ROBUST NON-LINEAR LYAPUNOV DEEP LEARNING CONTROL DESIGN FOR CHAOTIC SYSTEMS

by

AMR SALAH MAHMOUD

A dissertation submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY IN ELECTRICAL AND COMPUTER ENGINEERING

2022

Oakland University
Rochester, Michigan

Doctoral Advisory Committee:

Mohamed A. Zohdy, Ph.D., Chair
Brian Dean, Ph.D.
Darrell Schmidt, Ph.D.
Richard Olawoyin, Ph.D.

الحمدّ والشكر لك ربي لما وفقتني إليه، والحمد لك جلّ جلالك

I dedicate the final product of this dissertation to my parents who have supported and inspired me throughout this journey.

ACKNOWLEDGMENTS

I would like to express my special thanks and gratitude to my advisor and mentor Dr. Mohamed Zohdy (Coach) as for without him the work presented in this dissertation wouldn't have been possible without the mentoring, guidance, support, and encouragement. Besides my advisor I would like to express my deep and sincere gratitude to the members of the advisory committee Dr. Richard Olawoyin, Dr. Brian Dean and Dr. Darrell Schmid for their time and valuable input.

I would also like to thank my fellow lab mates and teaching assistant colleagues who through their discussions I was inspired to bring the ideas in this dissertation forth. Finally, I would like to thank my parents and sister for without their love, encouragement, support, and inspiration to begin and continue this journey this wouldn't have been possible.

Amr Salah Mahmoud

ABSTRACT

ROBUST NON-LINEAR LYAPUNOV DEEP LEARNING CONTROL DESIGN FOR CHAOTIC SYSTEMS

by

AMR SALAH MAHMOUD

Adviser: Mohamed Zohdy, Ph.D.

Despite their operational success, machine learning controllers lack theoretical guarantees in terms of system stability. In contrast, classic model-based controller design uses principled approaches such as Linear Quadratic Regulator (LQR) to synthesize stable controllers with verifiable proofs. In addition, deep learning controllers encounter feedback timing bottlenecks that increase exponentially with the system complexity. Deep learning is also dependent on the quality and diversity of the dataset to produce unbiased findings; therefore, the prediction of deep learning is not guaranteed. As a result, in this research, we develop and implement a guaranteed stability solution for safety critical and chaotic systems through the integration of Lyapunov Stability theory and deep machine learning. Three control methods are researched, leading to the development of the Deep Lyapunov-stable controller: the deep learning methodology, the Lyapunov control function, and controller parameters. In this research, we provide a generic method for synthesizing a Deep Lyapunov-stable control and a way to simultaneously confirm its stability. A unique Lyapunov control function is devised and shown to be effective in managing Duffing, Van der Pol, and Zohdy-Harb nonlinear

systems, but with restrictions on the system's oscillation frequency, initial conditions and disturbances. Subsequently, Dynamic Lyapunov Deep Learning is introduced to alleviate the Lyapunov control's shortcomings. Developing a deep learning architecture in combination with a customized Lyapunov control resolves the temporal delay and Lyapunov parameters calibration concern. Different datasets are also presented before establishing the one with the best accuracy. In addition to the dataset, the architecture of the deep learning model has a significant effect on the model's accuracy. A process for relearning is intended to accommodate the introduction of new system dynamics. Based on the correlation study, we also designed an optimization technique to improve the integration of the deep learning layer and controller layer. The proposed integration of Deep Learning and Lyapunov Control, referred to as Lyapunov Deep Learning (LDL) control, is applied in MATLAB / SIMULINK to the magnetic levitation chaotic non-linear system to demonstrate its effectiveness in addressing sudden changes in system behavior, the environment, and demands in comparison to other methods of control.

TABLE OF CONTENTS

ACKNOWLEDGMENTS iv

ABSTRACT v

LIST OF TABLES xi

LIST OF FIGURES xii

CHAPTER ONE
INTRODUCTION 1

1.1 Fundamentals of Complex Non-Linear System 1

1.1.1 Duffing System 3

1.1.2 Van der Pol System 3

1.1.3 Navier-Stokes Equations in Fluid Dynamics 4

1.1.4 Lotka-Volterra Equations 5

1.1.5 Lorenz Chaotic Systems 5

1.2 An Overview of Nonlinear Control 6

1.2.1 Gain Scheduling 8

1.2.2 Adaptive Control 8

1.2.3 Model Predictive Control 8

1.3 Lyapunov Control Function 11

1.4 Thesis Chapter Outline 12

CHAPTER TWO
LITERATURE REVIEW AND THESIS CONTRIBUTION 14

2.1 Traditional Non-Linear Control 14

TABLE OF CONTENTS—Continued

2.2 Model Predictive Control 16

2.3 Genetic Algorithms 19

2.4 Machine Learning Control 20

2.5 Contribution 23

CHAPTER THREE
LYAPUNOV STABILITY THEORY 25

3.1 Non-Linear Oscillators 25

3.2 Duffing Lyapunov Control and Analysis 27

3.3 PID Controlled Duffing System 28

3.4 Van der pol Lyapunov Control and Analysis 29

3.5 PID Controlled Van der pol System 31

3.6 Zohdy-Harb Lyapunov Control and Analysis 31

3.7 PID Controlled Zohdy Harb System 34

CHAPTER FOUR
NN PREDICTIVE CONTROL 36

4.1 Types of Machin Learning Control 38

4.2 Genetic Algorithms 39

4.3 Reinforcement learning 42

4.4 Neural Network Control 44

4.5 Model Predictive Control 46

4.6 NN Predictive control Zohdy-Harb System 47

4.7 Applying NN Predictive control Duffing 51

TABLE OF CONTENTS—Continued

4.8 Applying NN Predictive control Van der pol 52

CHAPTER FIVE
LYAPUNOV DEEP LEARNING CONTROL 55

5.1 Machine Learning Lyapunov Control 56

5.2 Duffing System 58

5.3 Deep Neural Network Architecture 60

5.3.1 Dataset (Trail 1) 62

5.3.2 Dataset (Trail 12) 64

5.3.3 Dataset (Trail 116) 66

5.4 Feature Engineering 67

5.5 Deep Neural Network Architecture 68

5.5.1 Vanishing Gradient Problem 68

5.5.2 Batch Normalization 69

5.5.3 Long Short-Term Memory 69

5.5.4 Dataset (Trail 144) 70

5.6 Lyapunov Deep Learning Control on Zohdy-Harb 71

CHAPTER SIX
MAGNETIC LEVITATION APPLICATION 77

6.1 Magnetic Levitation System Dynamics 79

6.1.1 State Space Representation 81

6.2 Design of the Lyapunov Controller 82

TABLE OF CONTENTS—Continued

6.3 Deep Learning Algorithm 83

6.3.1 Customized Deep Learning Architecture 85

6.3.2 Parameterized Complexity and Dynamic Programming 86

6.4 Maglev Dynamic LDL Control Results 89

6.5 Correlation Study and DL Algorithm Application Results 94

CHAPTER SEVEN
CONCLUSION AND FUTURE WORK 99

7.1 Conclusion 99

7.2 Future work 101

REFERENCES 102

LIST OF TABLES

Table 1	Duffing Oscillator parameters table	27
Table 2	Van der pol Oscillator parameters table	29
Table 3	Zohdy-Harb oscillator parameters table	32
Table 4	Maglev system parameters	90
Table 5	Lyapunov controller parameters	90
Table 6	Person equation parameter table	95

LIST OF FIGURES

Figure 1	Relationship between Dynamic, Nonlinear and Chaotic systems	2
Figure 2	Genetic Algorithms Applied to Reinforcement Learning Tasks	20
Figure 3	Phase plane solution of Duffing System under Lyapunov control	28
Figure 4	Phase Plane Solution of Duffing System under PID control	29
Figure 5	Phase plane solution for Van der pol system under Lyapunov control	30
Figure 6	Phase plane solution for Van der pol system under PID control	31
Figure 7	Phase plane solution of Zohdy-Harb system under Lyapunov control	33
Figure 8	Position output compared to reference signal under Lyapunov control	33
Figure 9	Phase plane solution for PID controlled Zohdy-Harb System	34
Figure 10	Position Output of PID controlled Zohdy-Harb System	35
Figure 11	Magnetic levitation system controller setup	45
Figure 12	Neural Network Components	46
Figure 13	Model Predictive Control process	47
Figure 14	(Left) Network Architecture and (right) Mean Square Error Diagram as the NN is trained	48
Figure 15	The Phase plane solution under NNMPC on Zohdy-Harb System	49
Figure 16	The position output (blue) compared to the reference Sine curve in (red) under NNPC	50
Figure 17	The Phase plane solution result under NNPC on Duffing System	51

LIST OF FIGURES—Continued

Figure 18	The position output (blue) compared to the reference curve in (red) under MPC	52
Figure 19	The Phase plane solution result under model predictive control on Van der pol System	53
Figure 20	The position output (blue) compared to the reference curve in (red) under MPC Duffing System	53
Figure 21	Network Algorithm	58
Figure 22	Network retraining flow chart	59
Figure 23	Neural network Architecture initial design	60
Figure 24	Dataset utilized in trail 1	63
Figure 25	The training and validation RMSE comparison of the dataset in trail 1	64
Figure 26	Dataset utilized in trail 12	65
Figure 27	The training and validation RMSE comparison of the dataset in trail 12	65
Figure 28	The training and validation RMSE comparison of the dataset in trail 116	66
Figure 29	The training and validation RMSE comparison of the dataset in trail 144	71
Figure 30	The system error goes to infinity as shown when the algorithm is not applied	72
Figure 31	The system error after using the neural network	73
Figure 32	Error uptick at t = 0.8867 due to the disturbance introduction	74
Figure 33	The Algorithm reacting to the sudden change by adjusting Beta1	74

LIST OF FIGURES—Continued

Figure 34	The Algorithm reacting to the sudden change by adjusting Beta2	75
Figure 35	Phase diagram after finding the optimal system parameters	75
Figure 36	Magnetic levitation system	81
Figure 37	Magnetic levitation system controller setup	85
Figure 38	Dynamic Deep Learning Algorithm	88
Figure 39	Custom Deep Learning NN Architecture layers	89
Figure 40	Phase portrait of the Maglev system with a reference sinusoidal wave of 40 rad/sec frequency under Lyapunov control	91
Figure 41	Lyapunov controlled position with reference to sinusoidal wave of 40 rad/sec frequency	91
Figure 42	Phase portrait of the Maglev system with a reference sinusoidal wave of 40 rad/s frequency under PID control	92
Figure 43	PID controlled position with reference to sinusoidal wave of 40 rad/s	92
Figure 44	Lyapunov controlled position with reference to sinusoidal wave of 4000 rad/s	93
Figure 45	The position with reference to a combined signal of sinusoidal and step function under PID control	94
Figure 46	Pearson correlation chart between the parameters and error	95
Figure 47	Maglev system with a reference sinusoidal wave of 4000 rad/sec under DLDL	96
Figure 48	DLDL controlled position with reference to sinusoidal wave of 4000 rad/s frequency	97
Figure 49	The position with reference to a combination of sinusoidal and step function	97

CHAPTER ONE

INTRODUCTION

In this research, we focus on advancing the field of nonlinear controllers. Over the past decade, researchers have concentrated on the linearization of nonlinear systems and the use of linear controllers such as PID with the hope that it will perform well enough for pole placement, control, root-locus, or other linear techniques to work. This might be attributed to the ease of implementation and the low time requirement of linear control. Controllers that require linearization, such as PID or LQR techniques lose some of the best properties of the system while linearizing. On the other hand, nonlinear controllers overcome this disadvantage by using the nonlinear model. New technologies have been increasing in complexity in an exponential manner in response to the market's overwhelming need for greater functionality, performance, and bandwidth there, for as much functionality as possible is integrated in new designs. As such, linear controllers will no longer be able to provide the desired outcome as complexities increase and demand for efficiency and lossless design increase.

1.1 Fundamentals of Complex Non-Linear Systems

In order to understand the best method to control a system, we should begin by understanding what modern complex nonlinear systems require. While it is simple to define linear functions, the term nonlinear encompasses anything else. As Stanislaw Ulam, a famous scientist in the field of mathematics and nuclear physics once explained, “Using a term like nonlinear science is like referring to the bulk of zoology as the study

of non-elephant animals."[1]. As Figure 1 shows, most Dynamic Systems are non-linear in nature and most Nonlinear systems are Chaotic in nature.

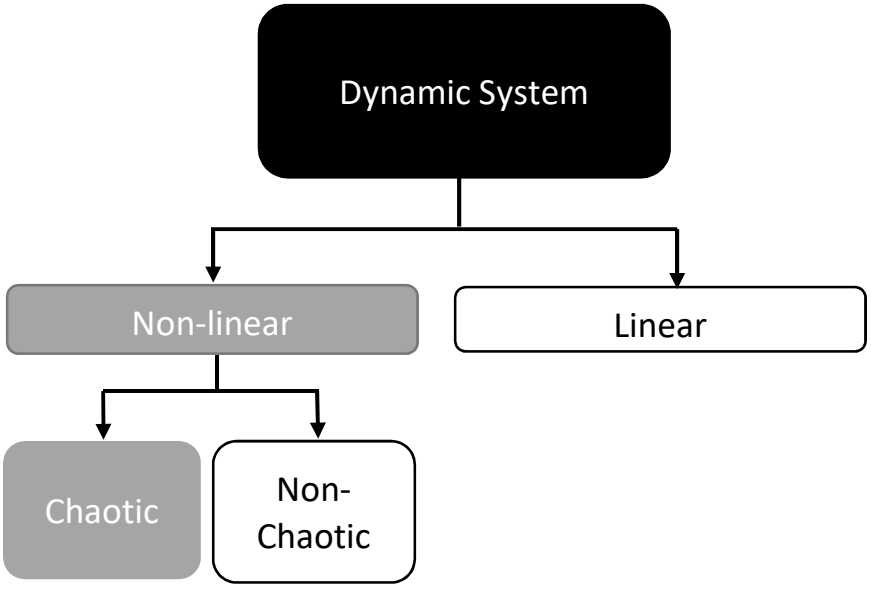

Figure 1 Relationship between Dynamic Systems, Nonlinear and Chaotic systems

Nonlinearity frequently emerges from the collective behavior of even the simplest systems, it is insufficient to combine the effects of the components merely linearly. Emergent phenomena, including chaos, solitons, fractals, and meta/multi-stability, are produced by the interactions between the components. Even if the underlying physics is deterministic, the ensuing dynamics can be very unpredictable and result in the formation of non-equilibrium patterns.

Methods of solution or analysis for problems involving nonlinear differential equations are situation specific. Lotka–Volterra, Navier–Stokes, Duffing and Van Der Pol equations are examples of nonlinear differential equations. One of the most challenging aspects of nonlinear issues is that it is typically impossible to combine

existing solutions to create new ones. A family of linearly independent solutions may be utilized through the superposition principle to derive generic solutions for linear problems, for instance. This is shown by one-dimensional heat transfer with Dirichlet boundary conditions, whose solution is represented as a time-dependent linear combination of sinusoids with varying frequencies; this makes solutions extremely versatile. Identifying several exact solutions to nonlinear equations is feasible, but the absence of a superposition principle precludes the creation of additional solutions. A further distinction between linear and nonlinear systems is that nonlinear dynamics can only be solved by using computers and simulating the dynamics. Nonlinear systems dynamics are several, but we present a few examples below, some of which will be utilized later in this research.

1.1.1 Duffing System

For the purpose of simulating damped and driven oscillators, a nonlinear second-order differential equation known as the Duffing equation is used. The Duffing system is named after Georg Duffing (1861–1944) [2]. In addition to being an example of a highly complex chaotic system, the frequency response of the Duffing system also exhibits the phenomena of jump resonance, which is a form of frequency hysteresis behavior. The Duffing equation describes the nonlinear oscillations of a mass connected to a nonlinear spring and a linear damper. Duffing Dynamic Differential Equation is presented in 1.

$$\ddot{x} + \varphi\dot{x} + \delta x + \gamma x^3 = cos\, t + u \tag{1}$$

1.1.2 Van der pol System

The Van der Pol oscillator was devised by Balthasar van der Pol, a Dutch electrical engineer and scientist at Philips. Van der Pol discovered stable oscillations,

which he subsequently termed relaxation oscillations. In addition, Van der Pol and his colleague, van der Mark, reported in the September 1927 edition of Nature that, at particular driving frequencies, an irregular noise could be heard, which was subsequently determined to be the outcome of deterministic chaos. Van der pol Dynamic Differential Equation is presented in 2.

$$\ddot{x} + \varphi(1 - x^2)\dot{x} + x = \cos t + u \tag{2}$$

1.1.3 Navier–Stokes Equations in Fluid Dynamics

The motion of viscous fluid substances may be understood via the use of a set of partial differential equations known as the Navier–Stokes equations. Claude-Louis Navier, a French engineer and scientist, and George Gabriel Stokes, a mathematician, both contributed to the naming of this phenomenon. Over the course of many decades, beginning in 1822 and continuing through 1842–1850, the ideas were gradually evolved [2, 3]. For Newtonian fluids, the Navier–Stokes equations quantitatively express momentum and mass conservation. Occasionally, they are accompanied with a state equation that links pressure, temperature, and density. The Navier–Stokes equations are valuable because they describe the physics of numerous phenomena that are of importance to science and engineering. They can be utilized to represent weather, ocean currents, water flow in a pipe, and air flow around a wing. The full and simplified Navier–Stokes equations aid in the design of aircraft and automobiles, the study of blood flow, the design of power plants, and the analysis of pollution, among several other applications. They can be used to research and model magnetohydrodynamics when combined with Maxwell's equations.

1.1.4 Lotka–Volterra Equations

In 1910, Alfred J. Lotka was the first person to describe the Lotka–Volterra predator–prey model as a component of the concept of autocatalytic chemical processes. This model was developed by Lotka and Volterra [4]. It is common practice to utilize first-order nonlinear differential equations when attempting to describe the dynamics of biological systems involving the interaction of two species, one of which acts as a predator while the other acts as prey. These equations are sometimes referred to as the predator–prey equations since they are generally recognized by that name. When attempting to represent the dynamics of natural populations of predators and prey, many models, including the Lotka–Volterra model and the Rosenzweig–MacArthur model, have been used.

Concerning the reliability of models that are dependent on prey or ratios, there has been a great deal of disagreement. It is generally accepted that Richard Goodwin carried out the first application of the Lotka–Volterra Equations 3 in either 1965 or 1967 [4]. In the hypothetical system, predators thrive so long as there is an ample supply of prey, but they run out of their food supply and finally die out. The number of animals that are hunted will eventually increase since there will be fewer predators. These activities continue in a cycle of population growth followed by population decrease.

$$\begin{aligned} \frac{dx}{dt} &= \alpha x - \beta xy \\ \frac{dy}{dt} &= \delta xy - \gamma y \end{aligned} \tag{3}$$

1.1.5 Lorenz Chaotic System

Edward Lorenz, a mathematician, and meteorologist initially explored the Lorenz system, a set of ordinary differential equations. The model is known for having chaotic

solutions for parameter values and beginning circumstances. An accumulation of chaotic solutions to the Lorenz system is what is known as the Lorenz attractor. The "butterfly effect" originates from the real-world implications of the Lorenz attractor, which state that in a chaotic physical system, in the absence of perfect knowledge of the initial conditions, even a disturbance in the air caused by a butterfly flapping its wings, our ability to predict its future course will always fail[5, 6]. This idea has made its way into popular culture, where it is used to describe the inability to accurately forecast the behavior of a system. This illustrates that physical systems may be completely predictable while yet keeping the unpredictability that is fundamental to their nature. When plotted in phase space, the shape of the Lorenz attractor itself looks like a butterfly. This resemblance is most apparent when looking at the attractor in its entirety. In its current form, the model may be represented by a set of three ordinary differential equations that are collectively referred to as the Lorenz equations.

1.2 An Overview of Nonlinear Control

Two frequent characteristics of novel control challenges are that the system's attractive operating range is not always close to equilibrium, necessitating explicit consideration of nonlinear effects in order to build a good controller. Even though physical modeling enables the precise identification of well-defined nonlinear systems the controller must contend with a large degree of uncertainty, owing mostly to due to a lack of familiarity with the system's specifications and an inability to measure the status of the entire system. This issue demonstrates the critical requirement for the development of controller tools that take on unpredictable nonlinear system behavior. When

considered conceptually, they may be generally categorized into Analytically and computationally oriented. An analytical model of the system, and controller design is the result of a methodical procedure that ensures a desired behavior. Because stability is a necessary but not sufficient requirement for this technique, it is commonly referred to as robust stabilization. It encompasses Lyapunov-based approaches, gain-assignment methods, and conventional robust and adaptive tools. On the other hand, computationally focused approaches do not require an analytical model and may be built based on a numerical model of the system to be controlled—for example, produced by the collection of vast quantities of data to approximate its behavior. The most visible examples of this school include neural network-based control, fuzzy control, and intelligent control. Recently, a second class of computationally focused methodologies has gained prominence, which is based on analytical models of the system. To attempt to replicate the evolution of linear systems. To account for nonlinear effects in theory, piecewise linear models are offered. Typically, an optimal control objective is defined, and the controller design challenge is to demonstrate that the optimization is possible for the given numerical values of the system model, e.g., that it can be translated into linear matrix inequalities and a control signal can be numerically produced. Two disadvantages exist with the optimum control strategy. To begin, the solutions are vulnerable to plant uncertainty, such as a lack of complete state measurement and parametric uncertainty, which are prevalent concerns in the majority, if not all, actual applications. Second, calculation of the optimal control law is only achievable for low-dimensional systems, casting doubt on the method's application to nonlinear systems. Additionally, there is not necessarily a compelling rationale, other than mathematical convenience, for expressing

the intended behavior of a dynamical system in terms of an optimization scalar criteria. While computationally oriented techniques benefit from rapidly developing computer technology, they focus on providing answers to specific issues rather than on explaining why, how, and when these solutions work. Therefor in this research we aim to comprehend the underlying process by which the system operates. The information is contained in the dynamics of the nonlinear system and disclosed by a thorough nonlinear analysis.

1.2.1 Gain Scheduling

Gain scheduling is a typical method for regulating nonlinear systems whose dynamics vary across operating conditions. Gain scheduling is utilized when a single set of controller gains does not offer the necessary performance and stability throughout the whole range of plant operating circumstances.

1.2.2 Adaptive Control

Adaptive control is an active field in the design of control systems to account for uncertainty. The major distinction between adaptive controllers and linear controllers is the adaptive controller's capacity to change itself to deal with unforeseen model uncertainties. Direct and indirect adaptive control are the two primary classifications. Indirect approaches estimate the plant's parameters and then utilize the predicted model data to calibrate the controller. Direct techniques are those in which the estimated parameters are utilized directly by the adaptive controller.

1.2.3 Model Predictive Control

MPC models anticipate the change in the system's dependent variables that will result from changes in the independent variables. The setpoints of regulatory PID

controllers (pressure, flow, temperature, etc.) or the final control element are often controller-adjustable independent variables (valves, dampers, etc.). We make use of independent variables that are not subject to the influence of the controller here in the role of disturbances. The dependent variables in these processes are additional measurements that either represent control goals or process constraints.

Model predictive control may be broken down into many subtypes, one of which is known as nonlinear model predictive control, or NMPC for short. NMPC makes use of nonlinear system models for prediction. The iterative solution of optimal control problems with a limited prediction horizon is required in NMPC, just as it is in linear MPC. In linear MPC, these problems have a convex solution, however in nonlinear MPC, the convexity of these problems is not guaranteed. Both the theoretical framework of NMPC stability and the numerical solution face challenges as a result of this[7].

Typically, the numerical solution of NMPC optimum control problems is based on direct optimal control techniques employing Newton-type optimization procedures in one of the following variants: direct single shooting, direct multiple shooting, or direct collocation.

NMPC algorithms often make use of the similarity between successive optimum control problems. This allows for an efficient initialization of the Newton-type solution approach by a properly shifted estimate from the previously calculated optimum solution. As a result, a significant amount of computation time may be saved as a result of this. Path following algorithms are algorithms that never attempt to iterate any optimization problem to the point where it converges, but instead only take a few iterations towards

the solution of the most recent NMPC problem, before proceeding to the next one, which is suitably initialized; see, exploit the similarity of subsequent problems even further[8].

NMPC is increasingly being applied to applications with high sampling rates, such as in the automotive industry, or even when the states are spread in space, thanks to the breakthroughs that have been made in controller hardware and computational algorithms, such as preconditioning. In the past, NMPC applications were predominantly used in the process and chemical industries, which had relatively slow sampling rates (Distributed parameter systems).

Recent aerospace applications of NMPC include tracking optimum terrain-following/avoidance trajectories in real time[9].

Model predictive control algorithm utilizes the following functions:

- An optimization algorithm
- A cost function J
- A dynamic model of the process
- Sliding mode control

Sliding Mode Control is an approach to nonlinear control that modifies the dynamics of a nonlinear system by applying a discontinuous control signal [10]. This control signal causes the system to slide over a cross-section of the system's normal behavior, which in turn modifies the dynamics of the system. Legislation to govern the input received from the state is not a time-continuous function. Instead, it can transition from one continuous structure to another in accordance with the position it now occupies in the state space. Control using a sliding mode is thus an example of control using a variable structure. The sliding-mode-control rule toggles between states according to the

sign of this distance as the criterion for transition. Since the control rule for the sliding mode is discontinuous, the sliding surface will be reached in a finite amount of time. As a result, the sliding-mode control applies a constant force in the direction of the sliding mode, which is the direction in which the trajectories will approach the sliding surface. After a trajectory has contacted the surface, it will glide over the surface and may, for example, go back toward its point of origin. As a result, the switching function may be thought of as being comparable to a topographic map that has a contour line of constant height that trajectories are required to follow[11].

1.3 Lyapunov Control Function

The need for increased autonomy and accuracy in robotics has forced the development of new methods of control systems. The global output feedback issue for robots has proven to be particularly difficult. The design of setpoint and tracking controllers for nonholonomic systems was influenced by mobile robots with wheels. Other applications that have challenged control designers include rigid link flexible joints in the late 1990s and higher dimensional and continuum robots in more recent years. According to the indirect technique, also known as the linearization method, the stability features of a nonlinear system are almost identical to those of its linearized approximation when the system is located in close proximity to a point where it has achieved equilibrium. The approach provides the theoretical foundation for utilizing linear control for physical systems, which are always essentially nonlinear. The physical systems themselves are always nonlinear. Because the direct approach is such an effective instrument for doing nonlinear system analysis, the term "Lyapunov analysis"

often but incorrectly refers to the direct method instead. Lyapunov-based approaches have the remarkable benefit of allowing both design and analysis inside a shared framework, with one step iteratively stimulating the other.

