Adaptive control of an active magnetic bearing flywheel system using neural networks

---

A dissertation presented to

The School of Electrical, Electronic & Computer Engineering

North-West University

In partial fulfilment of the requirements for the degree

Master Ingeneriae

in Computer and Electronic Engineering

by

Angelique Combrinck

Supervisor: Prof. G. van Schoor

Co-supervisor: Dr. P. A. van Vuuren

December 2010

Potchefstroom Campus

DECLARATION

I hereby declare that all the material used in this dissertation is my own original unaided work except where specific references are made by name or in the form of a numbered reference. The work herein has not been submitted for a degree to another university.

Signed: _____

Angelique Combrinck

SUMMARY

The School of Electrical, Electronic and Computer Engineering at the North-West University in Potchefstroom has established an active magnetic bearing (AMB) research group called McTronX. This group provides extensive knowledge and experience in the theory and application of AMBs. By making use of the expertise contained within McTronX and the rest of the control engineering community, an adaptive controller for an AMB flywheel system is implemented.

The adaptive controller is faced with many challenges because AMB systems are multivariable, nonlinear, dynamic and inherently unstable systems. It is no wonder that existing AMB models are poor approximations of reality. This modelling problem is avoided because the adaptive controller is based on an indirect adaptive control law. Online system identification is performed by a neural network to obtain a better model of the AMB flywheel system. More specifically, a nonlinear auto-regressive with exogenous inputs (NARX) neural network is implemented as an online observer.

Changes in the AMB flywheel system's environment also add uncertainty to the control problem. The adaptive controller adjusts to these changes as opposed to a robust controller which operates despite the changes. Making use of reinforcement learning because no online training data can be obtained, an adaptive critic model is applied. The adaptive controller consists of three neural networks: a critic, an actor and an observer. It is called an observer-based adaptive critic neural controller (ACNC).

Genetic algorithms are used as global optimization tools to obtain values for the parameters of the observer, critic and actor. These parameters include the number of neurons and the learning rate for each neural network. Since the observer uses a different error signal than the actor and critic, its parameters are optimized separately. When the actor and critic parameters are optimized by minimizing the tracking error, the observer parameters are kept constant.

The chosen adaptive control design boasts analytical proofs of stability using Lyapunov stability analysis methods. These proofs clearly confirm that the design ensures stable simultaneous identification and tracking of the AMB flywheel system. Performance verification is achieved by step response, robustness and stability analysis. The final adaptive control system remains stable in the presence of severe cross-coupling effects whereas the original decentralized PD control system destabilizes. This study provides the justification for further research into adaptive control using artificial intelligence techniques as applied to the AMB flywheel system.

Keywords: adaptive control, active magnetic bearing, neural network, system identification.

ACKNOWLEDGEMENTS

King of Kings, You have been my strength and inspiration through so many challenges. I would never have made it without You.

Prof. George van Schoor, thank you for being so much more than just a project supervisor. I appreciate all your support. Dr. Pieter van Vuuren, thank you for always asking the difficult questions. Your insight and guidance go a long way.

Marisha, thank you for all your love and understanding. Esmé and Marné, you both knew exactly what to say and when to say it. My parents, Henry and Ansie Combrinck, and my brothers, Drikus and Henry – even from a distance, I knew you were with me every step of the way. Thank you to all my friends and family.

I would also like to thank Necsa and PBMR Pty. (Ltd.) for their financial support.

"Let the morning bring me word of Your unfailing love, for I have put my trust in You. Show me the way I should go, for to You I entrust my life." – Psalm 143:8

CONTENTS

Chapter 1: Introduction 1

1.1 The basics of an AMB system 1

1.2 The challenges regarding AMB systems 2

1.3 Adaptive control 2

1.4 Artificial neural networks 3

1.5 AMB flywheel system 4

1.6 Research problem 5

1.7 Issues to be addressed and methodology 5

1.7.1 Adaptive control design 5

1.7.2 System identification 5

1.7.3 Modelling using neural networks 6

1.7.4 Online training 6

1.7.5 Implementation 6

1.7.6 Performance verification 6

1.8 Dissertation overview 7

Chapter 2: Literature overview 8

2.1 AMB control systems 8

2.1.1 Introduction 8

2.1.2 Characteristics of AMB systems 8

2.1.3 Advantages of AMBs 10

2.1.4 Shortcomings of AMBs 10

2.1.5 Applications of AMBs 10

2.1.6 Flywheel energy storage systems 11

2.1.7 AMB control methods 11

2.2 Adaptive control 15

2.2.1 Introduction 15

2.2.2 Adaptive control design methods 16

2.2.3 Adaptive control versus fixed control 18

2.2.4 System identification 18

2.2.5 System identification procedure 19
2.2.6 Neural networks as model set 20
2.3 Neural networks 22
2.3.1 Introduction 22
2.3.2 Neural network architectures 23
2.3.3 Activation functions 25
2.3.4 Training neural networks 25
2.3.5 Training and testing strategy 28
2.3.6 Parameter selection 28
2.3.7 Error function 28
2.3.8 Neural network control 28
2.4 Critical overview and conclusions 30

Chapter 3: Architecture selection 32

3.1 Introduction 32
3.2 Architecture features comparison 32
3.3 System identification 34
3.3.1 Prior knowledge 34
3.3.2 Experiment design 34
3.3.3 Obtaining data 36
3.3.4 Choosing a model set 37
3.3.5 Choosing a condition of fit 42
3.3.6 Calculating the model 44
3.3.7 Model validation 47
3.3.8 Model revision 49
3.3.9 Implementation 49
3.3.10 Results 52
3.4 Conclusions 65

Chapter 4: Adaptive control design 68

4.1 Introduction 68
4.2 Indirect adaptive control 69
4.2.1 Using neural networks 70
4.2.2 Reinforcement learning 71
4.3 Adaptive critic model 71

4.4 Control system design 72

4.4.1 Lyapunov stability analysis methods 73
4.4.2 Observer design 74
4.4.3 Actor design 75
4.4.4 Critic design 77
4.4.5 Observer-based ACNC 77
4.4.6 Adaptive laws 78

4.5 Derivative-free optimization 80

4.5.1 Genetic algorithms 80
4.5.2 Optimization process 81
4.5.3 Initialization and selection 81
4.5.4 Crossover and mutation 82

4.6 System identification loop 82

4.7 Performance verification 82

4.7.1 Step response 83
4.7.2 Robustness analysis 83
4.7.3 Stability analysis 84

4.8 Conclusions 85

Chapter 5: Simulation results 86

5.1 Implementation 86

5.1.1 Optimization using genetic algorithms 86
5.1.2 Initialization of network weights 89

5.2 Observer verification 89

5.3 Adaptive controller verification 93

5.3.1 Step response 93
5.3.2 Robustness analysis 96
5.3.3 Stability analysis 98

5.4 Results assessment 99

5.4.1 Step response 99
5.4.2 Robustness analysis 100
5.4.3 Stability analysis 100
5.4.4 Other comments 100

5.5 Conclusions 101

Chapter 6: Conclusions and recommendations..... 102

6.1 Conclusions 102

6.2 Recommendations for future work 104

6.2.1 Eliminate need for state information 104

6.2.2 Increase hidden layers 104

6.2.3 Higher sampling rate for observer 104

6.2.4 Improve network weights initialization 105

6.2.5 Enhance use of genetic algorithms 105

6.2.6 Proven method to select number of neurons 106

6.2.7 Adaptive controller order reduction 106

Appendix A: Project CD 107

References 108

LIST OF FIGURES

Figure 1-1: Schematic of electromagnetic suspension 1
Figure 1-2: Fly-UPS 4
Figure 1-3: Proposed adaptive control system 5
Figure 2-1: Direct adaptive control system 17
Figure 2-2: Indirect adaptive control system 17
Figure 2-3: System identification loop 19
Figure 2-4: Feed-forward single-layer network 22
Figure 2-5: Recurrent single-layer network 24
Figure 2-6: Neural network as function approximator 29
Figure 2-7: Neural networks for assistance 29
Figure 2-8: Neural networks for control 30
Figure 3-1: Simplified discrete-time block diagram of AMB system 32
Figure 3-2: Block diagram of single control loop 35
Figure 3-3: Given **SIMULINK®** control system model 36
Figure 3-4: Excitation points 37
Figure 3-5: FTD architecture 38
Figure 3-6: NARX architecture 38
Figure 3-7: LRN architecture 39
Figure 3-8: Window approach 40
Figure 3-9: Logsig and tansig activation functions 41
Figure 3-10: NARX configurations 46
Figure 3-11: Simplified block diagram 47
Figure 3-12: Frequency response up to 3 kHz 48
Figure 3-13: New system with neural model as plant 48
Figure 3-14: Excitation of AMB 1(x) 50
Figure 3-15: Architecture selection software execution loop 51
Figure 3-16: Excitation signal for AMB 1 52
Figure 3-17: Plant input for AMB 1 52
Figure 3-18: Plant output for AMB 1 53
Figure 3-19: Rotor movement 53
Figure 3-20: Data division 54
Figure 3-21: FTD training results 54
Figure 3-22: Accuracy of first row FTD networks 55
Figure 3-23: Accuracy of second row FTD networks 55

