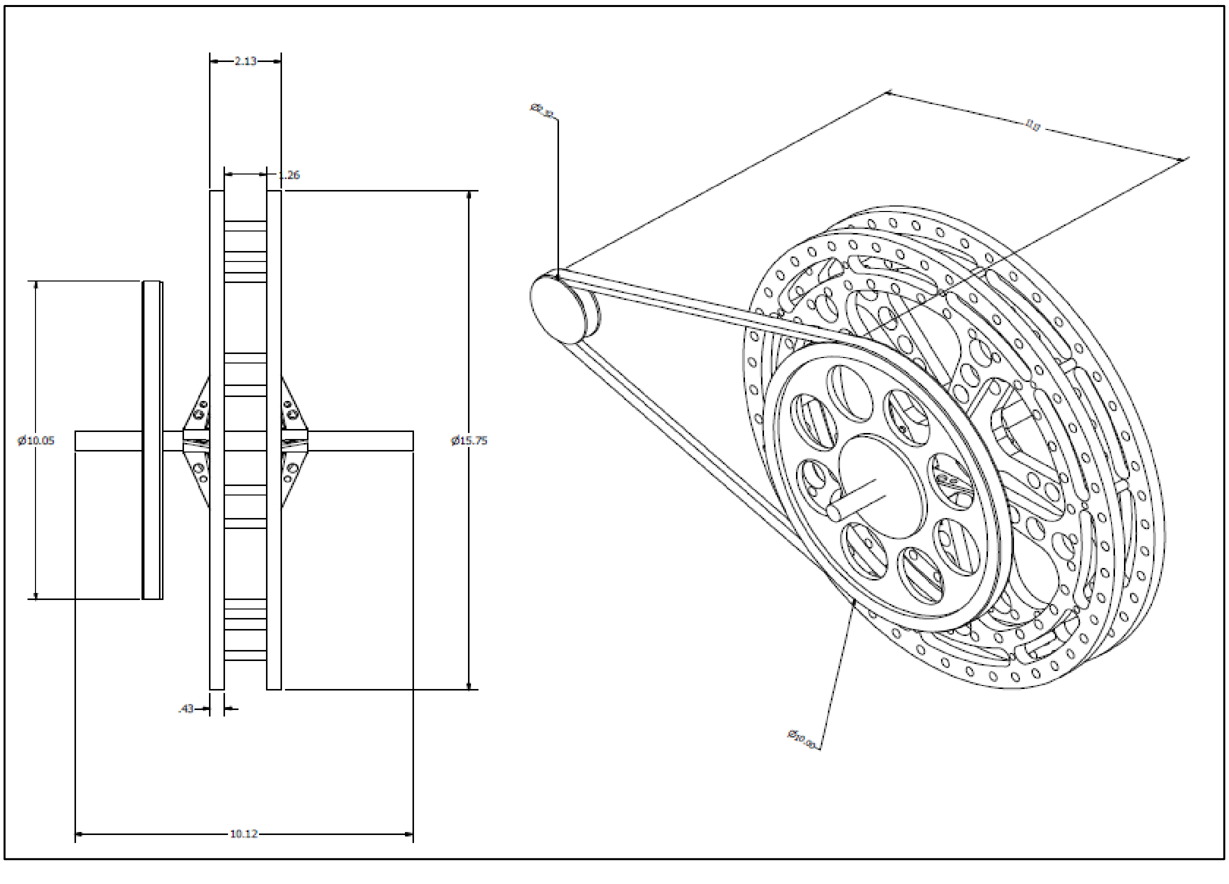
Servo motor dimensions



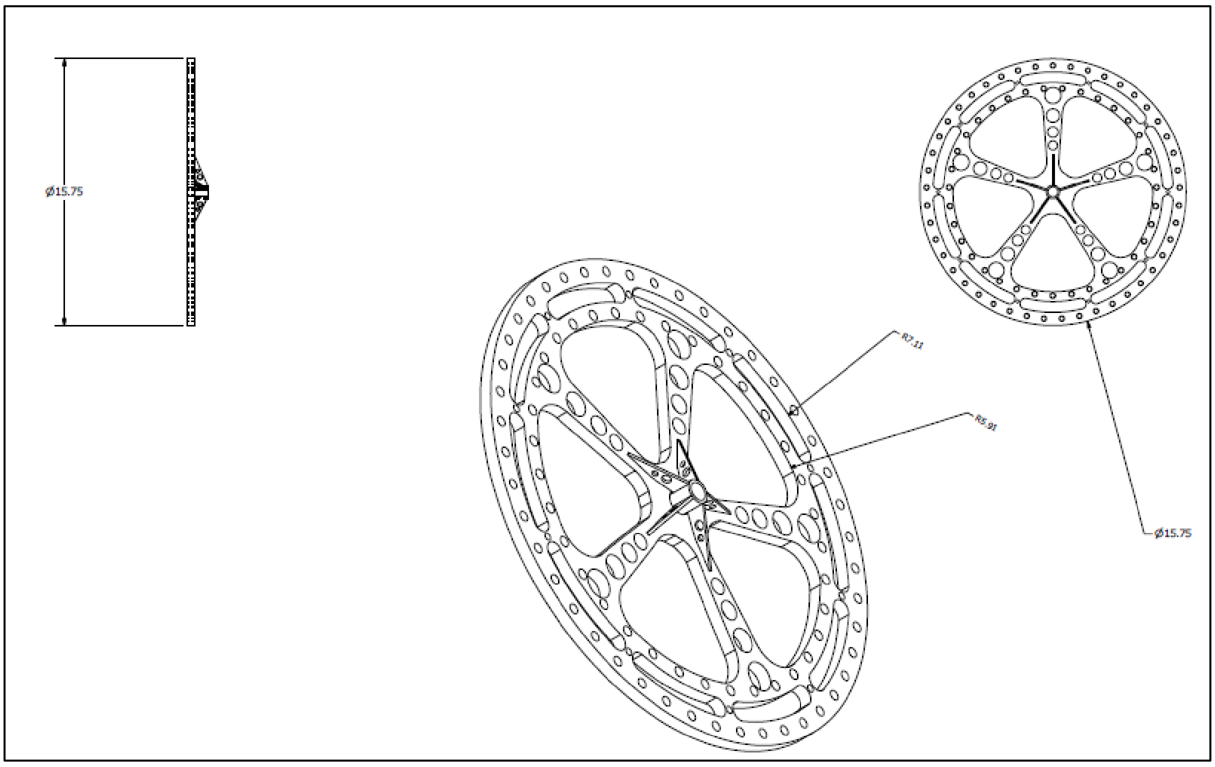
Ball radius sensor holder



Wheel assembly



Wheel Assembly continued



APPENDIX B: MATLAB/ SIMULINK CODE

```
%% BALL ON WHEEL (BOW) MATLAB SIMULATION CODE

%% NAME: BALL ON WHEEL SYSTEM MODEL AND CONTROLLER SIMULATION
%% AUTHOR: JM FERNANDES
%% DATE: FEB 2013

%% PARAMETERS FOR BALL ON WHEEL SYSTEM

%% DEFINITION OF CONSTANTS
g = 9.81;                                        %Gravitational acceleration
%% WHEEL PARAMETERS
Rw = 0.195;                                      % Radius of wheel
Mw = 2.5;                                        % Mass of wheel
Iw = (0.5*Mw)*((Rw)^2);                          % Moment of inertia of
                                                 wheel

%% BALL PARAMETERS
Mb = 0.1;                                        % Mass of Ball
Rb = 0.02465;                                    % Radius of the ball
Ib = (2/5)*(Mb)*((Rb)^2);                        % Moment of Inertia of ball
%% DEFINITION OF FUNDAMENTAL SYSTEM EQUATIONS
E = (Iw + ((2/5)*(Mb)*((Rw)^2)));
F = ((2/5)*(Mb)*((Rw)^2))+((2/5)*(Mb*Rb*Rw));
G = (7*((Rw)^2) + (14*Rw*Rb) + (7*((Rb)^2)));
H = (2*((Rw)^2)+(2*Rb));
I =  5*g*(Rw+Rb);
R = ((H)/ ((G*E)-(H*F)));
N = ((E*I)/((G*E)-(H*F)));
P = ((G)/((G*E)-(H*F)));
M = ((F*I)/((G*E)-(H*F)));
%% SYSTEM MATRICES (STATE SPACE)
Ja =  [0 1 0 1; N 0 0 0; 0 0 0 1; M 0 0 0];      %  Jacobian  linearized
                                                 system matrix
Jb =  [0; R; 0; P];                              % Linearized input matrix
Cy = [1 0 0 0];                                  % Output vector (to control
                                                 wheel angle theta 1)
Dy = 0;                                          % Disturbance
sys_ss = ss(Ja,Jb,Cy,Dy);                        % State space model of Ball
                                                 on Wheel system

%% SYSTEM PLOT
figure(1);step (sys_ss,'b');                     % Step response of system
xlabel('Time(s)');ylabel('Amplitude');
title('Setpoint and Actual output');
rank (ctrb(sys_ss));                             %  Investigate  system
                                                 controllability matrix
rank (obsv(sys_ss))                              %  Investigate  system
                                                 observability matrix
[num den] = ss2tf(Ja, Jb, Cy, Dy, 1);            % Converts state space to
                                                 transfer function
sys_tf = tf(num,den);                            %  Convert  to  transfer
                                                 function
sys_tf_2 = minreal(sys_tf);                      %  Eradicate  unnecessary
                                                 poles/zeros that have no
                                                 effect
```

```
figure(2);rlocus(sys_tf_2, 'r');          % Plot the root locus of
                                          uncompensated system
xlabel('S-plane');ylabel('im');
title('Root Locus Plot');
figure(3);bodeplot(sys_tf_2,'g');         % Plot Bode diagram
                                          (frequency domain)
xlabel('Freq(rad/sec))');ylabel('dBs');
title('Bode plot');
margin(sys_tf_2)                          % Determine phase and gain
                                          margin of uncompensated
                                          system
```

%% CONTROLLER IMPLEMENTATION SELECTION

% Default = 0. Set sel between 0-5 to select controller type: PD=0, PID=1,LEAD = 2, LEAD_LAG = 3, LQR = 4, NN-PID = 5

```
sel = 0;                                              %Controller
                                                      selector
```

%% TRADITIONAL CONTROLLER IMPLEMENTATION

%% 1. PD CONTROLLER

```
if sel == 0
num_pd = [2.4946 24.946];                             % PD numerator
                        den_pd = [1];                 % PD
                                                      denominator
sys_pd = tf(num_pd, den_pd);                          % Transfer
                                                      function of PD
mul_num = sys_pd*sys_tf_2;                            % Multiply PD
                                                      controller with
                                                      BOW system
[num_com,den_com] = tfdata(mul_num, 'v');             % Extract
                                                      numerator and
                                                      denominator
[num_cl den_cl] = cloop(num_com,den_com);             % Close the
                                                      loop
      sys_pd_cl = tf(num_cl,den_cl);                  % Transfer
                                                      function of
                                                      closed loop
                                                      system
      figure(4);step(sys_pd_cl,'b');                  % Step response
                                                      of system with
                                                      PD controller
xlabel('Time(s)');ylabel('Amplitude');
title('Setpoint and Actual output with PD controller');
C_PD_Z = c2d(sys_pd,0.001,'tustin');                  %% Sampling
                                                      time is 1 ms,
                                                      and Tustin's
                                                      bilinear
                                                      transformation
                                                      is used
```

%% PID CONTROLLER

```
elseif sel == 1
Kp = 134.4;                                           % Proportional
                                                      gain
Kd = 22.176;                                          % Derivative
gain
Ki = 112;                                             % Integral Gain
pid_num = [Kd Kp Ki];                                 % PID Numerator
pid_den = [1 0];                                      % PID
Denominator
```

```
      pid_num_tf = tf(pid_num,pid_den);                    % Transfer function of PID controller
mul_num_pid = pid_num_tf*sys_tf_2;                         % Combining PID controller with the BOW system
[pid_num_com,pid_den_com] = tfdata(mul_num_pid, 'v');      % Extract numerator and denominator from
[pid_num_cl pid_den_cl] = cloop(pid_num_com,pid_den_com);  % Close the loop
sys_pid_cl = tf(pid_num_cl,pid_den_cl);                    % Transfer function of closed loop system
figure(4);step(sys_pid_cl);                                % Closed loop step response with PID controller
xlabel('Time(s)');ylabel('Amplitude');
title('Setpoint and Actual output with PID controller');
C_PID_Z = c2d(pid_num_tf,0.0001,'tustin');                 %% Sampling time is 1 ms, and Tustin’s bilinear transformation is used
```

```
%% LEAD CONTROLLER
elseif sel == 2
lead_num = [5.04 112];                                     % Lead Numerator
                  lead_den = [0.0012 1];                   % Lead Denominator
lead_num_tf = tf(lead_num,lead_den);                       % Transfer function of Lead controller
      mul_lead = lead_num_tf*sys_tf_2;                     % Combining Lead controller with the BOW system
[lead_num_com,lead_den_com] = tfdata(mul_lead, 'v');       % Extract numerator and denominator
[lead_num_cl lead_den_cl] = cloop(lead_num_com,lead_den_com);   % Close the loop
sys_lead_cl = tf(lead_num_cl,lead_den_cl);                 % Transfer function of closed loop system
figure(4);step(sys_lead_cl);
xlabel('Time(s)');ylabel('Amplitude');
title('Setpoint and Actual output with LEAD compensator');
C_LEAD_Z = c2d(lead_num_tf,0.001,'tustin')                 %% Sampling time is 1 ms, and Tustin’s bilinear transformation is used
```

```
%%LEAD_LAG COMPENSATOR
elseif sel == 3
leadlag_num = [1.0994e-5 7.8528 49.08];                     % Numerator
leadlag_den = [7.8e-8 5.9e-4 1];                            % Denominator
leadlag_num_tf = tf(leadlag_num,leadlag_den);               %      Transfer
                                                            function      of
                                                            controller
mul_leadlag = leadlag_num_tf*sys_tf_2;                      %     Combining
                                                            controller  with
                                                            the BOW system
[leadlag_num_com,leadlag_den_com] = tfdata(mul_leadlag, 'v');     % Extract
                                                                  numerator
                                                                  and
                                                                  denominat
                                                                  or
[leadlag_num_cl  leadlag_den_cl]  =  cloop(leadlag_num_com,leadlag_den_com);
                                                                  %    Close
the loop
sys_leadlag_cl = tf(leadlag_num_cl,leadlag_den_cl)          %      Transfer
                                                            function      of
                                                            closed      loop
                                                            system
figure(4);step(sys_leadlag_cl);
xlabel('Time(s)');ylabel('Amplitude');
title('Setpoint and Actual output with LEAD LAG compensator');
C_LEADLAG_Z = c2d(leadlag_num_tf,0.001,'tustin')             % Digitize
```

```
%% LQR (LINEAR QUADRATIC REGULATOR)
elseif sel == 4
Q = [0 1 0 1;0 0 0 0 ; 0 1 0 1; 0 0 0 0];                   %   Q   matrix:
Cy'*Cy
R = 1;
                  K = lqr(Ja,Jb,Q,R);                       % Find feedback
                                                            control   matrix
                                                            K using 'lqr'
                  Ac = [(Ja-(Jb*(K)))];                     % Compensated A
                                                            matrix
                              Bc = [Jb];                    % Compensated B
                                                            matrix
Cc = [Cy];                                                  % Compensated C
matrix
Dc = [Dy];                                                  % Compensated D
matrix
sys_lqr = ss(Ac,Bc,Cc,Dc);                                  %   Compensated
system in state space                                                      %
Step response of compensated system
[num den] = ss2tf(Ac,Bc,Cc,Dc);                             % Converting to
tf
sys_lqr_tf = tf(num,den);                                   % Converting to
tf
lqr_d = c2d(sys_lqr_tf,0.0001,'tustin')                     % Digitize
figure(4);step(sys_lqr_tf);
xlabel('Time(s)');ylabel('Amplitude');
title('Setpoint and Actual output with LQR');
```

```
%%INTELLIGENT NN BASED CONTROL
%% NN-PID (Neural Network PID controller)
elseif sel == 5

ts=0.001;
```

```
n= 0.9;
alfa=0.000000001;
beta = 0.6;

% Parameter Initialization
u_1=0.0;u_2=0.0;u_3=0.0;
y_1=0;y_2=0;y_3=0;

%Digital system numerator and denominator
dsys=c2d(sys_tf_2,ts,'z');
[nn_num,nn_den]=tfdata(dsys,'v');

%Parameter initialization
w1_1 = 1;
w2_1 = -1;
w1_2 = 1;
w1_3 = 1;
w2_2 = -1;
w2_3 = -1;
u1_1 = 0;
u1_2 = 0;
u2_1 = 0;
x1_1 = 0;
x1_2 = 0;
x2_1 = 0;
uo = 0;
wo_1 = 0.1;
wo_2 =0.1;
wo_3 = 0.1;
xo = 0;
xo_1 = 0;
yout=0;
y_1 = 0;
error = 0;
u2_2prev = 0;
u2_3prev = 0;
error_prev=0;
h=0;
ts=0.001;
dwo_1_prev = 0;
dwo_2_prev = 0;
dwo_3_prev = 0;
b=0;

S=1; % Signal type
v=1; % 1= include disturbances, 0 = step command

for k=1:1:12000                                                     %Repetitions
time(k)=k*ts;

%%Setpoint generator
if S==1

   if k == 2000 && v == 1
   rin(k)= 1;%0.8; % Step Input
   b(k)=0.5;
   elseif k == 3000 && v == 1
   rin(k)= 1;%0.8; % Step Input
   b(k)=0.4;
   elseif k == 4000 && v == 1
```