The direct technique of Lyapunov is based on the physical principle that a system whose entire energy is being continually wasted must ultimately reach equilibrium. Given a scalar, nonnegative energy function V (t) for a system, it can be shown that if its time derivative V (t) 0, the system is stable in the sense of Lyapunov if the system states can be constrained for all future time to lie within a ball whose radius is proportional to the size of the initial system states[12].

CLF is an extension of the concept of the Lyapunov function V(x) to systems with control inputs. The standard Lyapunov function determines if a dynamical system is stable or asymptotically stable. Lyapunov stability denotes that if the system begins in a state in domain D, that state will persist forever. Additionally, for asymptotic stability, the state must converge to x=0. The control-Lyapunov function is used to determine if a system is asymptotically stabilizable. For each state x, there exists a control u(x,t) such that the system can be brought to the zero state by applying the control u [11, 13, 14].

1.4 Thesis Chapter Outline

Chapter Two: Provides an overview of the previous research completed in traditional linear control and why a transition in non-linear control is required to provide chaotic systems stability. Examples of research completed in non-linear controls are given with references to research results. A look into research completed in computational and data driven control types is provided with the previous results

obtained. Finally, a problem statement is clarified, and the contribution points are provided.

Chapter Three: Control Lyapunov function is introduced and discussed in this chapter. The Lyapunov function developed is utilized as a feedback signal for the purpose of control. The controller results are discussed in the chapter when utilized with the highly chaotic Duffing, Van Der Pol and Zohdy-Harb system. The novel control function results are compared to PID control results [15].

Chapter Four: Machine Learning Control is introduced and discussed in detail in this chapter. In addition, the integration of Deep Machine Learning into Lyapunov control to develop a novel Intelligent Nonlinear control is described. The different datasets and architectures developed during the research of the machine learning control accuracy [16].

Chapter Five: To tackle the drawbacks of Deep Learning such as the computational and time expense the novel Dynamic Deep Learning Control is introduced in this chapter. The Dynamic Learning Lyapunov Control (DLLC) is utilized on the Mag lev application. Maglev systems face instability and disturbances in a sensitive environment. The results are discussed and the metho is shown to be successful in controlling and adapting to the maglev system [17].

Chapter Six: In this chapter we will conclude the thesis work and provide a summary of the work completed and results. A summary of proposals to future work will also be included in this chapter.

CHAPTER TWO

LITERATURE REVIEW AND THESIS CONTRIBUTION

2.1 Traditional Non-Linear Control

Over the past few years there has been a significant growth of system complexities. Some complex systems desirable operating range is not close to equilibrium, demanding explicit consideration of nonlinear effects in order to design an effective controller. Even though physical modeling permits the exact identification of well-defined nonlinear systems, the controller must deal with a high degree of uncertainty due to a lack of knowledge with the system's specifications and an inability to monitor the system's status. This problem illustrates the vital need for the development of controller tools capable of handling unpredictable nonlinear system behavior. Conceptually, they can be classified into the following categories: Having an analytical and computational focus. A system's analytical model and controller design are the outcome of a systematic technique that ensures the desired performance. This technique is usually referred to as robust stabilization because stability is a necessary but insufficient criterion. It consists of Lyapunov-based strategies, gain-assignment techniques, and standard robust and adaptable tools [18].

In contrast, computationally focused techniques do not require an analytical model and can be constructed based on a numerical model of the system to be controlled, such as one generated by the collection of huge quantities of data approximating the system's behavior. The most prominent examples are control based on neural networks, fuzzy control, and intelligent control. Recently, other computationally focused techniques

based on analytical models of the system has gained importance. Piecewise linear models are available to account for nonlinear effects in theory. Typically, an optimal control objective is specified, and the controller design challenge is to demonstrate that the optimization is possible for the given numerical values of the system model, it can be translated into linear matrix inequalities and a numerical control signal can be generated. There are two downsides to the optimal control approach [19]. The solutions are susceptible to plant uncertainty, such as a lack of comprehensive state measurement and parametric uncertainty, which are prominent problems in the vast majority of, if not all, practical applications. Second, the optimal control law can only be calculated for low-dimensional systems, casting doubt on the applicability of the method to nonlinear systems. In addition, describing the anticipated behavior of a dynamical system in terms of an optimization scalar criterion does not necessitate a persuasive justification, other than mathematical convenience. While computationally focused strategies benefit from fast advancing computer technology, they are more concerned with giving solutions to specific problems than with explaining why, how, and when these solutions work. Therefore, the purpose of this research is to know the system's basic operating mechanism[18]. The information is included in the nonlinear system's dynamics and is revealed through a comprehensive nonlinear analysis.

Adaptive control is an active field in the design of control systems to account for uncertainty. The major distinction between adaptive controllers and linear controllers is the adaptive controller's capacity to change itself to deal with unforeseen model uncertainties. Direct and indirect adaptive control are the two primary classifications. Indirect approaches estimate the plant's parameters and then utilize the predicted model

data to calibrate the controller. Direct techniques are those in which the estimated parameters are utilized directly by the adaptive controller [20].

Consider the difficulty of controlling a nonlinear multivariable system that is constrained in its input and state by a combination of physical and operational constraints. The well-known systematic nonlinear control approaches, such as feedback linearization, lead to beautiful solutions; nevertheless, they are dependent on complicated design processes that do not scale well to large systems and are not intended to manage constraints in a systematic way[21].

2.2 Model Predictive Control

MPC models anticipate the change in the system's dependent variables that will result from changes in the independent variables. Nonlinear model predictive control, or NMPC, is a subtype of model predictive control that employs nonlinear system models for prediction[22]. The iterative solution of optimal control problems with a limited prediction horizon is required in NMPC, just as it is in linear MPC. In linear MPC, these problems have a convex solution, however in nonlinear MPC, the convexity of these problems is not guaranteed. Both the theoretical framework of NMPC stability and the numerical solution face challenges as a result of this [8].

The numerical solution of NMPC optimum control problems is based on direct optimal control techniques employing Newton-type optimization procedures in one of the following variants: direct single shooting, direct multiple shooting, or direct collocation. NMPC algorithms often make use of the similarity between successive optimum control problems. This makes it possible to effectively initiate the Newton-type solution

approach by making a guess that is correctly offset from the previously calculated ideal solution. As a result, a significant amount of time may be saved throughout the calculation process. There is completed research into path following algorithms or real-time iterations that never attempt to iterate any optimization problem to convergence [9, 23].

Non-linear Model Predictive Control (NMPC) is increasingly being applied to applications with high sampling rates, such as in the automotive industry, or even when the states are distributed in space, thanks to advancements in controller hardware and computational algorithms, such as preconditioning. An illustration of such an application is the situation in which the states are dispersed over space. NMPC applications were used most often within the process industries, which had lower sample rates [24].

Recent aerospace applications of NMPC include tracking optimum terrain-following/avoidance trajectories in real time. An optimization algorithm minimizing the cost function J using the control input u. An example of a quadratic cost function is given by [24]:

$$J = \sum_{i=1}^{N} w_{X_i}(r_i - x_i)^2 + \sum_{i=1}^{N} w_{u_i}\Delta u_i^2 \quad (4)$$

x_i: i^{th} controlled variable

r_i: i^{th} reference variable

u_i: i^{th} manipulated variable

w_{x_i}: weighting coefficient reflecting the relative importance of x_i

w_{u_i}: weighting coefficient penalizing relatively big changes in u_i

2.2 Sliding mode control

SMC is an approach to nonlinear control that modifies the dynamics of a nonlinear system by applying a discontinuous control signal (or, more formally, a set-valued control signal) that causes the system to "slide" over a cross-section of the system's normal behavior [25]. This causes the dynamics of the system to be altered in a way that is not linearly predictable. Legislation to govern the input received from the state is not a time-continuous function. Instead, it is able to transition from one continuous structure to another in accordance with the position it now occupies in the state space. Sliding mode control is hence an example of variable structure control [10].

The numerous control structures are built such that trajectories always advance toward a neighboring region with a different control structure; hence, the end trajectory will not reside wholly inside a single control structure. It will instead glide along the perimeters of the control structures. A sliding mode describes the motion of the system as it moves along these limitations, and the geometrical locus that is generated by the boundaries is referred to as the sliding surface. Any variable structure system, such as a system that is subject to SMC, may be seen as a special instance of a hybrid dynamical system within the framework of modern control theory. This is because the system flows through a continuous state space in addition to discrete control modes. Lyapunov control has been shown to be capable of effectively controlling extremely chaotic non-linear oscillators [26]. One of the key factors that contribute to the success or failure of any sort of control plan is the controller and system parameters. As a consequence of this, researchers have studied a number of different strategies in order to determine the

particular attributes that, once maximized, would give the greatest potential outcomes for the system [20].

2.3 Genetic Algorithms

The genetic algorithm is a technique for solving optimization issues that is based on natural selection as presented in Figure 2. This approach is used to solve both limited and unconstrained optimization problems [27]. At each stage of the process, the genetic algorithm chooses members of the existing population to serve as parents, and then employs those people to generate the offspring that will comprise the subsequent generation. The population evolves toward the best possible answer as time passes and new generations are born. Genetic algorithms can be used to solve a variety of optimization problems that are not well suited for standard optimization algorithms. These problems include complications in which the objective function is discontinuous, nondifferentiable, stochastic, or highly nonlinear. Genetic algorithms can be applied to solve these problems by using a population of individuals with these characteristics. The evolutionary algorithm may be used to solve issues in mixed integer programming, described as a kind of programming in which certain components can only have integer values [28].

Employing GAs comes with several drawbacks, the most notable of which are the inability to rapidly converge on a final solution and the incapacity to adapt to unknown system dynamics or unforeseen perturbations.

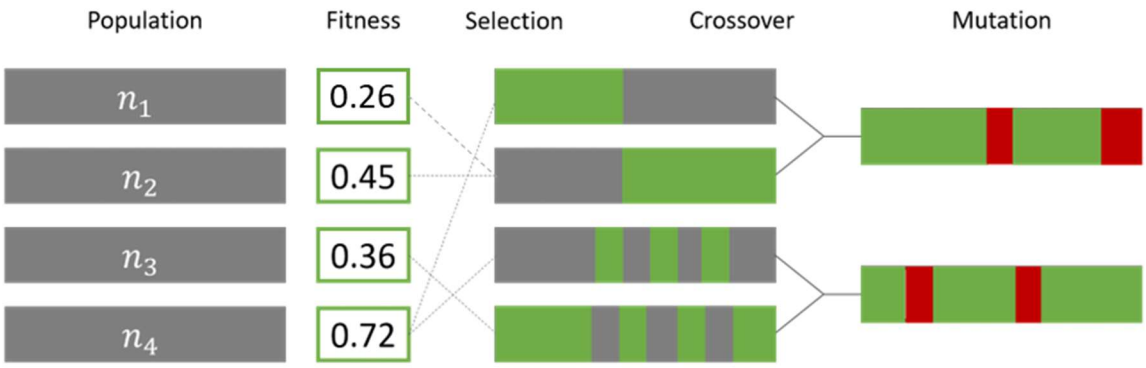

Figure 2 Genetic Algorithms Applied to Reinforcement Learning Tasks

Researchers investigated an array of methods, all with the goal of finding one that wouldn't compromise the system's adaptability while still capable of dealing with undetermined aspects of the system. A strategy that combines Fuzzy Control with GAs was investigated in [20], although system linearization is necessary in order to employ the method described in [20].

2.4 Machine Learning Control

The application of machine learning as a means of improving controller performance is yet another strategy that has lately become the subject of investigation. Most recently, there has been the development of neural Lyapunov control, which proposes the utilization of deep learning in order to locate the control and Lyapunov functions that are predicted to stabilize the system. The method described in [29] and [30] is appropriate for the task of determining which system settings would initially get the system to a state of stability while simultaneously lowering the amount of error it produced. The method described in [29] and [30] has a flaw in that it presupposes the system to be deterministic, time invariant, and affine in the control input. but in a real-life

scenario, external perturbations could lead to the failure of the system at any time while it is operating [31], [32]. The method that is described in [33], [34], and [35] makes an effort to forecast the control and Lyapunov functions that would result in the stability of the system, but it does so only under certain conditions in which the dynamics of the system are of a deterministic character.

Deep learning-based systems have a number of benefits, including the ease with which complex patterns may be detected, the ability to adapt and learn towards unknowns, and a higher degree of accuracy in predicting the outputs when compared to shallow neural networks. On the other hand, these systems also have a number of drawbacks, including the inability to adapt and learn towards unknowns and the inability to predict the outputs with a high degree of accuracy. One of these downsides is the fact that cyber-physical systems that are based on deep learning face a number of obstacles. For instance, in order to provide conclusions that are objective, DNNs put an emphasis on acquiring data that is accurate as well as diverse. In addition, DNNs do not provide any guarantees or assurances on the safety or applicability of a suggested solution or the findings of the study [17]. The solution to the problem of set bias was to provide the DNN extra data; however, this caused a delay in the amount of time that was necessary for the DNN to determine which method was most effective. Therefore, Finding the correct balance between more in-depth architectures and more practical regularization methods is one of the most difficult aspects of using DNN in CPS. This is one of the primary issues.

In spite of the fact that Lyapunov Control traditional methods are effective in managing system behaviors of a medium complexity, they are vulnerable when dealing

with cyber–physical systems of a high complexity [36]. It has been demonstrated that the control Lyapunov function, abbreviated as CLF, may be utilized to successfully regulate complex nonlinear duffing and Van der Pol systems [15, 33, 37]. The utilization of nonlinear controllers allows for a lossless control strategy to be implemented, which helps to prevent the linearization of the system. Additionally, the incorporation of deep learning enables the continual modification and selection of system and control parameters. Machine learning has been utilized in previous research in conjunction with what is typically referred to as a cost function; however, one of the drawbacks discovered was the amount of time required for the DNN to relearn during the process of updating the data set and also the time requirement to come up with the appropriate solution [38, 39]. This was found to be one of the most time-consuming aspects of the process. In addition, the number of needed parameters as well as the size of the dataset had an effect on the accuracy of the DNN as well as the amount of time it took to discover a solution, as shown in [40].

The deep learning data driven integration is presented to allow the non-linear component to adapt with the system unknown and unknown parameters. In addition, it is presented as a result of the ongoing trend of advancements in technology and microchips, the complexity of systems will continue to increase. According to an article that was published in the journal of the National Academies in 2016, it was stated that "today's practice of system design and implementation is often ad hoc and unable to support the level of complexity, scalability, security, safety, interoperability, and flexible design and operation that will be required to meet future needs.". While earlier research in [41, 42] focused on using machine learning to toggle the control law between two or more

possibilities, the current study will focus on the integration of Deep Learning while maintain the safety and fidelity of the nonlinear analytical controller.

Deep Learning is found to present some restrictions when it comes to computational and time expense [38]. This study also presents the usage of the Pearson correlation, as well as the practice of giving precedence to parameters with strong correlation to the system error. In addition, we place an emphasis on the application of parameterized complexity in order to analyze the dataset. This allows us to reduce the amount of depth that the DNN has while still producing correct results during the retraining process. An individualized structure composed of layers is presented in this research where the components can be repeated while maintaining the number of nodes in a single layer. To the best of our knowledge, the incorporation of the concept that was indicated earlier has not been covered in any of the earlier research that has been published.

2.5 Contribution

In this study, we research the use of Lyapunov control functions as a means to controlling highly chaotic systems. A novel Lyapunov controller is developed for each system under investigation. We further investigate the integration of deep machine learning into the Lyapunov control parameter selection, but it is found that over the shelf deep learning architectures were not able to converge to a feasible solution within the required time. Therefore, a novel deep learning architecture accompanied with an algorithm is developed.

The feature impact and dataset impact on the controller are studied. To tackle the drawbacks of Deep Learning such as the computational and time expense a method is proposed based on the parameterized complexity theory. The method evaluates the dataset's complexity and adjusts the DNN's depth accordingly. Other numerical aspects of the input instance, such as the correlation between the dataset parameters collected while the CPS system is operating or the rise or reduction in memory occupation, are examined in addition to the input instance's length when assessing the system's complexity. The initial data set consists of parameter modifications and their respective output effects. The deep neural network (DNN) collects an updated dataset and executes it based on the initial CPS information. Through a neural architecture search and meta-learning, we uncover compact high-performance deep learning architectures. Customized neural network architecture minimizes training time and computational requirements. If the delta error between the actual system error and the anticipated system error generated by the DNN after parameter substitution is more than 0.40, a new data set is recorded. The system begins adding to its current data set while retaining 40% of the older data set in memory. To tackle the deep learning computational and time expense with the increase of parameters a unique function for calculating the number of NN layers needs to relearn the effect of parameter modification on the model.

The presented Dynamic Deep Learning Control is proven to be successful in tackling the timing latency compared to regular Deep Learning Lyapunov Control and with the improved adaptiveness and ability tackle disturbances compared to Lyapunov control and PID control.

CHAPTER THREE

LYAPUNOV STABILITY THEORY

Given a control system, the first and most crucial concern regarding its features is whether it is stable or unstable, as an unstable control system is often ineffective and potentially hazardous. Qualitatively, a system is considered stable if beginning the system near its target operating point implies that it will remain near the point indefinitely. Every control system, whether linear or nonlinear, entails a stability issue that must be thoroughly investigated. The theory proposed by the Russian mathematician Aleksandr Mikhailovich Lyapunov in the late 19th century is the most practical and general method for studying the stability of nonlinear control systems. The 1892 publication The General Problem of Motion Stability by Lyapunov contains two approaches for stability analysis [43]. The linearization method and the direct method. The linearization method concludes about the local stability of a nonlinear system at an equilibrium point based on the stability properties of its linear approximation. The direct technique is not limited to local motion, and it finds the stability features of a nonlinear system by building a scalar function for the system and analyzing the time variation of that function. A control-Lyapunov function is an extension of the Lyapunov function to systems with control inputs.

3.1 Non-Linear Oscillators

There are a variety of uses for Duffing and Van der Pol Oscillators. For instance, the Duffing oscillator has been utilized to detect chirp signals [44]. It is also utilized extensively in the signal communication domain, such as in the secure communication

field [45] and weak signal identification [46]. As demonstrated by [47], it has also made its way into marine applications, such as ship propeller blade number recognition. Despite having fewer application examples than the Duffing oscillator, the Van der Pol oscillator has found its way into several sectors, including the medical industry, where it was utilized to represent the heart pulse, as demonstrated in [48]. As demonstrated in [49], another illustration is the application of the Van der Pol oscillator in the simulation of dust density wave fields. On the control of a Duffing Oscillator, some effort has been expended [50] employed a fuzzy sliding controller. The control methods were based on the Lyapunov stability theorem, and simulation results demonstrated that the system could be successfully controlled despite the existence of chaos. Using impulsive parametric perturbations, [51] proposed chaos control for a Duffing system. Using numerical simulations, the authors validated the viability of the suggested method, which is based on the Melnikov method [52]. The researchers of [53] studied the dynamic properties of a Van der Pol system with extra delay. The authors discovered that Hopf bifurcation arises from trivial equilibrium when the delay exceeds critical values. The authors then determined the link between the critical values and the system parameters.

The authors supported their claims with numerical evidence. [54] investigated chaos management in a Van der Pol system with a nonlinear force and two forcing excitations. The authors demonstrated their findings by numerical simulation.

By altering the phase difference and magnitude of the second excitation force, the scientists determined that chaotic motions are controllable. Lastly, Van der Pol system control was also accomplished utilizing bifurcation.

3.2 Duffing Lyapunov Control and Analysis

The mathematical model presented in equation 2 presents the Duffing Oscillator.

$$\ddot{x} + \varphi\dot{x} + \delta x + \gamma x^3 = \cos t + u \tag{5}$$

The Duffing oscillator can be regarded as a forced oscillator with a spring. This spring is dubbed a hardening spring when $\gamma > 0$ and a softening spring when $\gamma < 0$, This interpretation is acceptable only for small x. (Thompson and Stewart, 2002).

Table 1 Duffing Oscillator parameters table

$\boldsymbol{\Phi}$	controls the amount of damping
$\boldsymbol{\delta}$	controls the linear stiffness
$\boldsymbol{\gamma}$	controls the amount of non-linearity in the restoring force

Equation 5 depicts a forced Duffing oscillator in conjunction with an actuator. Error (e) is defined as $e = z_{des} - z$ and therefor, the 2nd derivative of the error e is $\ddot{e} = \ddot{z}_{des} - \ddot{z}$. Figure 3 presents the stable output of the Duffing phase solution under Lyapunov control.

A control Lyapunov candidate is selected such that

$$R = \frac{1}{2}(\beta_1 e + \beta_2 \dot{e}) \tag{6}$$

$$\dot{R} = (\beta_1 e + \beta_2 \dot{e}).(\beta_1 \dot{e} + \beta_2 \ddot{e}) \tag{7}$$

$$\dot{R} = -\delta R \tag{8}$$

$$\beta_2 \ddot{e} + \beta_1 \dot{e} = \ddot{z}_{des} - \ddot{z} + \beta_1 \dot{e} \tag{9}$$

Substituting in equation 6 we get

$$\beta_2(\ddot{z}_{des} - \ddot{z}) + \beta_1\dot{e} = -\frac{\delta}{2}(\beta_1 e + \beta_2\dot{e}) \quad (10)$$

Through substitution in $\ddot{z}$

$$\beta_2(\ddot{z}_{des} + \varphi\dot{x} + \delta x + \gamma x^3 - u) + \beta_1\dot{e} = -\frac{\delta}{2}(\beta_1 e + \beta_2\dot{e}) \quad (11)$$

To find the control law u

$$u = \frac{\delta}{2}\frac{\beta_1}{\beta_2}e + \frac{\delta}{2}\dot{e} + \ddot{z}_{des} + \varphi\dot{x} + \delta x + \gamma x^3 + \frac{\beta_1}{\beta_2}\dot{e} - \frac{\cos t}{\beta_2} \quad (12)$$

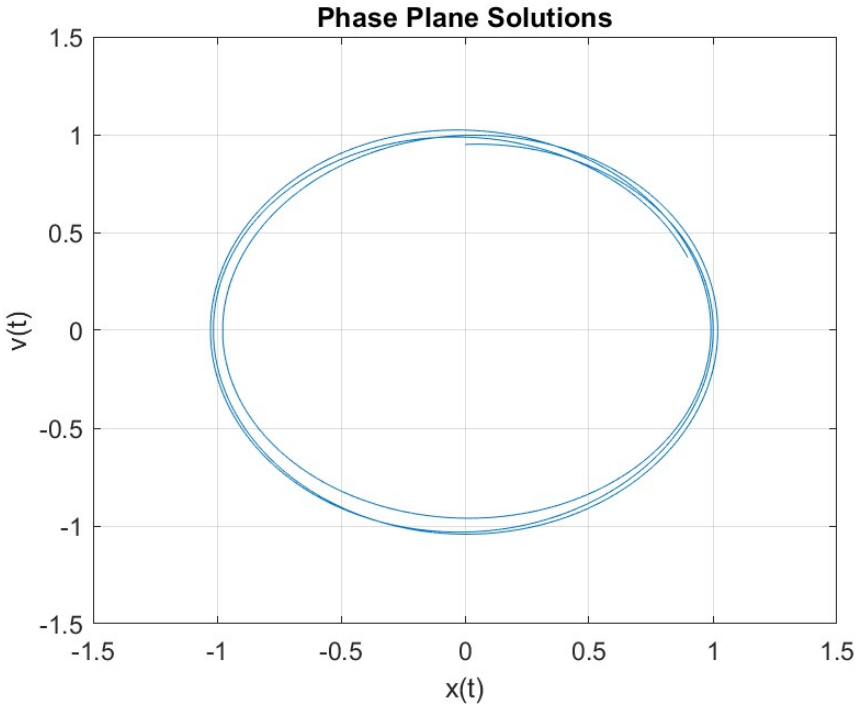

Figure 3 Phase plane solution of Duffing System under Lyapunov control

3.3 PID Controlled Duffing System

PID control is utilized in comparison to Lyapunov control to show Lyapunov's superiority in terms of control. Figure 4 PID control is shown to lag behind in managing the error for high frequency inputs.

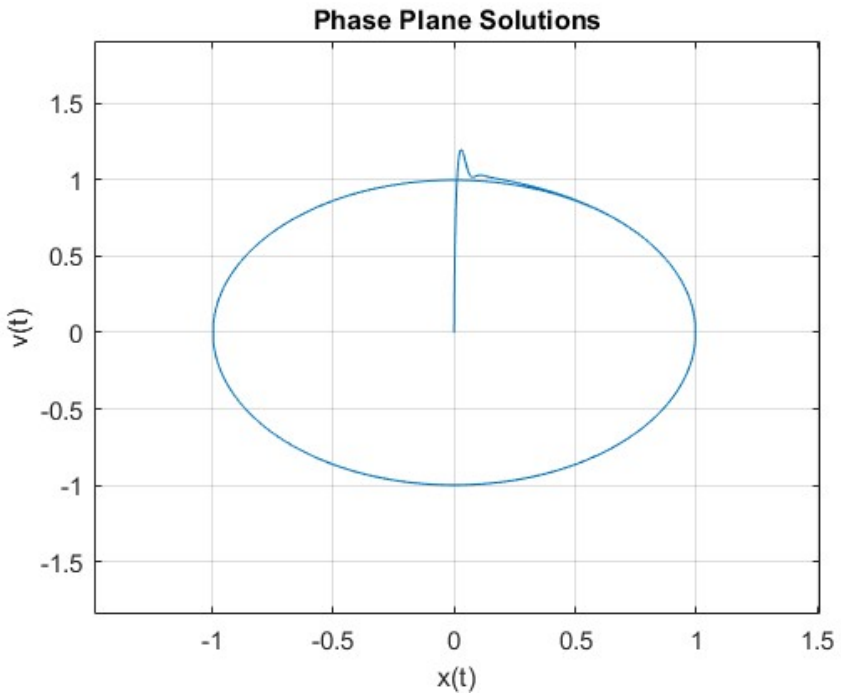

Figure 4 Phase Plane Solution of Duffing System under PID control

3.4 Van der pol Lyapunov Control and Analysis

The mathematical model presented in equation 13 presents the Van der pol Oscillator.

$$\ddot{x} + \varphi(1 - x^2)\dot{x} + x = cos\, t + u \tag{13}$$

The Van der pol oscillator can be regarded as a non-conservative, non-linearly damped oscillator that changes over time based on the second-order differential equation in 13. Figure 5 presents the stable output of the Van der pol phase solution under Lyapunov control.