Figure 3-24: Accuracy of final row FTD networks 56
Figure 3-25: FTD output 56
Figure 3-26: NARX training results 57
Figure 3-27: Accuracy of first row NARX networks 57
Figure 3-28: Accuracy of second row NARX networks 58
Figure 3-29: Accuracy of final row NARX networks 58
Figure 3-30: NARX output 59
Figure 3-31: LRN training results 59
Figure 3-32: Accuracy of first row LRN networks 59
Figure 3-33: Accuracy for 2 hidden layers 61
Figure 3-34: Original system step response 61
Figure 3-35: Rotor movement for original system with step input 62
Figure 3-36: FTD step response 62
Figure 3-37: Rotor movement for FTD with step input 63
Figure 3-38: NARX step response 63
Figure 3-39: Rotor movement for NARX with step input 64
Figure 3-40: NARX frequency response 64
Figure 3-41: NARX frequency response (zoomed) 65
Figure 4-1: Indirect adaptive control system (MIMO) 70
Figure 4-2: Discrete-time IAC with neural network (MIMO) 70
Figure 4-3: Adaptive critic model (MIMO) 72
Figure 4-4: Adaptive critic model with observer (MIMO) 72
Figure 4-5: Temporary control system design 72
Figure 4-6: NARX observer 75
Figure 4-7: Actor neural network 76
Figure 4-8: Critic neural network 77
Figure 4-9: Observer-based ACNC 78
Figure 4-10: Measurement of sensitivity data 84
Figure 5-1: Optimization of observer parameters 86
Figure 5-2: AMB 1(x) observer output at 0 r/min and $n = 2$ with PE 89
Figure 5-3: AMB 1(x) observer output at 2,200 r/min and $n = 2$ without PE 90
Figure 5-4: AMB 1(x) observer output at 4,000 r/min and $n = 2$ without PE 90
Figure 5-5: AMB 1(x) observer output at 8,300 r/min and $n = 2$ without PE 91
Figure 5-6: AMB 1(x) observer output at 31,400 r/min Hz and $n = 2$ without PE 91
Figure 5-7: AMB 1(x) observer output at 0 r/min and $n = 20$ with PE 92
Figure 5-8: AMB 1(x) observer output at 8,300 r/min and $n = 20$ without PE 92
Figure 5-9: Step response of PD controller and ACNC with $n = 6$ 94

Figure 5-10: Step response of PD controller and ACNC with n = 12 94
Figure 5-11: Step response of PD controller and ACNC with n = 20 95
Figure 5-12: Step response of PD controller and ACNC with n = 20 for 5 s 96
Figure 5-13: Worst-case sensitivity of PD controller and ACNC with small chirp signal 97
Figure 5-14: Worst-case sensitivity ACNC with large chirp signal 97
Figure 5-15: Worst-case gain and phase margins of PD controller and ACNC with n = 20 98
Figure 5-16: AMB 1(y) output for decentralized PD control 99
Figure 5-17: AMB 1(y) output for ACNC 100
Figure 5-18: Genetic algorithm results for 40 generations 101

LIST OF TABLES

Table 2-1: AMB control design methods 13
Table 2-2: Summary of neural networks 27
Table 3-1: Candidate architectures 33
Table 3-2: Comparison of BP algorithms 45
Table 3-3: Network parameters 60
Table 3-4: Network *mse* values on validation set 60
Table 3-5: Response correlation 65
Table 4-1: Suggested ranges for design parameters 79
Table 5-1: Selected ranges for parameters 88
Table 5-2: Optimal values for ACNC parameters 93

LIST OF ABBREVIATIONS

ACNC	Adaptive critic neural controller
AI	Artificial intelligence
AMB	Active magnetic bearing
ANN	Artificial neural network
BP	Backpropagation
CE	Certainty equivalence
COG	Centre of gravity
DAC	Direct adaptive control
DOF	Degrees-of-freedom
FESS	Flywheel energy storage system
FIS	Fuzzy inference system
FLS	Fuzzy logic system
FNN	Feed-forward neural network
FTD	Focused time-delay neural network
GA	Genetic algorithm
IAC	Indirect adaptive control
ITAE	Integral of time multiplied by absolute error
LIP	Linearity-in-the-parameters
LQ	Linear quadratic
LQG	Linear quadratic Gaussian
LTI	Linear time-invariant
LRN	Layered-recurrent neural network
MIMO	Multiple-input multiple-output
MLP	Multi-layer perceptron
NARMA	Nonlinear auto-regressive moving average
NARX	Nonlinear auto-regressive with exogenous inputs
NN	Neural network
PA	Power amplifier
PD	Proportional derivative
PE	Persistent excitation
PID	Proportional integral derivative
RBF	Radial basis function
RNN	Recurrent neural network

...continued on next page

SISO	Single-input single-output
SOM	Self-organizing Map
SVM	Support vector machine
UFA	Universal function approximation
UPS	Uninterruptible power supply
VSC	Variable structure control
WNN	Wavelet neural network

LIST OF SYMBOLS

α_i	Learning rate of neural network *i*
ζ	Damping ratio
E	Amount of electrical energy
$E(i)$	Mean estimation error of run *i*
f	Magnetic force
$f_{\max}$	Maximum frequency range
g_0	Nominal air gap length
g_m	Gain margin
$G_s(s)$	Sensitivity function
i	Instantaneous current
i_{0ref}	Bias current
i_{ref}	Reference current
I	Moment of inertia
j	Neural network design parameter
k_m	Electromagnetic constant
k_i	Force-current factor
k_s	Force-displacement factor
K_P	Proportional constant
K_D	Derivative constant
l	Length of flywheel
$L(s)$	Loop transfer function
m	Mass of flywheel
mse	Mean squared error

...continued on next page

M	Number of algorithm executions or runs
n	Number of state variables
n_i	Number of hidden layer neurons of network i
φ_m	Phase margin
φ_i	Activation function of network i
$P.O.$	Percentage overshoot
$\boldsymbol{Q}(k)$	Strategic utility function
r	Radius of flywheel
std	Standard deviation
τ	Time constant
T_s	Settling time
$\hat{\mathbf{V}}_i$	Hidden layer weight matrix of network i
ω	Rotational speed
ω_n	Natural frequency
ω_{gc}	Gain crossover frequency
ω_{pc}	Phase crossover frequency
$\hat{\mathbf{W}}_i$	Output layer weight matrix of network i
x_s	Rotor position
x_{ref}	Reference position of rotor
$\mathbf{x}(k)$	State vector in discrete time
$\hat{y}$	Estimate of plant output y

Chapter 1: Introduction

This chapter starts with a brief review of AMB system operation and the challenges associated with AMB system modelling and control. The focus then shifts to the use of neural networks as part of an adaptive control design in an effort to improve the control of an AMB system. A short discussion of the issues specific to this study and their proposed resolutions follow. This introduction concludes with an overview of the rest of this document.

1.1 The basics of an AMB system

Many challenges are associated with the modelling and control of AMB systems. Before anything can be said about these challenges, it is important to grasp the basic components and operation of a typical AMB system. An AMB system consists of a rotor, position sensors, a controller, power amplifiers and electromagnets. There is no contact between rotor and bearing because the rotor is suspended by magnetic forces. Figure 1-1 illustrates the principle of electromagnetic suspension [1].

1. A contact-free sensor measures the position of the rotor.
2. This position measurement is used by the controller to derive a control signal.
3. A power amplifier converts the control signal to a control current.
4. The control current generates a magnetic field in the actuating electromagnets.
5. Resulting magnetic forces ensure the rotor remains suspended.