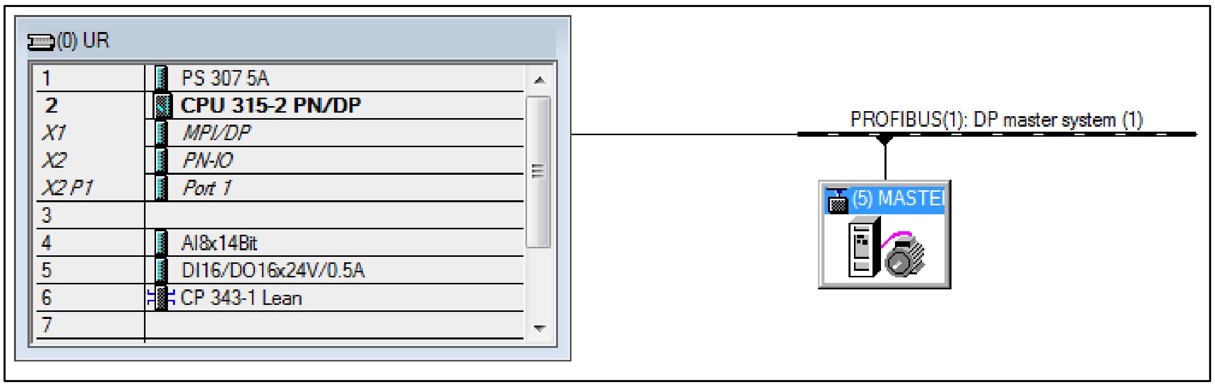
```
    rin(k)= 1;%0.5; % Step Input
     b(k)=0.3*0.0001*k;
    elseif k == 5500 && v == 1
    rin(k)=1;%1.5;%(0.001*k)/2; % Step Input
    %den(3)=1.01;
    b(k)=-0.2;
    elseif k == 7500 && v == 1
    rin(k)= 1;%0.5; % Step Input
    b(k)=-0.3*0.0001*k;
    elseif k == 9000 && v == 1
    rin(k)= 1;%0.5; % Step Input
    b(k)=-0.1;
    else
    rin(k)= 1;%0.5; % Step Input
    b(k)=0;
    end
    end

%Ball on wheel model in time domain
%Non-linear model in Time Domain
% System output:
yout(k)=(-nn_den(2)*y_1-nn_den(3)*y_2)+(nn_num(1)*u_1+nn_num(2)*u_2)
error(k)=rin(k)-yout(k);                                     % System Error

%% Network configuration
%%Input neurons
u1_1(k)  = rin(k);
x1_1(k)  = u1_1(k);

u1_2(k)  = yout(k)+b(k);
x1_2(k)  = u1_2(k);
%% Hidden layer neurons
%P-Neuron
u2_1(k)  = x1_1(k)*w1_1 + x1_2(k)*w2_1;
x2_1(k)  = u2_1(k);

%I-Neuron
u2_2(k)  = x1_1(k)*w1_2 + x1_2(k)*w2_2;
x2_2(k)  = u2_2prev + u2_2(k);

%D-Neuron
u2_3(k)  = x1_1(k)*w1_3 + x1_2(k)*w2_3;
x2_3(k)  = (u2_3(k) - u2_3prev);
%%
%%Output layer neurons
%Output neuron
uo(k)  = x2_1(k)*wo_1 + x2_2(k)*wo_2 + x2_3(k)*wo_3;
xo(k)  = uo(k);

%Mean-square error
E(k)  = 0.5*((yout(k)-rin(k))^2);
%%
%%Weight Update
%Hidden to Output layer
dk(k)  = (yout(k)- rin(k))*yout(k)*(1-yout(k))*0.5;
dwo_1(k)  = -n*dk(k)*x2_1(k)+ beta*dwo_1_prev;
wo_1 = wo_1 + dwo_1(k);
```

```
dwo_2(k)  = -n*dk(k)*x2_2(k)+ beta*dwo_2_prev;
wo_2 = wo_2 + dwo_2(k);


dwo_3(k)  = -n*dk(k)*x2_3(k)+ beta*dwo_3_prev;
wo_3 = wo_3 + dwo_3(k);


%Input to Hidden layer weight updates
dw1_1(k)   =  n*x1_1(k)*x2_1(k)*(1  -  x2_1(k))*(dk(k)*wo_1  +  dk(k)*wo_2  +
dk(k)*wo_3);
w1_1 = w1_1 + alfa*dw1_1(k);

dw1_2(k)   =  n*x1_1(k)*x2_2(k)*(1  -  x2_2(k))*(dk(k)*wo_1  +  dk(k)*wo_2  +
dk(k)*wo_3);
w1_2 = w1_2 + alfa*dw1_2(k);

dw1_3(k)   =  n*x1_1(k)*x2_3(k)*(1  -  x2_3(k))*(dk(k)*wo_1  +  dk(k)*wo_2  +
dk(k)*wo_3);
w1_3 = w1_3 + alfa*dw1_3(k);

dw2_1(k)   =  n*x2_1(k)*x2_1(k)*(1  -  x2_1(k))*(dk(k)*wo_1  +  dk(k)*wo_2  +
dk(k)*wo_3);
w2_1 = w2_1 + alfa*dw2_1(k);

dw2_2(k)   =  n*x2_1(k)*x2_2(k)*(1  -  x2_2(k))*(dk(k)*wo_1  +  dk(k)*wo_2  +
dk(k)*wo_3);
w2_2 = w2_2 + alfa*dw2_2(k);

dw2_3(k)   =  n*x2_1(k)*x2_3(k)*(1  -  x2_3(k))*(dk(k)*wo_1  +  dk(k)*wo_2  +
dk(k)*wo_3);
w2_3 = w2_3 + alfa*dw2_3(k);

kp(k)  = wo_1; %Equating Kp
ki(k)  = wo_2; %Equating Ki
kd(k)  = wo_3; %Equating Kd

% Variable updates/ previous values updates
y_1 = yout(k);
xo_1 = xo(k);
u2_2prev = u2_2(k);
u2_3prev = u2_3(k);
error_prev = error(k);
w1_1_1(k)  = w1_1;

dwo_1_prev = dwo_1(k);
dwo_2_prev =dwo_2(k);
dwo_3_prev =dwo_3(k);

u_3=u_2;u_2=u_1;u_1=xo(k);
y_3=y_2;y_2=y_1;y_1=yout(k);
end
%%NN-PID PLOTS

figure(4);
plot(time,rin,'r',time,yout,'b');                          %, time, xo, 'g');
xlabel('time(s)');ylabel('rin,yout');
title('Setpoint and actual output');
```

```
figure(5);
plot(time,error,'r');
xlabel('time(s)');ylabel('error');
title('System error');

figure(6);
plot(time,E,'g');
xlabel('time(s)');ylabel('E');
title('Mean-Square error');

figure(7);
subplot(311);
plot(time,kp,'r');
xlabel('time(s)');ylabel('kp');
subplot(312);

plot(time,ki,'g');
xlabel('time(s)');ylabel('ki');
subplot(313);

plot(time,kd,'b');
xlabel('time(s)');ylabel('kd');

end
```

APPENDIX C: PLC CODE

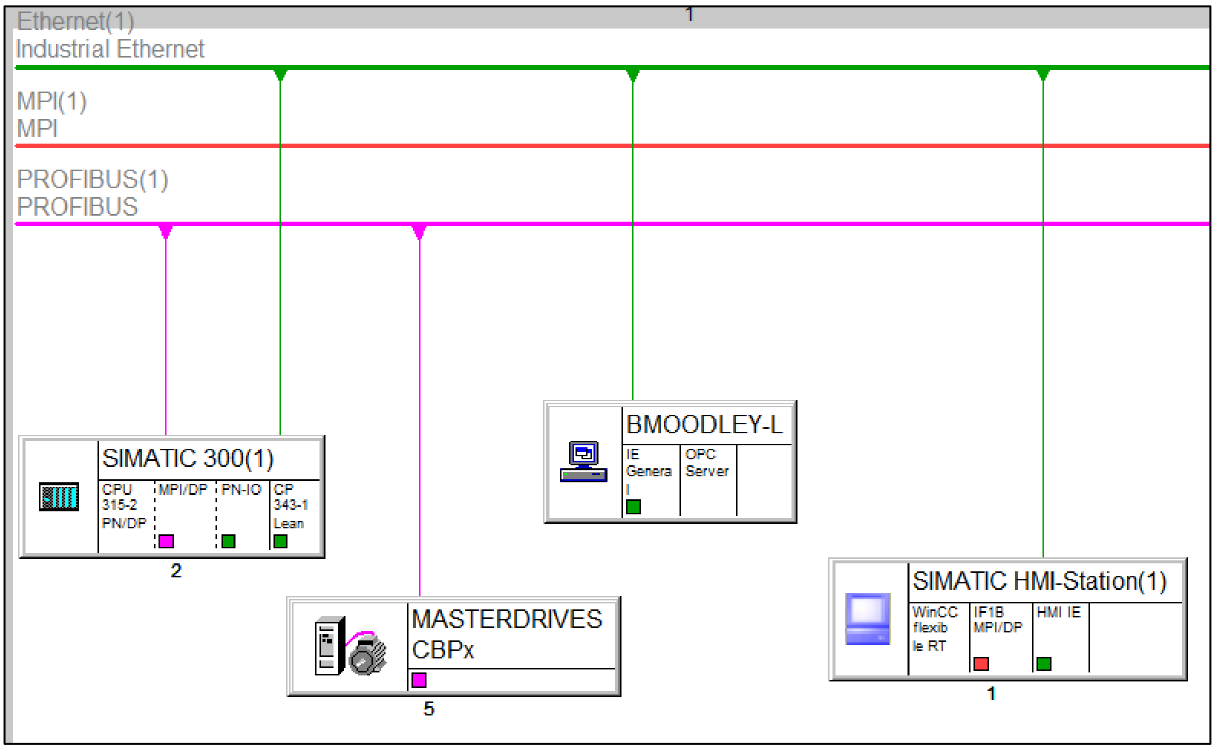
Hardware configuration



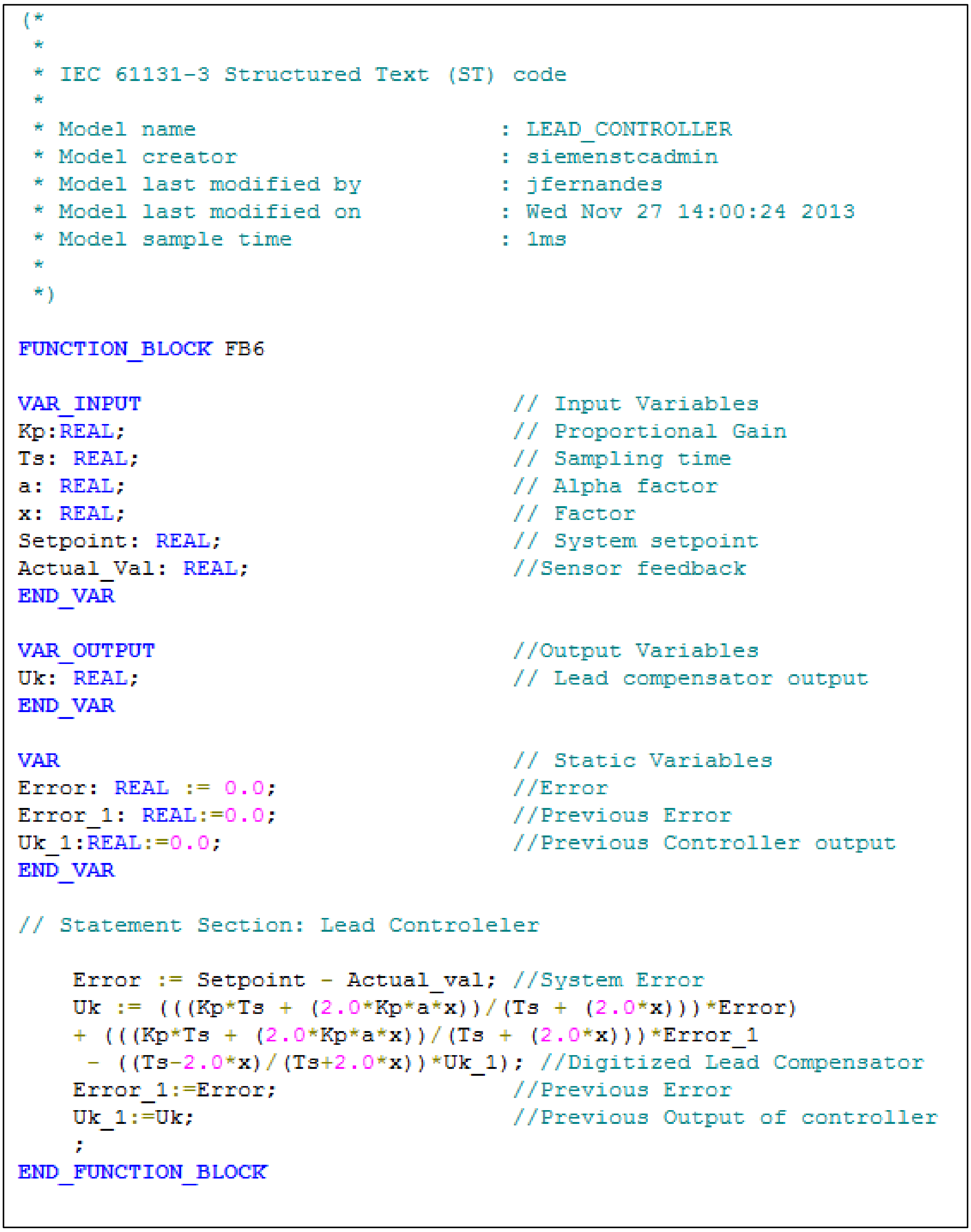
(0) UR

1	PS 307 5A
2	**CPU 315-2 PN/DP**
X1	*MPI/DP*
X2	*PN-IO*
X2 P1	*Port 1*
3	
4	AI8x14Bit
5	DI16/DO16x24V/0.5A
6	CP 343-1 Lean
7	

BOW System network setup

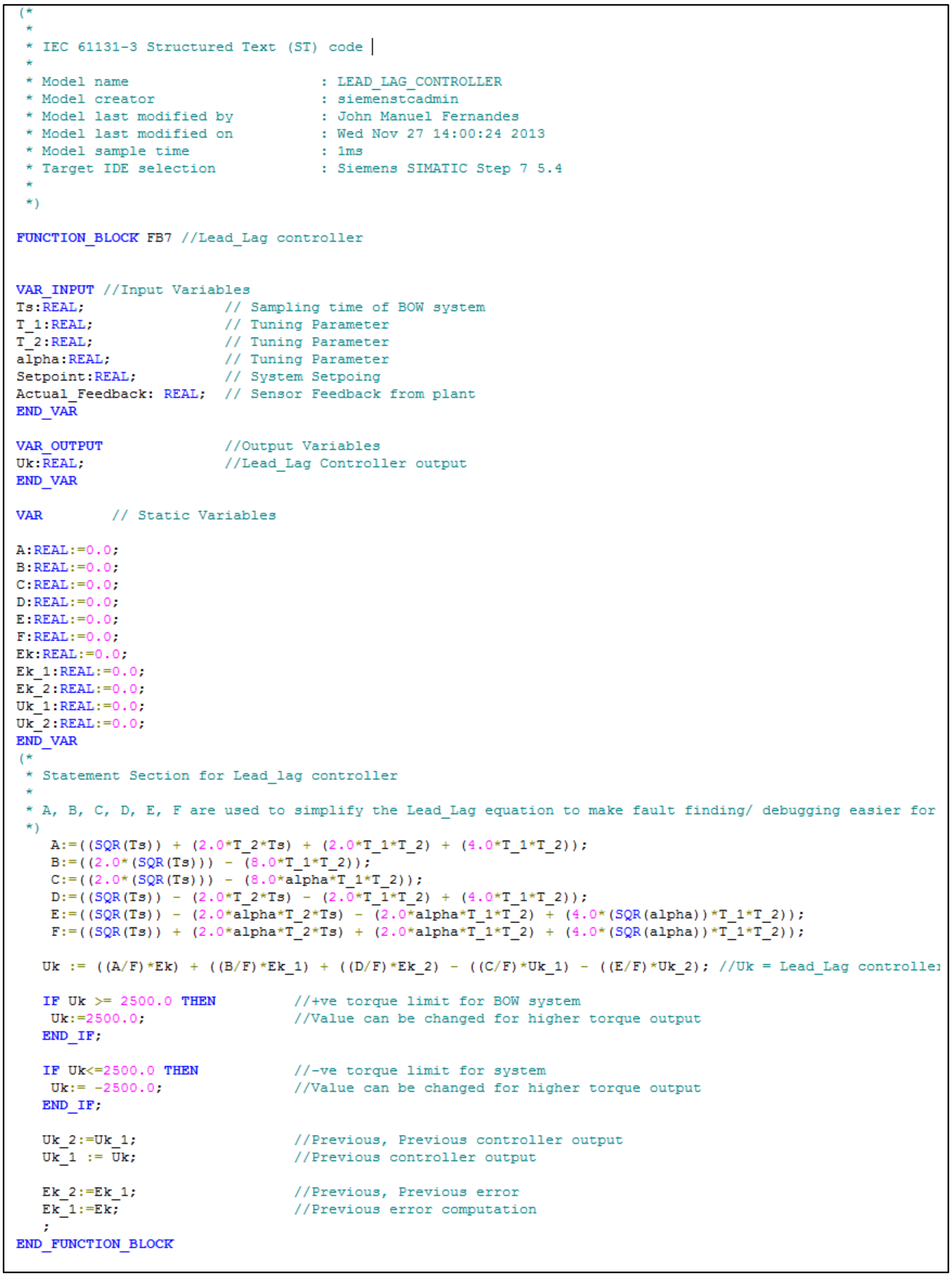


Lead compensator code implemented in SCL



```
(*
 *
 * IEC 61131-3 Structured Text (ST) code
 *
 * Model name                        : LEAD_CONTROLLER
 * Model creator                     : siemenstcadmin
 * Model last modified by            : jfernandes
 * Model last modified on            : Wed Nov 27 14:00:24 2013
 * Model sample time                 : 1ms
 *
 *)

FUNCTION_BLOCK FB6

VAR_INPUT                           // Input Variables
Kp:REAL;                            // Proportional Gain
Ts: REAL;                           // Sampling time
a: REAL;                            // Alpha factor
x: REAL;                            // Factor
Setpoint: REAL;                     // System setpoint
Actual_Val: REAL;                   //Sensor feedback
END_VAR

VAR_OUTPUT                          //Output Variables
Uk: REAL;                           // Lead compensator output
END_VAR

VAR                                 // Static Variables
Error: REAL := 0.0;                 //Error
Error_1: REAL:=0.0;                 //Previous Error
Uk_1:REAL:=0.0;                     //Previous Controller output
END_VAR

// Statement Section: Lead Controleler

    Error := Setpoint - Actual_val; //System Error
    Uk := (((Kp*Ts + (2.0*Kp*a*x))/(Ts + (2.0*x)))*Error)
    + (((Kp*Ts + (2.0*Kp*a*x))/(Ts + (2.0*x)))*Error_1
     - ((Ts-2.0*x)/(Ts+2.0*x))*Uk_1); //Digitized Lead Compensator
    Error_1:=Error;                 //Previous Error
    Uk_1:=Uk;                       //Previous Output of controller
    ;
END_FUNCTION_BLOCK
```