Table 2 Van der pol Oscillator parameters table

$\boldsymbol{\Phi}$	controls the amount of damping

A control Lyapunov candidate is selected such that

$$R = \frac{1}{2}(\beta_1 e + \beta_2 \dot{e}) \tag{14}$$

$$\dot{R} = (\beta_1 e + \beta_2 \dot{e}).(\beta_1 \dot{e} + \beta_2 \ddot{e}) \tag{15}$$

$$\dot{R} = -\delta R \tag{16}$$

$$\beta_2 \ddot{e} + \beta_1 \dot{e} = \ddot{z}_{des} - \ddot{z} + \beta_1 \dot{e} \tag{17}$$

Substituting in equation 14 we get

$$\beta_2(\ddot{z}_{des} - \ddot{z}) + \beta_1 \dot{e} = -\frac{\delta}{2}(\beta_1 e + \beta_2 \dot{e}) \tag{18}$$

Through substitution in $\ddot{z}$

$$\beta_2(\ddot{z}_{des} + \varphi(1 - x^2)\dot{x} + x - cos\, t - u) + \beta_1 \dot{e} = -\frac{\delta}{2}(\beta_1 e + \beta_2 \dot{e}) \tag{19}$$

To find the control law u

$$u = \ddot{z}_{des} + \varphi(1 - x^2)\dot{x} + x - cos\, t + \frac{\beta_1}{\beta_2}\dot{e} + \frac{\delta}{2}\frac{\beta_1}{\beta_2}e + \frac{\delta}{2}\dot{e} \tag{20}$$

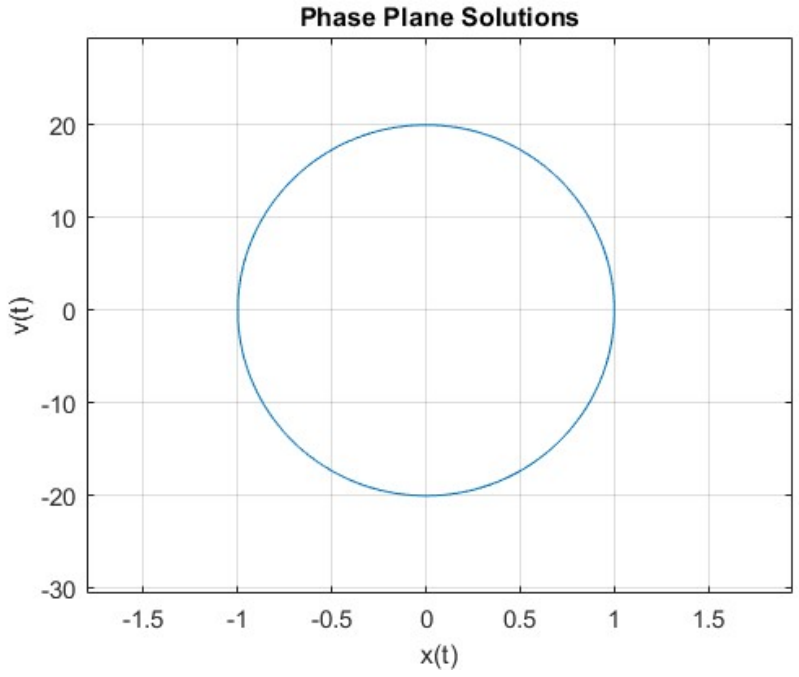

Figure 5 Phase plane solution for Van der pol system under Lyapunov control

3.5 PID Controlled Van der pol System

PID control is utilized in comparison to Lyapunov control to show Lyapunov's superiority in terms of control. Figure 6 PID control is shown to lag in managing the error for high frequency inputs.

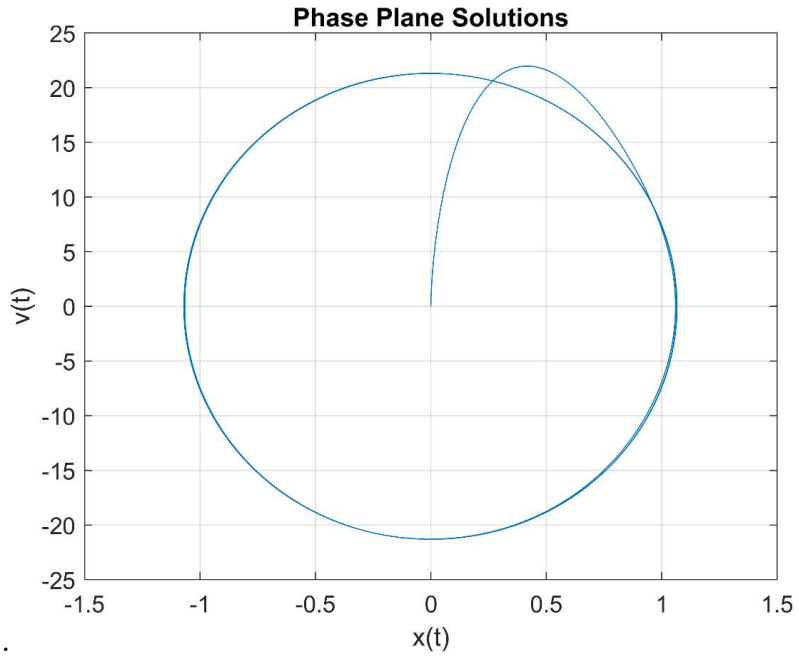

Figure 6 Phase plane solution for Van der pol system under PID control

3.6 Zohdy-Harb Lyapunov Control and Analysis

The mathematical model presented in equation 21 presents the Zohdy-Harb Oscillator [55].

$$\ddot{x} + \delta\dot{x} + \varphi(x^2\ddot{x} + \dot{x}^2x) + \gamma x^3 = P\cos\Omega t + u \tag{21}$$

The Zohdy-Harb oscillator integrates both nonlinearities of stiffness and damping, hence producing a more complicated and chaotic behavior to regulate. The system parameters that contribute to the control of damping and stiffness are described below. Figure 7 presents the stable output of the Zohdy-Harb phase solution under Lyapunov

control and Figure 8 presents the position output successfully following a sinusoidal wave.

Table 3 Zohdy-Harb Oscillator parameters table

$\boldsymbol{\Phi}$	controls the amount of damping and stiffness
$\boldsymbol{\gamma}$	controls the amount of non-linearity in the restoring force

Equation 21 depicts a Zohdy-Harb oscillator in conjunction with an actuator. Error (e) is defined as $e = z_{des} - z$ and therefor, the 2nd derivative of the error e is $\ddot{e} = \ddot{z}_{des} - \ddot{z}$.

A **control Lyapunov candidate** is selected such that

$$R = \frac{1}{2}(\beta_1 e + \beta_2 \dot{e}) \tag{22}$$

$$\dot{R} = (\beta_1 e + \beta_2 \dot{e}).(\beta_1 \dot{e} + \beta_2 \ddot{e}) \tag{23}$$

$$\dot{R} = -\delta R \tag{22}$$

$$\beta_2 \ddot{e} + \beta_1 \dot{e} = \ddot{z}_{des} - \ddot{z} + \beta_1 \dot{e} \tag{23}$$

Substituting in equation 22 we get

$$\beta_2(\ddot{z}_{des} - \ddot{z}) + \beta_1 \dot{e} = -\frac{\delta}{2}(\beta_1 e + \beta_2 \dot{e}) \tag{24}$$

Through substitution in $\ddot{z}$

$$\beta_2(\ddot{z}_{des} + \varphi\dot{x} + \alpha(x^2\ddot{x} + \dot{x}^2 x) + \gamma x^3 - u - cos\, t) + \beta_1 \dot{e} = -\frac{\delta}{2}(\beta_1 e + \beta_2 \dot{e}) \tag{25}$$

To find the control law u

$$u = \frac{\delta}{2}\frac{\beta_1}{\beta_2} e + \frac{\delta}{2}\dot{e} + \ddot{z}_{des} + \varphi\dot{x} + \alpha(x^2\ddot{x} + \dot{x}^2 x) + \gamma x^3 + \frac{\beta_1}{\beta_2}\dot{e} - \frac{cos\, t}{\beta_2} \tag{26}$$

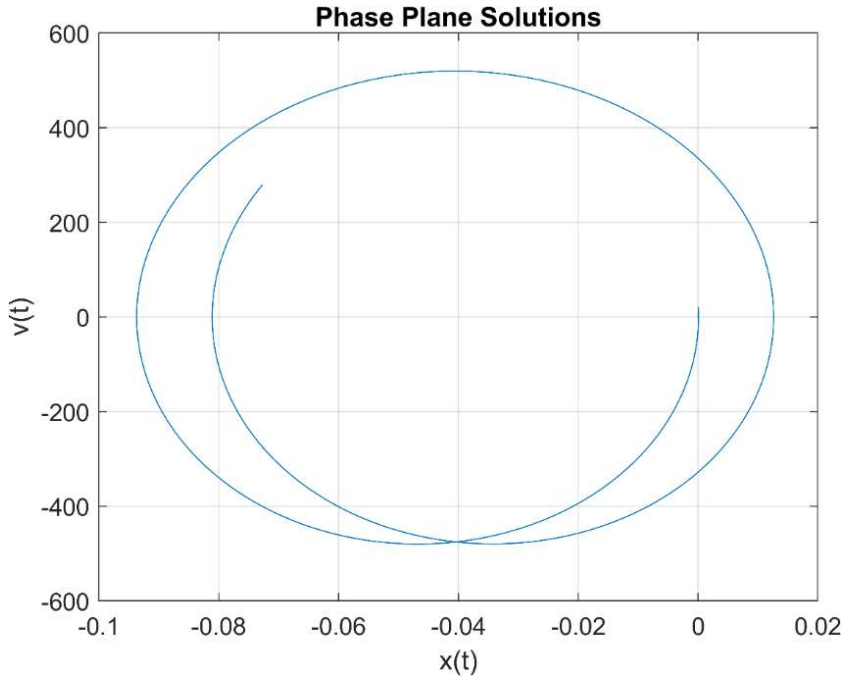

Figure 7 Phase plane solution of Zohdy Harb system under Lyapunov control

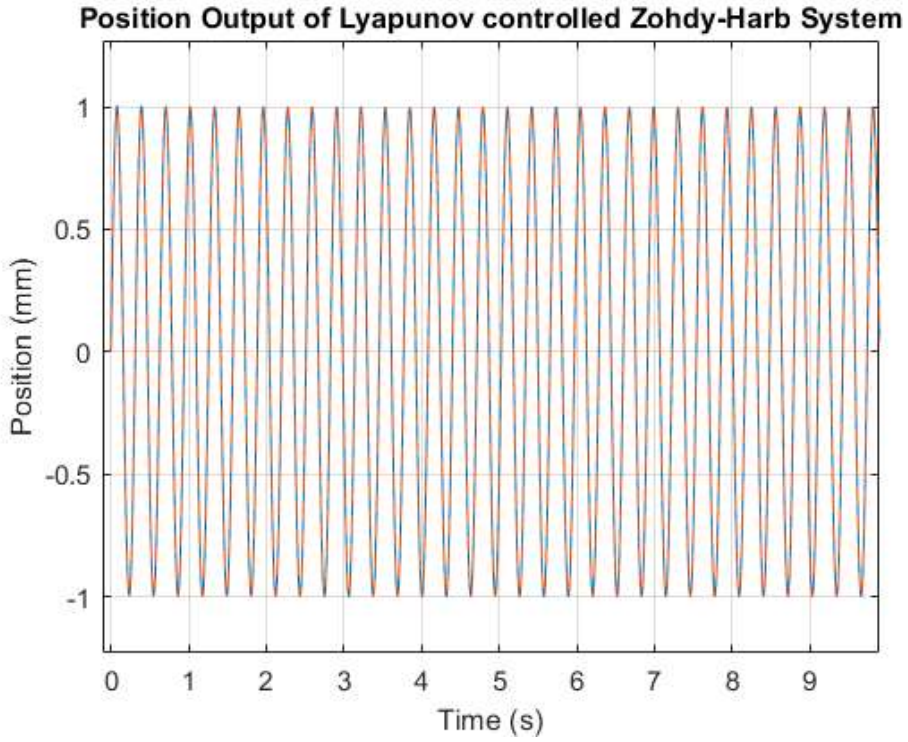

Figure 8 Position output compared to reference signal under Lyapunov control

3.7 PID Controlled Zohdy Harb System

PID control is utilized in comparison to Lyapunov control to show Lyapunov's superiority in terms of controls. Figure 9 and 10 PID control is shown to lag in managing the error.

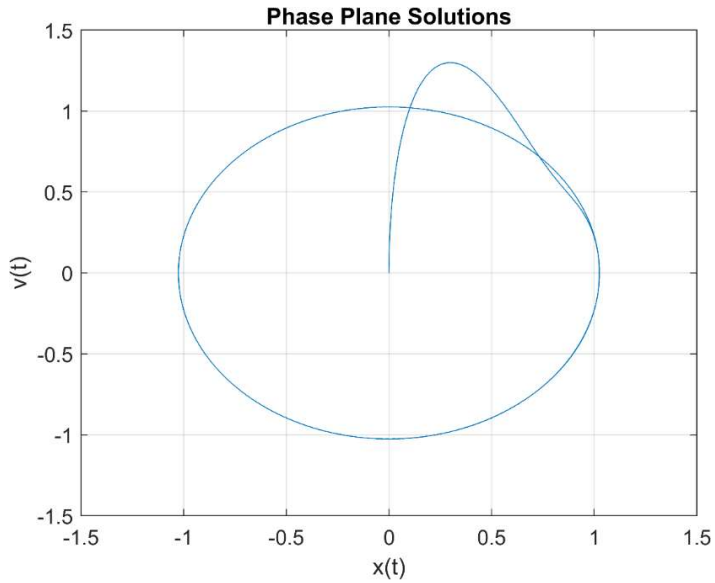

Figure 9 Phase plane solution for PID controlled Zohdy-Harb System

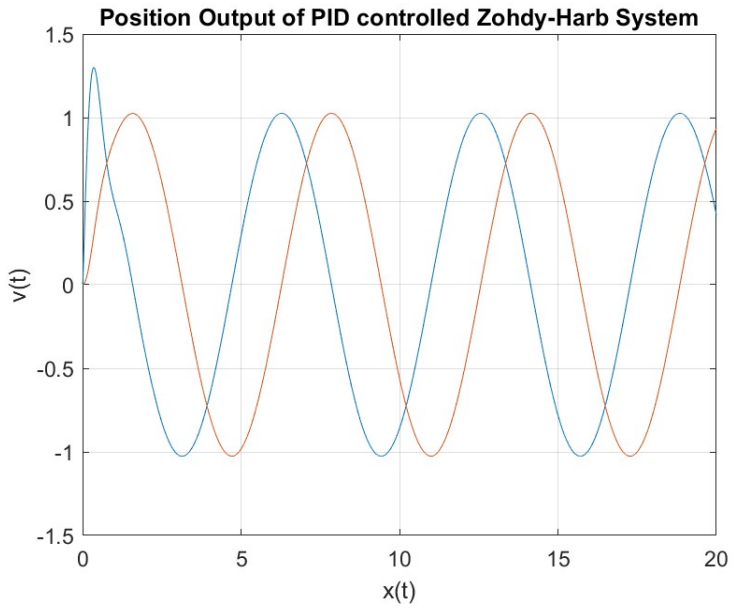

Figure 10 Position Output of PID controlled Zohdy-Harb System

In this chapter we reviewed the results of utilizing PID control in comparison to Lyapunov control. Lyapunov control was shown to be superior in terms of robustness of control and adaptation to disturbances. On the other hand, Lyapunov control was found to include multiple parameters that require fine tuning in order to get the best outcome. Compared to the 3 PID control parameters that require tuning, Lyapunov control might have more than 5 parameters that require tuning and depending on the system complexity the number of parameters can increase. As such we propose and investigate in the next chapters the integration of machine learning to fine tune the controller parameters initially and to continue it tuning depending on the system output and changes.

CHAPTER FOUR

NN PREDICTIVE CONTROL

Data driven control or else known as Machine learning control (MLC) is an area of machine learning, intelligent control, and control theory that solves optimum control problems using machine learning techniques. Principal applications are complicated nonlinear systems for which linear control theory approaches cannot be applied [56].

MLC combines three well-established disciplines: the theory of closed-loop feedback control, machine learning and regression, and turbulent fluid flow-characteristic nonlinear dynamical systems [56]. Over the past several decades, control theory has matured into a science with a solid theoretical foundation and robust numerical techniques. Significant developments have enabled the robust management of systems with sensor noise, external disturbances, and model uncertainty. Modern techniques from control theory have changed the engineering sciences and the industrial scene. Controlling systems with significantly nonlinear dynamics resulting in broadband frequency spectra, a high-dimensional state space, and huge time delays still presents obstacles [57]. MLC begins to address these difficulties by discovering effective control rules utilizing powerful machine learning techniques.

MLC has emerged with solutions to difficulties such as:

- Control parameter identification: If the structure of the control rule is known but the parameters are unknown, MLC translates to parameter identification [56]. A genetic method for improving the coefficients of a PID controller [58] or discrete-time optimal control is one example [27].

- Control design as a first-order regression problem: MLC approximates a generic nonlinear mapping from sensor signals to actuation commands if the sensor signals and optimum actuation command for each state are known. An illustration would be the calculation of sensor feedback from a known complete state feedback. A neural network is a typical method for performing this task[59].
- MLC can also discover arbitrarily nonlinear control rules that minimize the cost function of the plant. In this instance, neither a model, the structure of the control law, nor the optimal actuation command are required. The optimization is based only on the plant-measured control performance (cost function). For this aim, genetic programming is a highly effective regression method.
- Using reinforcement learning, the control rule may be continuously changed in response to measurable changes in performance (rewards).

In addition to methodological commonalities with other data-driven controls like artificial intelligence and robot control, MLC controls include neural network control, genetic algorithm-based control, genetic programming control, and reinforcement learning control. MLC has been effectively used in a wide variety of nonlinear control challenges, allowing researchers to investigate unknown and frequently surprising actuation processes.

Applications include:

- Satellite's ability to regulate their altitude [60].
- Temperature regulation in the building
- Feedback turbulence management is the eighth point[58, 61]
- A vehicle that can be driven remotely while submerged in water[62].

In this chapter we review the different types of machine learning control, present the data set and the results of utilizing only a data driven approach to non-linear control systems.

4.1 Types of Machin Learning Control

The review paper written by PJ Fleming and RC Purshouse provides a summary of a great number of further engineering MLC applications [63]. In the same vein as other generic nonlinear approaches, MLC does not come with any guarantees regarding its convergence, optimality, or resilience across a variety of operating circumstances.

Numerous turbulence control issues cannot be well characterized by linear models, have enormous state spaces, and suffer from temporal delays between actuators and sensors due to nonlinear convective fluid dynamic processes. Consider the aerodynamic drag reduction of a vehicle having actuators at the rear, pressure sensors scattered throughout the vehicle, and intelligent feedback control logic. While the numerical simulation of the underlying dynamics given by the Navier–Stokes equations takes days or weeks to complete, the control system requires actuation decisions to be made on a scale that is measured in milliseconds. Reduced-order models that combine nonlinearities, multiscale phenomena, and actuation effects have evaded many serious attempts and are likely to stay elusive for decades. It is possible that there is no suitable paradigm for robust control design. The bulk of turbulence control experiments utilize open-loop forcing, such as periodic blowing, incrementally adjust an open-loop vii approach, or stabilize an underlying laminar solution, such as a laminar boundary layer on an aircraft wing. In numerical research, feedback control that responds in real time to

dominating flow structures is possible, although in real-world trials with turbulent flows, it is uncommon.

4.2 Genetic Algorithms

The population of possible solutions to an optimization issue may improve over time thanks to the use of a genetic algorithm. In general, potential solutions are encoded in binary as strings of 0s and 1s, although other encodings are also feasible. Each prospective solution includes a set of properties that may be modified and changed in some way [64].

Typically, evolution begins with a population of randomly created individuals and is an iterative process, with the population in each iteration being referred to as a generation. In every generation, the fitness of every member of the population is evaluated; fitness is typically the value of the objective function in the optimization problem being addressed. The fittest people are randomly picked from the present population, and their genomes are transformed to create a new generation. In the subsequent algorithm iteration, the new generation of candidate solutions is utilized. Commonly, the method finishes after either a maximum number of generations has been created or a suitable fitness level for the population has been attained.

Standard genetic algorithms require a genetic representation of the solution domain and an evaluation function for the solution domain. Each possible solution is represented as an array of bits by convention. Similar functionality may be achieved with arrays of different kinds and structures. Due to their constant size, the components of these genetic representations may be readily aligned, which simplifies straightforward

crossover procedures. Variable length representations may also be employed; however, crossover implementation is more complicated in this instance. In genetic programming and evolutionary programming, tree-like and graph-like representations are investigated, but in gene expression programming, both tree-like and graph-like representations are investigated. After defining the genetic representation and fitness function, a genetic algorithm (GA) initializes a population of solutions and then improves it by repeatedly applying mutation, crossover, inversion, and selection operations. The population size varies based on the nature of the problem, but often encompasses many hundreds or thousands of potential solutions. Typically, the starting population is produced at random, providing for the whole spectrum of viable solutions. Sometimes, solutions are in regions where optimum solutions are likely to be discovered.

On the other hand, there are restrictions associated with the employment of a genetic algorithm in comparison to other optimization algorithms. Such as in artificial evolutionary algorithms, repeated fitness function assessment for complicated situations is often the most prohibitive and restricting component. Finding the ideal solution to complicated, high-dimensional, multimodal issues sometimes necessitates very costly assessments of fitness function. In real-world situations, such as structural optimization challenges, the whole simulation of a single function evaluation might take several hours to many days. Typical optimization techniques cannot address such problems. In this situation, it may be essential to forego an accurate assessment in favor of a computationally efficient approximation. Evidently, combining approximation models may be one of the most promising ways to employ GA persuasively to tackle complicated real-world issues.

Genetic algorithms are not very good at scaling effectively with increasing complexity. Thus, when the number of elements subjected to mutation is high, the size of the search space typically increases exponentially[65]. This makes it incredibly challenging to apply the method to issues such as the design of an engine, a home, or an airplane. For such issues to be amenable to evolutionary search, they must be represented in the simplest manner feasible. Consequently, evolutionary algorithms often encode ideas for fan blades rather than engines, building forms rather than complete construction plans, and airfoils rather than whole aircraft designs[66]. The second concern of complexity is how to preserve parts that have evolved to represent excellent solutions from additional damaging mutation, especially when their fitness evaluation depends on their ability to mix effectively with other parts. The best solution is relative to other options. In consequence, the stop condition is not always evident in every issue.

In genetic algorithms and genetic programming, diversity is essential since crossing across a homogenous population does not provide novel solutions. Due to a larger dependence on mutation, variety is not required in evolutionary strategies and programming.

It is challenging to operate on dynamic data sets, since genomes converge early on towards answers that may no longer be viable for subsequent data. Increasing genetic diversity and preventing early convergence are two strategies that have been proposed as potential solutions to this problem. These strategies can be implemented in one of two ways: either by increasing the probability of mutation whenever the solution quality decreases, or by periodically introducing entirely new, randomly generated elements into the gene pool. The so-called "comma technique" is another method that may be used to

implement evolution strategies and programming. In this method, parents are not kept and new parents are only selected from their children. This may be more effective for solving dynamic issues. There is no method for a genetic algorithm to converge on a solution when the sole fitness measure is a single right/wrong measure. In such circumstances, a random search may identify a solution just as rapidly as a GA. Nonetheless, if the success/failure experiment may be repeated with different outcomes, the ratio of successes to failures is a valid fitness metric.

Other optimization methods may be more effective than genetic algorithms in terms of speed of convergence for certain optimization issues and problem cases. Alternative and complementary algorithms include evolutionary techniques, evolutionary programming, simulated annealing, Gaussian adaptation, hill climbing, swarm intelligence, and approaches based on integer linear programming. Evolution techniques are also known as evolutionary programming. The degree of familiarity with the problem is a factor in determining whether genetic algorithms may be used to it; problems that are well-known often have better, more specialized solutions.

4.3 Reinforcement learning

Reinforcement learning (RL) is a subfield of machine learning concerned with determining how intelligent agents should operate in an environment to maximize the concept of cumulative reward. Reinforcement learning is one of the three fundamental paradigms of machine learning, along with supervised learning and unsupervised learning[67].

Reinforcement learning differs from supervised learning in that it does not need the presentation of labeled input/output pairings or the explicit correction of suboptimal behaviors. Instead, the emphasis is on striking a balance between exploration and exploitation. Combining the benefits of supervised and RL algorithms, partially supervised RL algorithms may incorporate the benefits of both types of algorithms[68].

Typically, the environment is given in the form of a Markov decision process (MDP), since several reinforcement learning methods for this context use dynamic programming approaches[69]. The primary distinction between classical dynamic programming techniques and reinforcement learning algorithms is that the latter do not presuppose prior knowledge of an accurate mathematical model of the MDP and target big MDPs for which precise approaches become impractical.

The issues of reinforcement learning have also been explored in the theory of optimal control, which is more concerned with the existence and characterization of optimum solutions and methods for their accurate calculation than it is with learning or approximation, particularly in the absence of a mathematical model of the environment. This is because learning or approximation is difficult to achieve when there is no mathematical model of the environment. In the fields of economics and game theory RL refers to a technique that may be used to illustrate how equilibrium could develop despite the presence of restricted rationality. The utilization of samples to maximize performance and the use of function approximation to cope with huge environments are the two components that contribute to the effectiveness of reinforcement learning. Because of these two fundamental aspects, reinforcement learning may be implemented in vast settings in the following contexts:

There is a model of the environment accessible, but there is no analytical solution currently available. The only method to get data about the environment is to engage in activity inside it[70]. The first two of these issues might be seen as planning issues, while the third could be interpreted as an authentic learning issue. On the other hand, reinforcement learning transforms both planning difficulties into challenges of machine learning:

- Nonlinear
- Unknown dynamics
- High Dimensional
- Limited measurements

4.4 Neural Network Control

Neural network control or Data-driven control systems are a large class of control systems in which the identification of the process model and/or the design of the controller are totally dependent on experimental plant data[71].

In many control applications, attempting to develop a mathematical model of the plant is a difficult and time-consuming task that requires the involvement of process and control experts. This issue is circumvented by data-driven approaches, which fit a system model to the experimental data collected and select it from a particular model class. This model can then be used by the control engineer to develop a suitable controller for the system. However, it is still challenging to construct a simple yet trustworthy model for a physical system that incorporates only the system dynamics that are relevant to the control specifications such as in Figure 11. Without requiring a specified model of the

system, direct data-driven approaches permit tuning of a controller belonging to a certain class. On this basis, it is also possible to simply weight process dynamics of interest within the control cost function, while excluding dynamics of no relevance[72].