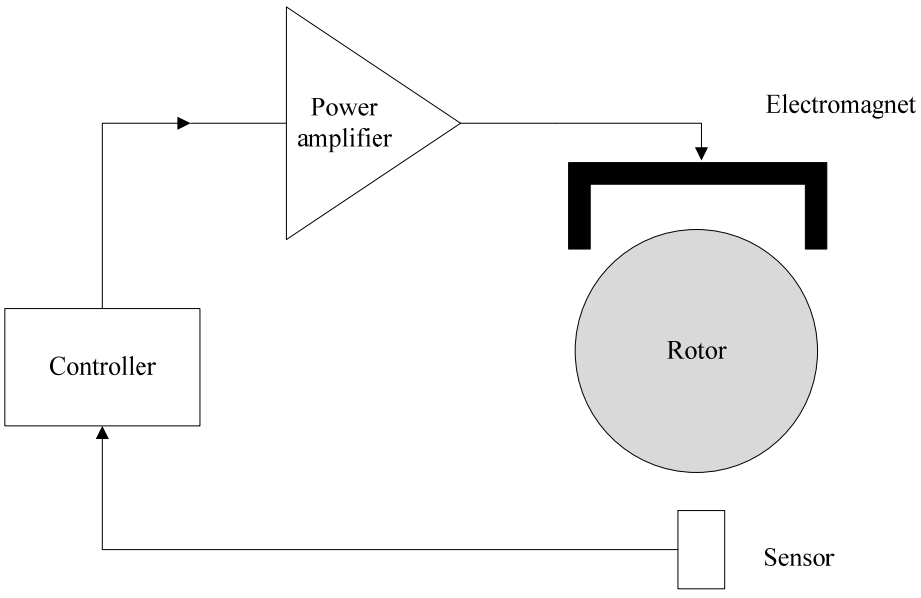

Figure 1-1: Schematic of electromagnetic suspension

1.2 The challenges regarding AMB systems

Apart from the fact that AMB systems are inherently unstable, they are also considered multivariable, nonlinear and dynamic systems. It is not important at this point to understand these system characteristics or know why they exist in an AMB system – it will be explained in the next chapter. Simply note that most of the challenges associated with the modelling and control of an AMB system can be attributed to the above-mentioned system characteristics [1].

Most existing mathematical models of AMB systems are poor approximations of reality which only adds uncertainty to the control problem. These models are usually linear approximations and their accuracy can only be ensured close to a specified operating point [1]. In model-based control design methods, the accuracy of the model has the most significant influence on the success of the design. It is therefore crucial to have a high quality model when using a model-based control design method [2].

Many of the control design methods applied to AMB systems result in fixed (non-adaptive) control systems and aim for a high level of robustness [1]. A system is defined as robust when it performs as desired in the presence of considerable plant uncertainty [3]. This plant uncertainty can be attributed to unknown phenomena which cannot be modelled. Some control design methods attempt to work despite the resulting modelling error whereas other methods aim to adapt to the associated uncertainty. In the latter case, the model used is either a good representation of the system under consideration or it is continually improved [2].

1.3 Adaptive control

It is clear at this point that a need for more accurate modelling and improved control of AMB systems exist. The basic idea behind adaptive control involves updating a model using previous estimates as well as new data to achieve improved control of the system under consideration [4]. According to Dumont and Huzmezan [5], the use of an adaptive controller is justified if a fixed controller cannot produce an acceptable trade-off between stability and performance. Consequently, complex systems with time-varying dynamics were identified by Dumont and Huzmezan [5] as suitable candidates for adaptive control applications.

Ordonez and Passino [6] also showed that adaptive control is best employed when considering uncertain nonlinear systems with time-varying structures. Based on the inherent characteristics of an AMB system, it is indeed a suitable candidate for adaptive control. An adaptive control design aims to adjust the controller based on updated model information. The resulting control system is adaptive to a changing environment. Structures well known for their ability to adapt to changes in their environment are neural networks [7].

1.4 Artificial neural networks

Research in artificial neural networks (ANNs) is one of the numerous branches of the field of artificial intelligence. ANNs are intelligent systems inspired by the workings of biological neural networks. The basic idea of an ANN is to reproduce the processing power as displayed by nervous systems [8]. Exactly what an ANN looks like and how it works, will be explained in the next chapter. The capabilities and advantages of an ANN (or simply NN) are of greater significance now.

Most applications of NNs are aimed at solving problems in function approximation, prediction, optimization, pattern classification, associative memory as well as control [9]. Conventional methods perform relatively well when used to solve these problems but they do so in properly constrained environments and performance is far from satisfactory without these constraints. ANNs have the flexibility and robustness to surpass the performance of conventional methods and provide alternative (and often better) ways to solve the above-mentioned problems [10].

A neural network can correctly approximate any function when provided with adequate input-output data and the structure of the network is properly chosen. This ability is called the universal function approximation (UFA) property and makes the NN a powerful modelling tool. Neural networks are also applicable to multivariable nonlinear problems such as the modelling and control of an AMB system [8].

Some researchers are convinced that neural networks are fault-tolerant to some extent. They believe that if a neural network should become damaged due to a faulty model or external disturbances, the performance of the network will degrade gracefully [11]. When properly trained, a neural network generalizes well to data it has never seen before i.e. it behaves as desired when presented with data different from the training set [8]. The many successful applications of neural networks to nonlinear adaptive control problems also encourage their use in this study [12-16].

A neural network called an observer[1] can be trained online to learn the characteristics of the plant in an AMB system model [12]. The UFA property of a NN implies that this observer network only needs enough input-output data from the plant as well as the appropriate structure to be able to model it. The observer network can supply an adaptive controller with information about the plant and enable the controller to make adjustments to its parameters as necessary. The controller itself might consist of one or more neural networks depending on the complexity of the control problem [8].

[1] Observer is a term commonly used to refer to the structure which estimates a model of an unknown system [8].

1.5 AMB flywheel system

The McTronX research group at the North-West University in Potchefstroom made an AMB system with a flywheel as the rotor available for this study. The system serves as an uninterruptible power supply (UPS) because the flywheel can store energy for later use. This AMB system is accordingly called the Fly-UPS [13].

The system was designed to deliver 2 kW of electricity for a period of 3 minutes during a power failure. The AMBs were designed to suspend the flywheel up to an operating speed of 31,400 r/min but due to safety reasons, the flywheel currently spins at a maximum speed of 8,300 r/min. It employs two radial AMBs and one axial AMB resulting in a 5-DOF[2] system as shown in figure 1-2 [13].

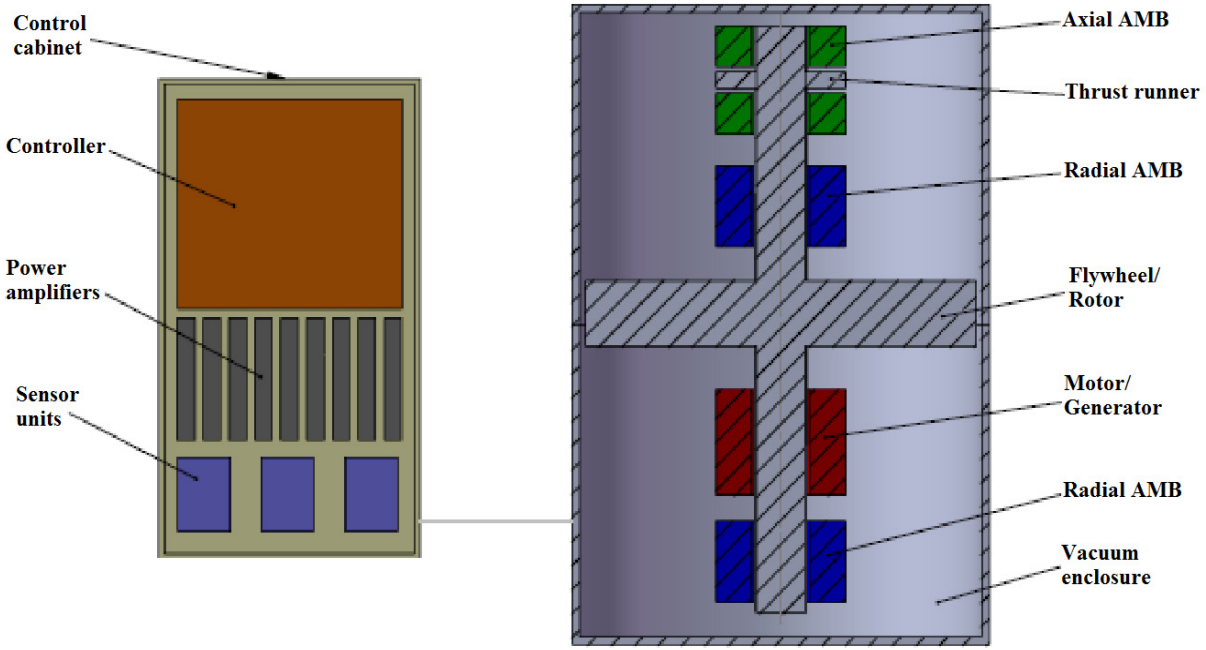

Figure 1-2: Fly-UPS

As can be seen in figure 1-2, the control cabinet contains the electronic system of the Fly-UPS. The electronic system consists of the controller, power amplifiers and sensor units. McTronX was also kind enough to provide a relatively accurate nonlinear flexible-rotor model of this AMB flywheel system which makes use of decentralized PD (proportional derivative) control. Only the 2 radial AMBs are included in the model resulting in a 4-DOF system. This is made possible because the axial and radial suspensions can be considered separately and independently under certain assumptions [1].