Lead-Lag compensator code implemented in SCL



```
(*
 *
 * IEC 61131-3 Structured Text (ST) code
 *
 * Model name                        : LEAD_LAG_CONTROLLER
 * Model creator                     : siemenstcadmin
 * Model last modified by            : John Manuel Fernandes
 * Model last modified on            : Wed Nov 27 14:00:24 2013
 * Model sample time                 : 1ms
 * Target IDE selection              : Siemens SIMATIC Step 7 5.4
 *
 *)

FUNCTION_BLOCK FB7 //Lead_Lag controller


VAR_INPUT //Input Variables
Ts:REAL;                 // Sampling time of BOW system
T_1:REAL;                // Tuning Parameter
T_2:REAL;                // Tuning Parameter
alpha:REAL;              // Tuning Parameter
Setpoint:REAL;           // System Setpoing
Actual_Feedback: REAL;   // Sensor Feedback from plant
END_VAR

VAR_OUTPUT               //Output Variables
Uk:REAL;                 //Lead_Lag Controller output
END_VAR

VAR        // Static Variables

A:REAL:=0.0;
B:REAL:=0.0;
C:REAL:=0.0;
D:REAL:=0.0;
E:REAL:=0.0;
F:REAL:=0.0;
Ek:REAL:=0.0;
Ek_1:REAL:=0.0;
Ek_2:REAL:=0.0;
Uk_1:REAL:=0.0;
Uk_2:REAL:=0.0;
END_VAR
(*
 * Statement Section for Lead_lag controller
 *
 * A, B, C, D, E, F are used to simplify the Lead_Lag equation to make fault finding/ debugging easier for
 *)
    A:=((SQR(Ts)) + (2.0*T_2*Ts) + (2.0*T_1*T_2) + (4.0*T_1*T_2));
    B:=((2.0*(SQR(Ts))) - (8.0*T_1*T_2));
    C:=((2.0*(SQR(Ts))) - (8.0*alpha*T_1*T_2));
    D:=((SQR(Ts)) - (2.0*T_2*Ts) - (2.0*T_1*T_2) + (4.0*T_1*T_2));
    E:=((SQR(Ts)) - (2.0*alpha*T_2*Ts) - (2.0*alpha*T_1*T_2) + (4.0*(SQR(alpha))*T_1*T_2));
    F:=((SQR(Ts)) + (2.0*alpha*T_2*Ts) + (2.0*alpha*T_1*T_2) + (4.0*(SQR(alpha))*T_1*T_2));

   Uk := ((A/F)*Ek) + ((B/F)*Ek_1) + ((D/F)*Ek_2) - ((C/F)*Uk_1) - ((E/F)*Uk_2); //Uk = Lead_Lag controlle

   IF Uk >= 2500.0 THEN         //+ve torque limit for BOW system
    Uk:=2500.0;                 //Value can be changed for higher torque output
   END_IF;

   IF Uk<=2500.0 THEN           //-ve torque limit for system
    Uk:= -2500.0;               //Value can be changed for higher torque output
   END_IF;

   Uk_2:=Uk_1;                  //Previous, Previous controller output
   Uk_1 := Uk;                  //Previous controller output

   Ek_2:=Ek_1;                  //Previous, Previous error
   Ek_1:=Ek;                    //Previous error computation
   ;
END_FUNCTION_BLOCK
```

NN-PID controller code implemented in SCL

```
(*
 *
 * IEC 61131-3 Structured Text (ST) code generated for NN_PID"
 *
 * Model name                       : NN_PID
 * Model creator                    : siemenstcadmin
 * Model last modified by           : jfernandes
 * Model last modified on           : Wed Nov 27 14:00:24 2013
 * Target IDE selection             : Siemens SIMATIC Step 7 5.4
 * Test Bench included              : No
 *
 *)

FUNCTION_BLOCK FB4 //NN_PID CONTROLLER

////////////////////////////////////////////////////////////////////////////////
(*Block Parameters*)
////////////////////////////////////////////////////////////////////////////////

VAR_INPUT                                  //NNPID input parameters
sp:REAL;                                   //System setpoint
act_out: REAL;                             //Actual plant output (for feedback)
Ts:REAL;                                   //System sampling time
eta:REAL;                                  //learning rate
alpha:REAL;                                //Inertial coefficient
reset:BOOL;                                //Controller reset
hidden_layer:INT;                          //Selection for the number of hidden layers
END_VAR

VAR_OUTPUT                                 //NNPID output parameters
NNPID_OUT:REAL;                            //Controller output
Ek_out :REAL;                              //Display of minimization error
Kp :REAL;                                  //Kp value from NN
Kd :REAL;                                  //Kd value from NN
Ki :REAL;                                  //Ki value from NN
Tester: REAL;                              //Test output
END_VAR

VAR                                        //Static Block Parameters with initialization values
x_i1: REAL:=  0.0;                         //Input to input neuron 1
x_i2: REAL:=  0.0;                         //Input to input neuron 2

w_i1_h1: REAL:=  1.0;                      //Initial weight between input I1 and H1
w_i2_h1: REAL:=  -1.0;                     //Initial weight between input I2 and H1

w_i1_h2: REAL:=  1.0;                      //Initial weight between input I1 and H2
w_i2_h2: REAL:=  -1.0;                     //Initial weight between input I2 and H2

w_i1_h3: REAL:=  1.0;                      //Initial weight between input I1 and H3
w_i2_h3: REAL:=  -1.0;                     //Initial weight between input I2 and H3

x_h1: REAL:=  0.0;                         //Input to hidden layer neuron H1 = system error (sp-act_out)
x_h2: REAL:=  0.0;                         //Input to hidden layer neuron H2 = system error
x_h3: REAL:=  0.0;                         //Input to hidden layer neuron H3 = system error

x_h3_prev: REAL:=  0.0;                    //Previous input to hidden neuron 3
x_h3_prev_prev: REAL:=0.0;
x_h3_prev_prev_prev:REAL :=0.0;

y_h1 :REAL:=  0.0;                         //output from hidden neuron 1
y_h2 :REAL:=  0.0;                         //output from hidden neuron 2
y_h3 :REAL:=  0.0;                         //output from hidden neuron 3

x_o1 :REAL:=  0.0;                         //input to output layer neuron

w_h1_o1:REAL:=  0.0;                       //weight between h1 and o1 connection
w_h2_o1: REAL:=  0.0;                      //weight between h2 and o1 connection
w_h3_o1: REAL:=  0.0;                      //weight between h3 and o1 connection

d_w_h1_o1 : REAL:=  0.0;                   //weight updates "delta w" for h1 to o1
d_w_h2_o1 : REAL:=  0.0;                   //weight updates "delta w" for h2 to o1
d_w_h3_o1 : REAL:=  0.0;                   //weight updates "delta w" for h3 to o1
d_w_h1_o1_prev :REAL:=  0.0;               // previous weight update(delta)
d_w_h2_o1_prev :REAL:=  0.0;               // previous weight update(delta)
d_w_h3_o1_prev :REAL:=  0.0;               // previous weight update(delta)
```

```
d_w_i1_h1: REAL:=  0.0;
d_w_i2_h1: REAL:=  0.0;
d_w_i1_h2: REAL:=  0.0;
d_w_i2_h2: REAL:=  0.0;
d_w_i1_h3: REAL:=  0.0;
d_w_i2_h3: REAL:=  0.0;

d_w_i1_h1_prev:REAL :=  0.0;
d_w_i2_h1_prev:REAL:=  0.0;
d_w_i1_h2_prev:REAL:=  0.0;
d_w_i2_h2_prev:REAL:=  0.0;
d_w_i1_h3_prev:REAL:=  0.0;
d_w_i2_h3_prev:REAL:=  0.0;

Ek: REAL;                              // error minimization

currTIME:S5TIME;
time_reset:BOOL;
time_set:BOOL;
time_out:BOOL;
END_VAR

/////////////////////////////////////////////////////////////////////////////////////////
(* Statement Section - SELECTION OF NN STRUCTURE*)
/////////////////////////////////////////////////////////////////////////////////////////



// nn structure 2:3:1, where the mille layer are the pid gains
// inputs to input layer neurons
x_i1:= sp;   //1st input neuron recieves sp as its input
x_i2:= act_out; //2nd input neuron recieves act_out (BOW feedback) as its input

// weights between input to hidden layer neurons
w_i1_h1 := -1.0;
w_i2_h1 := 1.0;

w_i1_h2 := -1.0;
w_i2_h2 := 1.0;

w_i1_h3 :=-1.0;
w_i2_h3 :=1.0;

//cosider stating loop here
// inputs to hidden layer neurons
x_h1 := (w_i1_h1*x_i1) + (w_i2_h1*x_i2); //input to hidden neuron 1
x_h2 := (w_i1_h2*x_i1) + (w_i2_h2*x_i2); //input to hidden neuron 2
x_h3 := (w_i1_h3*x_i1) + (w_i2_h3*x_i2); //input to hidden neuron 3

//outputs from hidden layer are PID gains
y_h1 := x_h1; //output from 1st hidden neuron is our proportional gain
y_h2 := (x_h2*Ts);//y_h2 + (x_h2*Ts);//Integral
y_h3 := (x_h3 - x_h3_prev)/Ts;//(x_h3 - x_h3_prev)/Ts;//derivative

//the input to the output neuron is given by:
x_o1 := (w_h1_o1*y_h1)+(w_h2_o1*y_h2)+(w_h3_o1*y_h3);

NNPID_OUT := -1.0*x_o1;

//BP LEARNING USING WINDROW-HOFF IS USED TO MINIMIZE THE ERROR WHILST UPDATING THE NETWORK WEIGHTS:

Ek := SQR(0.5*(x_i1-x_i2));
Ek_out := Ek;


IF Ek >= 2000.0  THEN

d_w_h1_o1 := -eta*(sp-act_out)*act_out*y_h1 + (alpha*d_w_h1_o1_prev); //delta
w_h1_o1 := w_h1_o1 + (alpha*d_w_h1_o1);

d_w_h2_o1 := -eta*(sp-act_out)*act_out*y_h2 + (alpha*d_w_h2_o1_prev);
w_h2_o1 := w_h3_o1 + (alpha*d_w_h2_o1);

d_w_h3_o1 := -eta*(sp-act_out)*act_out*y_h3 + (alpha*d_w_h3_o1_prev); //delta
w_h3_o1 := w_h3_o1 + (alpha*d_w_h3_o1);
```

```
//the input to the output neuron is given by:
x_o1 := (w_h1_o1*y_h1)+(w_h2_o1*y_h2)+(w_h3_o1*y_h3);

NNPID_OUT := -1.0*x_o1;

//BP LEARNING USING WINDROW-HOFF IS USED TO MINIMIZE THE ERROR WHILST UPDATING THE NETWORK WEIGHTS:

Ek := SQR(0.5*(x_i1-x_i2));
Ek_out := Ek;


IF Ek >= 2000.0  THEN

d_w_h1_o1 := -eta*(sp-act_out)*act_out*y_h1 + (alpha*d_w_h1_o1_prev); //delta
w_h1_o1 := w_h1_o1 + (alpha*d_w_h1_o1);

d_w_h2_o1 := -eta*(sp-act_out)*act_out*y_h2 + (alpha*d_w_h2_o1_prev);
w_h2_o1 := w_h3_o1 + (alpha*d_w_h2_o1);

d_w_h3_o1 := -eta*(sp-act_out)*act_out*y_h3 + (alpha*d_w_h3_o1_prev); //delta
w_h3_o1 := w_h3_o1 + (alpha*d_w_h3_o1);

//weight updates for input to hidden layers:
d_w_i1_h1 := eta*y_h1*(1.0 - y_h1)*x_i1 + alpha*d_w_i1_h1_prev;
w_i1_h1:= w_i1_h1 + alpha*d_w_i1_h1;

d_w_i2_h1 := eta*y_h1*(1.0 - y_h1)*x_i2 + alpha*d_w_i1_h1_prev;
w_i2_h1:= w_i2_h1 + alpha*d_w_i2_h1;

d_w_i1_h2 := eta*y_h2*(1.0 - y_h2)*x_i1 + alpha*d_w_i1_h2_prev;
w_i1_h2:= w_i1_h2 + alpha*d_w_i1_h2;

d_w_i2_h2 := eta*y_h2*(1.0 - y_h2)*x_i2 + alpha*d_w_i1_h2_prev;
w_i2_h2:= w_i2_h2 + alpha*d_w_i2_h2;

d_w_i1_h3 := eta*y_h3*(1.0 - y_h3)*x_i1 + alpha*d_w_i1_h3_prev;
w_i1_h3:= w_i1_h3 +alpha*d_w_i1_h3;

d_w_i2_h3 := eta*y_h3*(1.0 - y_h3)*x_i2 + alpha*d_w_i1_h3_prev;
w_i2_h3:= w_i2_h3 + alpha*d_w_i2_h3;

Tester := 0.0;

ELSE

Tester := 1.0;
NNPID_OUT:=0.0;

END_IF;


IF (x_o1 > 3000.0) THEN
x_o1 :=3000.0;
NNPID_OUT:=x_o1;

END_IF;

IF (x_o1 < -3000.0) THEN
x_o1 :=-3000.0;
NNPID_OUT:= x_o1;
END_IF;

// IF THE ERROR IS TOO GREAT FOR TOO LLONG THEN THE CONTROLLER MUST BE RESET.
// ALSO THE BEST GAINS NEED TO BE MAINTAINED.

IF Ek >= 500000.0 THEN
    time_set := true;
ELSE
    time_set := false;
END_IF;

Kp := w_h1_o1;
Kd := w_h2_o1;
Ki := w_h3_o1;
```

```
d_w_h1_o1_prev:= d_w_h1_o1;
d_w_h2_o1_prev:= d_w_h2_o1;
d_w_h3_o1_prev:= d_w_h3_o1;

//input to hidden layer weights

d_w_i1_h1_prev := d_w_i1_h1;
d_w_i2_h1_prev := d_w_i2_h1;
d_w_i1_h2_prev := d_w_i1_h2;
d_w_i2_h2_prev := d_w_i2_h2;
d_w_i1_h3_prev := d_w_i1_h3;
d_w_i2_h3_prev := d_w_i2_h3;

;

/////////////////////////////////////////////////////////////////////////////////////////////
(*NN_PID Master Reset*)
/////////////////////////////////////////////////////////////////////////////////////////////

IF reset = true  THEN                   //Master reset for controller. Contoller also gets reset if the e

x_i1:=  0.0;                            //Input to input neuron 1
x_i2:=  0.0;                            //Input to input neuron 2

w_i1_h1:=  1.0;                         //initial weight between input I1 and H1
w_i2_h1:=  -1.0;                        //initial weight between input I2 and H1

w_i1_h2:=  1.0;                         //initial weight between input I1 and H2
w_i2_h2:=  -1.0;                        //initial weight between input I2 and H2

w_i1_h3:=  1.0;                         //initial weight between input I1 and H3
w_i2_h3:=  -1.0;                        //initial weight between input I2 and H3

x_h1:=  0.0;                            //input to hidden layer neuron H1 = system error (sp-act_out)
x_h2:=  0.0;                            //input to hidden layer neuron H2 = system error
x_h3:=  0.0;                            //input to hidden layer neuron H3 = system error

x_h3_prev:=  0.0;                       //previous input to hidden neuron 3

y_h1 :=  0.0;                           //output from hidden neuron 1
y_h2 :=  0.0;                           //output from hidden neuron 2
y_h3 :=  0.0;                           //output from hidden neuron 3

x_o1 :=  0.0;                           //input to output layer neuron

w_h1_o1:=  0.0;                         //weight between h1 and o1 connection
w_h2_o1:=  0.0;                         //weight between h2 and o1 connection
w_h3_o1:=  0.0;                         //weight between h3 and o1 connection

d_w_h1_o1 :=  0.0;                      //weight updates "delta w" for h1 to o1
d_w_h2_o1 :=  0.0;                      //weight updates "delta w" for h2 to o1
d_w_h3_o1 :=  0.0;                      //weight updates "delta w" for h3 to o1
d_w_h1_o1_prev :=  0.0;                 // previous weight update(delta)
d_w_h2_o1_prev :=  0.0;                 // previous weight update(delta)
d_w_h3_o1_prev :=  0.0;                 // previous weight update(delta)

d_w_i1_h1:=  0.0;
d_w_i2_h1:=  0.0;
d_w_i1_h2:=  0.0;
d_w_i2_h2:=  0.0;
d_w_i1_h3:=  0.0;
d_w_i2_h3:=  0.0;

d_w_i1_h1_prev:=  0.0;
d_w_i2_h1_prev:=  0.0;
d_w_i1_h2_prev:=  0.0;
d_w_i2_h2_prev:=  0.0;
d_w_i1_h3_prev:=  0.0;
d_w_i2_h3_prev:=  0.0;

Kp := 0.0;
Kd := 0.0;
Ki := 0.0;

NNPID_OUT := 0.0;

END_IF;
```

PID controller code implemented in SCL

```
(*
 *
 * IEC 61131-3 Structured Text (ST) code
 *
 * Model name                        : PID_CONTROLLER WITH FILTER
 * Model creator                     : siemenstcadmin
 * Model last modified by            : jfernandes
 * Model last modified on            : Wed Nov 27 14:00:24 2013
 * Target IDE selection              : Siemens SIMATIC Step 7 5.4
 *
 *)

FUNCTION_BLOCK FB3
VAR_INPUT
    ssMethodType: INT;
    SETPOINT: REAL;
    FEEDBACK: REAL;
    Ki: REAL ;
    Kd: REAL ;
    Kp: REAL ;
END_VAR
VAR_OUTPUT
    PID_OUT: REAL;
END_VAR
VAR
    Kn: REAL := 1.0;
    Integrator_DSTATE: REAL;
    Filter_DSTATE: REAL;
    c_rtb_FilterCoeffici: REAL;
END_VAR
CASE ssMethodType OF
    0:

        (* InitializeConditions for DiscreteIntegrator: '<S2>/Integrator' *)
        Integrator_DSTATE := 0.0;

        (* InitializeConditions for DiscreteIntegrator: '<S2>/Filter' *)
        Filter_DSTATE := 0.0;
    1:

        (* Gain: '<S2>/Filter Coefficient' incorporates:
         *  DiscreteIntegrator: '<S2>/Filter'
         *  Gain: '<S2>/Derivative Gain'
         *  Inport: '<Root>/FEEDBACK'
         *  Inport: '<Root>/SETPOINT'
         *  Sum: '<S2>/Sum3'
         *  Sum: '<S2>/SumD' *)
        c_rtb_FilterCoeffici := (((SETPOINT - FEEDBACK) * Kd) - Filter_DSTATE) * Kn;

        (* Outport: '<Root>/PID_OUT' incorporates:
         *  DiscreteIntegrator: '<S2>/Integrator'
         *  Gain: '<S2>/Proportional Gain'
         *  Inport: '<Root>/FEEDBACK'
         *  Inport: '<Root>/SETPOINT'
         *  Sum: '<S2>/Sum'
         *  Sum: '<S2>/Sum1' *)
        PID_OUT := ((((SETPOINT - FEEDBACK)*Kp) + Integrator_DSTATE) + c_rtb_FilterCoeffici)*-1.0;

        (* Update for DiscreteIntegrator: '<S2>/Integrator' incorporates:
         *  Gain: '<S2>/Integral Gain'
         *  Inport: '<Root>/FEEDBACK'
         *  Inport: '<Root>/SETPOINT'
         *  Sum: '<S2>/Sum2' *)
        Integrator_DSTATE := ((SETPOINT - FEEDBACK) * Ki) + (Integrator_DSTATE)/1000.0;

        (* Update for DiscreteIntegrator: '<S2>/Filter' *)
        Filter_DSTATE := Filter_DSTATE + c_rtb_FilterCoeffici;
END_CASE;

IF PID_OUT > 2500.0 THEN
    PID_OUT := 2500;
END_IF;
IF PID_OUT < -2500.0 THEN
    PID_OUT := -2500;
    END_IF;

END_FUNCTION_BLOCK
```

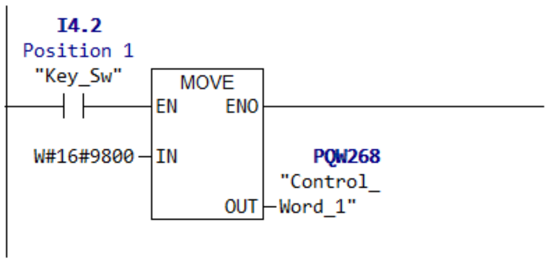
BOW system basic control function block

FB4 : BOW CONTROL PROGRAM

In this FB, the basic start/stop commands are programmed. Scaling is also done here as well as a few emergency conditions.