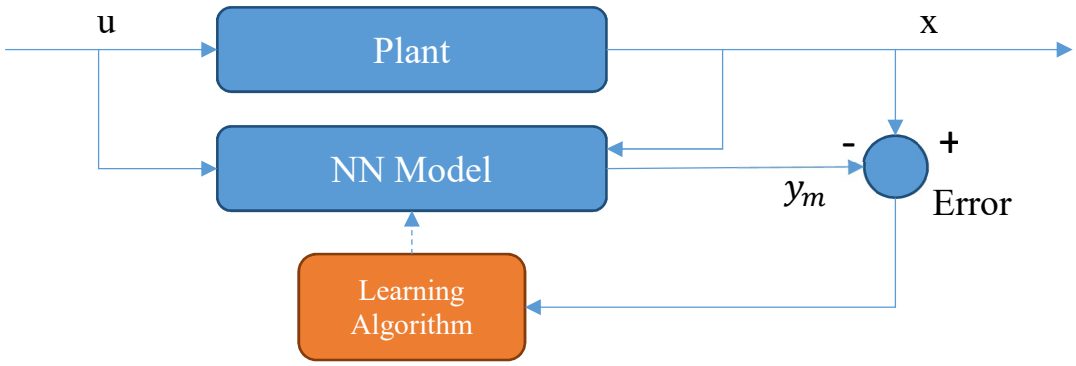

Figure 11 System Controller Setup

A neural network may be conceptualized as a mathematical function that translates an input set to a desired output set. Neural networks are composed of the following elements Figure 12:

- One input layer, x,
- One or more hidden layers,
- One output layer, ŷ,
- A set of weights and biases between each layer, W and b,
- Activation function for each hidden layer, σ.

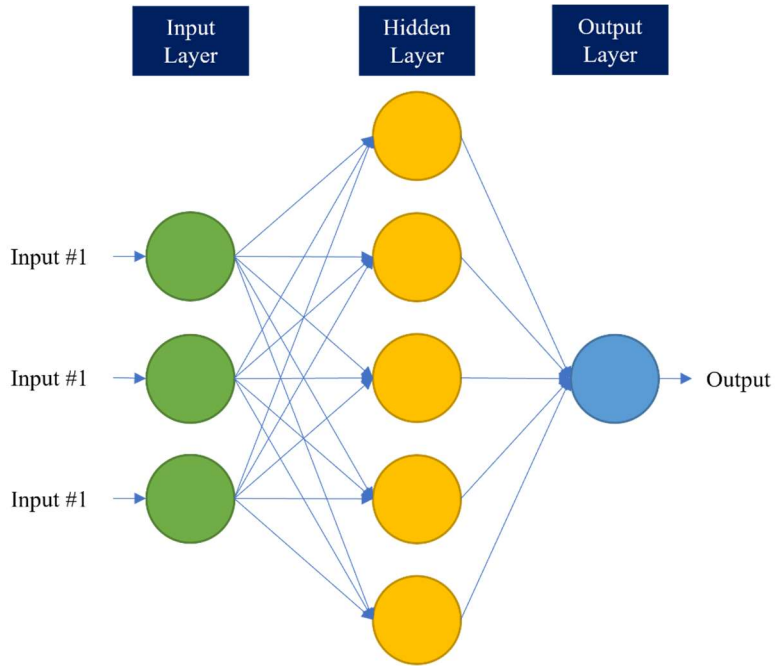

Figure 12 Neural Network Components

4.5 Model Predictive Control

Model Predictive Control works by iteratively optimizing a plant model over a finite time horizon. As shown in Figure 13 the plant state is sampled at time t, and a cost-minimizing control strategy for a relatively short time horizon in the future is determined using a numerical minimization algorithm at [t, t+T] [73]:

$$J = \sum_{i=1}^{N} w_{A_i}(r_i - x_i)^2 + \sum_{i=1}^{N} w_{b_i}\Delta u_i^2 \quad (27)$$

x_i: Variable under control at value i

r_i: Variable under refrence at value i

u_i: Variable under actuation at value i

w_{Ai}: weighting coefficient for x_i

w_{bi}: weighting coefficient for u_i

An online calculation is utilized to investigate state trajectories originating from the current state and determined through the solution of Euler-Lagrange equations a cost-minimizing control strategy until time t+T. Only the first step of the control strategy is executed, after which the plant state is sampled again and the calculations are completed, beginning with the new current state, providing a new control and anticipated state path. The prediction horizon continues to move forward therefore the given name receding horizon [74].

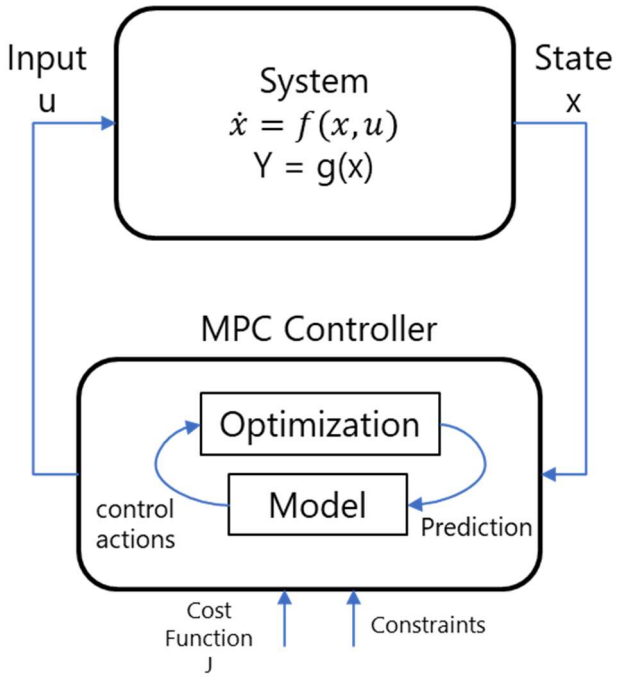

Figure 13 Model Predictive Control process

4.6 NN Predictive control Zohdy-Harb System

The neural network predictive controller on the other hand employs a neural network model of a nonlinear plant to forecast future plant performance. The controller then determines the control input that will optimize the performance of the plant over a

defined future time horizon. Model predictive control begins with the identification of the neural network plant model. The controller then uses the plant model to anticipate future performance.

Training a neural network to reflect the forward dynamics of the plant is the initial stage of model predictive control. The prediction error between plant output and neural network output is employed as the training signal for the neural network. The procedure is depicted in the following Figure 14:

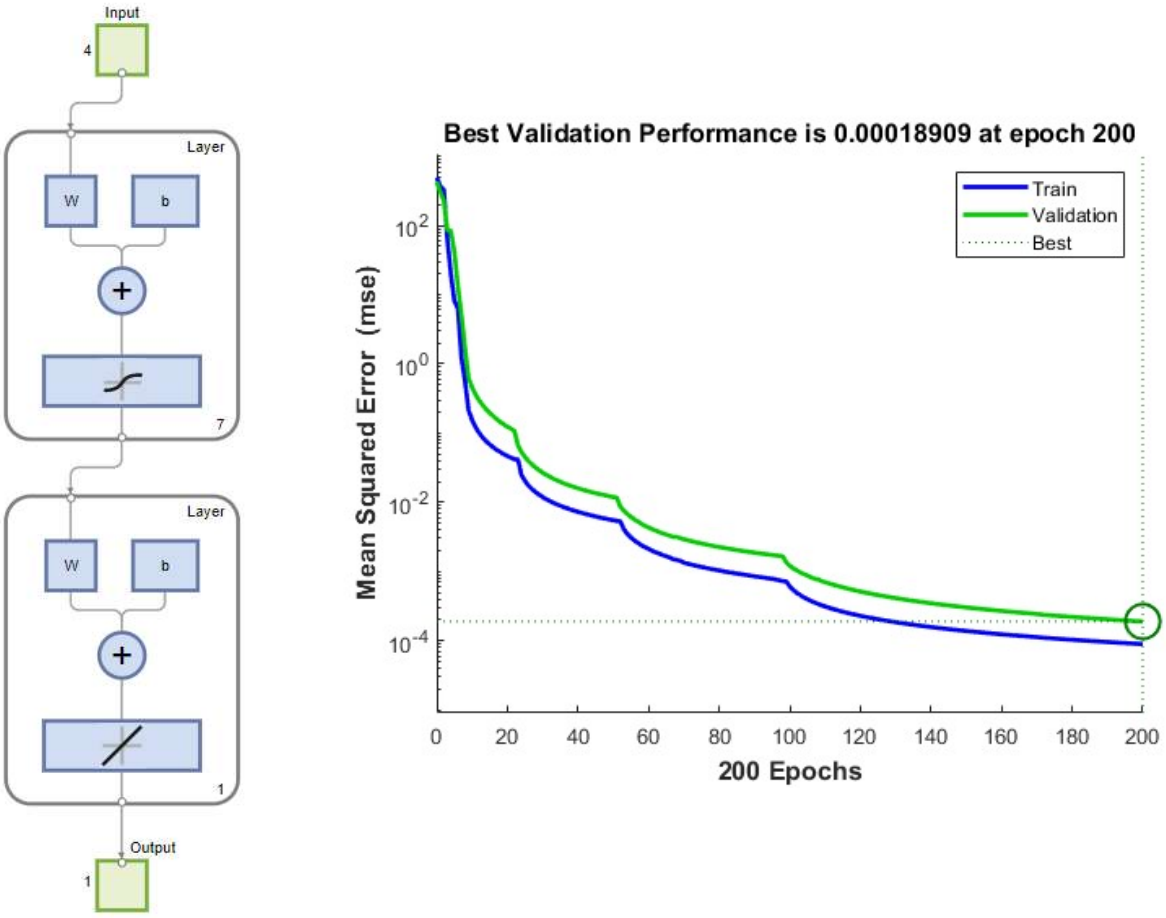

Figure 14 (Left) Network Architecture and (right) Mean Square Error Diagram as the NN is trained

The NN predictive control method is based on the receding horizon technique. The neural network model predicts the plant response over a specified time horizon. The p reductions are used by a numerical optimization program to determine the control signal that minimizes the following performance criterion over the specified horizon. where N1, N2, and Nu define the horizons over which the tracking error and the control increments are evaluated. The u′ variable is the tentative control signal, yr is the desired response, and ym is the network model response. The ρ value determines the contribution that the sum of the squares of the control increments has on the performance index. 8000 training samples were collected for training the network. 45 nodes with 1 hidden layer NN. A sampling interval of 0.2 seconds. The training parameters were 200 training epochs with Levenberg-Marquardt backpropagation method. The Levenberg-Marquardt technique was created to approximate second-order training speed without requiring the Hessian matrix to be calculated. When the performance function takes the shape of a sum of squares, it is referred to as a sum of squares performance function as is typical in training feedforward networks. Figures 15 and 16 present the NNPC output where the max error difference is 27 mm. On the other hand, when the scalar γ equals zero, this is effectively Newton's approach, which makes use of an estimated Hessian matrix. When a big value is used for, the calculation changes to a gradient descent with a short step size. This approximation to the Hessian matrix is used by the Levenberg-Marquardt method in the subsequent Newton-like update:

$$x_{S+1} = x_S - [J^T J + \gamma I]^{-1} J^T e \tag{28}$$

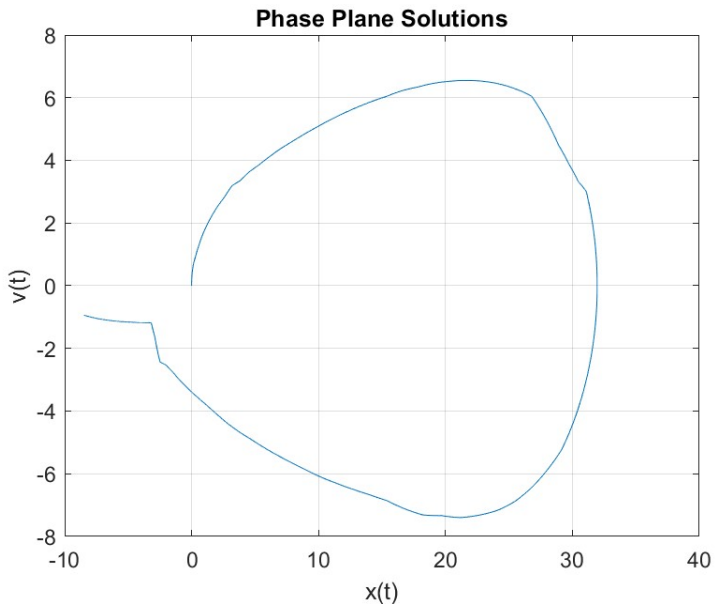

Figure 15 The Phase plane solution result under NNPC on Zohdy-Harb System

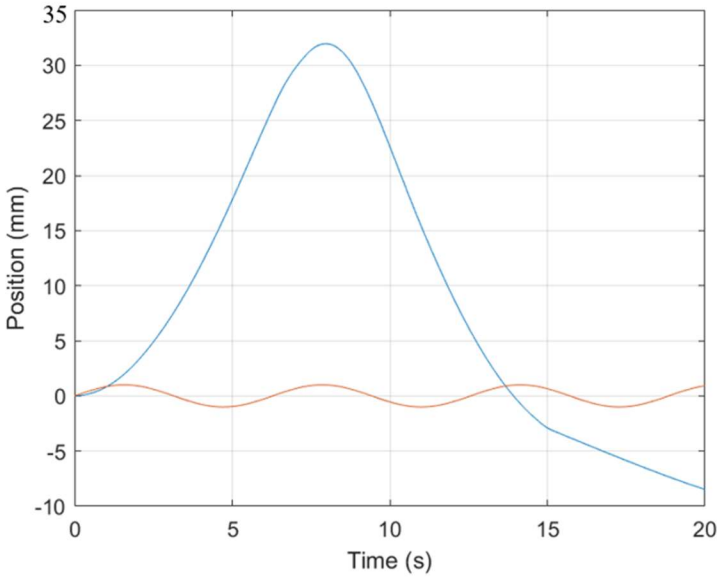

Figure 16 The position output (blue) compared to the reference Sin curve in (red)

Since Newton's approach is both quicker and more precise in the vicinity of an error minimum, the goal is to make the transition to Newton's method as soon as practically practicable. Therefore, after each successful step, is dropped (which results in a reduction in the performance function) and is only increased if a tentative step would

result in an increase in the performance function. In this manner, the performance function will consistently be improved during the course of the algorithm's iterations.

4.7 Applying NN Predictive control on Duffing System

In this section, we apply the NN predictive controller method to the Duffing nonlinear system and study the results compared to the PID and Lyapunov controller. Figure 17 presents the phase plane solution when applying NN predictive controller on the Duffing non-linear system where the system is shown to be unable to maintain precise output compared to reference Sine signal. Figure 18 presents the system as it doesn't accurately follow the reference signal as designed. The error increases at 14 seconds with the introduction of the duffing system nonlinear dynamics. Compared to PID and Lyapunov control NNPC performed the poorest in terms of accuracy and robustness.

4.8 Applying NN Predictive control Van der Pol

In this section, we apply the NN predictive controller method to the Van der pol nonlinear system and study the results compared to the PID and Lyapunov controller. Figures 19 and 20 show the simulation results when applying NN predictive control on Van Der Pol control. The controller was able to perform better in terms of accuracy compared to Duffing and Zohdy-Harb system but still the controller robustness and accuracy is substandard to PID and Lyapunov control.

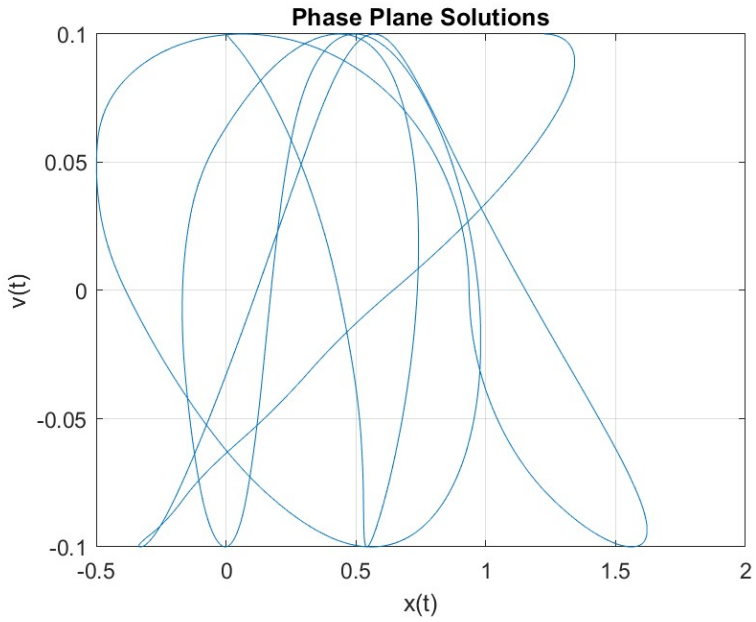

Figure 17 The Phase plane solution result under NNPC on Duffing System

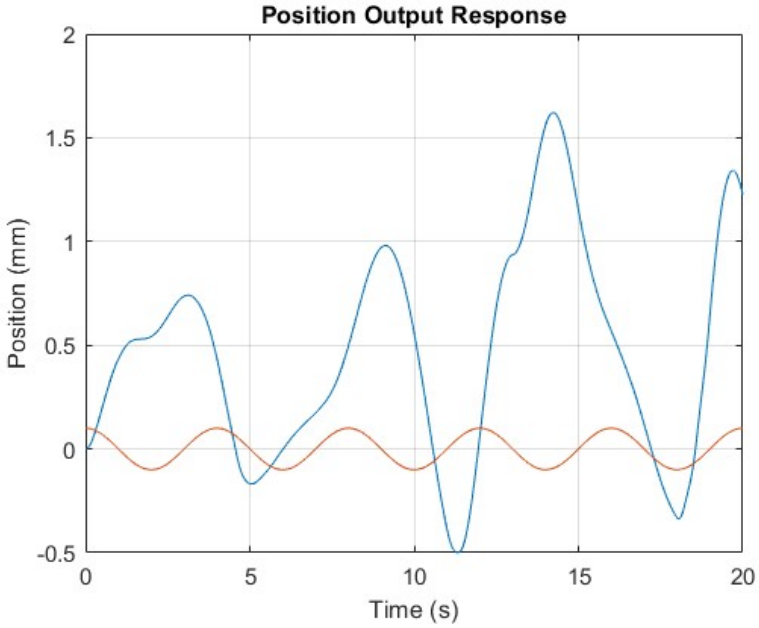

Figure 18 The position output (blue) compared to the reference curve in (red) under NNPC

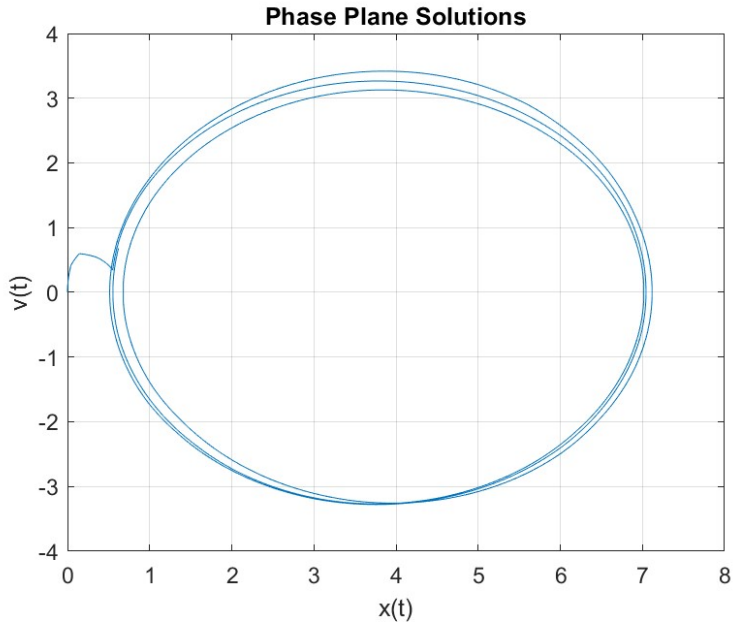

Figure 19 The Phase plane solution result under NNPC on Van der Pol System

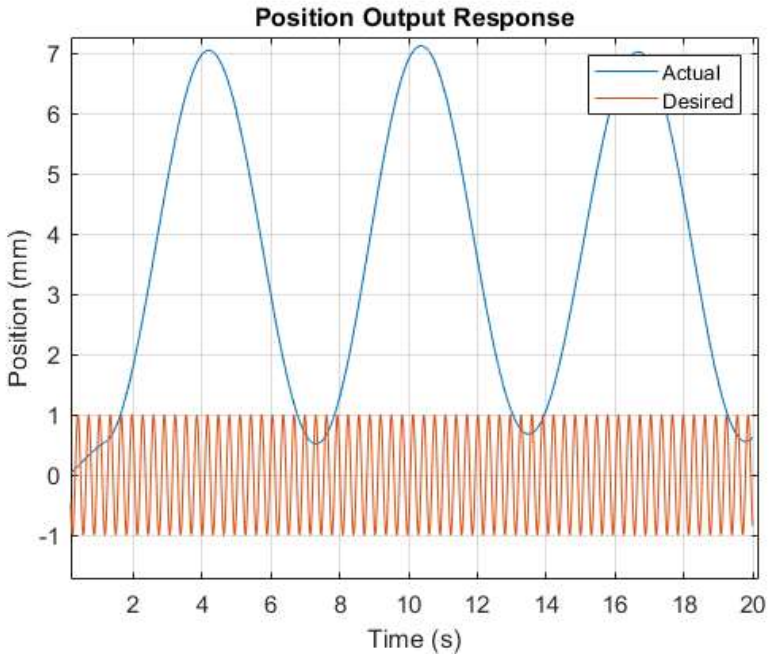

Figure 20 The position output (blue) compared to the reference curve in (red) under NNPC Duffing System

In this chapter we present the different approaches to data driven control. NN predictive control is applied on Duffing, Van der Pol and Zohdy-Harb systems. The approach is shown to be successful in learning the system model but when a sinusoidal input signal with high frequency is provided for control this approach was found to be unsuccessful. The results in this chapter would provide a comparison point to the integration of Deep Learning with Lyapunov nonlinear control.

CHAPTER FIVE

LYAPUNOV DEEP LEARNING CONTROL

In the previous chapter we examined different types of machine learning control and their benefits and drawbacks compared to conventional linear and non-linear control systems such as PID and Lyapunov control. In this chapter we present a resolution that would combine the upside of both conventional non-linear Lyapunov control and data driven machine learning control. The initial selection of system and controller settings does not always ensure continuing system stability and performance. This is primarily because of the introduction of unanticipated system disruptions or the absence of knowledge regarding the system's dynamics. In this study, we provide an innovative method for identifying early failure signs of non-linear highly chaotic systems and, as a result, make predictions about those systems. The strategy proposed keeps a constant watch on the signals coming from the system and the controller. The re-calibration of the system's parameters as well as the controller's is activated in response to a predetermined set of conditions designed to ensure the continued reliability of the system while minimizing any impact on its operating speed, expected result or minimum amount of computing power necessary. The parameter values that would work best are those that the deep neural model predicts. Combat the system instability that is to be anticipated. In order to provide evidence of the viability of the suggested strategy, it is implemented in the context of the combination of Duffing-Van der pol oscillators that is non-linear and complicated. Additionally, the strategy is evaluated using a variety of simulated conditions. System and controller settings are either initially chosen wrongly or,

alternatively, system parameters are altered while the system is operating, or new system parameters are introduced. While the program is being executed, the dynamics of the system are introduced so that efficiency and response time may be measured.

5.1 Machine Learning Lyapunov Control

In the Lyapunov Control method an exhaustive search issue may be used to describe the process of picking the appropriate controller parameters for the purpose of reducing the error produced by the model. In the Duffing Lyapunov Control and Analysis in Chapter 3 the following control law was deduced

$$\frac{\delta}{2}\frac{\beta_1}{\beta_2}e + \frac{\delta}{2}\dot{e} + \ddot{z}_{des} + \varphi\dot{x} + \delta x + \gamma x^3 + \frac{\beta_1}{\beta_2}\dot{e} - \frac{cos\, t}{\beta_2} \tag{29}$$

There is a significant effect on the controller output and error with the change of parameters β_1, $\beta_2$and δ. With PID control there are only 3 parameters to be tunned while with non-linear controllers such as Lyapunov while the output presents a much lower error there are several parameters that require tunning to reach peak controller performance. In this chapter we present a model for integrating deep machine learning with Lyapunov non-linear control. Initially, the model is executed while the error rate of the system is tracked. If the error is more than the set threshold, the neural network is asked to find control parameters that, given the present state of the system, would result in a reduction in the error produced by the system. The algorithm that governs the entire operation is outlined in the following paragraphs. The network shown in Figure 20 is first trained, and then it is used to make predictions about future Betas. These predictions include Beta 1 (β1) and Beta 2 (β2) if the system error (sys err) reaches a specific

threshold (0.8). The network is then provided with the present state of the system in terms of time (t), location (p), and velocity (v), as well as the randomly generated (β1) and its corresponding (β2), and it forecasts the amount of error that will occur in the system. If the anticipated error is lower than the actual error of the system, then the system is updated with the newly generated Betas that are theorized to minimize the error. This happens only if the expected error is lower than the actual error of the system.

At the end of each cycle, the newly created Betas and the system parameters that correspond to them are saved. After every 100 iterations, the system is checked for disturbances by computing the average of the system error that occurred over the preceding 100 iterations (avg sys err). If the current error is more than the threshold value, the system is considered to be perturbed. . In order to combine both types of control Figure 21 and Figure 22 describe the incorporation of the data received from the MLC as an input to the Lyapunov controller. The network is retrained using a subset of the old data (Memo) utilized in the training process by using the prior system average (prev sys avg). Prior training in addition to the information gathered from the preceding one hundred repeated attempts (new data). The reason for continuing to use the previous data as well as the additional information is to help avoid forgetting anything catastrophic such referred to in footnote [75]. The practice of recalling information from one's past is commonly referred to as memory that can update itself on its own and was first shown and experimented with in [76], [77], and [78]. The point being made here is that even in situations where there are disturbances; they may not last for very long, though, and as a consequence of this, we don't want the network to lose its capacity to forecast the system error in the event that the system settings are restored. To return to the norm after the

network was retrained on many occasions. A Memory ten percent of the previous training data to maintain network output safety.

```
while n > 2000 do
    sys err = current system error;
    if sys err > 0.8 then
        while True do
            s1 = randomly generated;
            s2 = - s1/8;
            predicted sys err = predict (t, V, p, v, s1,
             s2);
            if predicted sys err < sys err then
                Break;
            end
        end
        update controller (s1, s2);
        new data = Save (t, V, p, v, s1, s2,
         new–sys err);
        avg sys err = avg sys err + new sys err;
    end
    if n = = 0 then
        if avg sys–err / 100 > prev avg then
            retrain network (Memo + new data);
            Memo = Memo + 10% of the new data;
            prev sys avg = avg sys err;
            Clear new data;
            Clear avg sys err;
        end
    end
end
```

Figure 21 Network Algorithm

5.2 Duffing System

The mathematical model presented in equation 31 presents the Duffing Oscillator.

$$\ddot{x} + \varphi\dot{x} + \delta x + \gamma x^3 = \cos t + u \tag{30}$$

To find the control law u

$$u = \frac{\delta}{2}\frac{\beta_1}{\beta_2}e + \frac{\delta}{2}\dot{e} + \ddot{z}_{des} + \varphi\dot{x} + \delta x + \gamma x^3 + \frac{\beta_1}{\beta_2}\dot{e} - \frac{\cos t}{\beta_2} \tag{31}$$

In this section we combine the Lyapunov control discussed in chapter 3 to the deep learning machine learning architecture discussed in chapter 4 Figure 22. The

combination of both types of control would yield a highly adaptable Deep Learning non-linear control. Initially, the model is executed multiple times with varying β_1& β_2. The system error, velocity, displacement, and time are then gathered and utilized to construct a dataset. The dataset is applied to train the predictor model. If the current state of the system and the Betas that are being used result in a high error, a new pair of Betas are used on the predictor model to predict the behavior of the system; if the new Betas are predicted to result in a reduction of the error, then use them; otherwise, continually recommend new β_1& β_2. The trained predictor model forecasts the error for any given set of β_1& β_2 proposed by the algorithm. Several iterations of the Deep Neural Network Architecture and Dataset were made before reaching the architecture and dataset with the highest accuracy and least error. In this section we are going to present the different iterations to selection and results of each iteration. Since four hundred and sixty different iterations were made only iterations with significant impact to design will be presented. Figure 23 depicts the architecture of the final Deep Neural Network. There are 5 distinct blocks that make up the network. It was discovered that the performance of the DNN is negatively impacted by using a lesser number of layers but using a higher number of layers had no impact on the performance but did have a negative impact on the DNN's efficiency. A block is made up of three layers: a fully connected convolution layer, a batch normalization layer, and a rectified linear unit (RELU) in Equation 32. The fully connected convolution layer comes first.

$$\text{output} = \max(0.0,\text{input}) \tag{32}$$

Because it ensures that activations are neither excessively high or low, the batch normalization layer enables us to employ greater learning rates.