[2] The number of degrees of freedom (DOF) refers to the number of motions of a body in space. A rigid body has 6 DOF of motion. Two radial AMBs each enable movement in the x- and y-axis directions and the axial AMB enables movement in the z-axis direction [1].

1.6 Research problem

The primary objective of this study is to obtain an adaptive controller for the Fly-UPS system using online system identification[3], adaptive control design methods and neural networks. The neural networks will be used to continually adapt an AMB plant model online and acquire an adaptive controller for the 4-DOF AMB flywheel system. It is hoped that by using information from a more accurate model to adapt the parameters of the controller, the overall system control will be improved. System performance will be verified using step response, robustness and stability analysis. The proposed control system can be seen in figure 1-3 where the system reference input is represented by $r(t)$, the control signal by $u(t)$ and the system output by $y(t)$.

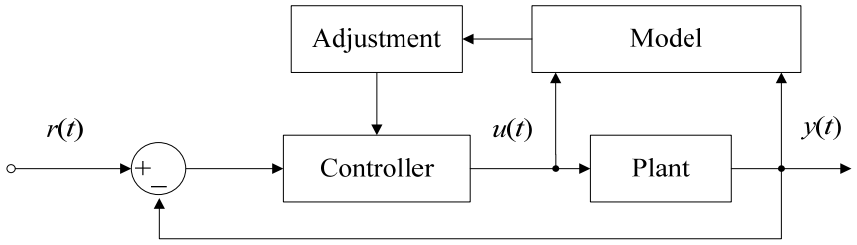

Figure 1-3: Proposed adaptive control system

1.7 Issues to be addressed and methodology

This section focuses on the most important issues or challenges to be faced during the course of this study. The proposed resolution of each issue is also discussed.

1.7.1 Adaptive control design

An adaptive control design method which allows for the use of model information to update the controller parameters is required. The design method is also required to continually adapt the plant model using online system identification. The algorithm used to update the model and the controller parameters should do so while ensuring the stability of the closed-loop system. A comparison of the various existing designs might reveal the most appropriate method. Only design methods with comprehensive analytical proofs will be considered.

1.7.2 System identification

The first step of system identification is obtaining data with adequate information about the system under consideration [14]. The nonlinear flexible-rotor model of the AMB flywheel system provided by McTronX will be used to obtain the necessary data required for system identification. This model encompasses all of the nonlinear characteristics of the 4-DOF AMB flywheel system and includes gyroscopic and unbalance effects [13].

---

[3] System identification is defined as constructing mathematical models of unknown systems using input-output data [14].

A way of comparing the new model with the original plant is also essential. Since the behaviour of the rotor is speed-dependent, a comparison in the time domain will not suffice. Analysis of the new model's frequency response will be compared with the frequency response of the original plant.

1.7.3 Modelling using neural networks

Another concern is the type of neural network to be used for online system identification. The network architecture should have the ability to model the AMB system characteristics as accurately as possible. A proper review of the various network architectures will be essential. This review will help to limit the number of possible architectures to those which are recommended for system modelling and allow for dynamic behaviour.

The candidate architectures have to be properly tested to allow for a fair comparison. This comparison will enable the selection of the most suitable architecture to model the plant of the AMB flywheel system. A quick offline system identification experiment can be used to test each of the architectures and expose their respective strengths and weaknesses.

1.7.4 Online training

An issue arises when considering the time scale of different processes in an adaptive control scheme. Three different time scales usually exist: underlying closed-loop dynamics, plant identification and plant parameter variations. If the identification is not faster than the plant variations, instability can occur [15]. This issue will be resolved by choosing an appropriate training algorithm and sampling rate for the adaptive controller.

1.7.5 Implementation

The offline system identification experiment will require quick and easy implementation. **MATLAB®** contains a neural network software package, the *Neural Network Toolbox*™, which can be used to experiment with different network architectures and training algorithms. The adaptive controller has to be implemented in the same environment as the original model provided by McTronX to allow for easy comparison. Consequently, **SIMULINK®** will be used to develop the simulation model of the adaptive control system.

1.7.6 Performance verification

A way of verifying the performance of the resulting adaptive control system is necessary. The stability and robustness of the original decentralized PD control system and the adaptive control system will be determined. This will enable a comparison between the original and the adaptive system and provide a way of verifying the performance of the adaptive control system.

1.8 Dissertation overview

Chapter 2 provides a broad overview of the relevant literature. The fields of AMB systems, adaptive control and neural networks are the main focus. AMB systems in general and the AMB flywheel system in particular are discussed first. The chapter continues with a discussion on adaptive control of nonlinear systems and a brief comparison between adaptive and fixed control methods. The final section introduces artificial intelligence and more specifically, neural networks and neural network control.

Chapter 3 covers the details of an offline system identification experiment comparing different neural network architectures. The estimation error and accuracy of each network are used to perform the comparison. The aim is to obtain the most suitable architecture to model the plant section of the AMB flywheel system. The neural model is then used in the original system as the plant and the frequency response of the closed-loop system is determined. The frequency response of the original system is compared to the frequency response of this new system to validate the neural model.

Chapter 4 starts by describing the system under consideration and reviewing indirect adaptive control using neural networks. The use of reinforcement learning in an online adaptive control system is motivated. The focus then shifts to the details of an adaptive critic controller. A discussion on the chosen adaptive control design follows and finally, optimization of the controller parameters using genetic algorithms is explained.

Chapter 5 contains the implementation and simulation results of the chosen adaptive controller. The observer optimization results are presented first. Its response at various rotor speeds is shown and compared with the original system output. The step response of the new control system is compared with the step response of the original PD control system. Performance verification continues with robustness and stability analysis of each control system.

Chapter 6 starts by drawing conclusions using results from the previous chapters. Important issues addressed during this study are highlighted and the outcome of proposed methodologies discussed. Concluding remarks continue by discussing the difference in performance between the decentralized PD control system and the adaptive control system. The chapter closes with recommendations for future work aimed at improving the adaptive controller as applied to the AMB flywheel system.

Chapter 2: Literature overview

The objective of this chapter is to provide a broad overview of the three fields associated with this study: active magnetic bearings, adaptive control and neural networks. AMB systems in general and the AMB flywheel system in particular are discussed first. The chapter continues with a discussion on adaptive control of nonlinear systems and a brief comparison between adaptive and fixed control methods. The final section introduces artificial intelligence and more specifically, neural networks and neural network control.

2.1 AMB control systems

2.1.1 Introduction

This section provides a short summary of AMB control systems. The characteristics of AMB systems as well as their advantages and shortcomings are discussed. The application areas of AMBs, including flywheel systems, are reviewed and conventional AMB control methods are discussed.

2.1.2 Characteristics of AMB systems

The terms used most to describe AMB systems are: unstable, nonlinear and dynamic. Before explaining why these terms are attributed to an AMB system, it is important to understand what they mean.

If an LTI (linear time-invariant) system were to be subjected to a bounded input or disturbance and its response is also bounded, it is called a stable system. In mathematical terms, all the poles of the system transfer function have to be on the left-hand side of the imaginary axis for a system to be stable. An unstable LTI system would be exactly the opposite: an unbounded response or poles on the right-hand side of the imaginary axis [3].

Stability analysis of nonlinear time-varying systems is much more involved and no generally applicable method exists [8]. A linear system is said to satisfy the principle of superposition and the property of homogeneity. It simply implies that a linear combination of inputs leads to the same linear combination of outputs in a linear system. This is not the case for a nonlinear system and the relationship between the inputs and outputs may not be clearly observable. One or more parameters vary with time in a time-varying system and adds to the complexity of the system's behaviour [3].

A system is called static when the output does not depend explicitly on time. The current value of the output depends only on current values of external signals in such a system. In a dynamic system the current value of the output also depends on previous values of external signals [14]. Each of these terms with regard to AMB systems will now be explained.