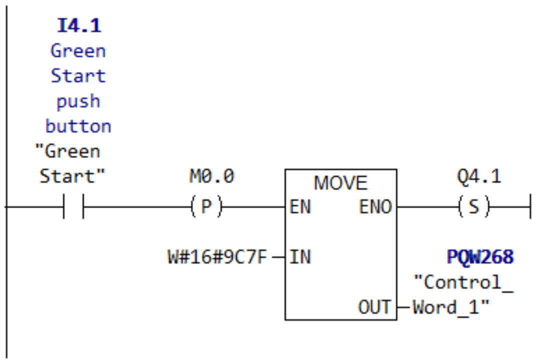
Network 1: Drive start

9C00 HEX TO STOP/OFF



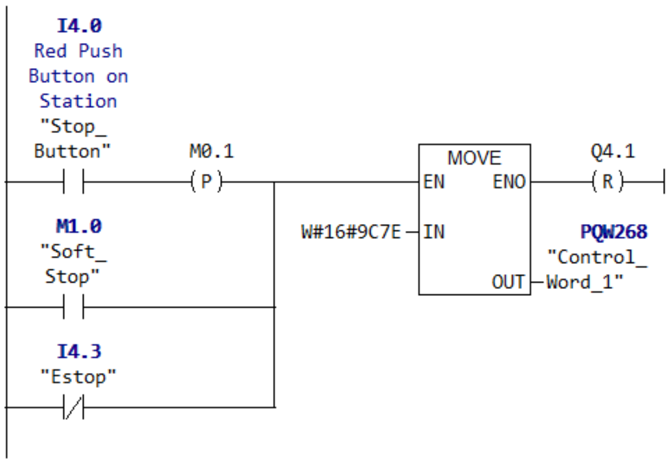
Network 2: Drive Start

Pushing the Green push button on the control box puts the drive in to the start state. The drive can also be started and stopped via the HMI.



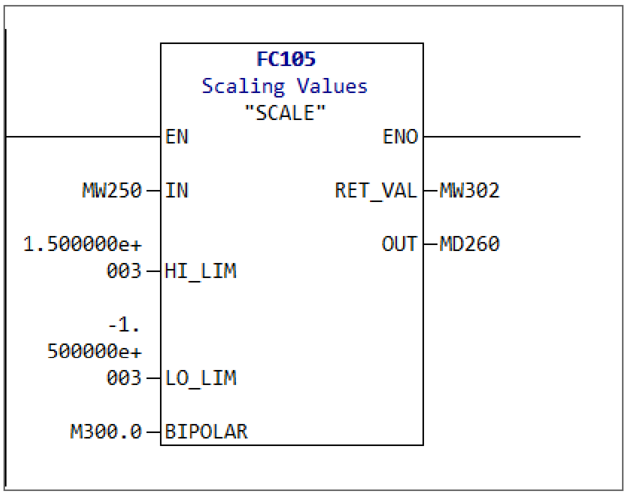
Network 3: Drive Stop

Pushing the Red push button on the control (found on black control box), will put the drive in an innert state. Caution should be taken however becasue the drive is still active. The drive can also be stopped via the HMI.



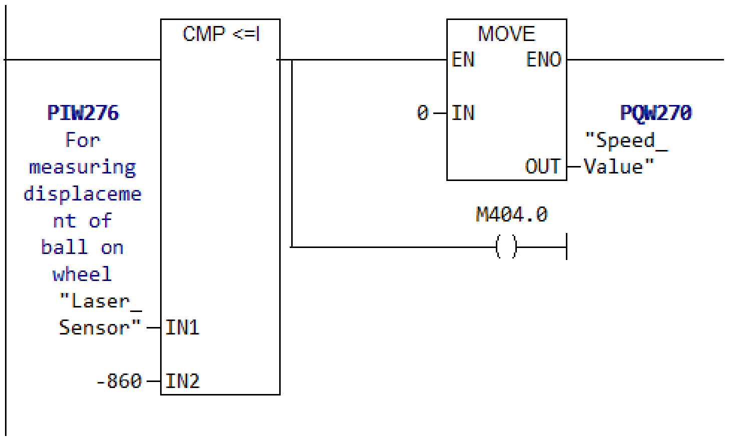
Network 6 : Title:

Comment:



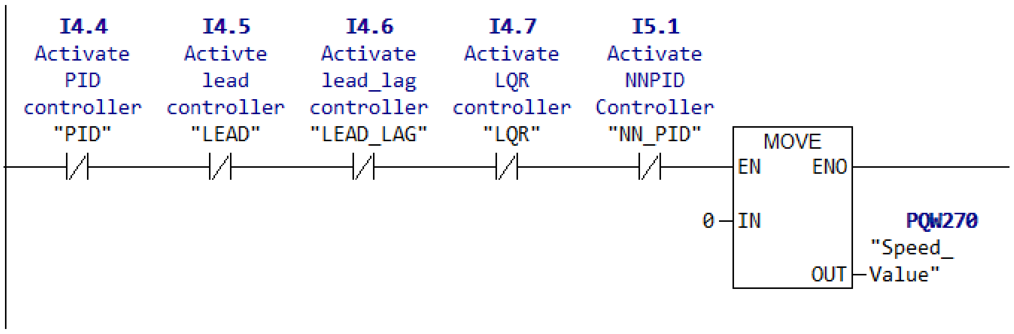
Network 7 : NO BALL ON THE WHEEL (PROTECTION)

IF THE SYSTEM EXCEEDS THE DEFINED LIMITS OR NO BALL IS ON THE WHEEL THE SYSTEM MUST STOP AUTOMATICALLY. M500.0 IS DUMMY BIT



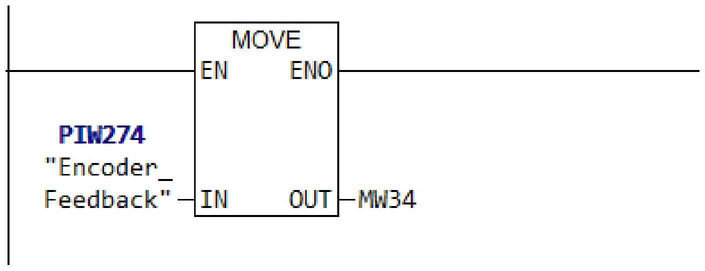
Network 8 : Title:

If no control algorithm is selected, the system must be in stationary.



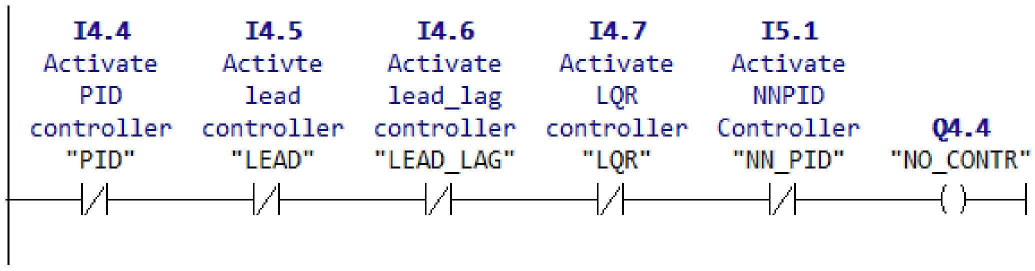
Network 9 : Motor Actual Speed.

Feedback from drive



Network 10 : No Control Algorithm Selected

No control algorithm selected - turns on LED "NO_CONTR"

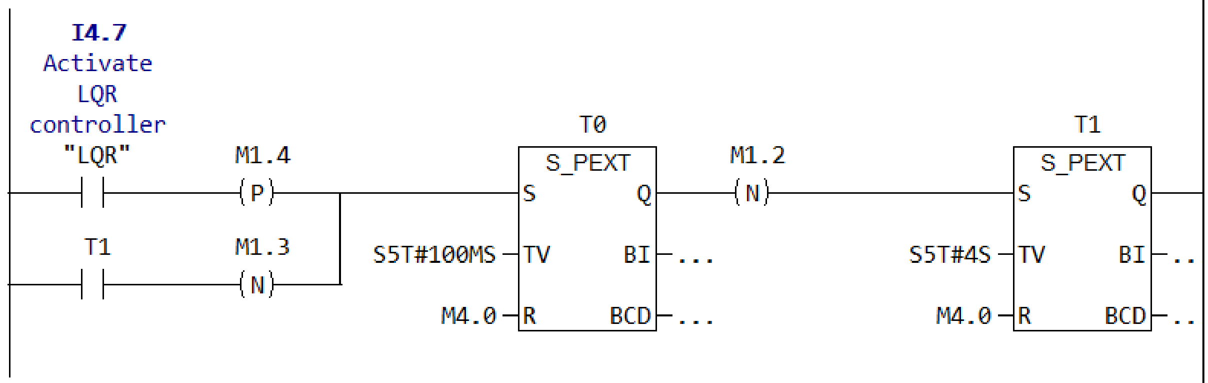


FB5 : BOW_DISTURBANCE_GENERATOR

```
This FB is used to indroduce automated disturbances to the system in
order to test the ability of the implemented controller. Disturbance signals
include:
1) Pulsed signal


Disturbances can only be introduced via the HMI
```

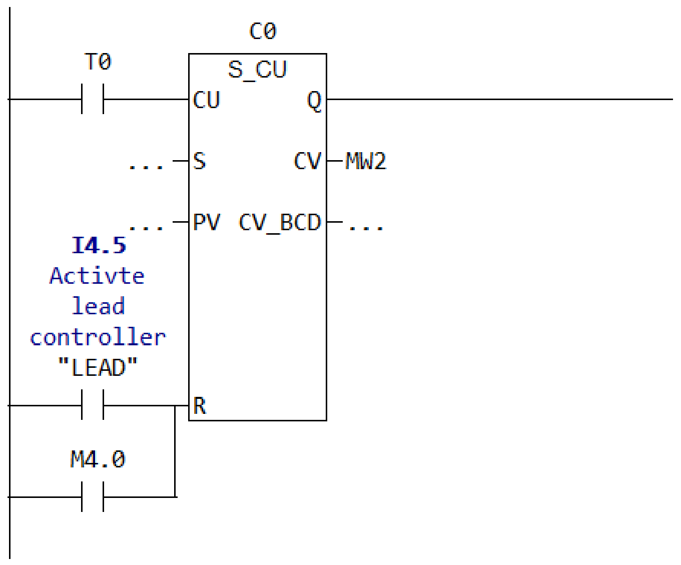
Network 1: Title:

```
User can select a predetermined pule length for the pulses.
```



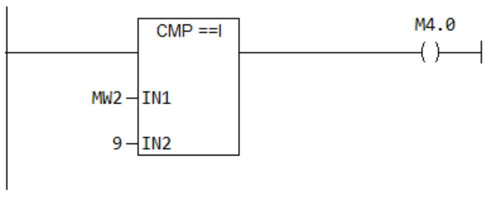
Network 2: Title:

```
User can select how many disturbance pulses he/ she wants via the HMI.
```



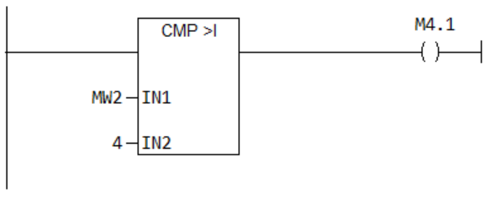
Network 3 : Disturbance must stop normal operation

Comment:



Network 4 : Disturbance must stop normal operation

Comment:

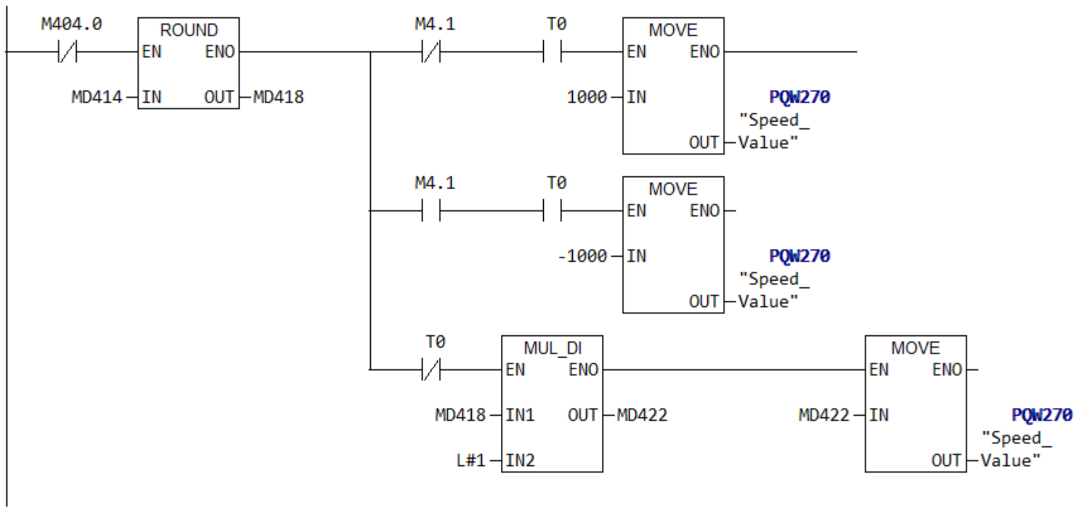


Network 5 : SPEED COMMAND TO DRIVE

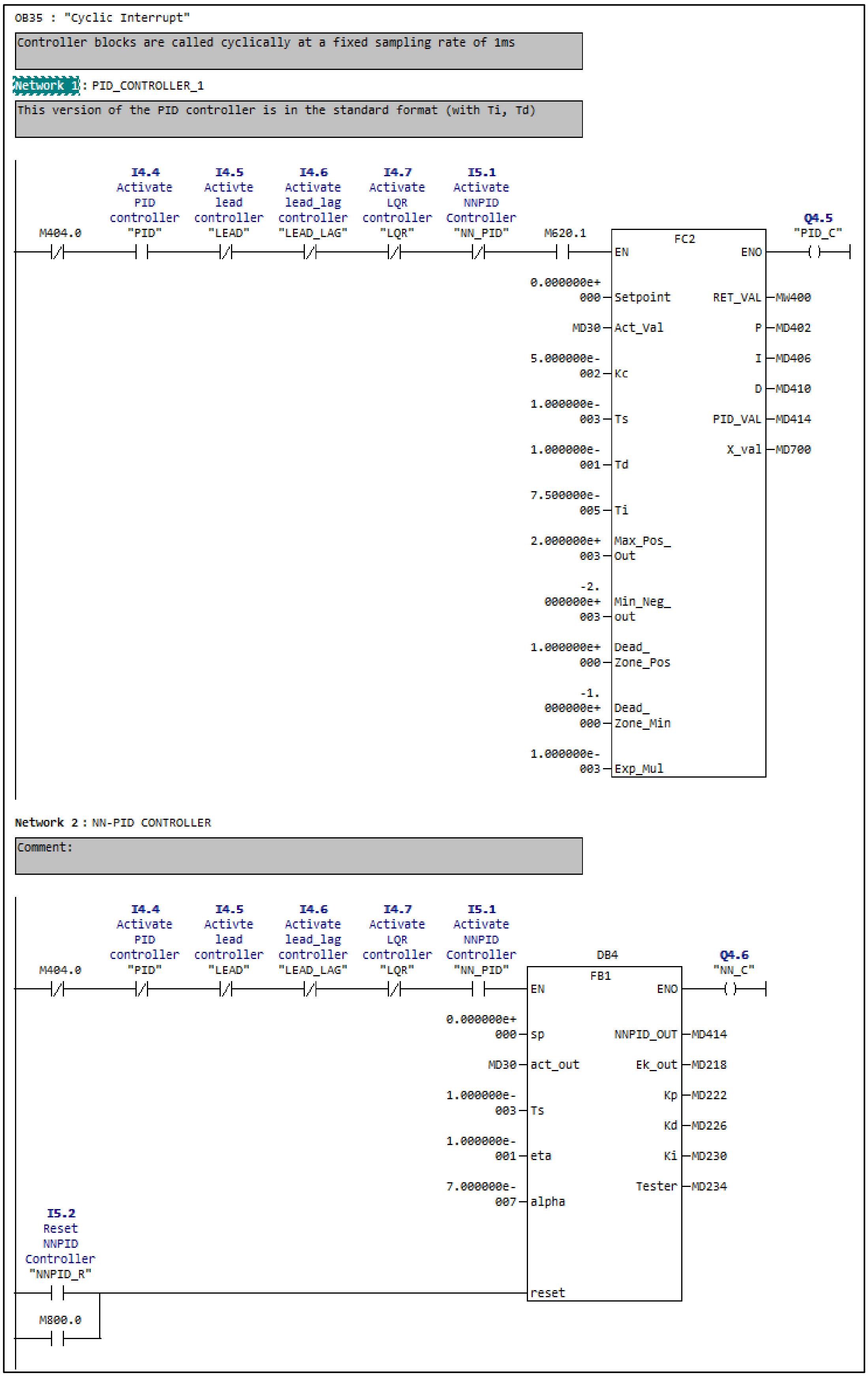
PUT BACK MD414 FOR CUSTOM MADE PID RUNNING IN OB35

MD250 - PID output from Matlab SIMULINK

M500.0 = DUMMY BIT



Cyclic Interrupt (OB35) – calling control algorithms


OB35 : "Cyclic Interrupt"