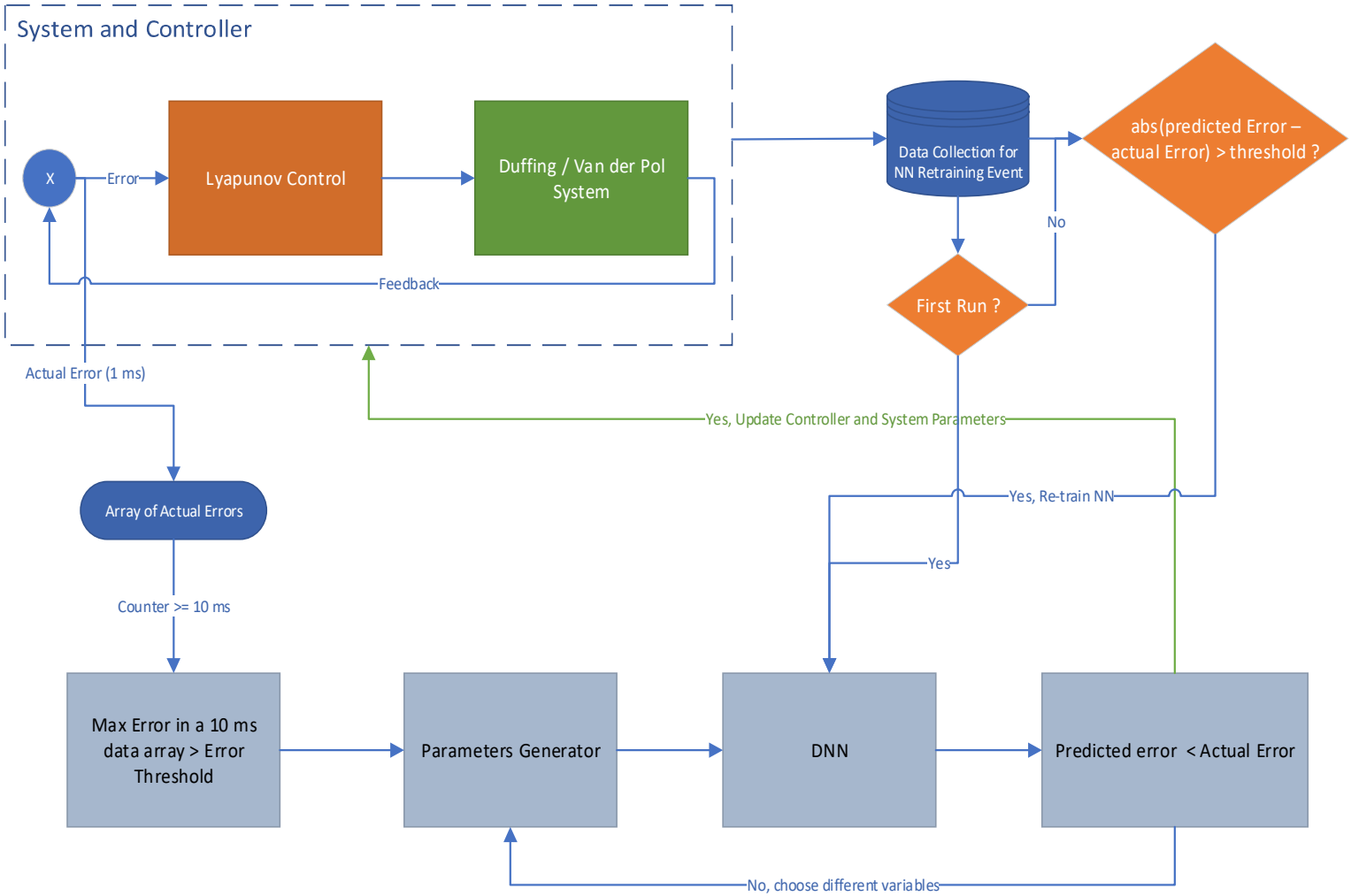

Figure 22 Network retraining flow chart

5.3 Deep Neural Network Architecture

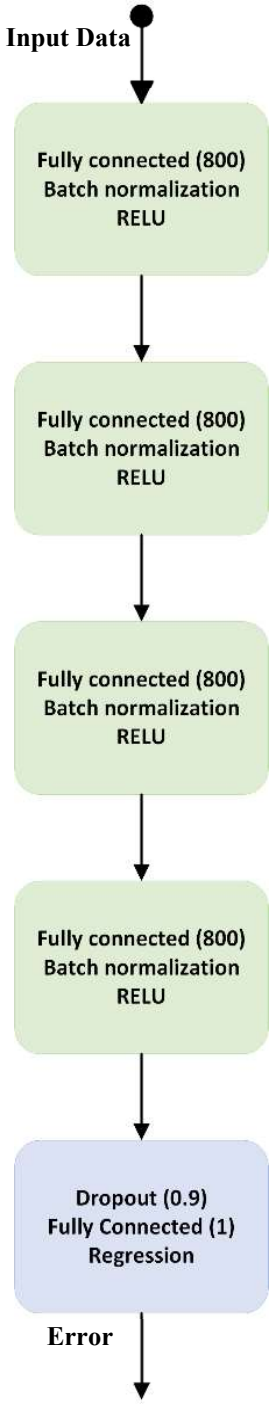

Figure 23 Neural network Architecture initial design

The RELU is almost a linear function since it is a piece-wise linear function that is composed of two linear components. This makes it very close to being a linear function. This feature increases its capability to preserve many of the properties that make linear models easy to optimize with gradient-based methods and the properties that make linear models generalize well. In addition, this feature increases its capability to preserve many of the properties that make linear models easy to optimize. In addition to the fully connected convolution layer, the final block consists of a dropout layer as well as a regression layer that is used to anticipate error.

The issue of overfitting was the motivation behind the development of the regularization technique known as dropout [79]. The overfitting problem occurs when the neural network learns every minor detail in the training data. Because of this behavior, the network will have a high level of accuracy when processing the data used for training, but it will have a very low level of accuracy when processing the data used for testing. This indicates that the network is unable to generalize on data that was not anticipated. The dropout technique is being considered as a potential solution to this issue. The dropout approach will randomly ignore a certain number of the layer's outputs. As a consequence of this, every update that is made to a layer while it is being trained is done so with a different "perspective" on the layer that is configured. The mean squared error loss is what is calculated by the regression layer. In equation 34, a function known as F is taught to the network through training.

$$F : (t, x, v, \beta_1, \delta) \rightarrow e \tag{32}$$

Where t is time, x is the position, v is the velocity, β_1 and δ are controller parameters.

5.3.1 Dataset (Trail 1)

During the first trail of building the Deep Neural Network training dataset the following parameters were collected

1. Time
2. Velocity Output
3. Stability

The parameter δ is changed after each time reset and restart of the system. Datapoints were collected at a one millisecond intervals. The dataset consisted of 10,000 data points divided into a training portion of sixty percent and a validation portion of forty percent as presented in Figure 24.

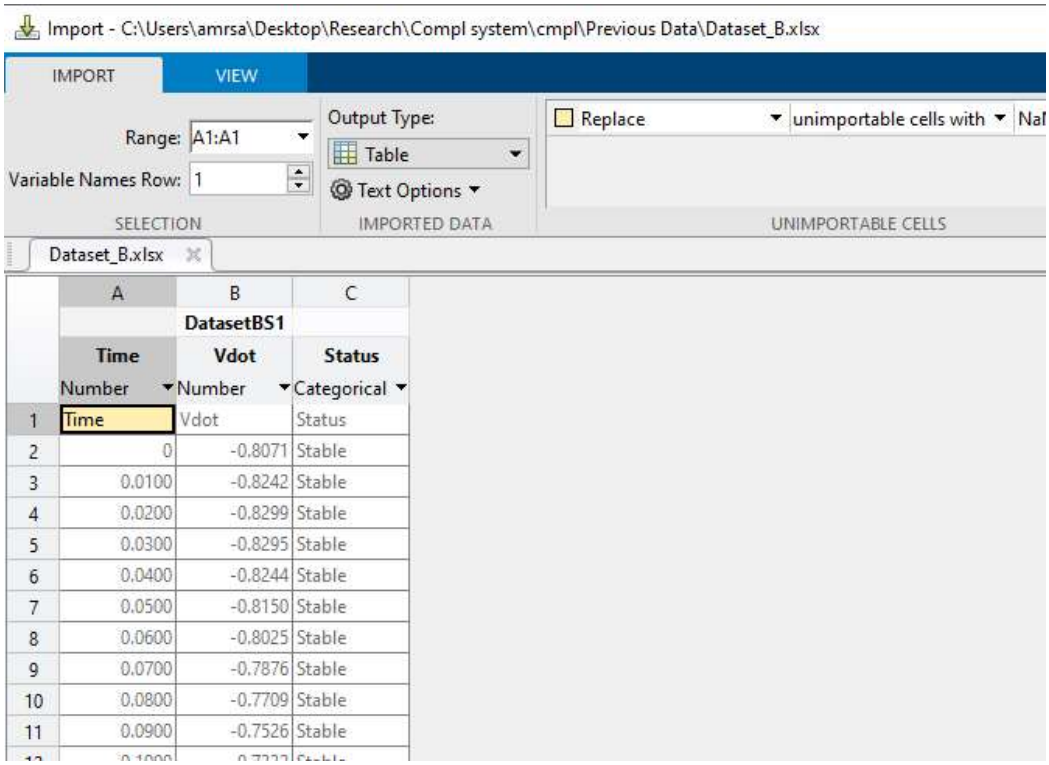

Figure 24 Dataset utilized in trail 1

Consistent revisions to the dataset were made according to the error rate of change and error gradient re-calculated every 10 seconds to activate the Deep Learning Network error estimation. 250 Epochs per learning iterations with 3 iterations in total. The lack of data features was the main attribute found to affect the outcome. Over the next dataset collection, we will focus on the number of features collected and their effect on the error output.

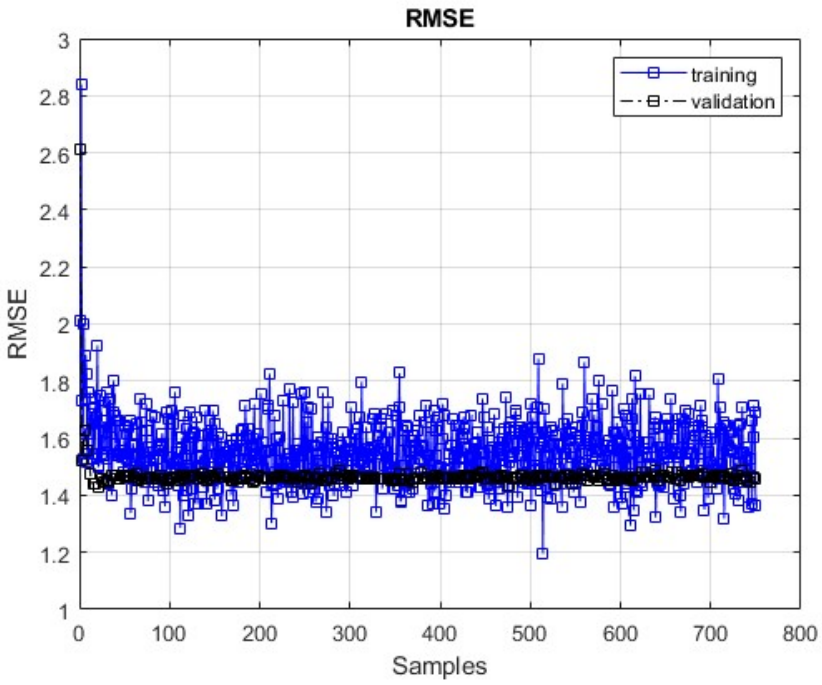

Figure 25 The training and validation RMSE comparison of the dataset in trail 1

5.3.2 Dataset (Trail 12):

In trail 12 the number of epochs is reduced to 50 with 3 iterations due to the lack of learning during the remaining epochs during the previous trials. The following parameters were sampled at ten millisecond intervals:

1. Time
2. Control Signal (u)

3. Position Output
4. Velocity Output
5. Output Error (Numeric)
6. β_1 & β_2

Dataset1

493x7 table

	1 Time	2 Position	3 Velocity	4 Beta1	5 Beta2	6 u	7 Error
1	0.0566	0.2977	6.0088	3.2000	-0.8000	21.3750	-0.9170
2	0.0818	0.4403	5.1892	3.2000	-0.8000	-26.1950	-0.8559
3	0.0927	0.4940	4.6507	3.2000	-0.8000	-50.6650	-0.8169
4	0.1139	0.5797	3.3968	3.2000	-0.8000	-102.4600	-0.7240
5	0.1351	0.6366	1.9354	3.2000	-0.8000	-150.8300	-0.6147
6	0.1560	0.6606	0.3423	3.2000	-0.8000	-182.4400	-0.4982
7	0.1741	0.6536	-1.1257	3.2000	-0.8000	-189.1300	-0.3965
8	0.1930	0.6174	-2.7139	3.2000	-0.8000	-173.1900	-0.2956
9	0.2123	0.5491	-4.3599	3.2000	-0.8000	-136.7800	-0.2047
10	0.2193	0.5165	-4.9548	3.2000	-0.8000	-120.3400	-0.1761
11	0.2437	0.3709	-6.9517	3.2000	-0.8000	-60.9520	-0.0997
12	0.2631	0.2227	-8.2577	3.2000	-0.8000	-23.3720	-0.0689
13	0.2880	0.0064	-8.8355	3.2000	-0.8000	2.5153	-0.0644
14	0.3102	-0.1823	-8.0029	3.2000	-0.8000	12.4730	-0.0751
15	0.3308	-0.3346	-6.7537	3.2000	-0.8000	23.6720	-0.0861
16	0.3490	-0.4472	-5.6278	3.2000	-0.8000	41.6680	-0.0979
17	0.3725	-0.5626	-4.1932	3.2000	-0.8000	77.9070	-0.1215
18	0.3919	-0.6322	-2.9710	3.2000	-0.8000	114.6000	-0.1507

Figure 26 Dataset utilized in trail 12

The Root Mean Square Error (RMSE) is a good measure of the distribution of prediction points from the regression line.

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N} (x_i - \hat{x}_i)^2}{N}} \tag{33}$$

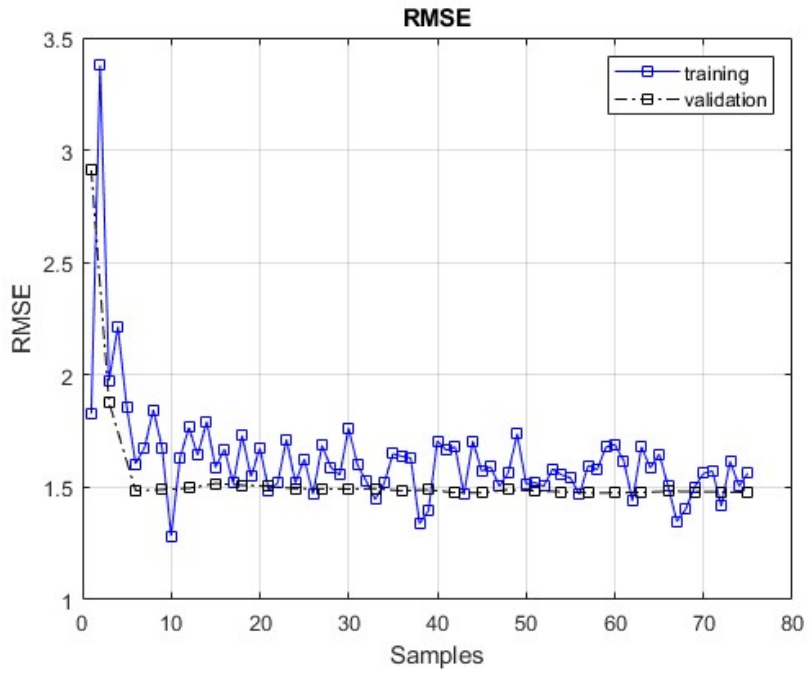

Figure 27 The training and validation RMSE comparison of the dataset in trail 12.

As shown in Figure 26 since the validation data and training data are overlapping at the end points no overfitting or underfitting has occurred in trail B.

5.3.3 Dataset (Trail 116):

In trail 116 the number of epochs is reduced to 30 epochs with 3 iterations. The time element is excluded from the dataset to ascertain the relation between the parameters and error directly without the time dimension. The following parameters were sampled at ten millisecond intervals:

1. Control Signal (u)
2. Position Output
3. Velocity Output
4. Output Error (Numeric)
5. β_1 & β_2

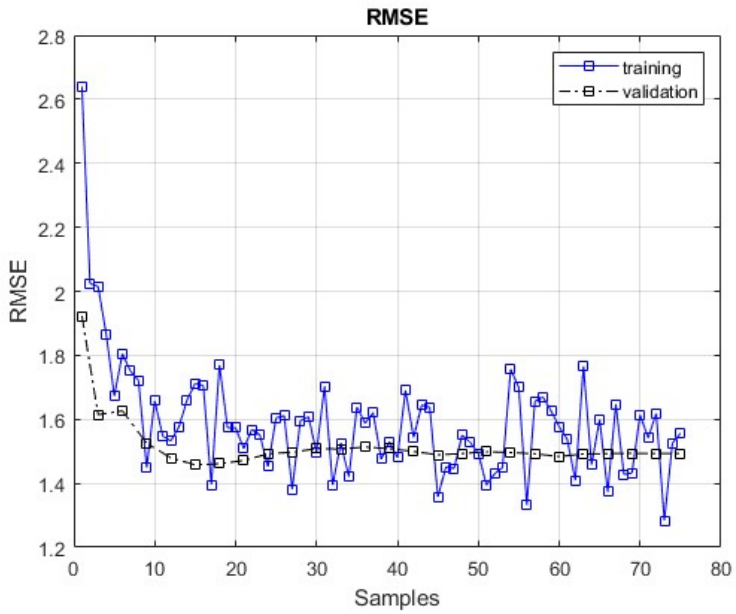

Figure 28 The training and validation RMSE comparison of the dataset in trail 116

Removing the time input had a worsening effect compared to previous trails. Time was found to be required to allow the DNN to learn the effect of time on the system dynamics. As time goes by in an oscillatory system motion repeats itself but for continuously changing dynamics repetition is not a factor. Therefore, time would be included in the following trials.

5.4 Feature Engineering

Over the previous trials we have learned that the rate of learning of the network is not improving with regular data manipulation of adding and removing more data points. Therefor more research was required into what is called feature engineering. Feature engineering entails modifying the facts into forms that have a stronger connection to the fundamental objective that is to be taught. Adding value to your existing data and improving the efficiency of your machine learning models are both possible outcomes of

feature engineering when it is carried out correctly. On the other side, if you use poor features, you might need to construct models that are significantly more complicated in order to attain the same level of performance.

5.4.1 Handling Outliers

In a dataset, outliers are numbers that stand out because they are abnormally outside the bound of a series or maintained output and are not likely to occur in typical circumstances. Because these outliers have the potential to negatively affect the network prediction, different approaches are employed to handle outliers such as:

- Elimination: Where the datapoints in the distribution that contain outliers are eliminated from further consideration. However, if there are any outliers across numerous variables, those datapoints are left as is to maintain the data integrity.
- Capping is when the maximum and minimum values are capped and then replaced with an arbitrary number, or a value drawn from a variable distribution. Capping can also be referred to as limiting.

5.4.2 Scaling

Feature scaling also referred to as feature normalization involves adjusting the scales of features using the following scaling technique:

Min-Max Scaling: This method includes rescaling all of a feature's values within the range of 0 to 1 in order to achieve the desired results.

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{34}$$

As presented in equation 37, Xmax and Xmin are the maximum and the minimum values of the feature respectively. The value that was at the lowest end of the original

range will be assigned the value 0, the value that was at the highest end of the range will be assigned the value 1, and the values in the middle will be scaled properly. This technique was employed in the parameter values beta 1 and beta 2.

5.5 Deep Neural Network Architecture

5.5.1 Vanishing Gradient Problem

The vanishing gradient problem is observed when training a deep neural network and the recurrent neural network is unable to transport valuable gradient information from the output terminal of the model to the layers near the input terminal[80]. As a consequence of this, models with many layers either cannot learn from the dataset they are given or prematurely converge on a solution that is not optimal. In general, models with many layers are unable to learn from datasets.

5.5.2 Batch normalization

Batch normalization is a strategy that is used to make the training of artificial neural networks quicker and more stable. This is accomplished by normalizing the inputs to the layers of the network by re-centering and re-scaling the data. In 2015, Sergey Ioffe and Christian Szegedy were the ones who initially proposed the resolution. There is much debate over the factors that contribute to its success[81]. It was thought that it may help minimize the problem of internal covariate shift, which is when parameter initialization and changes in the distribution of the inputs of each layer alter the learning rate of the network. It was believed that it could do this. As of late, a number of researchers have posited that batch normalization does not, in fact, lessen the effect of internal covariate

shift; rather, it smooths the objective function, which ultimately results in an improvement in performance [82].

5.5.3 Long short-term memory

The term "long-short term memory" (LSTM) is derived from the idea that a regular recurrent neural network (RNN) possesses both "long-term memory" and "short-term memory." It is analogous to how physiological changes in synaptic strengths store long-term memories that the connection weights and biases in the network change once per episode of training. On the other hand, the activation patterns in the network change once per time-step, which is analogous to how the moment-to-moment change in electric firing patterns in the brain stores short-term memories[83].The "long short-term memory" (LSTM) architecture is intended to give RNNs with a short-term memory that is durable enough to withstand thousands of timesteps. Because there may be gaps of undetermined length between significant occurrences in a time series, LSTM networks are ideally suited for the tasks of categorizing, processing, and making predictions using data from time series. The vanishing gradient problem that can arise during the training of conventional RNNs inspired the development of LSTMs as a solution to the issue. When compared to RNNs, hidden Markov models, and other sequence learning approaches, LSTM's relative insensitivity to gap length is an advantage that makes it superior un a variety of applications[84].

5.5.4 Dataset (Trail 144):

In trail 144 the methods in sections 5.4 and 5.5 are employed. The time element is reincluded in the dataset. The following parameters were sampled at ten milliseconds:

1. Control Signal (u)

2. Position Output
3. Velocity Output
4. Output Error (Numeric)
5. β_1 & β_2

The root mean square error in Figure 28 is shown to decrease by 1 during the validation of the dataset compared to trail 116.

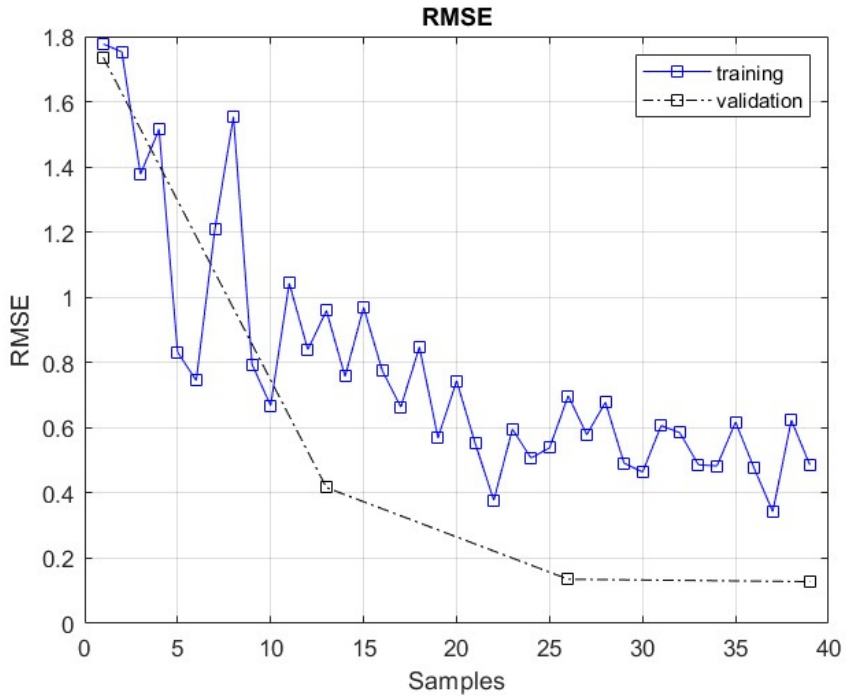

Figure 29 The training and validation RMSE comparison of the dataset in trail 144

5.6 Lyapunov Deep Learning Control on Zohdy-Harb

In this section we review the results of applying Lyapunov Deep Learning on the Zohdy-Harb non-linear system. The mathematical model presented in equation 38 presents the Zohdy-Harb Oscillator [55].

$$\ddot{x} + \delta\dot{x} + \varphi(x^2\ddot{x} + \dot{x}^2x) + \gamma x^3 = P\cos\Omega t + u \qquad (35)$$

And the Lyapunov control law is found in equation 39

$$u = \frac{\delta}{2}\frac{\beta_1}{\beta_2}e + \frac{\delta}{2}\dot{e} + \ddot{z}_{des} + \varphi\dot{x} + \alpha(x^2\ddot{x} + \dot{x}^2x) + \gamma x^3 + \frac{\beta_1}{\beta_2}\dot{e} - \frac{cost}{\beta_2} \quad (36)$$

The Deep NN in Figure 23 and the training data in Figure 29 are utilized to control the control Lyapunov parameters $\beta_1 and\ \beta_2$. The system is shown in Figure 30 to become unstable and fail within 0.3 seconds after simulation start due to the use of unfit Beta 1 = 100 and Beta 2 = 0.6, whereas in Figure 31 we show that the proposed deep neural network aids the Lyapunov controller in finding the control law parameters leading to system stabilization and keeping the system error under the set threshold.