An AMB has a negative mechanical stiffness which causes a pole in the right-hand side of the complex domain and makes the AMB inherently unstable. The nonlinearity can be attributed to the dependency of the magnetic force in an AMB on the size of the air gap between the electromagnet and the rotor as well as on the current in the coil. Equation (2.1) shows how this magnetic force (*f*) is inversely proportional to the square of the size of the air gap (*x*) as well as proportional to the square of the current (*i*) in the coil [1].

$$f = k_m \frac{i^2}{x^2} \tag{2.1}$$

Saturation and hysteresis in the ferromagnetic material of the magnets also contribute to the nonlinearity of the AMB. An AMB is considered dynamic because magnetic saturation and hysteresis also make the system's behaviour time-dependent and builds memory into the system [1].

When AMBs are used to suspend a spinning rotor, the system becomes much more complex. Rotating systems exhibit speed-dependent dynamic behaviour. Consider a spinning rotor with I_{z0} its inertia about the axis of rotation *z* and ω the rotational speed in rad/s. According to (2.2), when the speed of the rotor increases, a gyroscopic term **G** becomes bigger [1].

$$\mathbf{G} \propto I_{z0}\omega \tag{2.2}$$

When the rotor is more or less disc-like, its inertia I_{z0} about the axis of rotation *z* becomes large and consequently, **G** also becomes bigger. Gyroscopic effects are characterized by **G** and contribute to the dynamic behaviour of rotor vibrations. These vibrations are called unbalance effects because they are caused by rotor unbalance [1]. A rotor also becomes more and more flexible as the speed increases and gives rise to flexible modes [16].

The 4-DOF AMB control system under consideration for this study uses two radial AMBs to direct the position of a flexible rotor in both the x- and y-axes. Each set of electromagnets (a set for each AMB in each direction) is controlled independently. Non-contact sensors are arranged perpendicularly on the x- and y-axes and are used to determine the position of the rotor. This configuration of magnets and sensors results in a four-input four-output system and characterizes the AMB system as multivariable [1].

An AMB system is further characterized by various sources of uncertainty. The actuator[4] stiffness is easily affected by static rotor load and anything which changes the air gap in the actuator. Its high level of sensitivity makes it the most significant source of uncertainty in an AMB system [1].

[4] Actuator refers to the combination of the power amplifier and electromagnetic bearing [1].

The damping of flexible modes is sensitive to operating conditions, temperature and aging and makes it another important source of uncertainty. Other sources of uncertainty include the gain between the sensors and controller, rotor speed in some systems and in the case of model-based control, the model used to approximate the plant [1].

2.1.3 Advantages of AMBs

The most significant advantage of an AMB is the fact that it suspends a rotor using only magnetic fields without any contact from liquid or ball bearings as found in conventional systems. This allows for high rotational speeds whilst avoiding contaminating wear and greatly reduces maintenance costs. Operation costs are also reduced because AMBs have 5 to 20 times lower bearing losses than conventional bearings at high operating speeds [1].

Rotor unbalance vibrations are reduced because the active feedback control of the system can evaluate and change the system damping and stiffness characteristics. The AMB system provides measurement information online which allows for immediate diagnosis and improved system reliability [1].

2.1.4 Shortcomings of AMBs

Developing an AMB system sometimes involves demanding software and the associated hardware costs can be high. Designing an AMB for a specific application requires extensive knowledge of mechanics, electronics and control in addition to knowledge about the specific application area [1].

Retainer bearings are used to prevent damage to the rotor-bearing system if the AMB becomes overloaded or it malfunctions. These bearings support the spinning rotor until it comes to a complete standstill or the system regains control of the rotor. The retainer bearings are problematic because their design depends on the specific application. Even though solutions to this design problem exist, it still requires extra attention [1].

2.1.5 Applications of AMBs

Industrial systems such as jet engines, compressors, heart pumps and flywheel energy storage systems are required to meet stringent specifications: high speed, no mechanical wear, clean environment, low vibration and high precision. The advantages of AMBs make them useful for a number of industrial systems [17]. The main application areas for magnetic bearings include: vacuum systems, machine tools, medical devices, and turbo-machinery [1].

The system under consideration for this study, a flywheel energy storage system, is an example of a vacuum system. Magnetic bearings are used to suspend the flywheel in a vacuum without lubrication – a system with no mechanical wear or contamination [18].

Energy losses due to air resistance do not exist in a vacuum and magnetic bearings have relatively low power requirements. All of these advantages are especially important in a system geared towards energy storage [18].

2.1.6 Flywheel energy storage systems

Electricity cannot be stored in its original form. The energy has to be converted into a form which can be stored: mechanical, chemical or thermal. The flywheel can be used in an energy storage system to store energy mechanically. A typical flywheel energy storage system (FESS) transfers energy to and from the flywheel using a motor/generator configuration [19].

Electricity supplied to the stator winding of the motor is converted to torque and causes the flywheel to spin and gain kinetic energy. The generator converts kinetic energy stored in the flywheel to electricity by applying a torque. The motor/generator set is usually one machine functioning as a motor or generator depending on the phase angle [20]. Equation (2.3) explains that the amount of energy (E) stored depends on the moment of inertia (I) of the flywheel and the square of the rotational speed (ω) of the flywheel [18].

$$E = \frac{1}{2} I\omega^2 \tag{2.3}$$

In (2.4), the moment of inertia depends on the length (l), mass (m) and radius (r) of the flywheel [18].

$$I = \frac{lmr^2}{2} \tag{2.4}$$

According to (2.3), the ability of the flywheel to store energy can be improved by increasing the moment of inertia or by spinning the flywheel at higher speeds. Spinning a flywheel at very high speed results in substantial self-discharge of the flywheel due to air resistance and bearing losses. These losses in energy are usually minimized by operating the flywheel at very low pressure (in a vacuum chamber) and using magnetic bearings to suspend the flywheel. The resulting FESS has high overall efficiency [18].

2.1.7 AMB control methods

Controlling an AMB system is challenging for a number of different reasons. An AMB system requires feedback stabilization because it is open-loop unstable. SISO (single-input single-output) control design methods have proven inadequate even though these methods are easier to implement. The only solution is to use more complex multivariable or MIMO (multiple-input multiple-output) control design methods [1].

All the system states have to be accessible for state feedback control design methods. Only a limited number of signals are available in industrial AMB systems, rendering state feedback control design methods impossible. The number of possible control design methods for an AMB system is further limited to output feedback control or methods including the estimation of unavailable states [1].

The following different types of AMB control exist: current control, voltage control and flux-feedback control [16]. The control task remains the design of a controller capable of producing the correct command signal in each case. Even though current control has fewer advantages than voltage control, it is the control method of choice in the industry. Voltage control requires more complex control algorithms than current control and makes it less practical to implement. A number of technical challenges, including the avoidance of magnetic saturation, still surround the implementation of self-sensing magnetic bearings. For this reason and because the implementation platform uses current control, the focus of this study will remain on conventional current control [1].

Designing a controller for an AMB system becomes especially difficult when a flexible rotor is under consideration. A flexible rotor has a significantly wider mechanical bandwidth than a rigid rotor. This simply means that a flexible rotor can have much larger gain than the rigid rotor at high frequencies. The dynamic behaviour of the feedback controller becomes a key issue. The AMB flywheel system is subject to various excitation sources of which rotor unbalance is the most common. When the frequency of the excitation source coincides with a natural frequency of the rotor-bearing system, certain resonance phenomena occur. Each natural frequency has an associated mode shape[5] [1].

At each resonant frequency or critical speed, the rotor vibrations cause the rotor to be bent to the corresponding mode shape. These critical frequencies give rise to the rigid and flexible modes of the rotor. The rotor exhibits little or no deformation at the rigid modes but significant deformation can be observed at flexible modes [1]. The flywheel is modelled as a flexible rotor in the given simulation model since all metals become flexible at high rotational speeds. This means that rigid and flexible modes are included in the model. In the AMB flywheel system, the first two critical frequencies are rigid modes and the third critical frequency corresponds to the first flexible mode [21].

In many industrial AMB applications, the sensors and actuators are not collocated axially along the rotor. This allows the existence of flexible modes with a node between a sensor-actuator pair. These modes add to the dynamic behaviour of the system when their frequencies are within the controller bandwidth. These two flexibility effects have to be addressed in the design of an AMB controller for a flexible rotor [1].

[5] A mode shape is defined as an instantaneous image of the rotor deflection curve at maximum vibration strain [1].