Controller blocks are called cyclically at a fixed sampling rate of 1ms

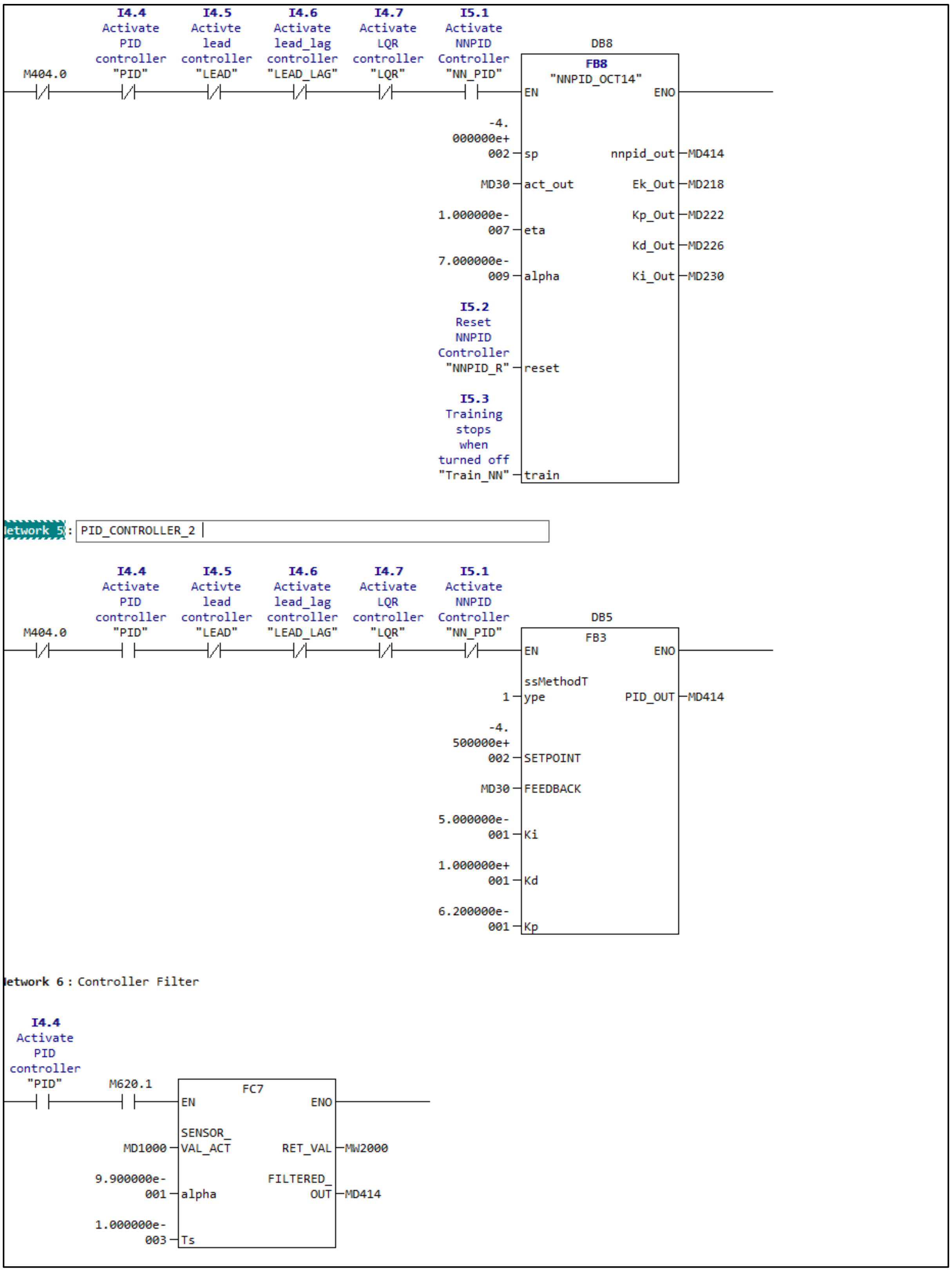
Network 1 : PID_CONTROLLER_1

This version of the PID controller is in the standard format (with Ti, Td)

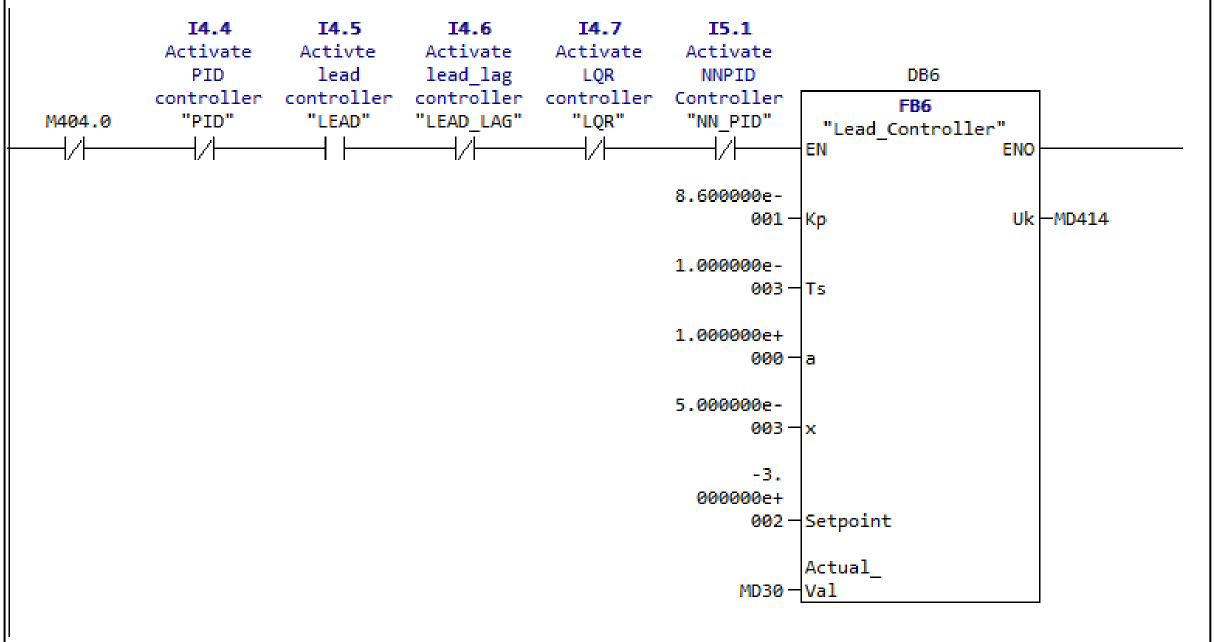
Network 2 : NN-PID CONTROLLER

Comment:

M404.0
I4.4 Activate PID controller "PID"
I4.5 Activte lead controller "LEAD"
I4.6 Activate lead_lag controller "LEAD_LAG"
I4.7 Activate LQR controller "LQR"
I5.1 Activate NNPID Controller "NN_PID"
M620.1
FC2
EN
ENO
Q4.5 "PID_C"
0.000000e+000 Setpoint
MD30 Act_Val
5.000000e-002 Kc
1.000000e-003 Ts
1.000000e-001 Td
7.500000e-005 Ti
2.000000e+003 Max_Pos_Out
-2.000000e+003 Min_Neg_out
1.000000e+000 Dead_Zone_Pos
-1.000000e+000 Dead_Zone_Min
1.000000e-003 Exp_Mul
RET_VAL MW400
P MD402
I MD406
D MD410
PID_VAL MD414
X_val MD700
M404.0
I4.4 Activate PID controller "PID"
I4.5 Activte lead controller "LEAD"
I4.6 Activate lead_lag controller "LEAD_LAG"
I4.7 Activate LQR controller "LQR"
I5.1 Activate NNPID Controller "NN_PID"
DB4
FB1
EN
ENO
Q4.6 "NN_C"
0.000000e+000 sp
MD30 act_out
1.000000e-003 Ts
1.000000e-001 eta
7.000000e-007 alpha
NNPID_OUT MD414
Ek_out MD218
Kp MD222
Kd MD226
Ki MD230
Tester MD234
I5.2 Reset NNPID Controller "NNPID_R"
M800.0
reset

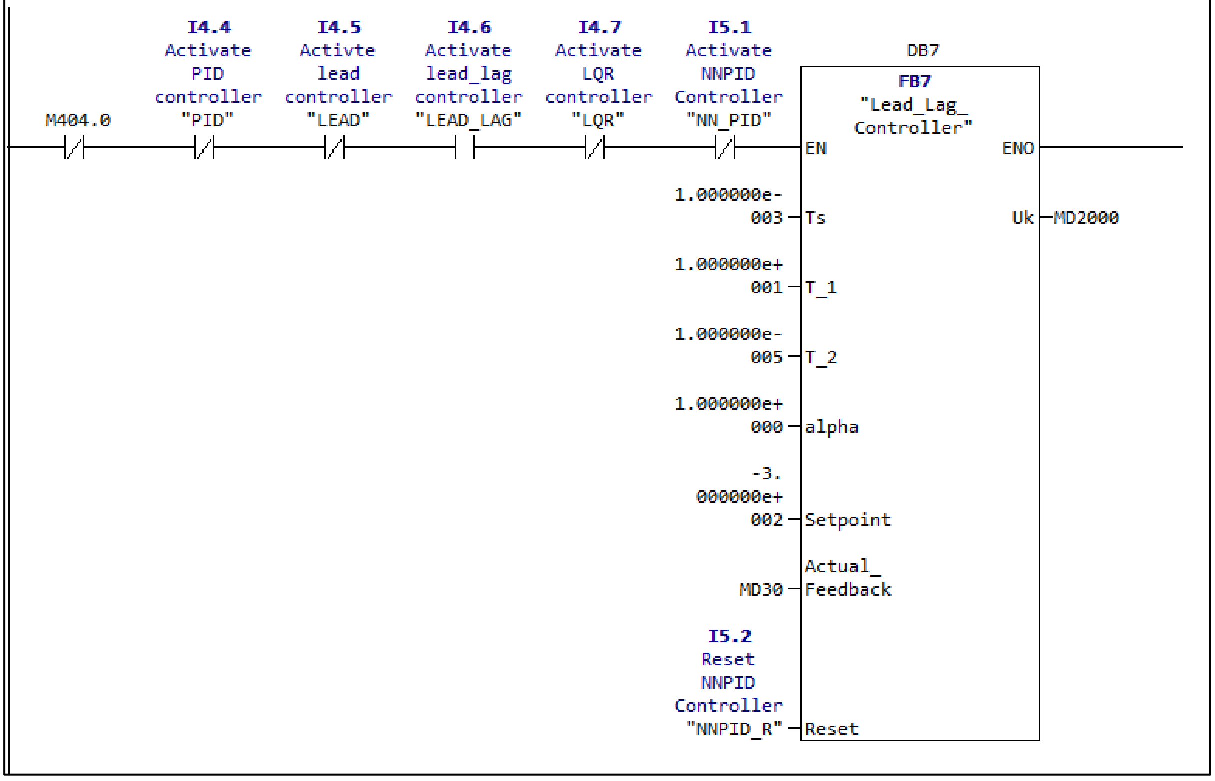

I4.4 Activate PID controller "PID"
I4.5 Activte lead controller "LEAD"
I4.6 Activate lead_lag controller "LEAD_LAG"
I4.7 Activate LQR controller "LQR"
I5.1 Activate NNPID Controller "NN_PID"
M404.0
DB8
FB8 "NNPID_OCT14"
EN
ENO
-4.000000e+002 sp
MD30 act_out
1.000000e-007 eta
7.000000e-009 alpha
nnpid_out MD414
Ek_Out MD218
Kp_Out MD222
Kd_Out MD226
Ki_Out MD230
I5.2 Reset NNPID Controller "NNPID_R" reset
I5.3 Training stops when turned off "Train_NN" train
letwork 5: PID_CONTROLLER_2
I4.4 Activate PID controller "PID"
I4.5 Activte lead controller "LEAD"
I4.6 Activate lead_lag controller "LEAD_LAG"
I4.7 Activate LQR controller "LQR"
I5.1 Activate NNPID Controller "NN_PID"
M404.0
DB5
FB3
EN
ENO
1 ssMethodType
PID_OUT MD414
-4.500000e+002 SETPOINT
MD30 FEEDBACK
5.000000e-001 Ki
1.000000e+001 Kd
6.200000e-001 Kp
letwork 6 : Controller Filter
I4.4 Activate PID controller "PID"
M620.1
FC7
EN
ENO
MD1000 SENSOR_VAL_ACT
RET_VAL MW2000
9.900000e-001 alpha
FILTERED_OUT MD414
1.000000e-003 Ts

Network 7 : Lead Controller



M404.0
I4.4 Activate PID controller "PID"
I4.5 Activtve lead controller "LEAD"
I4.6 Activate lead_lag controller "LEAD_LAG"
I4.7 Activate LQR controller "LQR"
I5.1 Activate NNPID Controller "NN_PID"
DB6
FB6
"Lead_Controller"
EN
ENO
8.600000e-001 Kp
Uk MD414
1.000000e-003 Ts
1.000000e+000 a
5.000000e-003 x
-3.000000e+002 Setpoint
MD30 Actual_Val

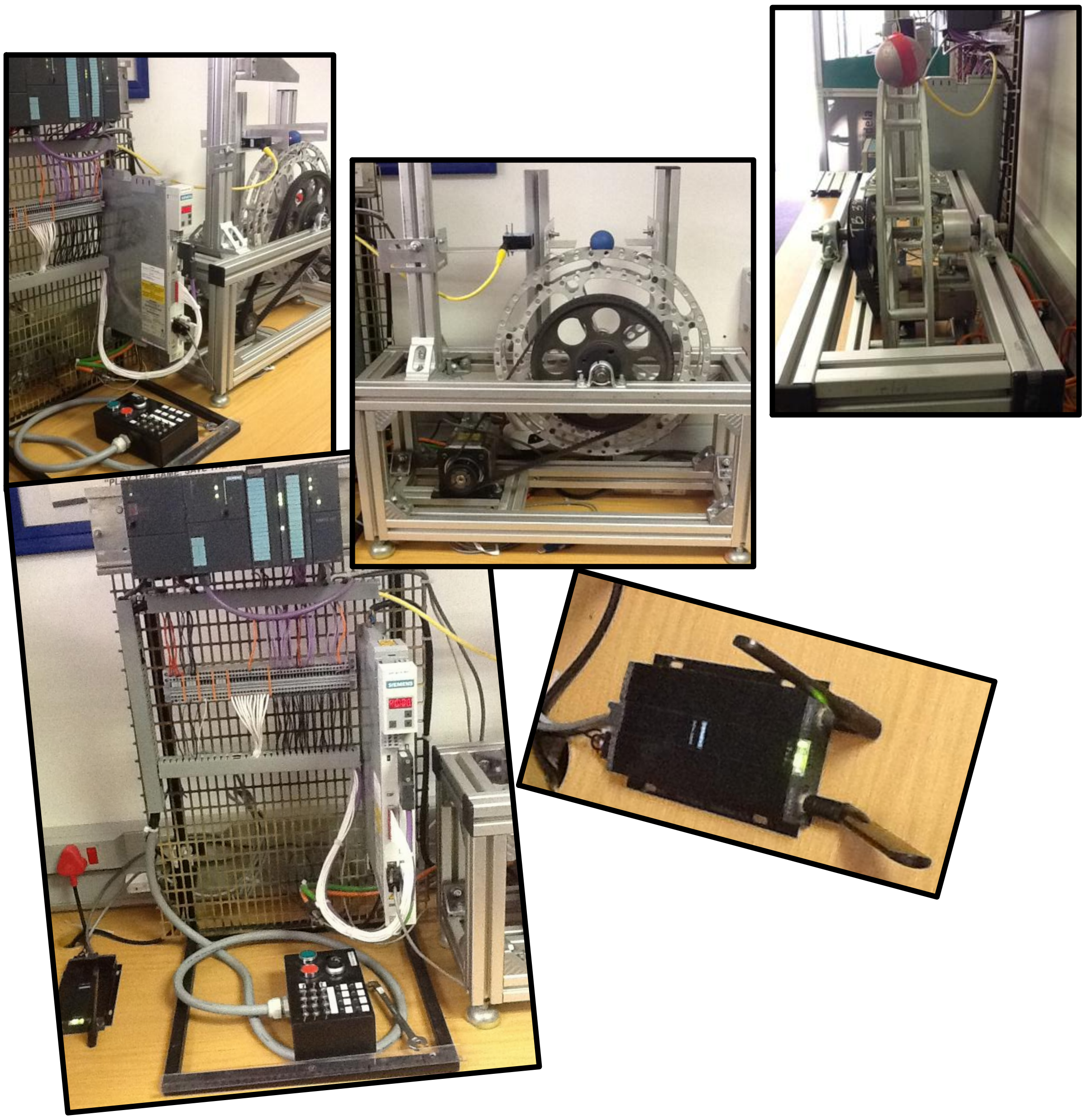
Network 8 : Lead Lag Compensator



M404.0
I4.4 Activate PID controller "PID"
I4.5 Activtve lead controller "LEAD"
I4.6 Activate lead_lag controller "LEAD_LAG"
I4.7 Activate LQR controller "LQR"
I5.1 Activate NNPID Controller "NN_PID"
DB7
FB7
"Lead_Lag_Controller"
EN
ENO
1.000000e-003 Ts
Uk MD2000
1.000000e+001 T_1
1.000000e-005 T_2
1.000000e+000 alpha
-3.000000e+002 Setpoint
MD30 Actual_Feedback
I5.2 Reset NNPID Controller "NNPID_R" Reset