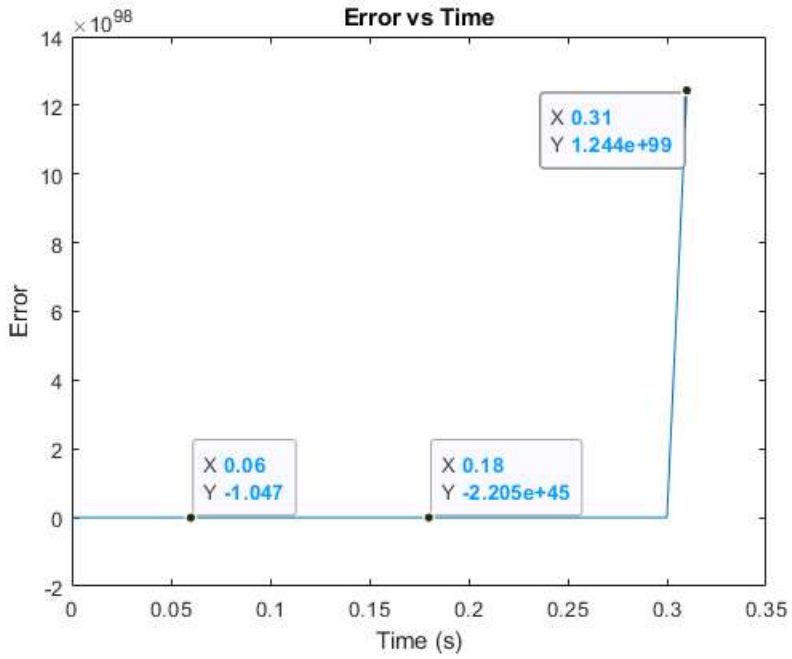

Figure 30 The system error goes to infinity as shown when the algorithm is not applied

The findings demonstrate the neural network's efficiency in forecasting the error given the beta1 and beta2, as well as the algorithm's effectiveness in preventing the system from failing and continuously improving its performance by maintaining the

system error as low as feasible. Maintaining a constant beta1 and beta2, on the other hand, leads in large system error and the system may finally fail.

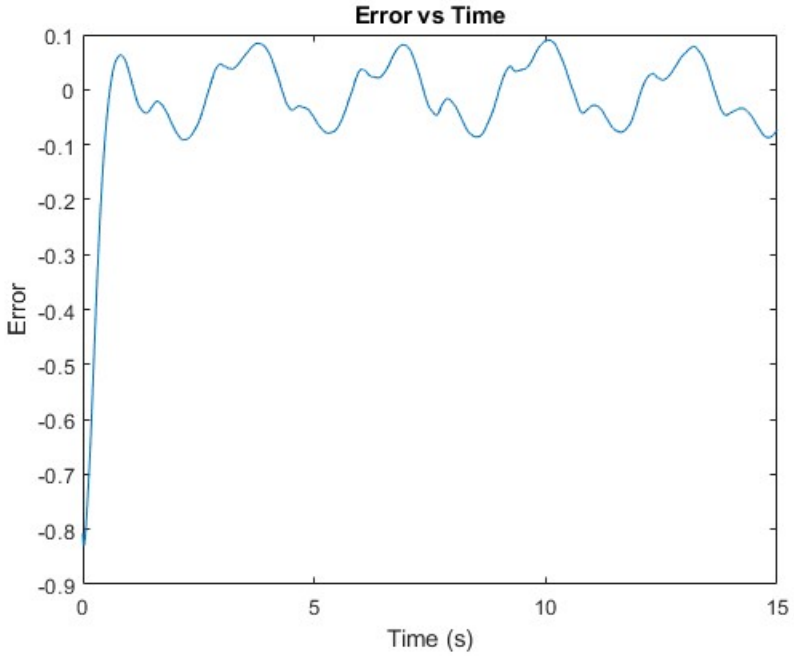

Figure 31 The system error after using the neural network

5.6.1 Introduction of step disturbance

To expand on the suggested solution's ability to deal with mid-system instability or unexpected changes. Figure 32 presents the decline in error at 0.8867 on the intervention of the DL algorithm in changing the control parameters. The system is introduced to step disturbance at 1 sec while the system is running. The deep network was able to anticipate the appropriate parameters to re-establish system and controller stability within 0.4 ms, as seen in Figures 33 and 34. Figure 35 shows that Lyapunov Deep Learning is shown to successfully react to changes in the system disturbances or during unknown initial conditions.

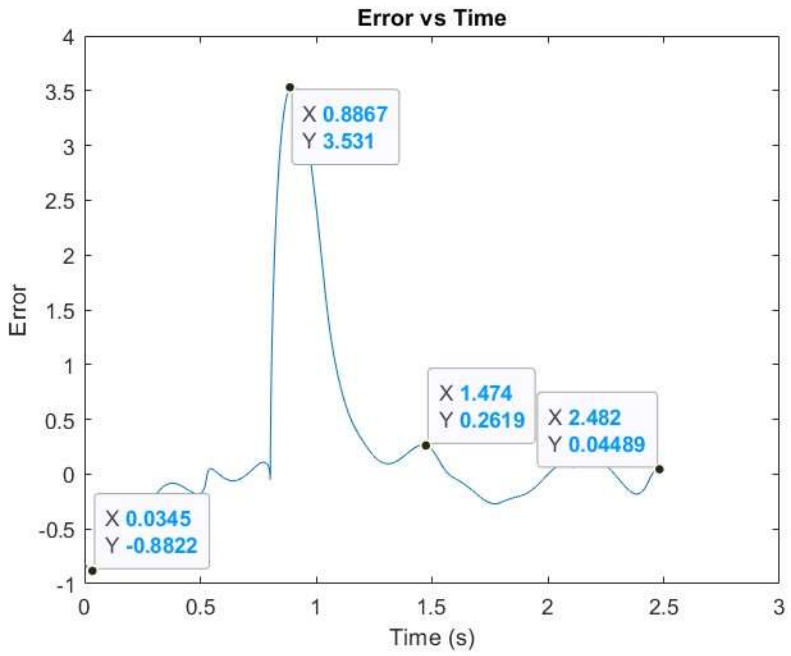

Figure 32: Error uptick at t = 0.8867 due to the disturbance introduction

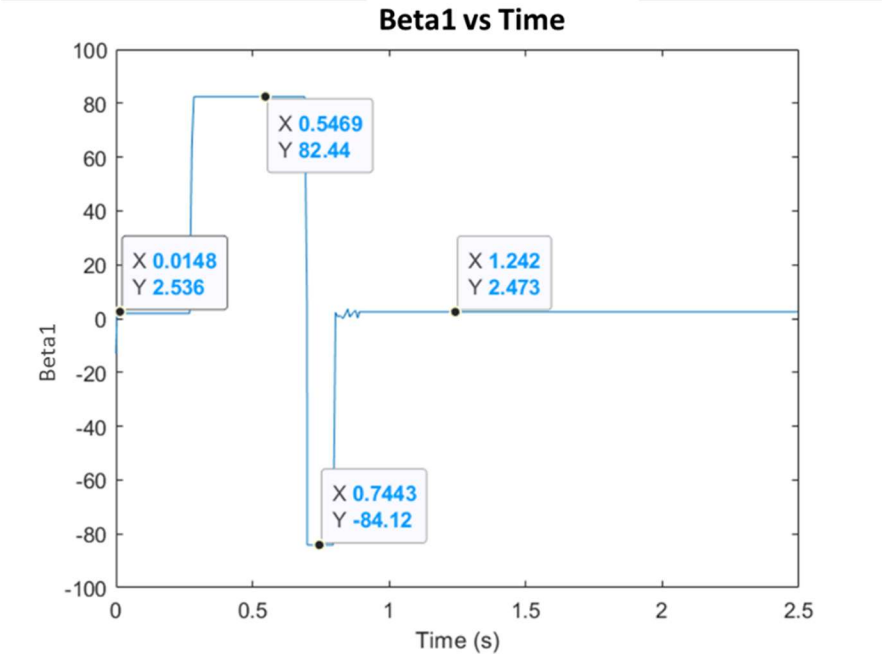

Figure 33 The Algorithm reacting to the sudden change by adjusting Beta1

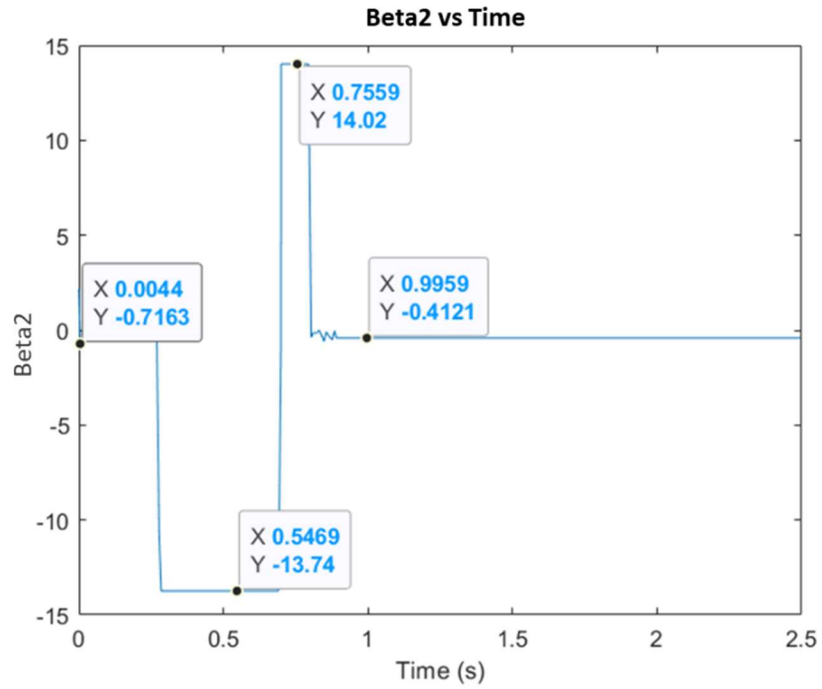

Figure 34 The Algorithm reacting to the sudden change by adjusting Beta2

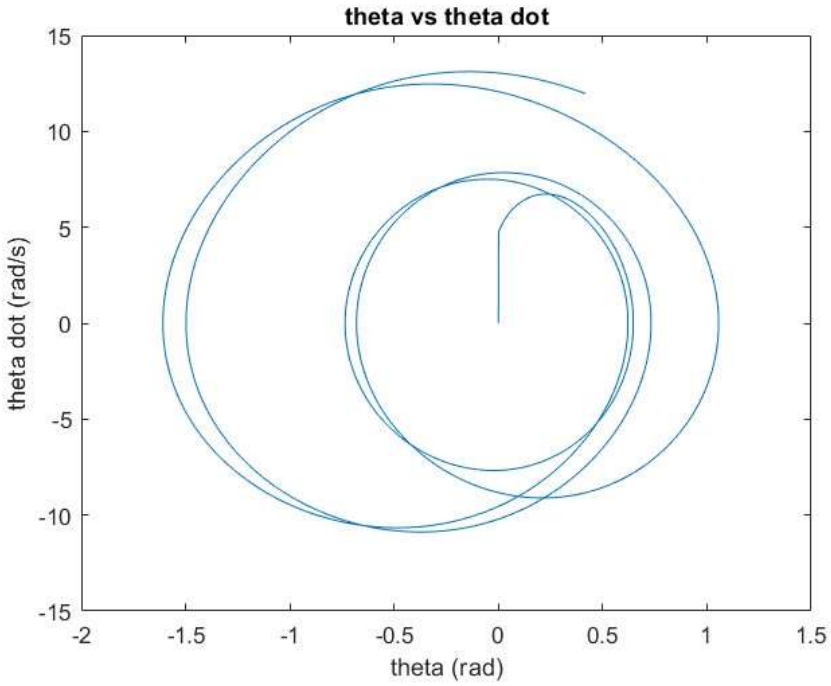

Figure 35 Phase diagram after finding the optimal system parameters

In this chapter we presented the deep learning network architecture and the method of integration with Lyapunov Control. We also explore the different phases of dataset development. Initial datasets included separate records for each model run and the stability of the system during that run would be the output of the DNN, but the accuracy of such dataset was proven to insufficient to update the parameters and the direction to take into account. Therefore, after further research and development a new combined dataset was proposed with the variation of the controller parameters combined with the error. The final dataset was found to substantially bring down the average validation RMSE to 0.225 which is a leap compared to previous results of 1.6. Therefor the dataset at trail 144 was selected to act as the template for upcoming model development.

CHAPTER SIX

MAGNETIC LEVITATION APPLICATION

Magnetic levitation (maglev) has emerged as a viable solution to the current demand for faster and more effective modes of transportation, and its contact-free technology is already finding uses in space and the military in addition to allowing for high speed, safe transit alternatives to railways. The demand for faster and more efficient modes of transportation is expected to continue growing in the coming years. Measurements of position, velocity, and acceleration are gathered so that a magnetic levitation system's feedback control loop can function properly. Following the completion of the appropriate measurements and processing, the signals are then introduced into a feedback loop. It is necessary to utilize contactless transducers in a Maglev system in order to accurately record position, velocity, and acceleration [85]. A Lyapunov nonlinear control can be used to monitor the output of the transducer, and the distance that separates the item and the rails can be adjusted. Efficiency is one of the most essential design criteria that has been taken into consideration because batteries have a finite quantity of power and are employed in applications related to transportation both now and in the future. Energy harvesting, also known as energy scavenging, is the process of gathering and transforming energy from the surrounding environment into usable electrical energy. This process was incorporated into the Maglev system described in [86-88]. During the course of the previous decade, several sources of energy, such as wind, solar radiation, thermal radiation, vibrations, and, most recently, magnetic energy harvesters, were successfully recycled in an effective manner. Magnetic energy harvester

systems require a controller, actuators, and coils in order to collect and manage the energy that is recovered from the magnetic field. Coils are also required. In order to function properly, the coil-magnet components of the harvester must comply with Faraday's law of electromagnetic induction [89].

As a result of the ongoing trend of advancements in technology and microchips, the complexity of CPS systems will continue to increase. According to an article that was published in the journal of the National Academies in 2016, it was stated that "today's practice of CPS system design and implementation is often ad hoc and unable to support the level of complexity, scalability, security, safety, interoperability, and flexible design and operation that will be required to meet future needs." [90]While earlier research in [41, 42] focused on using machine learning to toggle the control law between two or more possibilities, the current study will focus on this topic.

As a result of the study in [16], we have come to the conclusion that computational solutions like deep learning have some restrictions when it comes to the amount of time that is available [38].

This chapter presents the usage of the Pearson correlation, as well as the practice of giving precedence to parameters with strong correlation in connection to errors in the algorithm. In addition, we place an emphasis on the application of parameterized complexity in order to analyze the dataset. This allows us to reduce the amount of depth that the DNN has while still producing correct results. An individualized structure composed of layers is presented here. To the best of our knowledge, the incorporation of the concept that was indicated earlier has not been covered in any of the earlier research that has been published. In this chapter we also present a novel approach based on the

parameterized complexity theory to estimate the complexity of the dataset and accordingly change the depth of the DNN. Rather than quantifying the system complexity purely in terms of its input length, other numerical properties of the input instance are considered such as the correlation between the dataset parameters collected while the CPS system is running, or the memory occupation increase or decrease. The initial data set is a collection of parameter variations and their effects on the output. An updated dataset is collected via the deep neural network (DNN) and runs based on the initially defined CPS information. We identify high-performance compact deep learning architectures through a neural architecture search and meta learning. The neural network architecture is customized to minimize the training time and computational power required. A new data set is recorded if the delta error between the actual system error and the predicted system error generated by the DNN after the substitution with the proposed parameters is greater than 0.4. The system begins to add to its current data set while keeping in memory 40% of the older data set.

A function to calculate the number of DNN layers required to relearn the parameters tuning effect on the model. In addition, a novel control Lyapunov function is presented, and the results are compared to a PID-controlled Maglev system from [91]. The proposed controller is shown to successfully stabilize the system under different disturbances and reference signal changes safely without going into an infinite loop.

6.1 Magnetic Levitation System Dynamics

In this section, a typical Magnetic Levitation system plant model is investigated, and the equations describing the system dynamics are presented [92]. Following that, the

energy harvesting portion of the model dynamics is described [93]. The following set of differential equations can be used to represent the nonlinear model.

$$m\ddot{x} - \sigma\dot{x} = \frac{i^2 k_c}{(x-x_0)^2} - mg \tag{37}$$

m: mass of the ball.

g: gravity.

x: displacement.

$\ddot{x}$: acceleration.

i: current.

σ: damping constant [N/m.s].

The displacement and current position of the ball in the magnetic field is controlled by electric current supplied and governed by Equation (2).

$$F_s = \frac{I(t)}{U(t)} = \frac{k_a}{T_s+1} \tag{38}$$

i(t): current at time t.

U(t): voltage at time t.

Ka: coil inductance.

Ts: time constant.

The system in Equation (38) presents a traditional magnetic levitation system dynamic. While in the system in equation (40) presents an energy harvesting magnetic levitation model dynamics.

$$x + mg + xK_{mag_b} - K_{mag_t} + S\dot{q} + (c_m + c_e)\dot{x} = 0 \tag{39}$$

m: mass of the ball.

g: gravity.

x: displacement.

$\ddot{x}$: acceleration.

i: current.

Alpha: magnetic force constant.

$$L_s\ddot{q} + R_s\dot{q} + \frac{q}{C_s} - S\dot{x} = ecos(\omega t) \tag{40}$$

v: voltage input.

R: resistance.

L: inductance

6.1.1 State Space Representation

The below state space equation is aimed towards controlling the position of the Ferrous ball. In equation 39, Alpha is the summation of $c_m + c_e$ and taking x1 = Displacement, x2 = Velocity, x3 = i. Therefore, Equations (37) and (38) can be written as a matrix format while considering position as output.

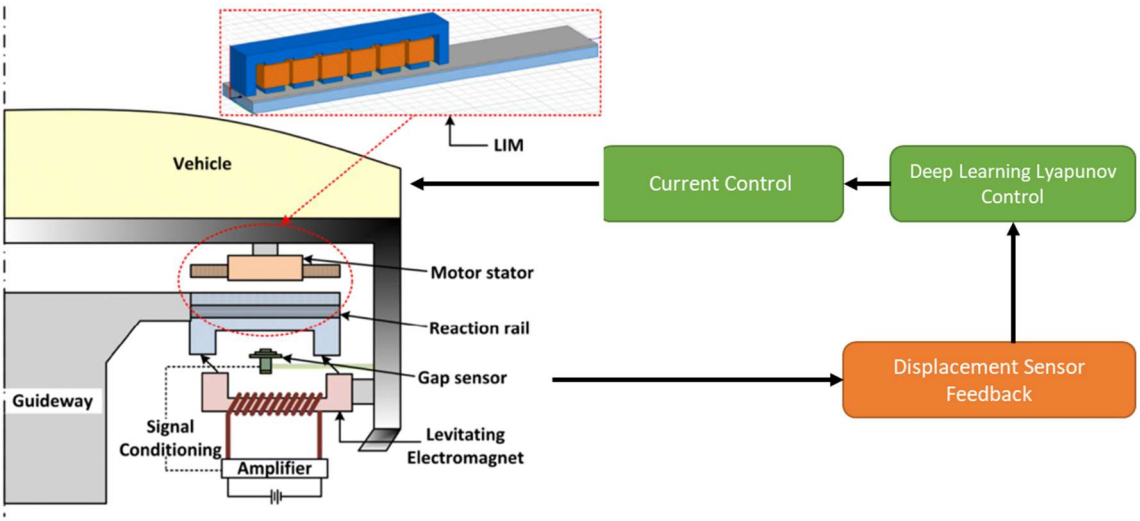

Figure 36 Magnetic levitation system

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\\ \dot{x}_3\end{bmatrix} = \begin{bmatrix} x_2 \\ g - \frac{\alpha}{m}\left(\frac{x_3}{x_1}\right)^2 \\ -\frac{R}{L}x_3 + \frac{2\alpha}{L}\frac{x_2 x_3}{x_1^2}\end{bmatrix} + \begin{bmatrix}0\\0\\\frac{1}{L}\end{bmatrix} \quad (41)$$

$$y_{Position} = [1 \quad 0 \quad 0] \quad (42)$$

A variation in the y matrix is made to change the intended controlled output

$$y_{Velocity} = [0 \quad 1 \quad 0] \quad (43)$$

$$y_{Current} = [0 \quad 0 \quad 1] \quad (44)$$

6.2 Design of the Lyapunov Controller

Lyapunov control has proven successful in managing complex nonlinear oscillators to a certain extent of oscillator frequency ω = 2.5 Hz [15]. In order to improve the system stability at a higher ω, deep learning was introduced in [16]. The deep learning algorithm is taught the relationship between the system parameters change, the controller parameter change, and the output error slope. If the algorithm detects a sudden change in slop a parameter update is triggered, and the deep learning algorithm substitutes the current parameters with updated parameters that are expected to bring the system to stability. The application of control Lyapunov functions was developed by Z. Artstein and E. D. Sontag in the 1980s and 1990s[94]. Control Lyapunov functions are utilized to determine the stability of a system or a system ability to regain stability. A control Lyapunov function u is selected such that the function is globally positive definite and the time derivative of the control function $\dot{u}$ is negative definite and globally exponentially stable.

$$\mathrm{u} = \frac{1}{2}(\dot{\mathrm{e}} + \alpha \mathrm{e})^2 \quad (45)$$

taking the time derivative of u

$$\dot{u} = (\dot{e} + \alpha e)(\ddot{e} + \alpha \dot{e}) \tag{46}$$

$$\dot{u} = \frac{\gamma}{2} u \tag{47}$$

such that the error

$$e = yd - y \tag{48}$$

yd is the desired state and y is the actual state. upon substitution in 46 we get the control law U The Lyapunov function for the system in equation 37 is derived for the magnetic levitation model as

$$U = \frac{km}{2}(\dot{e} * \alpha e) + \alpha \dot{e} m + gm + K_{fv}\dot{x} - \frac{i^2 * K_c}{(x - x_o)^2} + \ddot{y}m \tag{49}$$

The Lyapunov function for the system in 39 is derived for the Energy Harvesting Magnetic Model as

$$U = -\ddot{y_d} - \alpha_4 \ddot{y} - \omega_4 y + \alpha_3 \frac{\ddot{x} + x + \omega_3 x^3 + \alpha_2 \dot{y}}{-\alpha_1} - N\cos(\theta t) - \alpha \dot{e} - \frac{k}{2}(\dot{e} + \alpha e) \tag{50}$$

6.3 Deep Learning Algorithm

The research presented in [36] showed that the controller parameters had a significant effect on the desired system outcome and inaccuracy. Consequently, the research in [16] effectively provided a deep learning strategy that would permit one of the controller parameters to change while the system is operating, thereby mitigating unexpected changes in system stability. One of the downsides discovered through the survey in [95] is that the system learning/re-learning process was found to be lengthy, and if more parameters require updates simultaneously to achieve system stability, it would take even longer due to the high variances in data in the data-set. The system

would fail because the updated parameters are not found within the allotted period. Researchers have demonstrated success with the Lyapunov stability function in dynamic DNN weights [96]. In addition, Optimization has gained popularity for locating network architectures that are computationally efficient to train while still performing well on certain classes of learning problems [92], [93], and [97] as well as for taking advantage of the fact that many datasets are similar and therefore information from previously trained models can be reused. Despite the fact that this tactic is frequently quite effective, the current deficit in studies, such as [98] concentrate additional emphasis on the memory footprint reduction of the model. Meta learning or neural architecture search is computationally costly on its own, as it requires training several models on a varied collection of datasets [92]. While the cost has been steadily decreasing over the past few years and is now almost on par with the cost of conventional training [98]. The size of the data set that was used to train the initial model is an important limitation that must be adhered to when learning. It has been demonstrated, for example, that the performance of image recognition is heavily influenced by image biases and that the performance of transfer learning drops by 45 percent when these biases are removed. Observes an 11%–14% decline in performance [98], despite the use of one-of-a-kind data sets that were developed deliberately to emulate their training data. Switching to alternative forms of machine learning, some of which may still be undiscovered or underappreciated, is yet another strategy for getting over the computational constraints posed by deep learning. As a result, in this chapter, we combine various strategies in order to get around the data complexity link to computing cost.

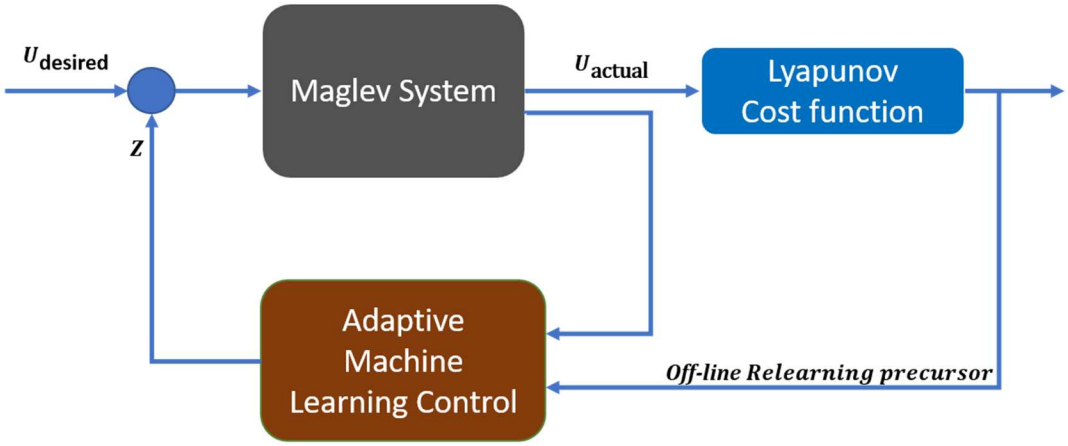

Figure 37 Magnetic levitation system controller setup

6.3.1 Customized Deep Learning Architecture

The input layer of the network architecture begins by receiving a pre-collected data set consisting of 38,000 points. These points include the time, V(cost function), velocity, location, parameters (k, k_{fv}, k_c, α_3), reference, and error. In the second step of the process, we build a convolution layer to compute the correlation between the different groups of input data. In contrast to a standard convolution layer, a completely connected convolution layer has a smaller impact on the correlation losses that occur. The third layer is a batch normalization layer, and its purpose is to both stabilize the learning process and cut down on the total number of training epochs that are necessary to properly train the network. It is necessary to have a fourth activation layer in order to stabilize the learning process and cut down on the number of training epochs that are needed to train deep neural networks.