The possibility exists for a flexible rotor to become unstable in speed ranges other than the design range. A popular approach for solving this problem is to design controllers for different speed ranges and switch between them as necessary. This is called gain scheduling [16]. Another approach is called linear parameter varying (LPV) control and consists of designing a controller functionally dependent on the rotor speed [1]. Table 2-1 provides a summary of different AMB control methods and compares their respective strengths and weaknesses:

Table 2-1: AMB control design methods

Method	Properties	Strengths	Weaknesses	Other comments
Proportional integral derivative (PID) control including PD control	Output feedback	Low controller order [22]	System dynamics need to be well understood [1], [23]	Widely used – most industrial applications use decentralized PID control [17]
		Design process ensures high feasibility [1]	Requires extensive knowledge and experience from designer [1], [23]	COG coordinate control superior over decentralized scheme [1]
	Other forms include COG (centre of gravity) coordinate, collocated, non-collocated and mixed [1]	Extends without difficulty [1]	Might become insignificant as robust design methods become better [1]	Many successful applications in decentralized schemes [1], [22], [24]
		Easily applied to decentralized (SISO) control schemes [1], [17]	Controller parameter selection can be time consuming without tuning software [1]	Often combined with genetic algorithms and fuzzy logic systems: [25], [26], [27]
		Suited to industrial applications [1], [28]	Potential instability under certain conditions [1], [29]	
H_∞ control	Output feedback	High robustness [30], [31]	Requires state-space description [1], [30]	Rarely used in industrial applications [1]
		Works well for MIMO control schemes and complex plants [1]	Large computational resources - high order and complexity [1]	Mathematically abstract [1]
	Design using frequency domain weighting functions [32]	Weighting functions determined by specifications [1], [30]	Very sensitive though unbalance performance better than μ-synthesis [1]	Tries to minimize model uncertainty, steady-state error, noise and disturbances [30]
μ-synthesis	Output feedback	Design follows natural reasoning [1]	Negative unbalance performance [1]	Few examples of industrial applications [1]
		Directly address sensitivity issues [1], [33]	Requires state-space description [1]	
	Design using frequency domain weighting functions [32]	Works well for MIMO control schemes and complex plants [1]	Large computational resources - high order and complexity [1]	Powerful theory for independent uncertainties [33]
		Weighting functions determined by specifications [1]		Mathematically abstract [1]

Method	Properties	Strengths	Weaknesses	Other comments
Passive control	Output feedback	Closed-loop stability ensured even with modelling errors [1]	Non-ideal dynamics, plant nonlinearities and digital control threaten passivity property [1]	Not feasible under certain conditions – relative degree of system cannot be higher than one [34]
	Passivity property of system used [1]	Used to design zero-bias control laws [35]		
Pole-placement	State feedback	Closed-loop system dynamics directly treated [1], [36]	All system states have to be available [37]	Familiar method but rarely used [1]
	Place zeros of characteristic equation at desirable locations [36]		Choosing closed-loop system poles difficult without required skill [1]	Performance comparable to sliding mode control [37]
			Treatment of bandwidth limitations and sensor noise challenging [1]	
Linear quadratic (LQ) control	State feedback	Weighting matrices address bandwidth limitations [1]	All system states have to be available [36]	Familiar method but rarely used [1]
	Quadratic cost function is minimized [36]		Selection of weighting matrices require skill and experience [1]	No significant advantages over decentralized control with rigid rotor [1]
Linear quadratic Gaussian (LQG) control	Output feedback	Observer used to estimate non-measurable system states [36]	Selection of weighting matrices require skill and experience	Familiar method but rarely used [1]
	Quadratic cost function is minimized [36]		Sensitivity problems due to uncertainties in observer dynamics [1]	No significant advantages over decentralized control with rigid rotor [1]
			Exact information of plant dynamics needed [1], [36]	Modification necessary for flexible rotor [38]
Artificial intelligence (Fuzzy logic, neural networks)	Systems are intended to exhibit adaptive capabilities in changing environments and possess knowledge of a human expert in a specific field [7]	Different AI approaches can be combined in complimentary fashion [7]	Many AI techniques suffer from curse of dimensionality [7]	Many examples in literature of AI techniques aiding conventional AMB control design methods
		Neural networks can recognize patterns and continually adapt to changes in environment [8]	Determining optimal weights for network can be a difficult task i.e. non-trivial [8]	Very few industrial AMB applications of neural network control: [39], [40]
		Fuzzy systems incorporate human knowledge and have the ability to reason [7]	Classic fuzzy systems cannot adapt to changing environments [7]	Fuzzy logic very often combined with neural networks, genetic algorithms and PD/PID control [7], [25], [41]
Controllers using genetic algorithms	Global optimization method [7]	Genetic algorithms (GAs) are derivative-free global optimization methods [7]	Derivative-free optimization methods take long with large search spaces [7]	GAs often used to tune controller parameters for other methods: [25], [42], [31]

<table>
<tr><th>Method</th><th>Properties</th><th>Strengths</th><th>Weaknesses</th><th>Other comments</th></tr>
<tr><td rowspan="3">Sliding mode control</td><td rowspan="2">State feedback</td><td>Order of controller can be reduced [43]</td><td>All system states have to be available [1]</td><td rowspan="2">Many successful applications in industry: [29], [37], [43], [44]</td></tr>
<tr><td>Disturbance rejection [43]</td><td>System dynamics need to be well understood [44]</td></tr>
<tr><td rowspan="2">Decoupled design procedure [37]</td><td>Low sensitivity to parameter variations and unmodelled dynamics [37], [43]</td><td>Existence of sliding mode controller by static hyperplane design may not always exist [45]</td></tr>
<tr><td>Easy to implement [43]</td><td>In case of high stiffness, uncontrollable chattering renders system unstable [1]</td><td>VSC – Variable structure control [46]</td></tr>
<tr><td rowspan="2">Backstepping</td><td>State feedback</td><td rowspan="2">Applicable to systems which are not feedback linearizable [34]</td><td rowspan="2">All system states have to be available [47]</td><td rowspan="2">Many applications in adaptive schemes: [34], [47], [48], [49]</td></tr>
<tr><td>Virtual controller uses latter states to stabilize previous one [49]</td></tr>
</table>

It should be clear at this point that designing an AMB controller is no easy task. Some researchers find the idea of an automated approach desirable. This approach consists of combining automated control design methods with system identification to achieve a self-tuning or adaptive system. The benefits would include significantly decreased design effort and development time [1]. Adaptive control systems are fine examples of such an automated approach.

2.2 Adaptive control

2.2.1 Introduction

Every living organism has the fundamental ability to adjust to changes in its environment. Adaptive control borrows this idea from nature to improve the performance of control systems. A way of monitoring system performance and converting it to some sort of performance index is essential. The final step consists of using the performance index to adjust the controller parameters and achieve an adaptive controller [50]. This is the description of an adaptive control system from the year 1958.

Over 50 years later, researchers have still not agreed on a formal definition of adaptive control but the main elements seem to be the same [51]. Astrom and Wittenmark [28] have provided a definition which has become widely accepted: a controller with adjustable parameters and a mechanism for adjusting the parameters.

Dumont and Huzmezan [5], Sastry and Bodson [4] and Tao [52] also tried to define adaptive control. The following important details were gleaned from their attempts:

- An adaptive control system consists of a process, a controller and an adaptive law[6].
- The process to be controlled is called the plant and its parameters can be known or unknown.
- The controller parameters are updated by the adaptive law to improve performance.

What has changed in the last 50 years? Apart from an exponential increase in the computational power of hardware, adaptive control theory has extended to the control of nonlinear systems [8]. This makes sense because the majority of physical systems are nonlinear with unknown or partially known dynamics.

Nonlinear systems with unknown or partially known dynamics, including AMB systems, are termed complex according to the definition of a complex system by Narendra, Feiler and Tian [53]. Complex systems can exhibit time-varying behaviour for a number of different reasons. Input and disturbance signals (including noise) may have different characteristics at different points in time. System parameters may also vary with time due to environmental changes [50].

Many researchers believe that complex systems require deeper understanding to improve their control [4]. In many industrial applications, a careful investigation into the causes of system or environmental changes might not be possible or simply too expensive. Adaptive control is highly suited to such circumstances [28].

2.2.2 Adaptive control design methods

With an increase in understanding of complex systems, new control design methods are developed. This is problematic and beneficial at the same time. The number of available tools to improve system control become large and makes the selection of an appropriate design method difficult [54].

Some design methods, like gain scheduling, aim for satisfactory performance over a wide range of operating conditions [54]. Another method, called model reference adaptive control, tries to ensure that the controlled plant exhibits behaviour close to that of a reference model [4]. Self-tuning regulators use estimates of plant parameters to calculate the controller parameters [8].

The relevant literature discusses many adaptive control design methods yet it seems that they are all based on two basic algorithms: direct and indirect [4]. Tao [52] also found that any adaptive control scheme can be developed using either a direct or indirect adaptive law.

---

[6] An adaptive law is defined as the algorithm used to adjust or tune the parameters of a controller [8].