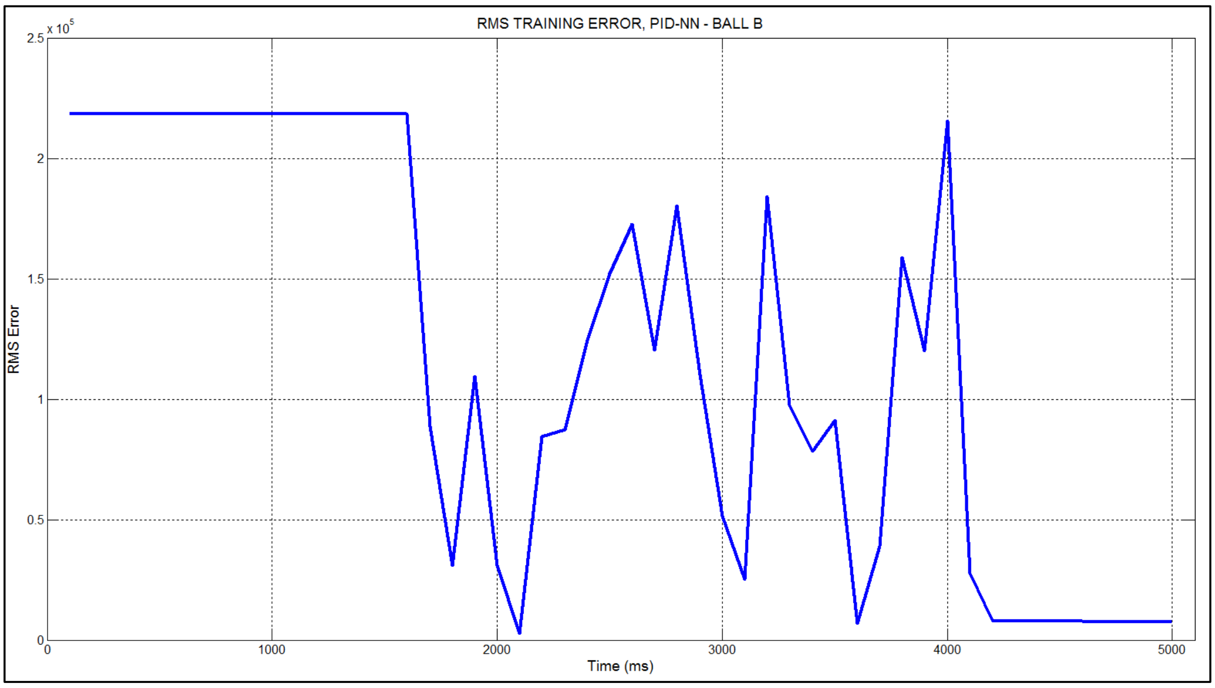
APPENDIX D: IMAGES OF BOW SYSTEM

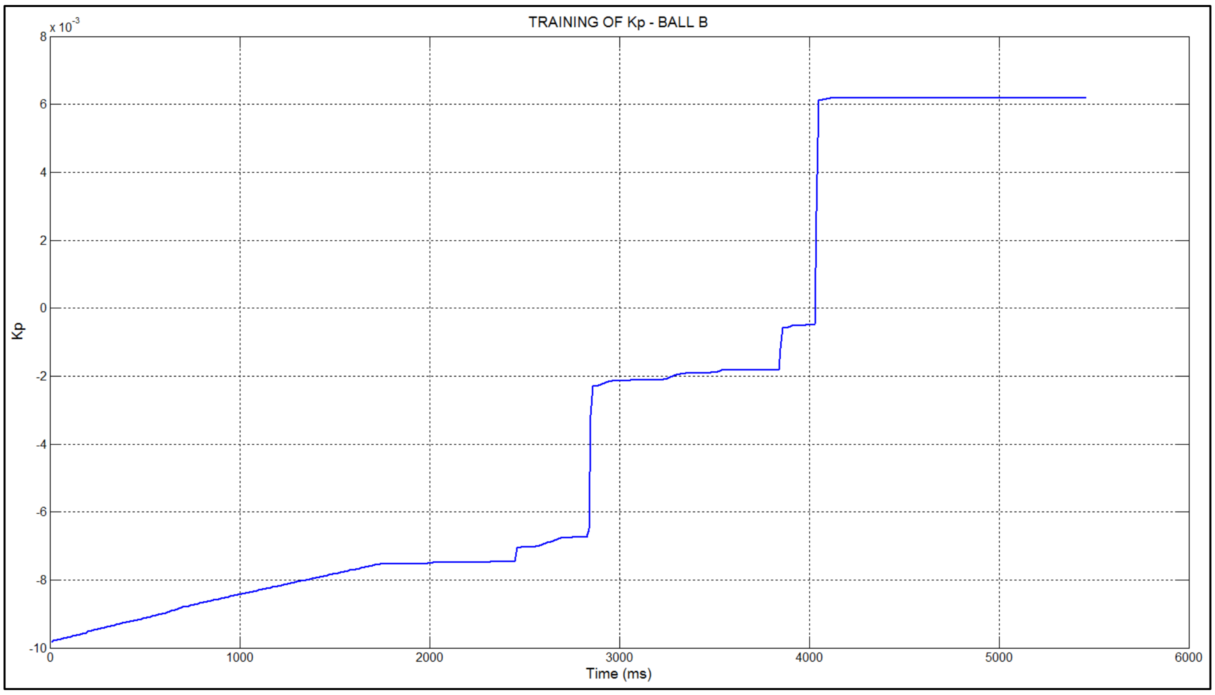
Various views of Ball on Wheel plant

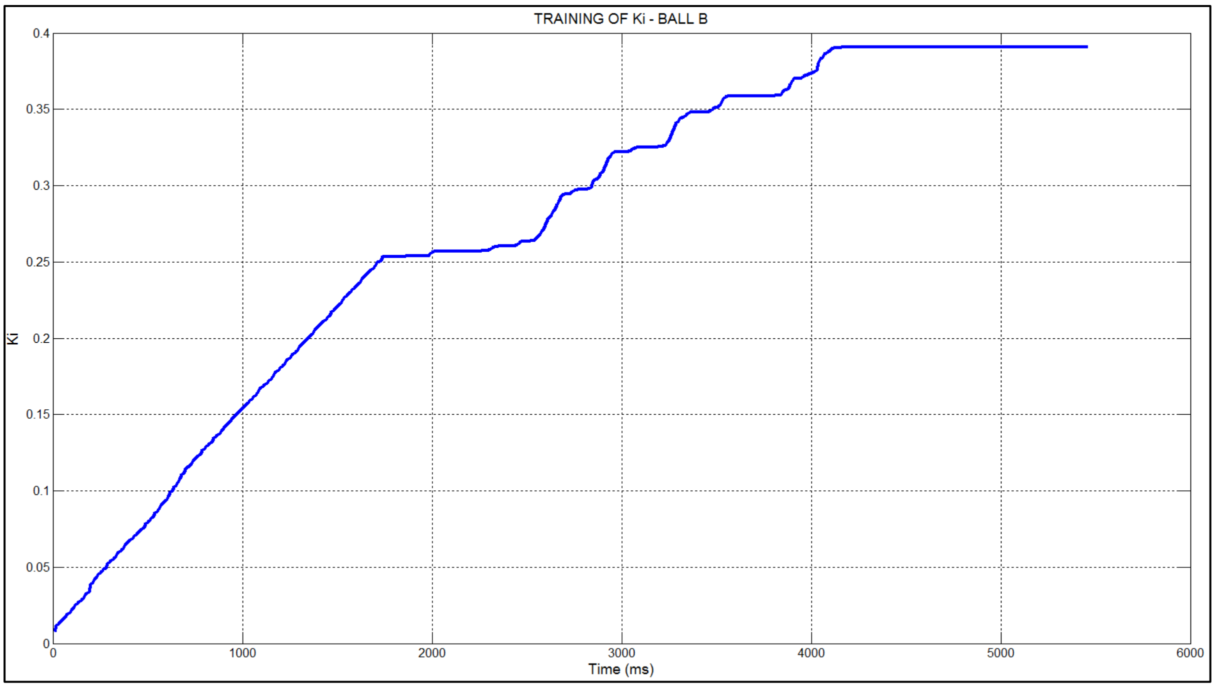


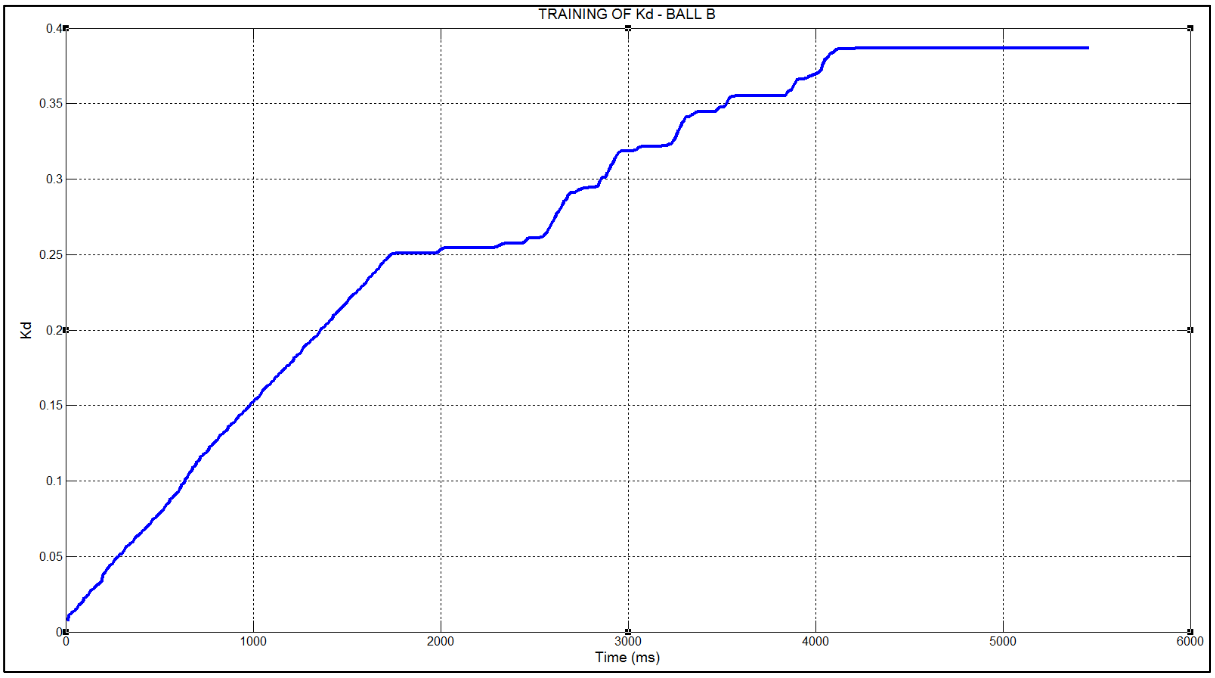
APPENDIX E: NN-PID TRAINING PLOTS

Ball B training plots

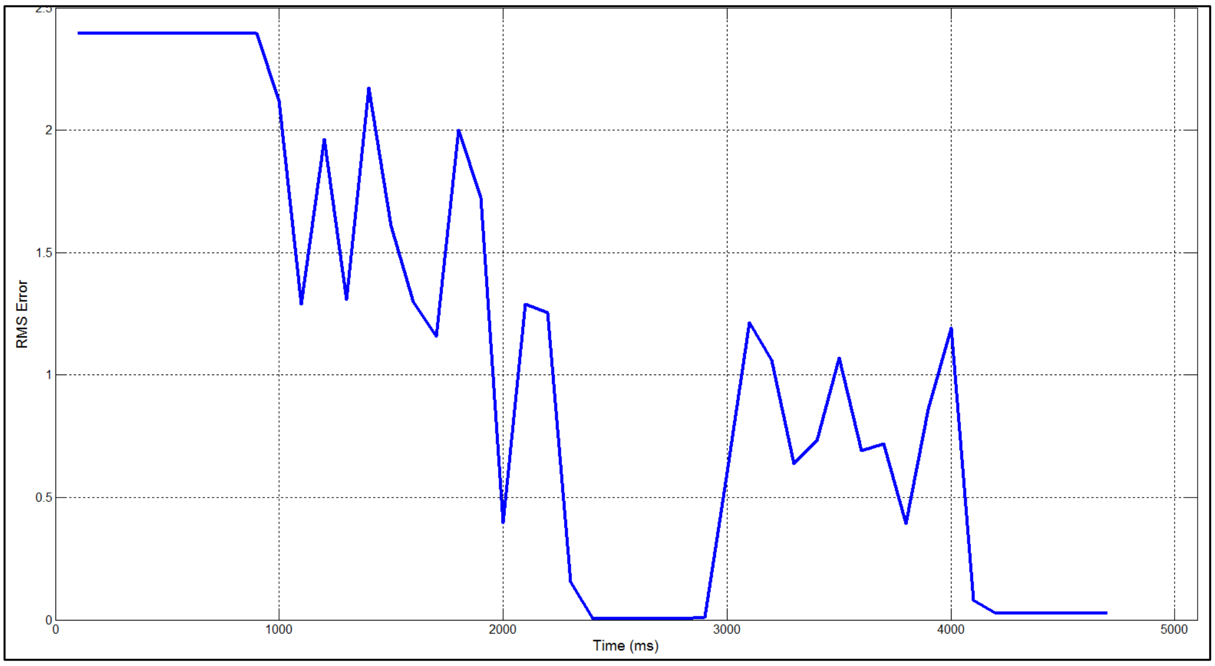


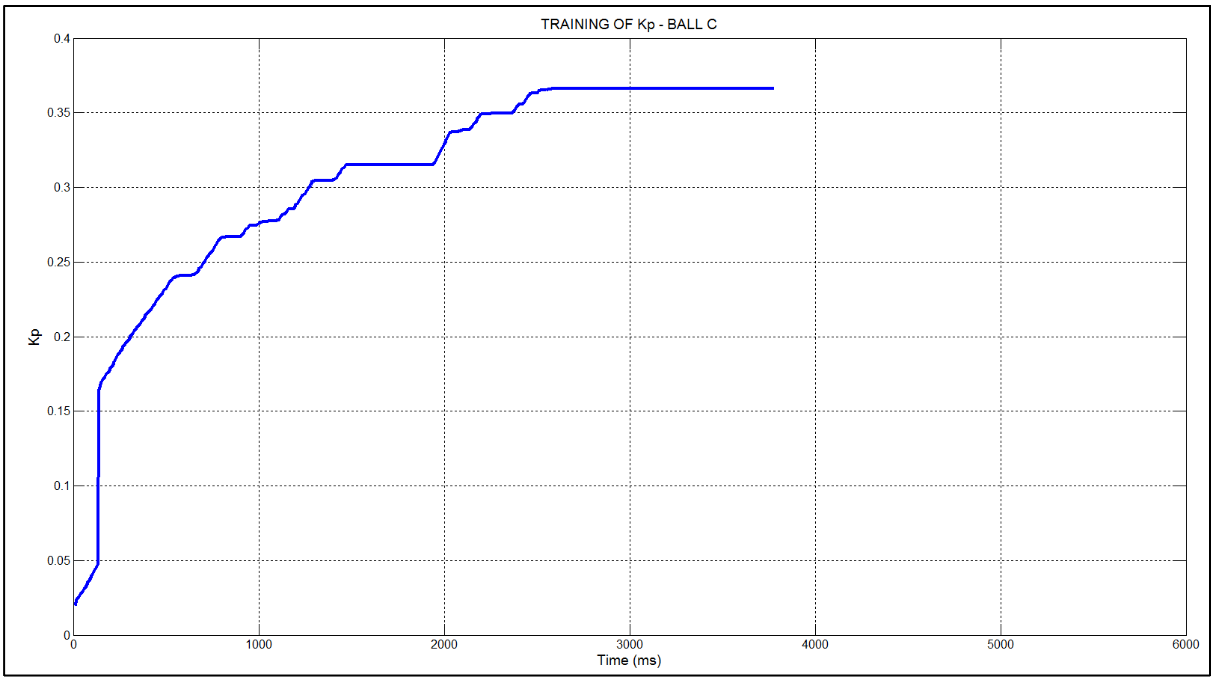


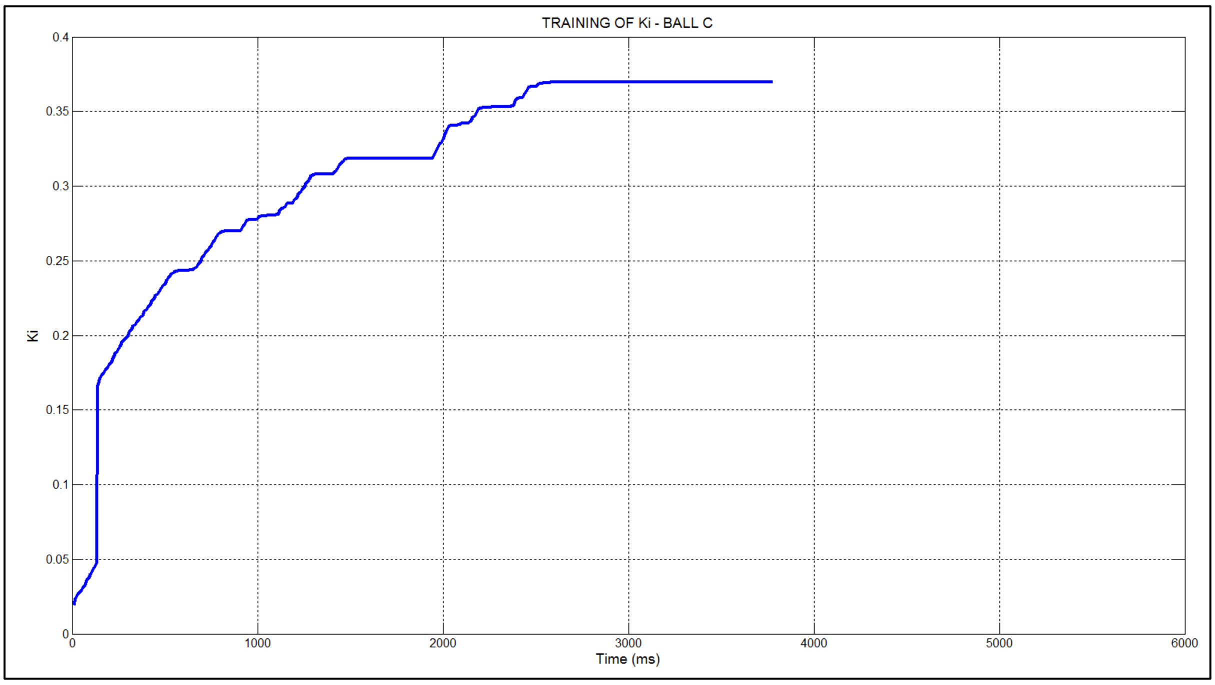

TRAINING OF Ki - BALL B
Ki
Time (ms)
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0
1000
2000
3000
4000
5000
6000

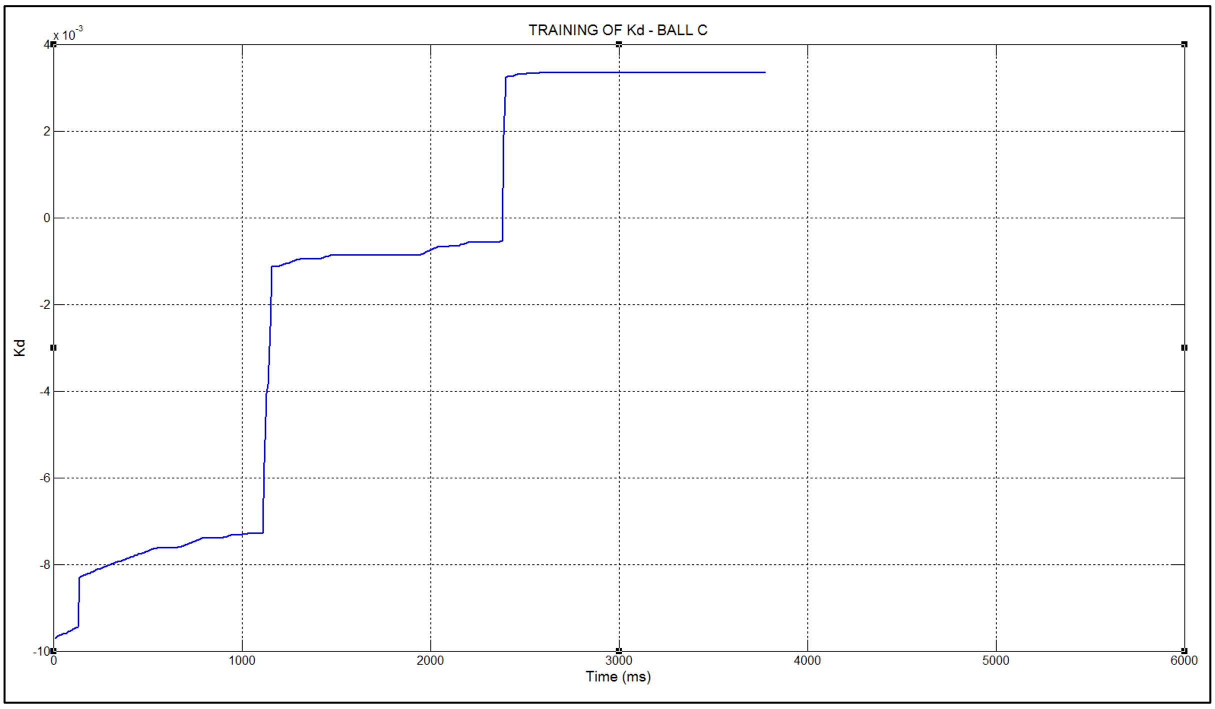

TRAINING OF Kd - BALL B
Kd
Time (ms)
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0
1000
2000
3000
4000
5000
6000