In order to fix the overfitting issue, a fifth dropout layer has been added. When networks have a high level of accuracy on the training data set but a very low level of accuracy on the test data set, this is an example of over-fitting. After everything else has

been finished, the regression layer is added so that the losses can be computed, and the node weights may be readjusted accordingly. The methods necessary to acquire the output are outlined in Algorithm 1, which can be found here.

6.3.2 Parameterized Complexity and Dynamic Programming

Parameterized complexity was developed with the intention of offering an alternative approach to the resolution of intractable computer issues [96]. Other numerical characteristics of the input are taken into account when determining the complexity of an algorithm, as opposed to basing it exclusively on the length of the data that it processes. For instance, the vertex cover problem is what's known as an NP-hard problem, which indicates that it cannot be solved in the traditional sense. This issue can be remedied in the amount of time that is equal to O(n^2) *f*k, provided that the run-time is further defined in terms of the vertex cover size k. It is said that anything is tractable if the value of k is relatively low in contrast to the total number of problems that need to be solved. This refined idea of effectiveness is known as fixed-parameter tractability, and it has become a canon of computational complexity [99]. A difficult problem can be broken down into a series of simpler decision problems using dynamic programming. Typically, these include overlapping values that can be modified, but what's more important is that the local values can be mixed in a controlled manner.

In most cases, the process will entail recursion of some kind [100, 101]. Using this line of reasoning, we have determined that there are two elements that contribute to the complexity of our data collection. The number of data points is represented by the factor n. On the other hand, factor f is defined as the number of parameters or features that are defined in the data set. As the correlation grows, the relationship between the

parameter and managing the error is easier to determine. The assumption made earlier led to the utilization of the time complexity is presented in equation 52.

$$O(n^2) *f*k \tag{51}$$

Where n is the number of features and f is the number of test points and k is a constant of 2.8 x 10-4. Taking as an example the dataset with 37,000 data points and 7 features, the projected run time for the DNN would be 37 minutes. It was discovered that five deep learning layers gave 97% accuracy after a run time of 6 minutes. In the Maglev system 100,000 datasets were collected. Therefore, the number of layers in the deep neural network was increased from 5 to 6 by adding one more fully connected convolution layer. This brought the accuracy of our predictions up to 97% or higher.

Figure 38 presents the algorithm flow diagram for integrating the Maglev system model and Lyapunov control with the DNN running in the background with the continues stream of new data to support relearning. This integration is accomplished by combining the DNN with the continuous stream of new data. Figure 31 illustrates the created DNN architecture, in which 38,000 data points are shown propagating through the input and hidden layers as the system starts to acquire additional data points. Only 10% of the data set is updated if the DNN's ability to accurately predict errors starts to become less reliable in order to preserve the safety of the control method. Artificial intelligence works in unison with the control and the system at all times. If the error slope exceeds a predetermined threshold that is greater than or equal to one, the neural network is prompted to suggest new control parameters with the intention of lowering the overall error level of the system.

Algorithm 1: Dynamic Deep Learning Algorithm.

```
Memory = 40% of the old training data previous_system_average = system error average of the training data
  initialization;
while n > 4000 do
    system_error > previous_system_error;
    if system_error > 0.5 then
        while True do
            Simulation_Time = get_param('maglev','SimulationTime');
            Parameters_Matrix=[k, α, k_fv, k_c, α3, system_error];
            Correlation_Matrix = corrcoef(A);
            K = minK + rand*(maxK-minK);
            α = minα + rand*(maxα-minα);
            k_fv = mink_fv + rand*(maxk_fv-mink_fv);
            k_c = mink_c + rand*(maxk_c-mink_c);
            α3 = minα3 + rand*(maxα3-minα3);
            predicted_system_error = predict (k, α, k_fv, k_c, α3);
            if predicted_system_error < system_error then
                Break;
            end
        end
        update_parameters (k, α, k_fv, k_c, α3);
        new_parameters = Save (k, α, k_fv, k_c, α3, new_sys_err);
        average_system_error = average_system_error + new_system_error;
    end
    if n == 0 then
        if average_system_error / 100 > previous_average then
            retrain_network(Memo + new_data);
            Memo = Memo + 10% of the new_data;
            average_system_average = average_system_error;
            Clear newly_collected_data;
            Clear average_system_error;
        end
    end
end
```

Figure 38 Dynamic Deep Learning Algorithm

The network shown in Figure 39 is first trained, and then it is utilized to make predictions regarding new values for (k, k_{fv}, k_c, and α_3). Memory usage, also known as self-refreshing memory, was initially described, and experimented with in [76-78]. This idea is often referred to as "self-refreshing memory." Even if perturbations do take place, it's possible that they won't last long enough for the network to lose its ability to foresee system errors if the system's parameters return to normal after the network has been retrained multiple times. This is the thinking behind this hypothesis.

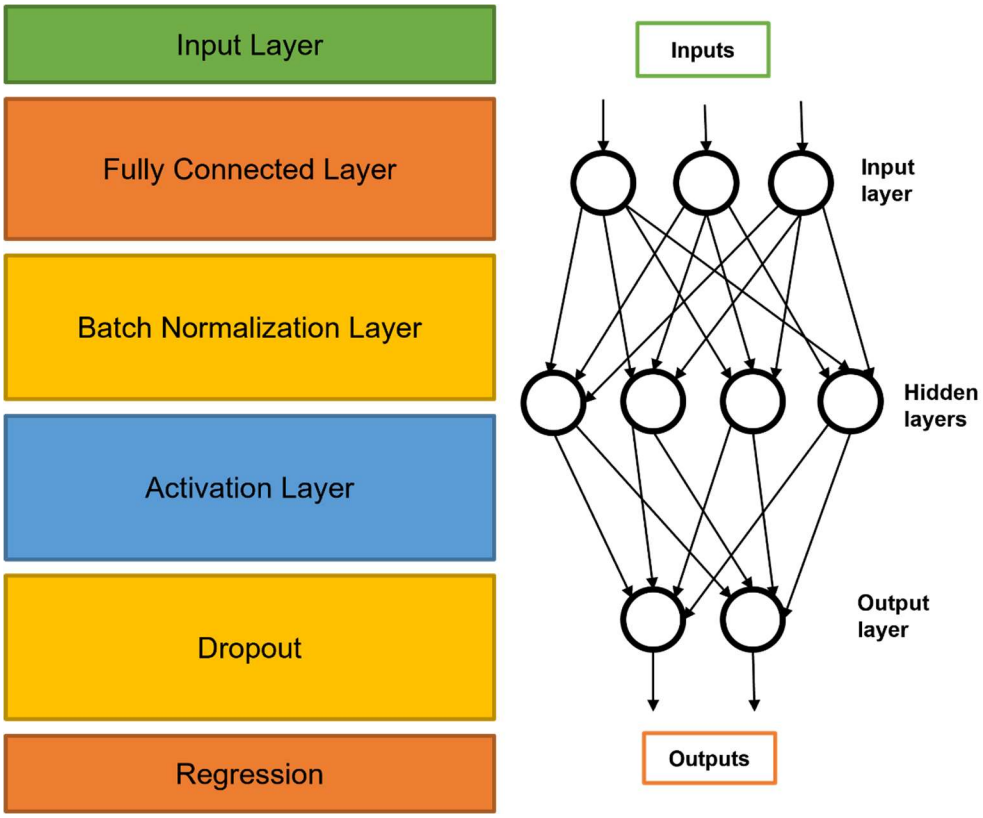

Figure 39 Custom Deep Learning NN Architecture layers

6.4 Maglev Dynamic LDL Control Results

Initially The system is put through its paces with the Lyapunov control in place, as well as without the DNN. The findings are compared to those obtained from a system that utilized PID control but did not include the DNN. An input reference sine wave signal and an input frequency of 40 rad/s were used in the testing of both controllers. The findings are documented in the following Figures: (Figures 40–43). The system is found to be stable with the system parameters set to the values in Table 1 and the controller parameters set to the initial static values in Table 2, until the frequency of oscillations of the reference signal is increased over 4000 rad/sec, at which point the system error begins

to increase and both controllers have shown that their static parameters are unable to compensate for the change in. The system is found to be stable with the controller parameters set to the initial static values in Table 4.

Table 4 Maglev system parameters

Parameter	**Value**
Mass (Kg)	0.1
Gravity (m/s^2)	9.8
Rs (Ohm)	1
Cs(F)	0.5
Ls (Henry)	0.4
e	0.002
S	0.4
ω	2

Table 5 Lyapunov controller parameters

Parameter	Value
K	0.1
α_1	9.8
K_{fv}	2
K_c	0.001

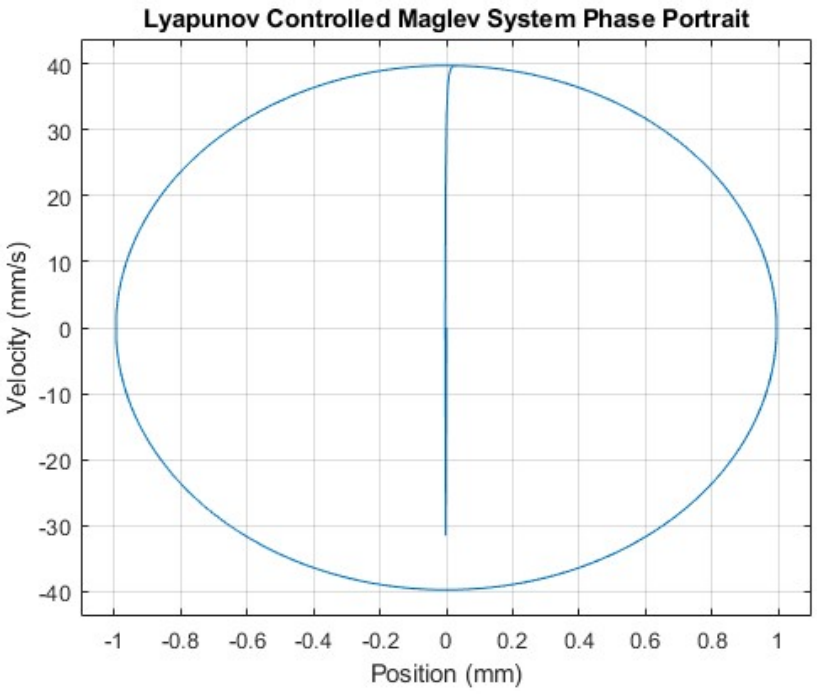

Figure 40 Phase portrait of the Maglev system with a reference sinusoidal wave of 40 rad/sec frequency under Lyapunov control

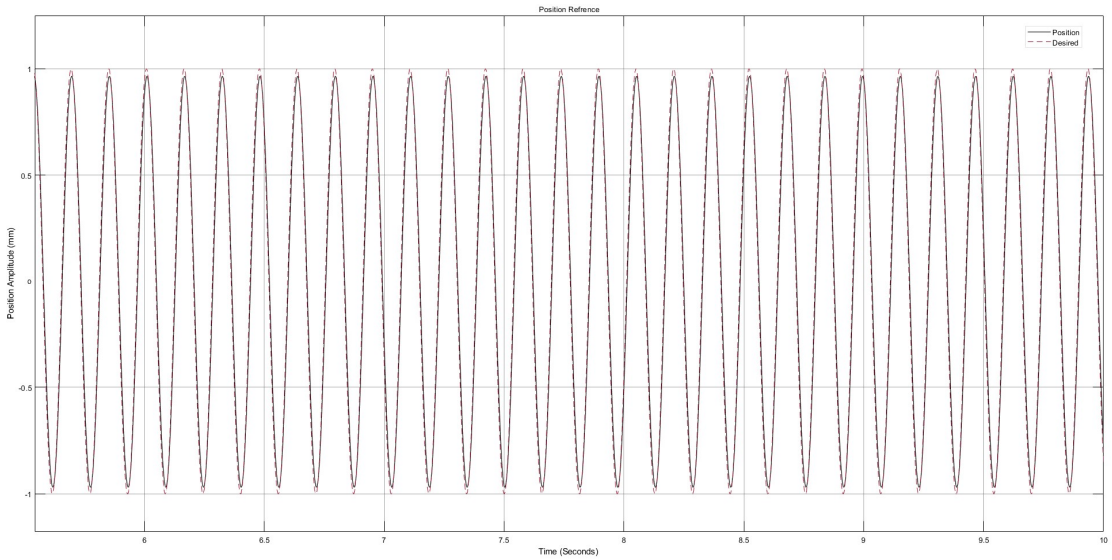

Figure 41 Lyapunov controlled position with reference to sinusoidal wave of 40 rad/sec frequency

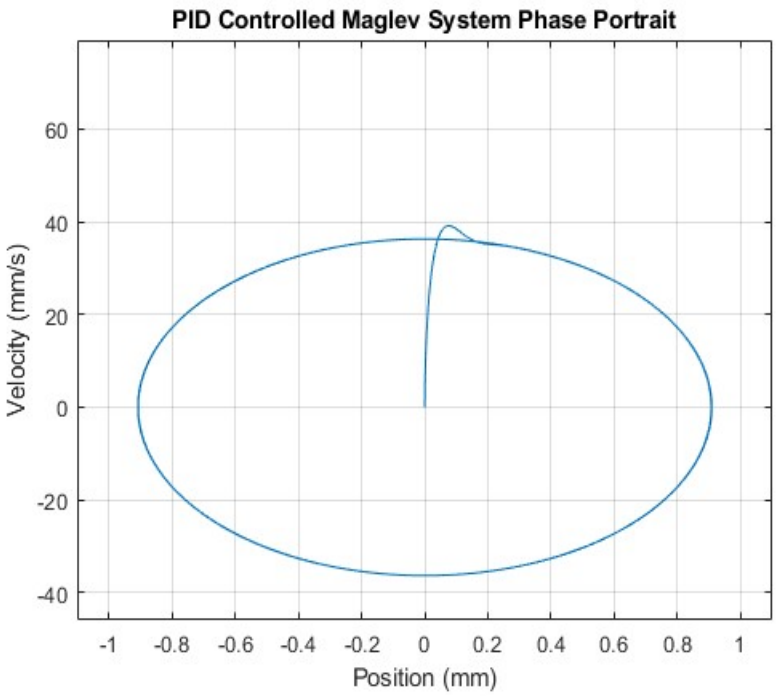

Figure 42 Phase portrait of the Maglev system with a reference sinusoidal wave of 40 rad/s frequency under PID control

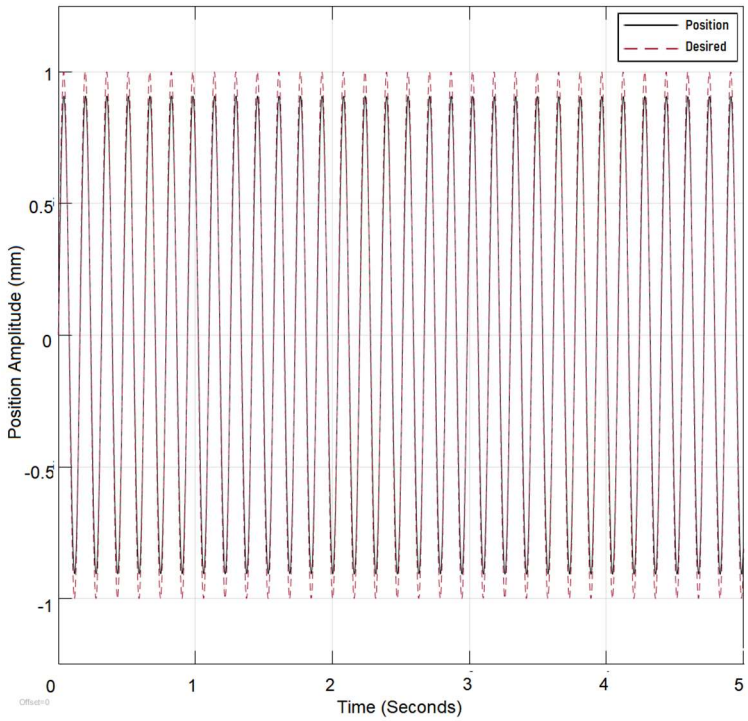

Figure 43 PID controlled position with reference to sinusoidal wave of 40 rad/s

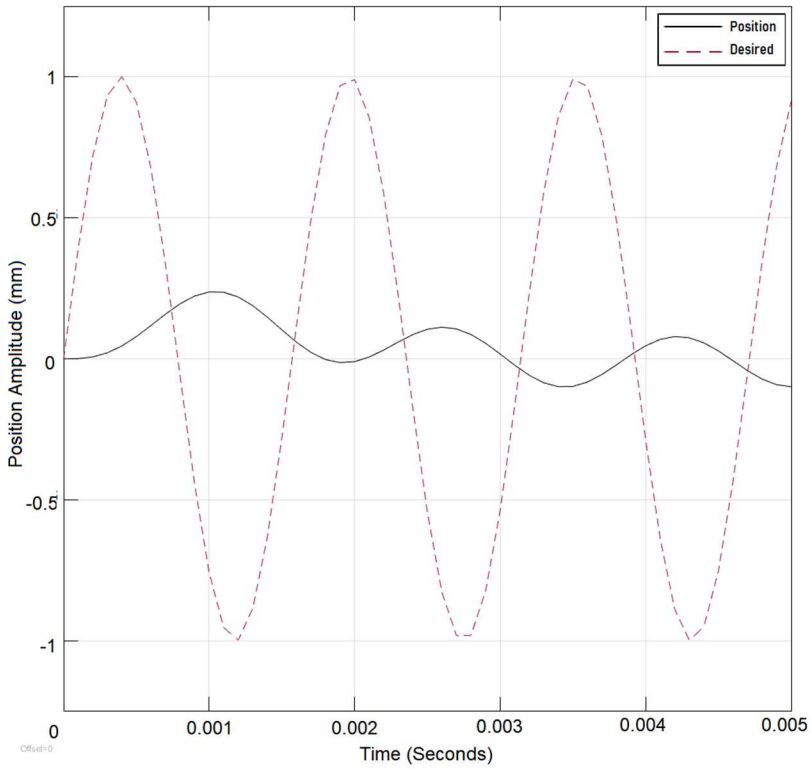

Figure 44 Lyapunov controlled position with reference to sinusoidal wave of 4000 rad/s

It is demonstrated in Figure 45 that the PID controller is less effective while operating under switching signal conditions. As soon as the desired reference signal is altered from a sinusoidal to a step function, there is a discernible rise in the amount of output errors. This is because the constant parameters need to undergo some adjustments in order to accommodate the signal transformation. As a result, the use of deep learning is able to address abrupt changes in the reference signal, which is demonstrated in Section 3.2, where the algorithm's reaction to the reference signal is displayed.

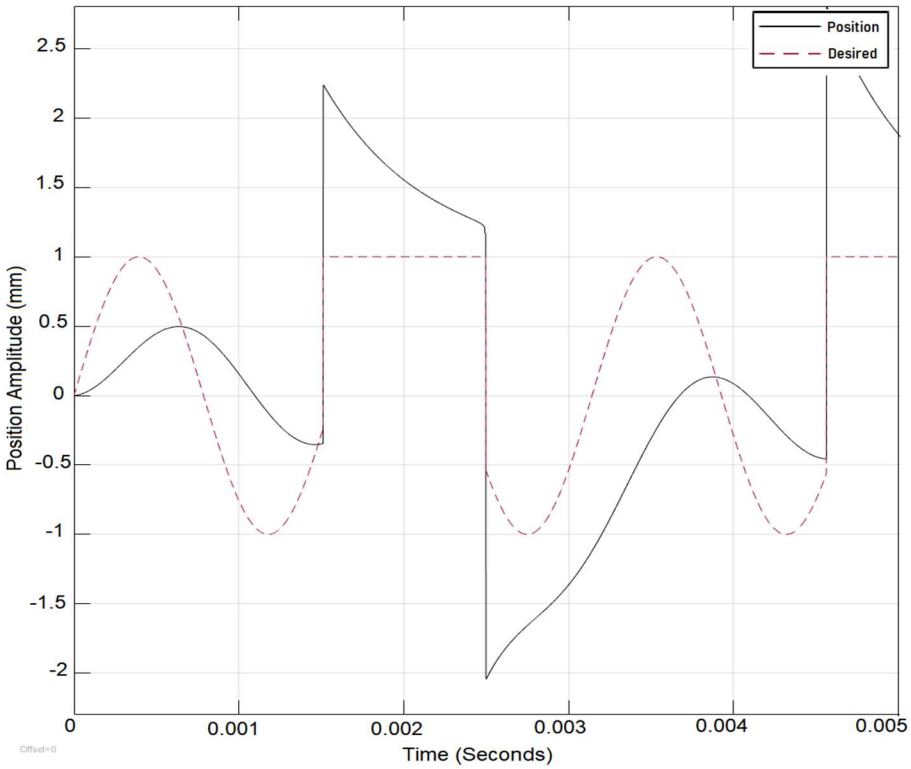

Figure 45 The position with reference to a combined signal of sinusoidal and step function under PID control

6.5 Correlation Study and DL Algorithm Application Results

Over time, the controller parameters were found to have varying effects on the error variation. As the dataset grew from 38,000 to 100,000 data points and the number of parameters under the purview of the DNN grew from one parameter K to five parameters (k, α, k_{fv}, k_c, α_3), It was discovered that the parameter with the strongest correlation to the error had the most influence on the error reduction. Therefore, a Pearson correlation study between the effect of parameter change and error was conducted. Figure 46 depicts the Pearson correlation coefficient (PCC), which measures the linear correlation between two sets of data. This demonstrates that the ratio of the covariance of two variables to the product of their standard deviations always falls between -1 and 1 [88]. The priority is

then assigned proportionately to each parameter. The correlation data are updated whenever a fresh data set is introduced, or the deep learning network is prompted to retrain. The sequence in which the parameters are changed is also examined by altering the parameters in a variety of sequences while recording the error vector.

$$r = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2 \sum_{i=1}^{n}(y_i-\bar{y})^2}} \tag{52}$$

Table 6 Person equation parameter table

r	correlation coefficient
x_i	values of the x-variable in a sample
$\bar{x}$	mean of the values of the x-variable
y_i	values of the y-variable in a sample
$\bar{y}$	mean of the values of the y-variable

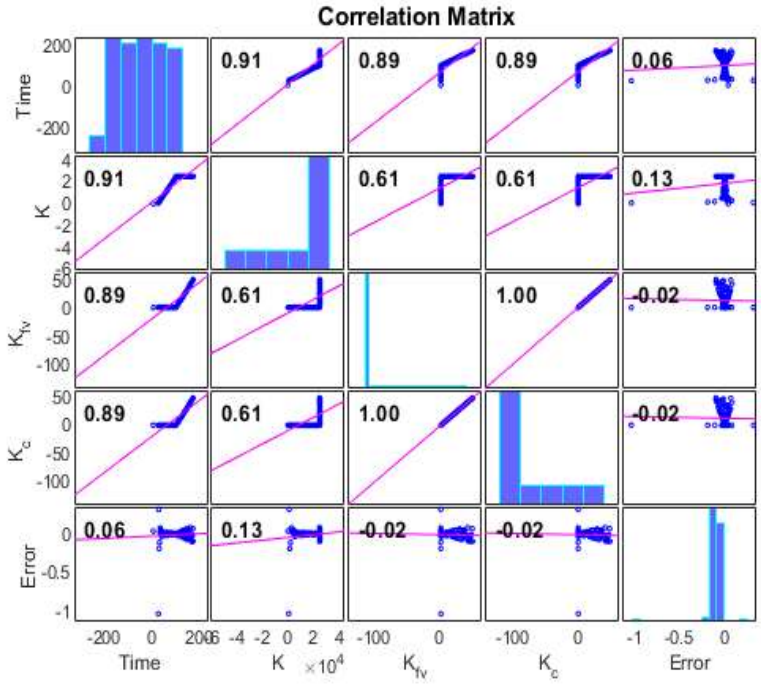

Figure 46 Pearson correlation chart between the parameters and error

As shown in Figure 46, the largest correlation is between K, and as the error in parameter K is varied, the error either decreases or increases depending on the state of the system; thus, the priority of change to control is given to k. To account for change, the correlation is continually calculated as described in the preceding paragraph. Figures 47 and 48 illustrate the effect of using the Dynamic Lyapunov Deep Learning (DLDL) control to stabilize the output of a high-frequency system. Figure 49 depicts the transition of the reference signal from sinusoidal to step function and back to sinusoidal, with the inaccuracy between the reference and output position signals under control. In response to a change in the reference signal type, the deep learning network updates the controller parameter K from 20 to 2700. If the adjustment in K does not restore system stability, the algorithm evaluates the second-highest parameter correlation to the error, followed by the third-highest parameter correlation.