In a direct adaptive control (DAC) design the parameters of the controller are updated straight from the adaptive law. The adaptive law is based on the output error between the plant output $y(t)$ and the reference model output $y_m(t)$. Model reference adaptive control is an example of a design method using a direct adaptive control algorithm [52]. Figure 2-1 illustrates a DAC system.

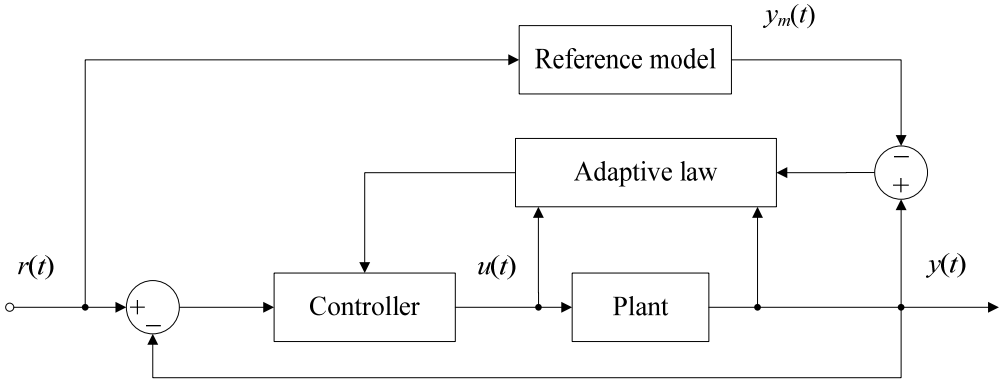

Figure 2-1: Direct adaptive control system

In an indirect adaptive control (IAC) design the controller parameters are calculated from model approximations. These approximations are obtained from a model of the plant using system identification. The adaptive law uses an approximation error representing the mismatch between the plant output $y(t)$ and the online model approximations from the observer [52]. Self-tuning regulators employ an indirect adaptive law [8]. Figure 2-2 illustrates an IAC system.

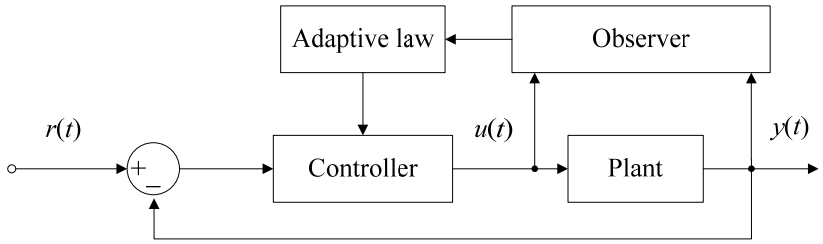

Figure 2-2: Indirect adaptive control system

Two approaches to system identification in adaptive control systems exist. System identification may take place offline at first to obtain an initial model for the online phase. In the online phase, called recursive system identification, the model is updated using previous estimates as well as new data [4]. Offline system identification is not compulsory, but it definitely improves the rate of convergence of the controller parameters online [8].

Since existing AMB models are poor approximations of reality, this study includes a phase dedicated to system identification. The adaptive control design method will accordingly make use of an indirect algorithm as the adaptive law.

It should be clear at this point that an adaptive control design method should include steps to recognize the changes to be adapted to, a means of adjustment and the criteria for adjustment [50]. All of these choices allow for a large number of possible designs and the numerous methods found in relevant literature suggest the same.

Apart from gain scheduling, model reference adaptive control and self-tuning regulators, researchers have also produced the following: adaptive fuzzy control [2], adaptive control using neural networks [8], adaptive backstepping [47], model predictive control [52], adaptive inverse control [55] and robust adaptive control [56].

2.2.3 Adaptive control versus fixed control

Selecting and implementing an adaptive control design method can be extremely difficult. This effort is well justified when comparing the advantages of adaptive control with fixed control. An adaptive controller is designed to adapt online to parametric, structural and environmental uncertainties while still ensuring satisfactory system performance. This is perhaps the most significant difference between adaptive and fixed controllers: a fixed controller is based solely on prior information[7], whereas an adaptive controller is also based on posterior information[8] [5].

A well-designed adaptive controller is also more robust than a fixed controller because it remains stable in the presence of unmodelled dynamics and external disturbances [52]. Fixed models can only exactly represent a fixed number of environments. A control system using fixed models requires a large number of models to ensure a small steady-state error [57].

2.2.4 System identification

A system can be described as an entity in which different variables interact to create observable signals called outputs. External signals called inputs or disturbances affect the system in such a way that their influence can be clearly observed in the outputs. System identification is defined as determining a mathematical model of an unknown target system using data – inputs and outputs – obtained from the system itself [14]. System identification has three main purposes [7]:

- The identified model can be used to predict the target system's behaviour.
- The model might be able to explain relationships between the input-output data of the target system.
- The model can be used to design a controller for the target system.

[7] Prior information is any data or knowledge obtained from the system or model offline [5].
[8] Posterior information is data or knowledge obtained from the system or model online i.e. while in operation [5].

There are two critical steps involved in system identification. The first step is known as structure identification. Prior knowledge about the system under consideration needs to be applied to determine candidates for the best model. The second step is called parameter estimation or identification. At this point, the structure of the model is known and optimization techniques can be applied. The optimization techniques are used to calculate the best set of parameter values which describe the system properly [7].

2.2.5 System identification procedure

A significant amount of time and effort goes into successful system identification, especially when a trial-and-error approach is utilized. To exhaust all the possible solutions and combinations of models and their structures hardly seems like a practical approach. The system identification loop proposed by Ljung [14] will be used to resolve this issue. The loop consists of collecting data, choosing a model set and picking the best model in the set. If the chosen model does not pass validation tests, it is discarded and the steps of the procedure are revised. The system identification loop is illustrated in figure 2-3.

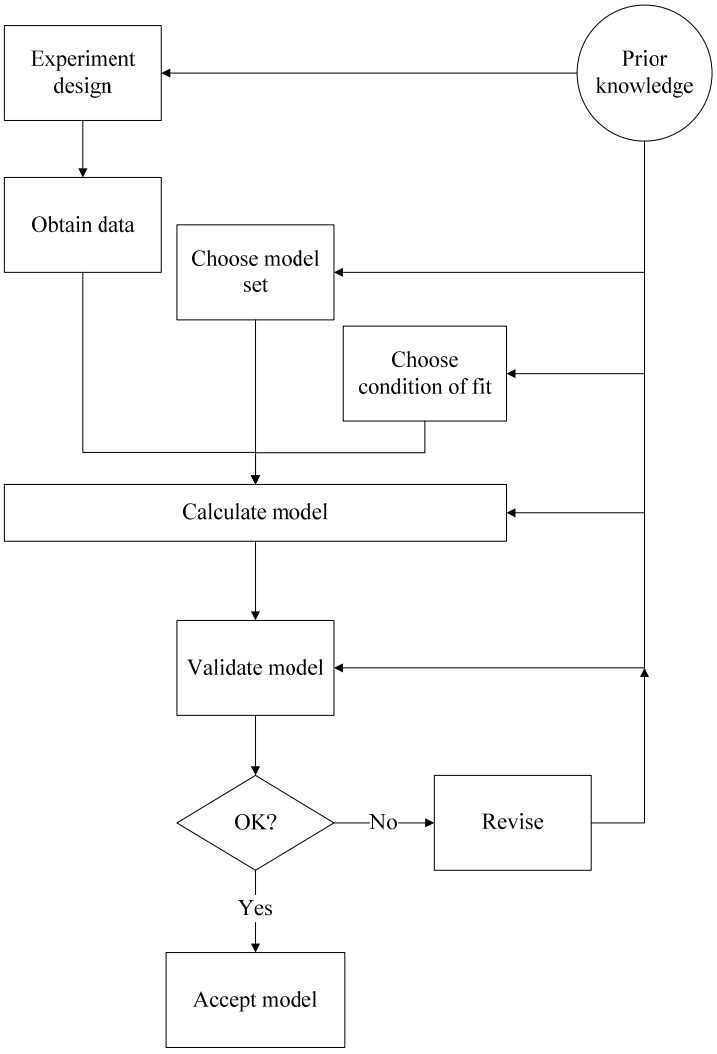

Figure 2-3: System identification loop

The data can be obtained either from the normal operation of the system under consideration or from a specifically designed identification experiment. In the experiment, the user may decide how signals are measured and specify the types of input signals. The motivation for an experiment is to ensure that the resulting data contains the maximum amount of information necessary for successful identification [14].

Choosing the appropriate type of model or model structure is important because trying to identify for example, a nonlinear system using a linear model would yield poor results. Prior information about the specific problem and formal properties of models are used to make this choice. The next step is evaluating the models based on how well they perform when trying to replicate the original data set according to a chosen condition or criterion. A suitable method i.e. independent of model structure will have to be devised [14].