Ball C training plots

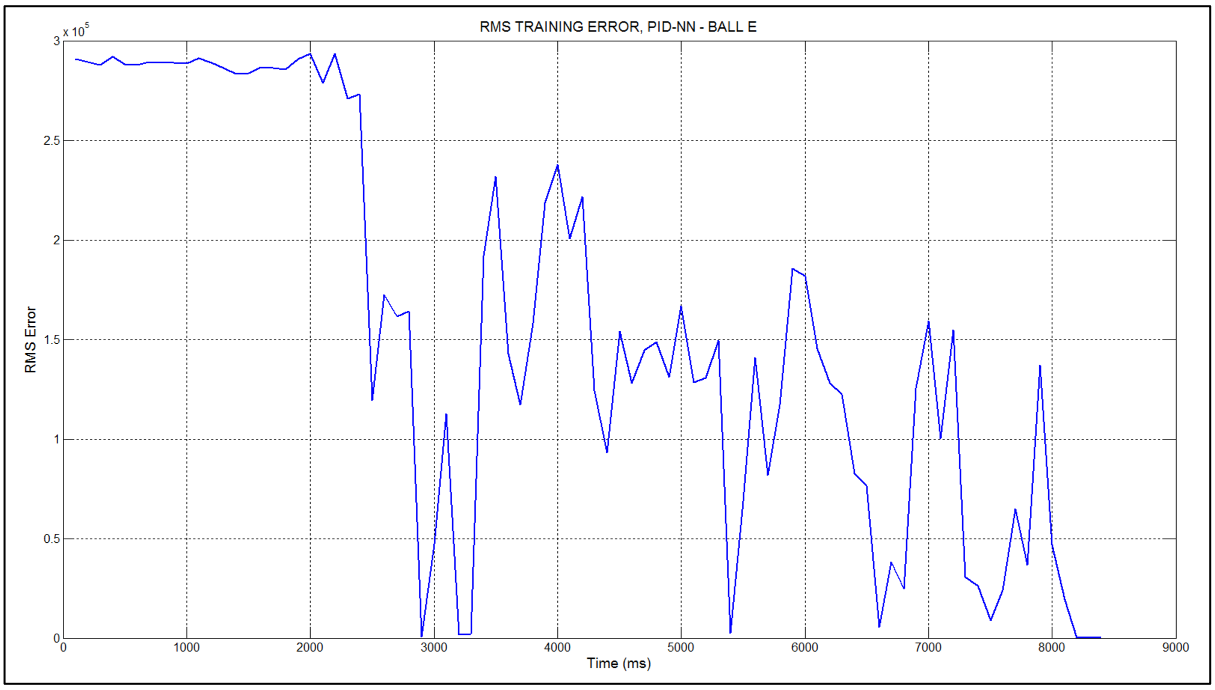


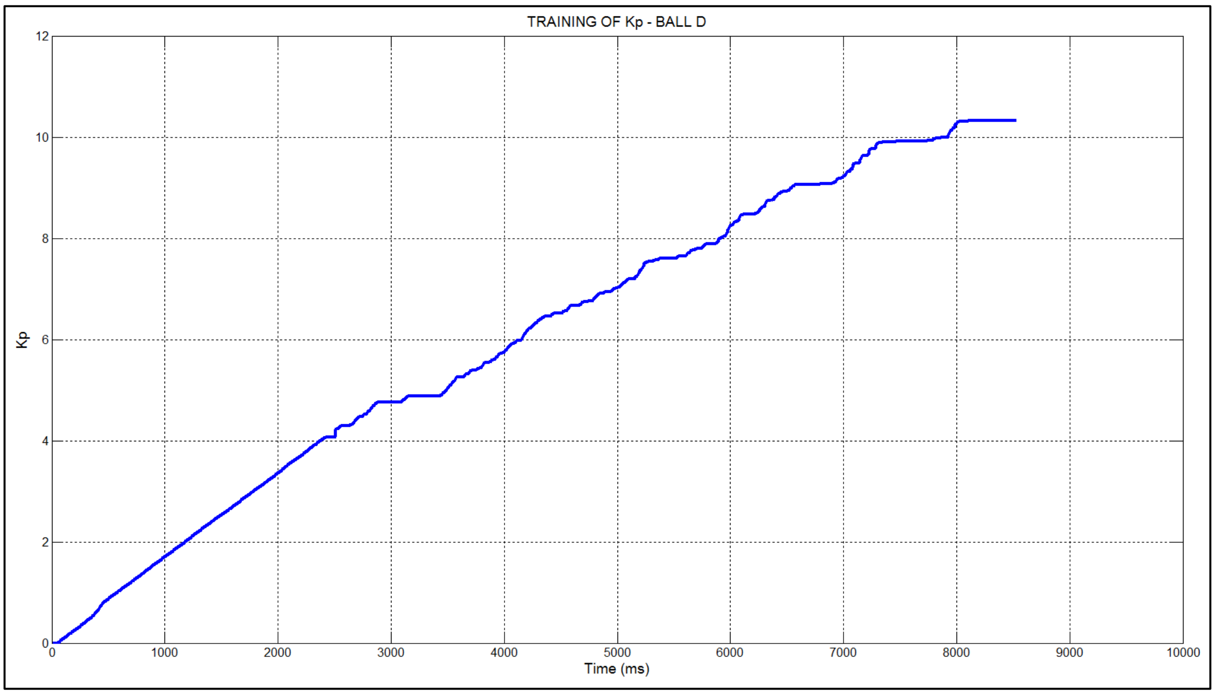


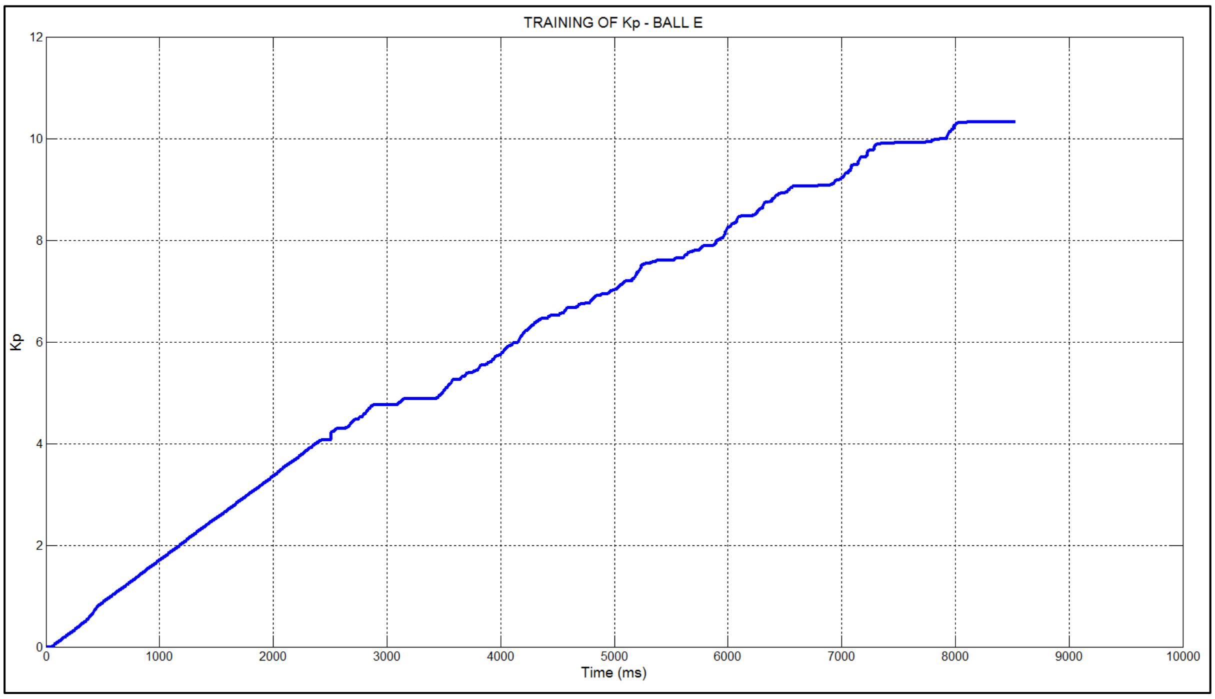

TRAINING OF Ki - BALL C
Ki
Time (ms)
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0
1000
2000
3000
4000
5000
6000


TRAINING OF Kd - BALL C
x 10^-3
Kd
Time (ms)
4
2
0
-2
-4
-6
-8
-10
0
1000
2000
3000
4000
5000
6000

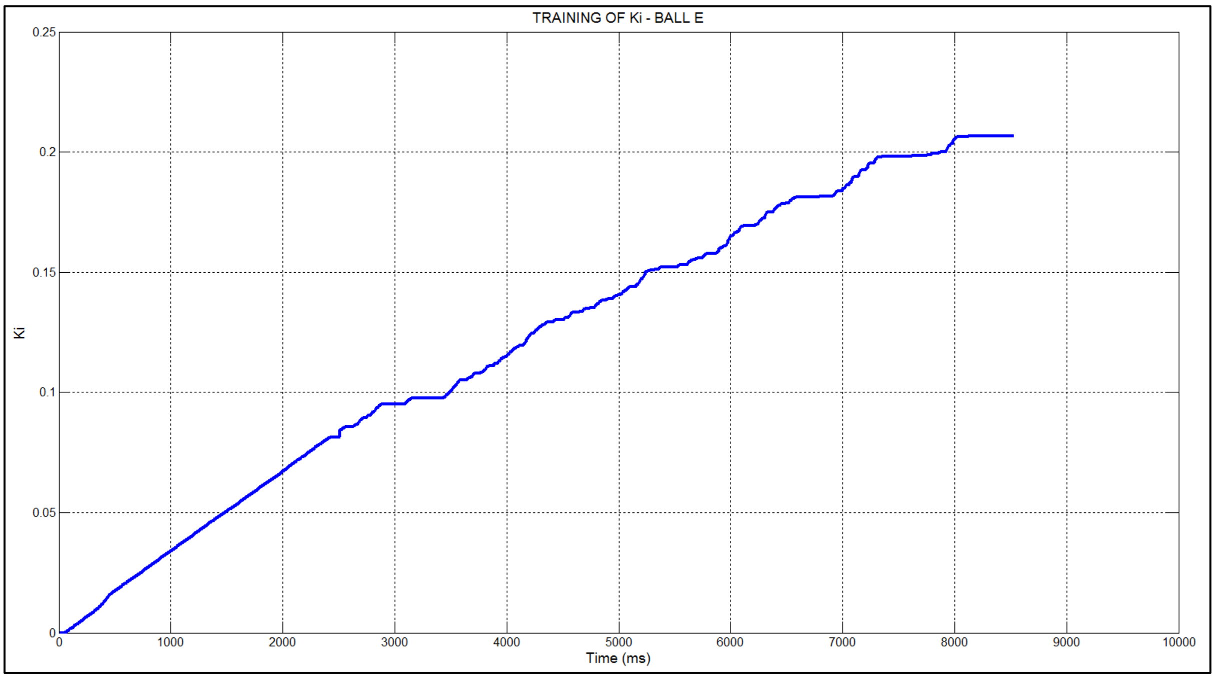
Ball E training plots



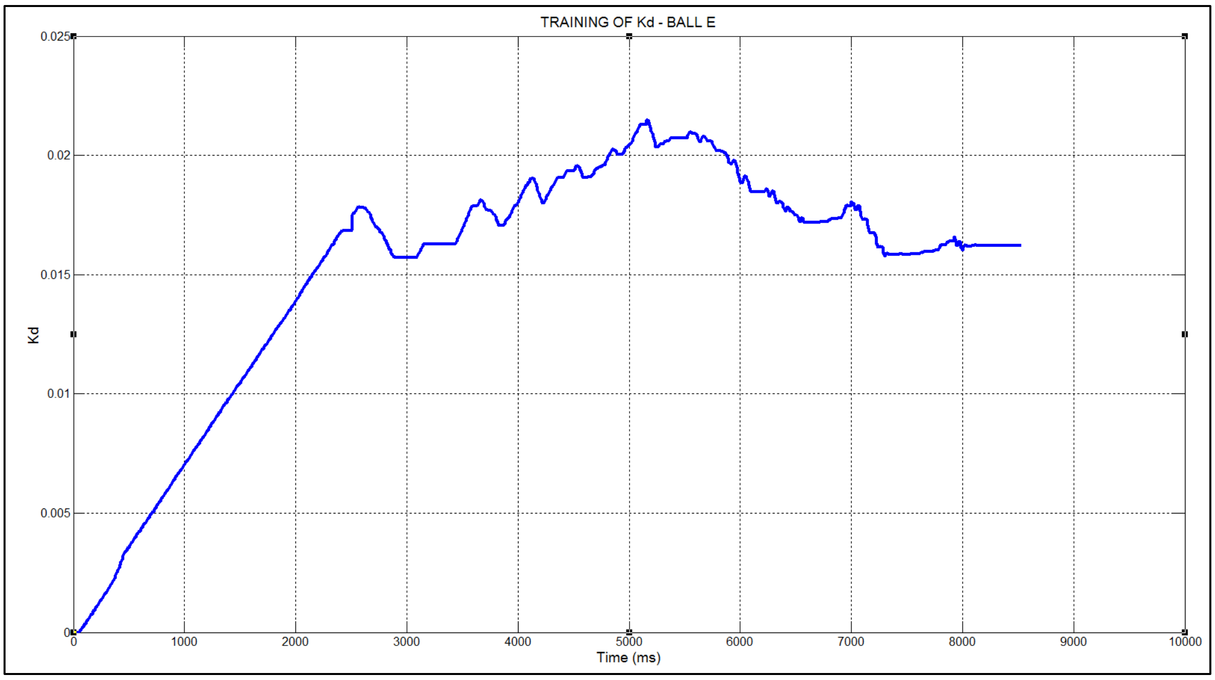




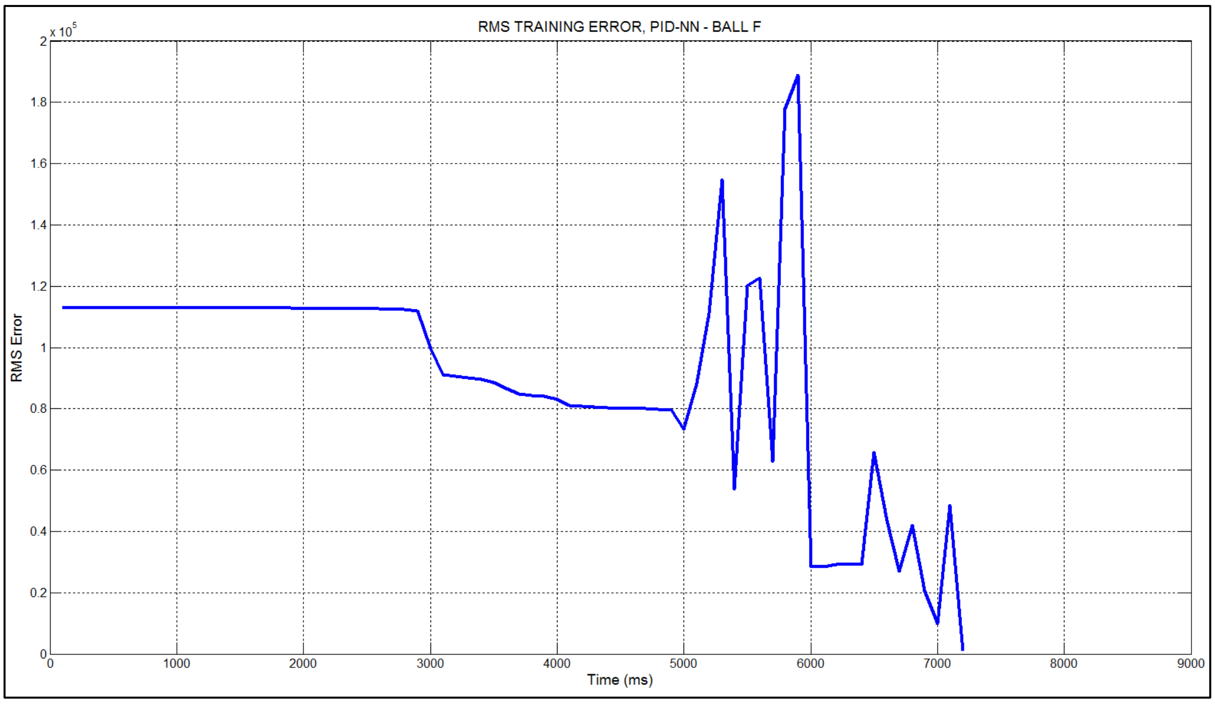
TRAINING OF Kp - BALL E
Kp
Time (ms)
0
2
4
6
8
10
12
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000