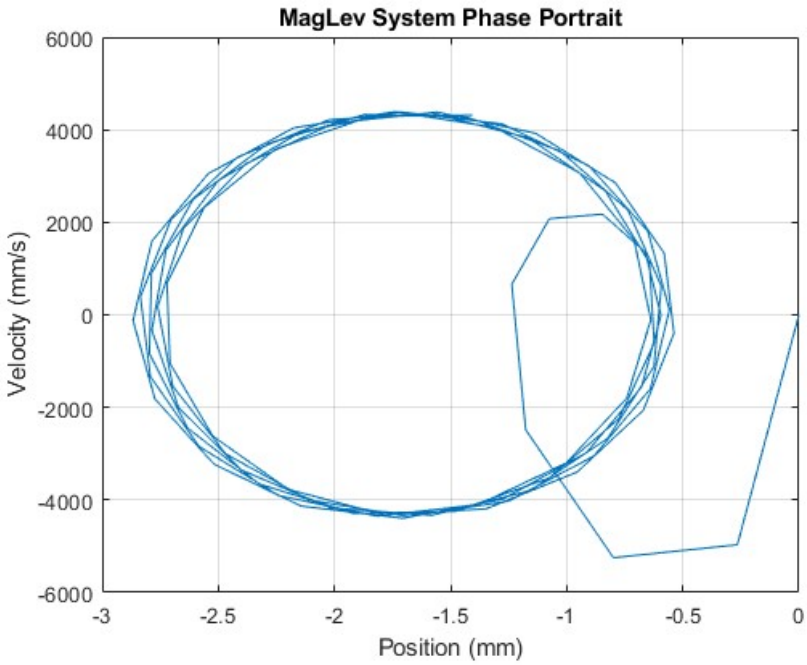

Figure 47 Maglev system with a reference sinusoidal wave of 4000 rad/sec under DLDL

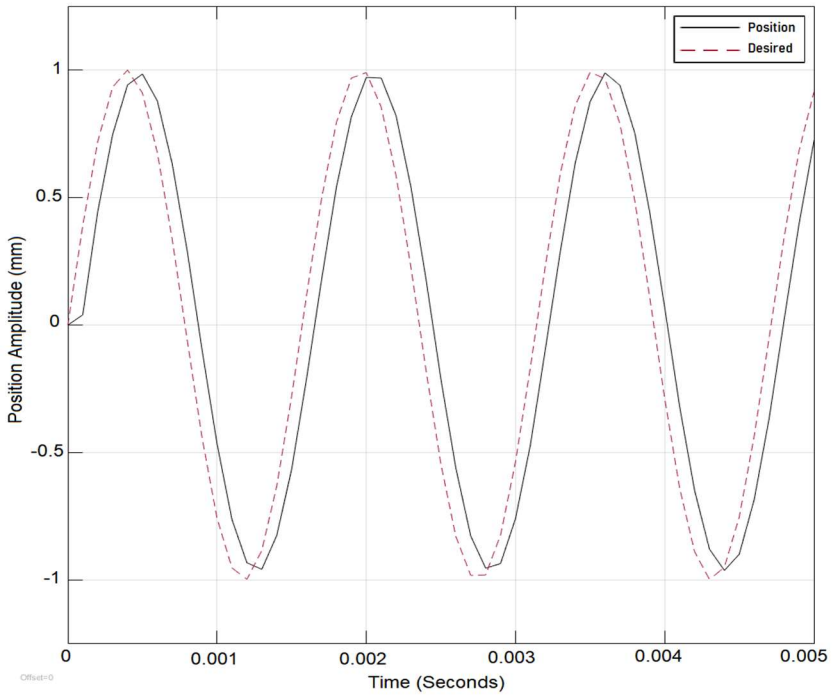

Figure 48 DLDL controlled position with reference to sinusoidal wave of 4000 rad/s frequency

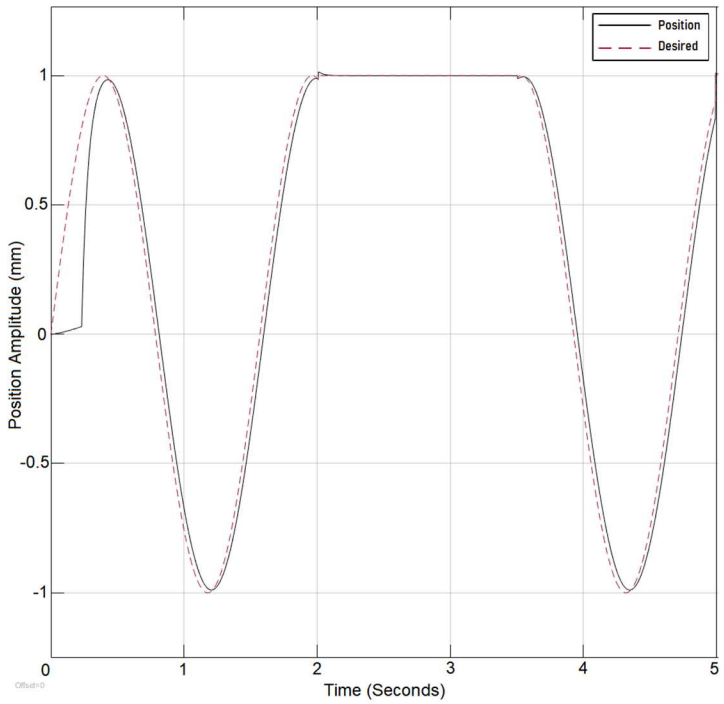

Figure 49 The position with reference to a combination of sinusoidal and step function

In this chapter we presented the implementation of DLDL on the magnetic levitation non-linear system. The integration of Lyapunov Control, Deep Learning and Dynamic Layering is shown to achieve safe parameter recommendations and control within 0.1 ms to 8 s from triggering the DNN depending on the number of parameters that require change and the type of nonlinear dynamics introduced. The algorithm is shown to yield successful results after testing different scenarios of switching inputs or applying high frequency input.

CHAPTER SEVEN

CONCLUSION AND FUTURE WORK

7.1 Conclusion

In this research we studied the effect of utilizing Lyapunov non-linear stability theory as a form of non-linear control. The developed control is tested on Duffing, van der pol and Zohdy-Harb oscillators exhibiting chaotic behavior. The outcome of the simulation is compared to PID control and Model Predictive Control. It was found that utilizing the Lyapunov control function as a feedback control signal yielded the best results from timing to stability to accuracy of control. Compared to PID control Lyapunov control was able to provide high accuracy feedback within timing bounds of 0.2 to 0.4 seconds depending on nonlinear system complexity. The drawback of Lyapunov control was found to be that parameter tuning had substantial effect on the controller agility and the system output. Moreover, it was found that by incrementing the frequency of the Duffing, Van der pol or Zohdy-Harb oscillation above 4000 Hz the controller required re-tuning to adjust. Therefor we began to study the integration of deep learning with the nonlinear Lyapunov control to determine the optimal parameters for system stability.

The study demonstrates that the integration of deep learning enables the agility and robustness of the controller through the selection of the optimal initial parameters for the controller, as well as recalibration of the parameters if disturbances or new dynamics are added to the system. On the other hand, deep learning was found to be computationally expensive with the increase of the number of parameters and hidden

layers. The combination between Lyapunov and Deep Learning yielded outstanding results given one or two parameters only required change but with the increase of the number of parameters to more than 2 the system was unable to converge to the optimal parameters within time. Therefor the novel Dynamic Deep Learning control was introduced along with enhanced custom architecture to tackle large numeric data. Different data sets were studied and presented as well as different network architectures. Their effect on the Root Mean Square Error was demonstrated in this research. Dynamic deep learning allows a change in the number of layers based on the Data set complexity and gives priority to the highest parameter with correlation to the error using the Person Correlation theory. Dynamic deep learning integration with Lyapunov control was found to be successful in tackling the timing constraint.

Therefore, the proposed approach is able to generate safe parameter recommendations within 0.1 ms to 8 s following the activation of the DNN, depending on the number of parameters requiring modification and the nonlinear dynamics introduced. After evaluating various input switching and high frequency input settings, it is demonstrated that the combination of Lyapunov Non-linear control and Deep Learning produces safe and agile control that could be utilized within fully autonomous system. The proposed control method is also simulated and tested on a Magnetic Levitation non-linear system. The Maglev application is selected due to its wide use in the medical, transportation and space exploration fields. The controller is shown to be successful in adapting to high frequency changes and the introduction of disturbances, which are two of the main issues faced by the previously mentioned fields.

7.2 Future work

Future work considerations include the study of other Neural Network architectures such as dual network approaches Generative Adversarial Network (GAN) , the network would be capable of determining which control method would have the most impact on stabilizing the system and reducing errors while adjusting for parameters. The impact of adjusting the control method based on the observed type of instability can be examined. Unsupervised types of machine learning architectures can be investigated. In addition, a further study objective is to investigate machine learning based observers to estimate unknown state measurement.

Traditional state observers are built based on the modeled system. Therefor a typical observer performance will depend on the accuracy of the mathematical model of a system. In future research in addition to adjusting the control law and parameters a machine learning data driven state observer can be implemented to act as a precursor to the system future state. Adjusting the control mechanism beforehand based on the observed state, this would require additional processing power and data sets but can be used in conjunction with parameter modification to achieve system stability as a last option. Furthermore, the use of deep learning in the evaluation of system complexity can be studied in addition to other types of non-linear control, which, when paired with the deep learning techniques used in this study, could yield positive results.

References

[1] J. G. De Gooijer, *Elements of Nonlinear Time Series Analysis and Forecasting*. Springer International Publishing, 2017.

[2] J. M. T. Thompson and H. B. Stewart, *Nonlinear Dynamics and Chaos*. Wiley, 2002.

[3] D. McLean, *Understanding Aerodynamics: Arguing from the Real Physics*. Wiley, 2012.

[4] H. I. Freedman, *Deterministic mathematical models in population ecology*. Marcel Dekker Incorporated, 1980.

[5] P. BergS, Y. Pomeau, and C. Vidal, "Order within chaos: towards a deterministic approach to turbulence," ed: Wiley and Sons, New York, 1984.

[6] K. M. Cuomo and A. V. Oppenheim, "Circuit implementation of synchronized chaos with applications to communications," *Physical review letters,* vol. 71, no. 1, p. 65, 1993.

[7] M. Arnold and G. Andersson, "Model predictive control of energy storage including uncertain forecasts," in *Power Systems Computation Conference (PSCC), Stockholm, Sweden*, 2011, vol. 23: Citeseer, pp. 24-29.

[8] C. E. Garcia, D. M. Prett, and M. Morari, "Model predictive control: Theory and practice—A survey," *Automatica,* vol. 25, no. 3, pp. 335-348, 1989.

[9] R. Findeisen and F. Allgöwer, "An Introduction to Nonlinear Model Predictive Control In CW Scherer and JM Schumacher, editors, Summerschool on The Impact of Optimization in Control," *Dutch Institute of Systems and Control, DISC,* 2001.

[10] V. I. Utkin, "Sliding mode control design principles and applications to electric drives," *IEEE transactions on industrial electronics,* vol. 40, no. 1, pp. 23-36, 1993.

[11] A. Isidori and W. Kang, "H/sub/spl infin control via measurement feedback for general nonlinear systems," *IEEE Transactions on Automatic Control,* vol. 40, no. 3, pp. 466-472, 1995.

[12] A. Behal, W. Dixon, D. M. Dawson, and B. Xian, *Lyapunov-Based Control of Robotic Systems*. CRC Press, 2009.

[13] R. A. Freeman and P. V. Kokotović, "Robust Nonlinear Control Design (illustrated, reprint ed.)," ed: Birkhäuser, 2008.

[14] H. K. Khalil, *Nonlinear control*. Pearson New York, 2015.

[15] M. Alghassab, A. Mahmoud, and M. A. Zohdy, "Nonlinear control of chaotic forced Duffing and van der pol oscillators," *International Journal of Modern Nonlinear Theory and Application,* vol. 6, no. 01, p. 26, 2017.

[16] A. Mahmoud, Y. Ismaeil, and M. Zohdy, "Continuous Lyapunov Controlled Nonlinear System Optimization Using Deep Learning with Memory," *International Journal of Control Science and Engineering,* vol. 10, no. 2, pp. 23-30, 2020.

[17] A. Mahmoud and M. Zohdy, "Dynamic Lyapunov Machine Learning Control of Nonlinear Magnetic Levitation System," *Energies,* vol. 15, no. 5, p. 1866, 2022.

[18] H. G. Kwatny and G. Blankenship, *Nonlinear Control and Analytical Mechanics: A Computational Approach*. Birkhäuser Boston, 2000.

[19] F. Allgöwer and A. Zheng, *Nonlinear model predictive control*. Birkhäuser, 2012.

[20] F.-C. Chen and H. K. Khalil, "Adaptive control of nonlinear systems using neural networks," *International journal of control,* vol. 55, no. 6, pp. 1299-1317, 1992.

[21] T. A. Johansen, "Introduction to nonlinear model predictive control and moving horizon estimation," *Selected topics on constrained and nonlinear control,* vol. 1, pp. 1-53, 2011.

[22] W. H. Kwon, A. Bruckstein, and T. Kailath, "Stabilizing state-feedback design via the moving horizon method," *International Journal of Control,* vol. 37, no. 3, pp. 631-643, 1983.

[23] D. Q. Mayne and H. Michalska, "Receding horizon control of nonlinear systems," in *Proceedings of the 27th IEEE Conference on Decision and Control*, 1988: IEEE, pp. 464-465.

[24] E. F. Camacho and C. B. Alba, *Model predictive control*. Springer science & business media, 2013.

[25] Y. Jing, H. Sun, L. Zhang, and T. Zhang, "Variable speed control of wind turbines based on the quasi-continuous high-order sliding mode method," *Energies,* vol. 10, no. 10, p. 1626, 2017.

[26] F. Mahini, L. DiWilliams, K. Burke, and H. Ashrafiuon, "An experimental setup for autonomous operation of surface vessels in rough seas," *Robotica,* vol. 31, no. 5, pp. 703-715, 2013.

[27] Z. Michalewicz, C. Z. Janikow, and J. B. Krawczyk, "A modified genetic algorithm for optimal control problems," *Computers & Mathematics with Applications,* vol. 23, no. 12, pp. 83-94, 1992.

[28] W.-D. Chang, "Nonlinear system identification and control using a real-coded genetic algorithm," *Applied Mathematical Modelling,* vol. 31, no. 3, pp. 541-550, 2007.

[29] A. J. Taylor, V. D. Dorobantu, H. M. Le, Y. Yue, and A. D. Ames, "Episodic learning with control lyapunov functions for uncertain robotic systems," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019: IEEE, pp. 6878-6884.

[30] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011: JMLR Workshop and Conference Proceedings, pp. 627-635.

[31] S. Ross, G. J. Gordon, and J. A. Bagnell, "No-regret reductions for imitation learning and structured prediction," in *In AISTATS*, 2011: Citeseer.

[32] A. D. Ames, K. Galloway, K. Sreenath, and J. W. Grizzle, "Rapidly exponentially stabilizing control lyapunov functions and hybrid zero dynamics," *IEEE Transactions on Automatic Control,* vol. 59, no. 4, pp. 876-891, 2014.

[33] H. Dai, B. Landry, L. Yang, M. Pavone, and R. Tedrake, "Lyapunov-stable neural-network control," *arXiv preprint arXiv:2109.14152,* 2021.

[34] H. Ravanbakhsh and S. Sankaranarayanan, "Learning control lyapunov functions from counterexamples and demonstrations," *Autonomous Robots,* vol. 43, no. 2, pp. 275-307, 2019.

[35] S. M. Richards, F. Berkenkamp, and A. Krause, "The lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems," in *Conference on Robot Learning*, 2018: PMLR, pp. 466-476.

[36] M. Lohstroh, P. Derler, and M. Sirjani, *Principles of Modeling*. Springer, 2018.

[37] T. El-Mezyani *et al.*, "Evaluation of nonlinearity and complexity in SSTs systems," in *2012 IEEE International Systems Conference SysCon 2012*, 2012: IEEE, pp. 1-7.

[38] N. C. Thompson, K. Greenewald, K. Lee, and G. F. Manso, "The computational limits of deep learning," *arXiv preprint arXiv:2007.05558,* 2020.

[39] X. Liu *et al.*, "Privacy and security issues in deep learning: A survey," *IEEE Access,* vol. 9, pp. 4566-4593, 2020.

[40] M. Zarei, Y. Wang, and M. Pajic, "Statistical verification of learning-based cyber-physical systems," in *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control*, 2020, pp. 1-7.

[41] S. Li, C. K. Ahn, J. Guo, and Z. Xiang, "Neural network-based sampled-data control for switched uncertain nonlinear systems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems,* vol. 51, no. 9, pp. 5437-5445, 2019.

[42] R. C. Rego and F. M. de Araújo, "Learning-based robust neuro-control: A method to compute control Lyapunov functions," *International Journal of Robust and Nonlinear Control,* vol. 32, no. 5, pp. 2644-2661, 2022.

[43] A. M. Lyapunov, "The general problem of motion stability," *Annals of Mathematics Studies,* vol. 17, no. 1892, 1892.

[44] C. R. Bermudez-Gomez, R. Enriquez-Caldera, and J. Martinez-Carballido, "Chirp signal detection using the Duffing oscillator," in *CONIELECOMP 2012, 22nd International Conference on Electrical Communications and Computers*, 2012: IEEE, pp. 344-349.

[45] A. A. Zaher, "Secure communication using Duffing oscillators," in *2011 IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*, 2011: IEEE, pp. 563-568.

[46] J. Li and Y. Shen, "The Study of Weak Signal Detection Using Duffing Oscillators Array," in *2009 IEEE Circuits and Systems International Conference on Testing and Diagnosis*, 2009: IEEE, pp. 1-4.

[47] Y. Wang, H. Li, and W. Dai, "Application of Duffing oscillator in ship propeller blade number recognition," in *2016 IEEE/OES China Ocean Acoustics (COA)*, 2016: IEEE, pp. 1-5.

[48] D. Peluffo-Ordóñez, J. Rodríguez-Sótelo, E. Revelo-Fuelagán, C. Ospina-Aguirre, and G. Olivard-Tost, "Generalized Bonhoeffer-van der Pol oscillator for modelling cardiac pulse: Preliminary results," in *2015 IEEE 2nd Colombian Conference on Automatic Control (CCAC)*, 2015: IEEE, pp. 1-6.

[49] K. O. Menzel, T. Bockwoldt, O. Arp, and A. Piel, "Modeling dust-density wave fields as a system of coupled van der Pol oscillators," *IEEE Transactions on Plasma Science,* vol. 41, no. 4, pp. 735-739, 2012.

[50] C.-L. Kuo, N.-S. Pai, S.-M. Liang, and S.-H. Hu, "Fuzzy Sliding-Model Control for Synchronization of an Uncertain Duffing-Holmes System," in *2008 IEEE International Symposium on Knowledge Acquisition and Modeling Workshop*, 2008: IEEE, pp. 104-107.

[51] A. Jimenez-Triana, W. K.-S. Tang, G. Chen, and A. Gauthier, "Chaos control in Duffing system using impulsive parametric perturbations," *IEEE Transactions on Circuits and Systems II: Express Briefs,* vol. 57, no. 4, pp. 305-309, 2010.

[52] J. Guckenheimer and P. Holmes, *Nonlinear oscillations, dynamical systems, and bifurcations of vector fields*. Springer Science & Business Media, 2013.

[53] R. Chuan-bo, Z. Zhen, and L. Lin, "Bifurcation and Chaos control of Van der pol system with delay," in *2011 Chinese Control and Decision Conference (CCDC)*, 2011: IEEE, pp. 957-963.

[54] Z. Yang, T. Jiang, and Z. Jing, "Chaos Control in Duffing-Van Der Pol System," in *2010 International Workshop on Chaos-Fractal Theories and Applications*, 2010: IEEE, pp. 106-110.

[55] A. M. Harb, A. A. Zaher, A. A. Al-Qaisia, and M. A. Zohdy, "Recursive backstepping control of chaotic Duffing oscillators," *Chaos, Solitons & Fractals,* vol. 34, no. 2, pp. 639-645, 2007.

[56] T. Bäck and H.-P. Schwefel, "An overview of evolutionary algorithms for parameter optimization," *Evolutionary computation,* vol. 1, no. 1, pp. 1-23, 1993.

[57] T. Duriez, S. L. Brunton, and B. R. Noack, *Machine learning control-taming nonlinear dynamics and turbulence*. Springer, 2017.

[58] N. Benard, J. Pons-Prat, J. Periaux, G. Bugeda, J.-P. Bonnet, and E. Moreau, "Multi-input genetic algorithm for experimental optimization of the reattachment downstream of a backward-facing-step with surface plasma actuator," in *46th AIAA Plasmadynamics and lasers conference*, 2015, p. 2957.

[59] C. Lee, J. Kim, D. Babcock, and R. Goodman, "Application of neural networks to turbulence control for drag reduction," *Physics of Fluids,* vol. 9, no. 6, pp. 1740-1747, 1997.

[60] D. C. Dracopoulos and A. J. Jones, "Neuro-genetic adaptive attitude control," *Neural Computing & Applications,* vol. 2, no. 4, pp. 183-204, 1994.

[61] S. L. Brunton and B. R. Noack, "Closed-loop turbulence control: Progress and challenges," *Applied Mechanics Reviews,* vol. 67, no. 5, 2015.

[62] J. Javadi-Moghaddam and A. Bagheri, "An adaptive neuro-fuzzy sliding mode based genetic algorithm control system for under water remotely operated vehicle," *Expert Systems with Applications,* vol. 37, no. 1, pp. 647-660, 2010.

[63] P. J. Fleming and R. C. Purshouse, "Evolutionary algorithms in control systems engineering: a survey," *Control engineering practice,* vol. 10, no. 11, pp. 1223-1241, 2002.

[64] S. W. Fentress, "Exaptation as a means of evolving complex solutions," Citeseer, 2005.

[65] K. L. Sadowski, P. A. Bosman, and D. Thierens, "On the usefulness of linkage processing for solving MAX-SAT," in *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, 2013, pp. 853-860.

[66] M. Bodor, R. Santos, T. Gerven, and M. Vlad, "Recent developments and perspectives on the treatment of industrial wastes by mineral carbonation—a review," *Open Engineering,* vol. 3, no. 4, pp. 566-584, 2013.

[67] T. Miura, C. Braendle, A. Shingleton, G. Sisk, S. Kambhampati, and D. L. Stern, "A comparison of parthenogenetic and sexual embryogenesis of the pea aphid Acyrthosiphon pisum (Hemiptera: Aphidoidea)," *Journal of Experimental Zoology Part B: Molecular and Developmental Evolution,* vol. 295, no. 1, pp. 59-81, 2003.

[68] B. J. Pandian and M. M. Noel, "Control of a bioreactor using a new partially supervised reinforcement learning algorithm," *Journal of Process Control,* vol. 69, pp. 16-29, 2018.

[69] M. Wiering and M. Van, "Otterlo. Reinforcement learning," *Adaptation, learning, and optimization,* vol. 12, no. 3, 2012.

[70] R. Williams, "A class of gradient-estimation algorithms for reinforcement learning in neural networks," in *Proceedings of the International Conference on Neural Networks*, 1987, pp. II 601-II 608.

[71] A. S. Bazanella, L. Campestrini, and D. Eckhard, *Data-driven controller design: the H2 approach*. Springer Science & Business Media, 2011.

[72] H. Hjalmarsson, M. Gevers, S. Gunnarsson, and O. Lequin, "Iterative feedback tuning: theory and applications," *IEEE control systems magazine,* vol. 18, no. 4, pp. 26-41, 1998.

[73] M. Nikolaou, "Model predictive controllers: A critical synthesis of theory and industrial needs," 2001.

[74] J. Berberich, J. Köhler, M. A. Müller, and F. Allgöwer, "Linear tracking MPC for nonlinear systems—Part I: The model-based case," *IEEE Transactions on Automatic Control,* vol. 67, no. 9, pp. 4390-4405, 2022.

[75] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of learning and motivation*, vol. 24: Elsevier, 1989, pp. 109-165.

[76] A. Robins, "Catastrophic forgetting, rehearsal and pseudorehearsal," *Connection Science,* vol. 7, no. 2, pp. 123-146, 1995.

[77] A. Robins, "Consolidation in neural networks and in the sleeping brain," *Connection Science,* vol. 8, no. 2, pp. 259-276, 1996.

[78] B. Ans and S. Rousset, "Neural networks with a self-refreshing memory: knowledge transfer in sequential learning tasks without catastrophic forgetting," *Connection science,* vol. 12, no. 1, pp. 1-19, 2000.

[79] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research,* vol. 15, no. 1, pp. 1929-1958, 2014.

[80] S. Basodi, C. Ji, H. Zhang, and Y. Pan, "Gradient amplification: An efficient way to train deep neural networks," *Big Data Mining and Analytics,* vol. 3, no. 3, pp. 196-207, 2020.

[81] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, 2015: PMLR, pp. 448-456.

[82] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, "How does batch normalization help optimization?," *Advances in neural information processing systems,* vol. 31, 2018.

[83] L. E. Jeffrey, "Finding structure in time," *Cognitive science,* vol. 14, no. 2, pp. 179-211, 1990.

[84] O. Calin, *Deep Learning Architectures: A Mathematical Approach*. Springer International Publishing, 2020.

[85] A. El Hajjaji and M. Ouladsine, "Modeling and nonlinear control of magnetic levitation systems," *IEEE Transactions on industrial Electronics,* vol. 48, no. 4, pp. 831-838, 2001.

[86] K. Kecik, A. Mitura, S. Lenci, and J. Warminski, "Energy harvesting from a magnetic levitation system," *International Journal of Non-Linear Mechanics,* vol. 94, pp. 200-206, 2017.

[87] S. Priya, "Advances in energy harvesting using low profile piezoelectric transducers," *Journal of electroceramics,* vol. 19, no. 1, pp. 167-184, 2007.

[88] Z. Li, L. Zuo, G. Luhrs, L. Lin, and Y.-x. Qin, "Electromagnetic energy-harvesting shock absorbers: design, modeling, and road tests," *IEEE Transactions on vehicular technology,* vol. 62, no. 3, pp. 1065-1074, 2012.

[89] S. Priya and D. J. Inman, *Energy harvesting technologies*. Springer, 2009.

[90] I. Horváth, Z. Rusák, and Y. Li, "Order beyond chaos: Introducing the notion of generation to characterize the continuously evolving implementations of cyber-physical systems," in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 2017, vol. 58110: American Society of Mechanical Engineers, p. V001T02A015.

[91] I. Ahmad, M. Shahzad, and P. Palensky, "Optimal PID control of magnetic levitation system using genetic algorithm," in *2014 IEEE International Energy Conference (ENERGYCON)*, 2014: IEEE, pp. 1429-1433.

[92] R. T. Rocha, J. M. Balthazar, A. M. Tusset, S. L. T. de Souza, F. C. Janzen, and H. C. Arbex, "On a non-ideal magnetic levitation system: nonlinear dynamical behavior and energy harvesting analyses," *Nonlinear Dynamics,* vol. 95, no. 4, pp. 3423-3438, 2019.

[93] K. Kecik and M. Kowalczuk, "Effect of Nonlinear Electromechanical Coupling in Magnetic Levitation Energy Harvester," *Energies,* vol. 14, no. 9, p. 2715, 2021.

[94] E. D. Sontag, "A ‘universal’construction of Artstein's theorem on nonlinear stabilization," *Systems & control letters,* vol. 13, no. 2, pp. 117-123, 1989.

[95] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once-for-all: Train one network and specialize it for efficient deployment," *arXiv preprint arXiv:1908.09791,* 2019.

[96] J. M. Ali, M. A. Hussain, M. O. Tade, and J. Zhang, "Artificial Intelligence techniques applied as estimator in chemical process systems–A literature survey," *Expert Systems with Applications,* vol. 42, no. 14, pp. 5915-5931, 2015.

[97] O. S. Patil, D. M. Le, M. L. Greene, and W. E. Dixon, "Lyapunov-derived control and adaptive update laws for inner and outer layer weights of a deep neural network," *IEEE Control Systems Letters,* vol. 6, pp. 1855-1860, 2021.

[98] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *International conference on machine learning*, 2018: PMLR, pp. 4095-4104.

[99] J. B. Clark and D. R. Jacques, "Practical measurement of complexity in dynamic systems," *Procedia Computer Science,* vol. 8, pp. 14-21, 2012.

[100] R. G. Downey and M. R. Fellows, "Parameterized approximation," in *Fundamentals of Parameterized Complexity*: Springer, 2013, pp. 623-644.

[101] V. M. Gomes, J. R. Paiva, M. R. Reis, G. A. Wainer, and W. P. Calixto, "Mechanism for measuring system complexity applying sensitivity analysis," *Complexity,* vol. 2019, 2019.