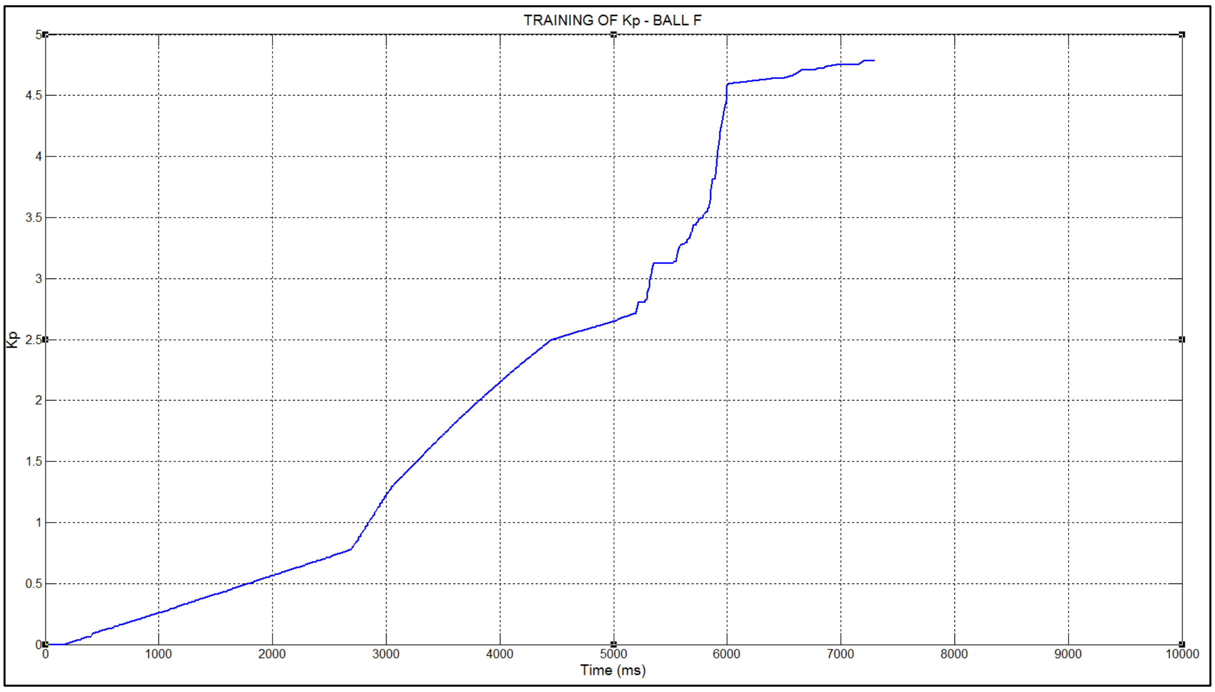


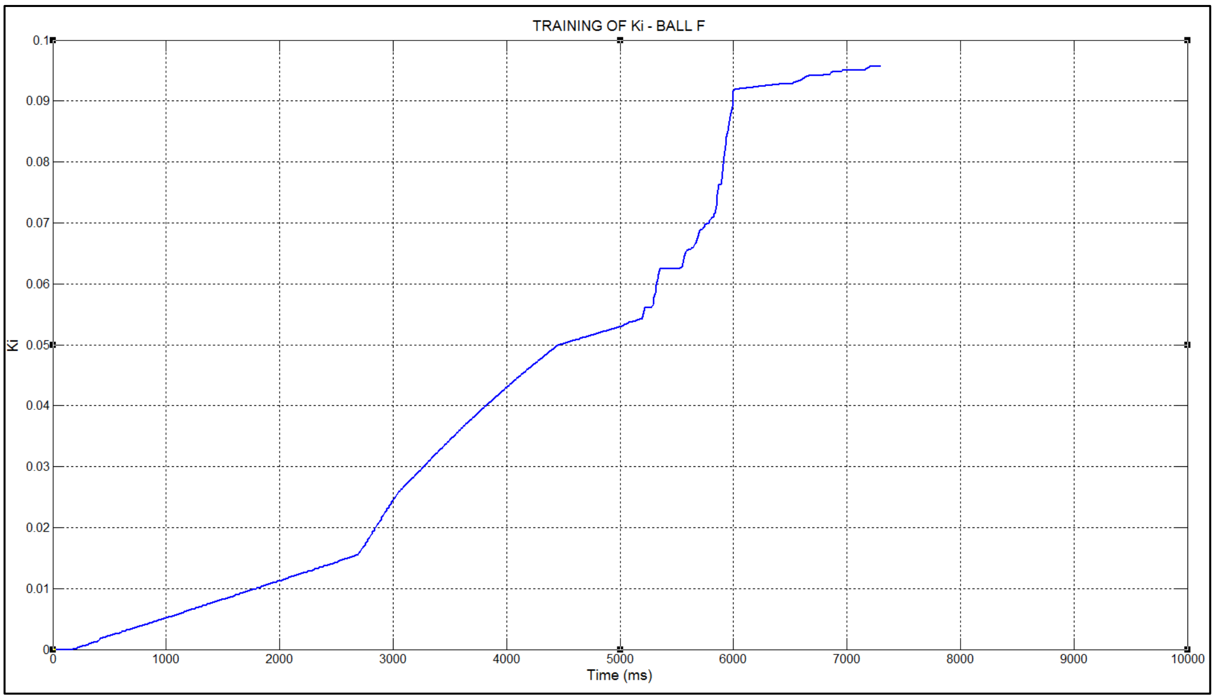
TRAINING OF Ki - BALL E
Ki
Time (ms)
0
0.05
0.1
0.15
0.2
0.25
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000

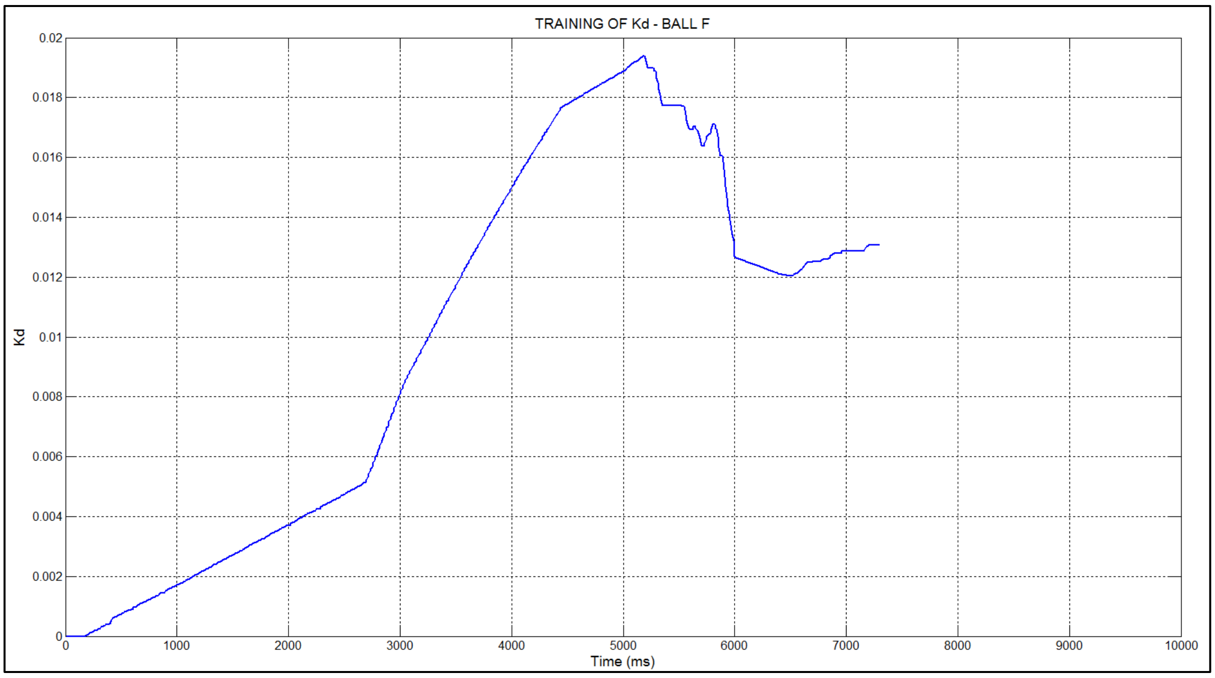


Ball F training plots

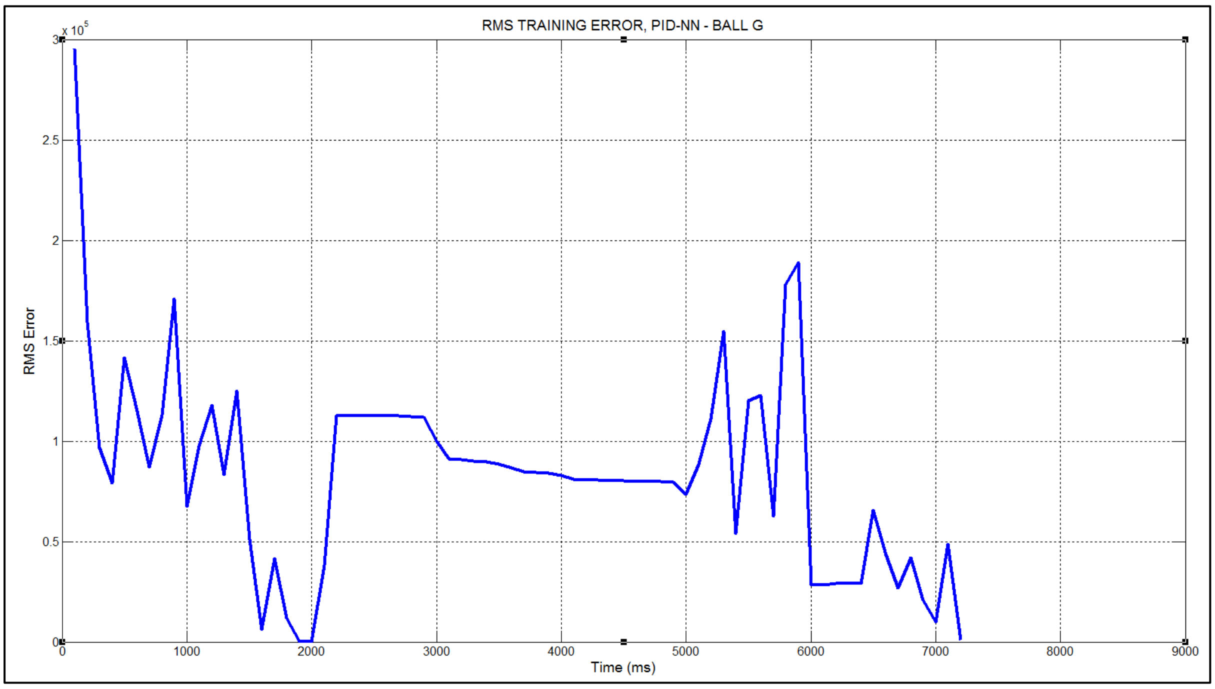


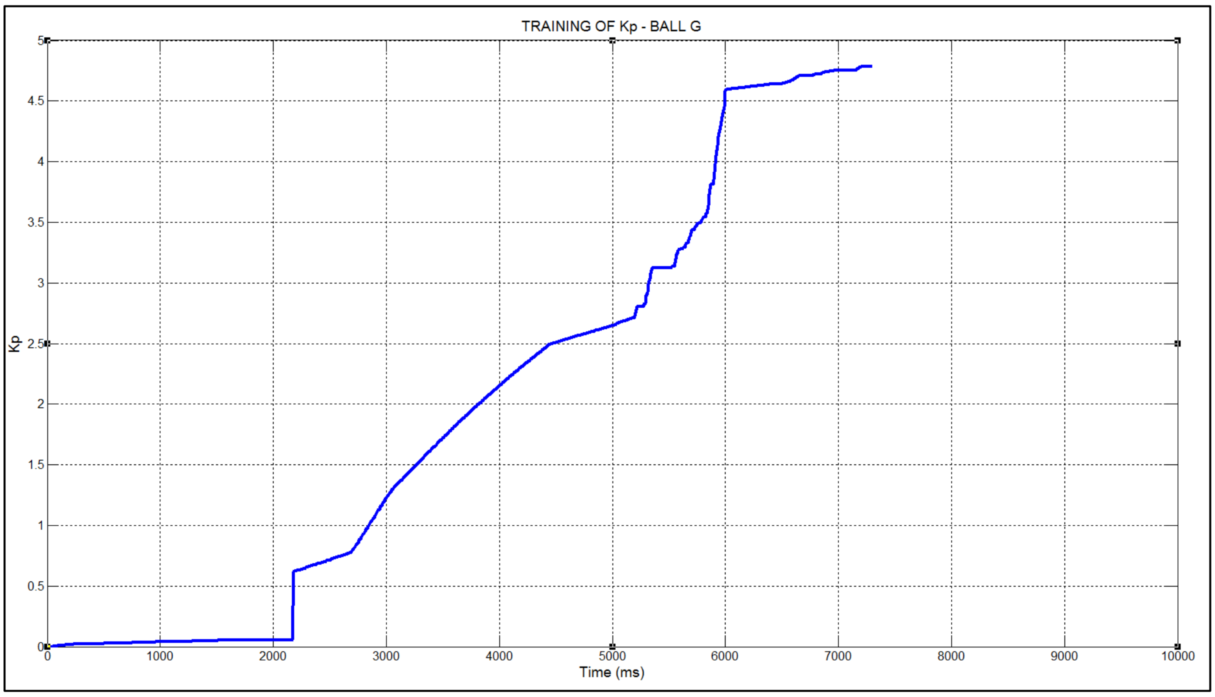

TRAINING OF Kp - BALL F
Kp
Time (ms)
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000

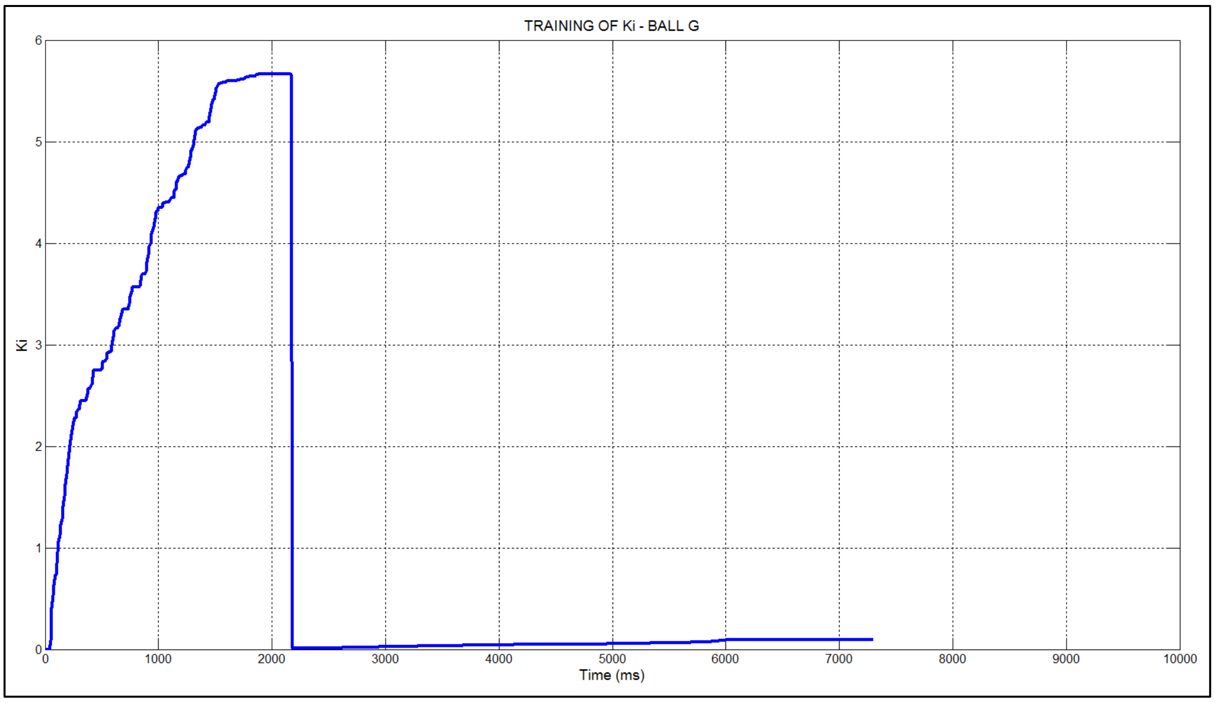

TRAINING OF Ki - BALL F
Ki
Time (ms)
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.1
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000

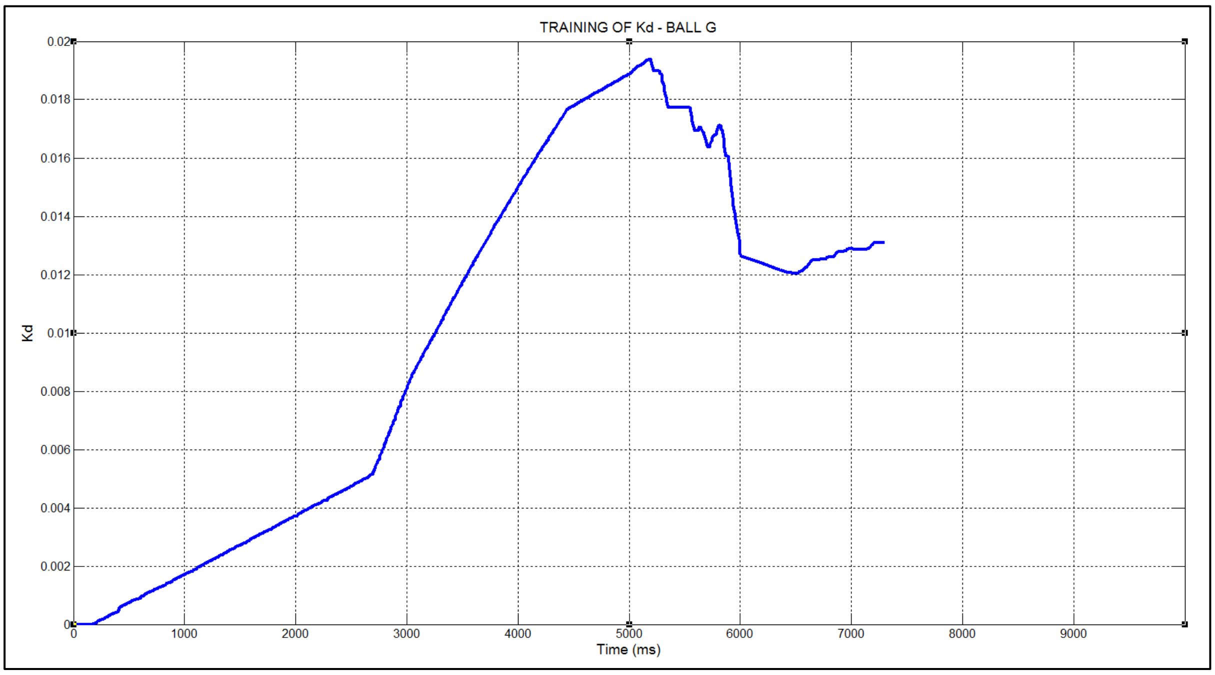


Ball G training plots




TRAINING OF Kp - BALL G
Kp
Time (ms)
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000


TRAINING OF Ki - BALL G
Ki
Time (ms)
0
1
2
3
4
5
6
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000


TRAINING OF Kd - BALL G
Kd
Time (ms)
0.02
0.018
0.016
0.014
0.012
0.01
0.008
0.006
0.004
0.002
0
0
1000
2000
3000
4000
5000
6000
7000
8000
9000