After performing these three steps, the chosen model needs to be validated. Validation is a necessary step because prior to doing so, the chosen model is the one which best fits the data according to a chosen condition. It has not been tested to see whether it is valid for its intended use, how well it relates to unobserved data or if it conforms to prior knowledge. The model will be discarded if it performs poorly but will be accepted and used if it performs in a satisfactory manner. There are various reasons why a model might perform poorly and be discarded [14]:

- The technique used to find the best model according to specified conditions failed.
- The specified conditions were not chosen properly.
- The set of models was inappropriate and did not contain suitable candidates for the system under consideration.
- The data set did not contain enough information to ensure the selection of superior models.

Models which are discarded because of poor performance should not be ignored. Their results can be used in combination with prior information as guidance for selecting new candidate models.

2.2.6 Neural networks as model set

A later section is devoted to more detail on neural networks. This section only aims to explore neural networks as a modelling tool and motivate their selection as the candidate model set for system identification in this study. The two most common AI techniques used for modelling are neural networks and fuzzy logic systems [2].

A fuzzy logic system or fuzzy inference system (FIS) possesses the capability to reason and learn in an environment characterized by ambiguity and imprecision. However, a FIS cannot adapt to changes in its environment without some sort of adaptive mechanism [7].

Neural networks are well known for their ability to adapt to changing environments when used in an online approach [7]. Research clearly reveals that both FISs and NNs are regarded as universal function approximators [2]. The performance of a FIS depends greatly on the construction of the system itself. This implies that the universal function approximation property only guarantees the existence of an optimal fuzzy system – it does not state how such a system is to be found [58].

To find an optimal fuzzy system, an arbitrarily large number of rules could be used initially and then reduced as fine-tuning progresses [58]. The same holds true for a neural network and its number of neurons. When properly trained, a neural network generalizes well to data it has never seen before i.e. it behaves as desired when presented with data different from the training set [8].

Some researchers are convinced that both FISs and NNs are fault-tolerant to some extent. If a rule were to be deleted in a FIS's rule base, it would not significantly impact the system's operation because of its parallel and redundant architecture. However, the system's performance does slowly worsen [7]. If a neural network should become damaged due to a faulty model or external disturbances, the performance of the network will degrade gracefully [11].

The greatest weakness of a fuzzy system can be found in its rule base. The curse of dimensionality can be clearly observed when the dimension of the input space increases. As the number of input variables increases, the number of rules increases exponentially and the fuzzy system can quickly become very complex [59]. The same holds true for a neural network, its input space and its number of neurons [7].

NNs are believed to have the flexibility and robustness to surpass the performance of conventional methods and provide alternative (and often better) ways to solve certain problems [10]. Fuzzy systems are constructed based on the uncertainties in the inputs and outputs of the system and thus are also considered robust systems [60].

It should be clear that both neural networks and fuzzy logic systems have useful advantages when trying to control an AMB system. Even though both structures are applicable to nonlinear control problems, neural networks are selected as the candidate model set. Neural networks have one specific advantage crucial to adaptive control - the ability to adapt to changing environments. Literature abounds with successful applications of fuzzy logic in AMB control but only a few examples of neural networks. Neural network controllers have been used for disturbance rejection and vibration suppression [39] and as robust AMB controllers [61]. Further investigation into the application of neural networks in AMB control is well justified.

2.3 Neural networks

2.3.1 Introduction

Research in artificial neural networks (ANNs) is one of the numerous branches of the field of artificial intelligence. Support vector machines (SVMs) and fuzzy logic systems (FLSs) are also included. An intelligent system is an entity which exhibits one or more cognitive capabilities such as learning, decision-making and classification [8].

ANNs are intelligent systems inspired by the functionality of biological neural networks i.e. the brain. The basic idea of an ANN is to reproduce the processing power as displayed by nervous systems [8]. Note that from this point forward, weights will be used to refer to the values of a network's weights and biases.

The basic components of a neural network are interconnected neuron-like elements. These elements, also called neurons, are multiple-input single-output units where the output is typically a weighted sum of the inputs. By adjusting these weights and updating the architecture, the network is adapted to model a desired function and in this manner the neural network achieves learning. The neurons are grouped in layers and can be either fully or partially connected [62]. Figure 2-4 illustrates a fully connected single-layer neural network.

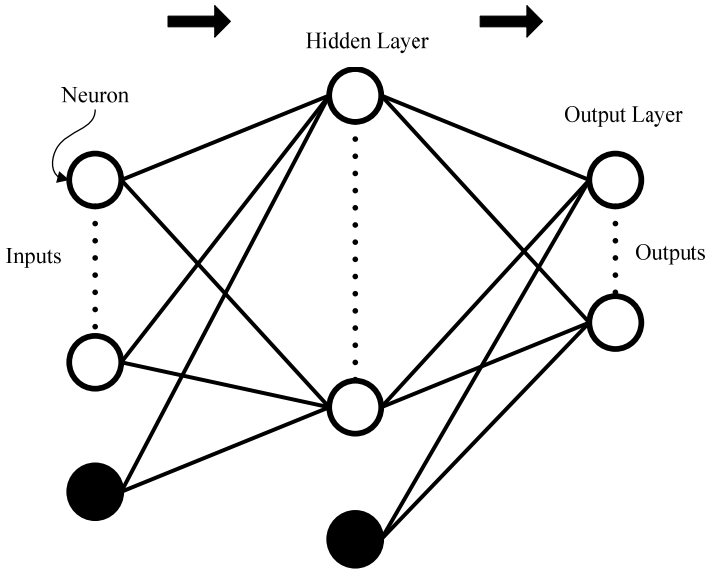

Figure 2-4: Feed-forward single-layer network

Note that the network in figure 2-4 is called a single-layer network because it has only a single layer of hidden neurons. Other naming conventions would call the same network a two-layer network given that it has two layers of weights [63]. The first naming convention will be used throughout this document. The direction of the arrows indicates the flow of information through the network.

Most applications of ANNs are aimed at solving problems in function approximation, prediction, optimization, pattern classification, associative memory as well as control. Conventional methods perform relatively well when used to solve these problems but they do so in properly constrained environments and performance is far from satisfactory without these constraints. ANNs have the flexibility and robustness to surpass the performance of conventional methods and provide alternative (and often better) ways to solve the above-mentioned problems [10].

Several steps are involved in the design process of a neural network. The following choices are considered the most important in this process:

- The neural network architecture as well as the type of activation function used by the neurons.
- The training algorithm which usually depends on the chosen architecture.
- The parameters specific to the chosen architecture and training algorithm.
- The training and testing strategy.
- The error or performance function.

Making the appropriate choices during the design process determines the success of solving the given problem. The most relevant information to consider during each of these choices will now be presented.

2.3.2 Neural network architectures

A neural network's architecture refers to the connection pattern used between the network's layers and their neurons. Research clearly shows that architectures can be divided into two broad classes: feed-forward and recurrent architectures. In a feed-forward neural network (FNN) the information flows in one direction only i.e. from the inputs through the hidden layer(s) to the outputs. A recurrent neural network (RNN) has feedback connections and the flow of information occurs in more than one direction [8].

2.3.2.1 Feed-forward neural networks

In general, FNNs are static because they have no feedback elements and contain no delays – properties which cause their response to an input to be independent of any previous network state. The most common FNN is the multi-layer perceptron (MLP) which has multiple hidden layers each with their own number of neurons. Referring to figure 2-4, note that the layers between the inputs and the outputs are called hidden layers. FNNs also include single-layer perceptron networks, radial basis function networks (RBFs), self-organizing maps (SOMs) and wavelet neural networks (WNNs). A WNN allows for fast convergence rates during training of the network because of its properties of finite support and self-similarity, but it can only be used for static problems due to its feed-forward network architecture [64].

RBFs and SOMs also have feed-forward architectures and thus also are not able to model internal system dynamics [65]. Dynamic feed-forward networks do exist but the dynamics are only present at the input of the network due to the inclusion of a tapped delay line. This type of network is called a focused time-delay neural network. When the tapped delay lines are distributed throughout the network, it is accordingly called a distributed time-delay network. Note that information still only flows in one direction i.e. forward [9].

2.3.2.2 Recurrent neural networks

The neurons in a recurrent neural network (RNN) form a directed cycle i.e. the flow of information is in more than one direction. This is achieved by introducing feedback into a FNN in the form of connections between different layers as well as connections among neurons in the same layer. RNNs are able to exhibit dynamic behaviour due to this introduction of feedback into the network as shown in figure 2-5 [8].

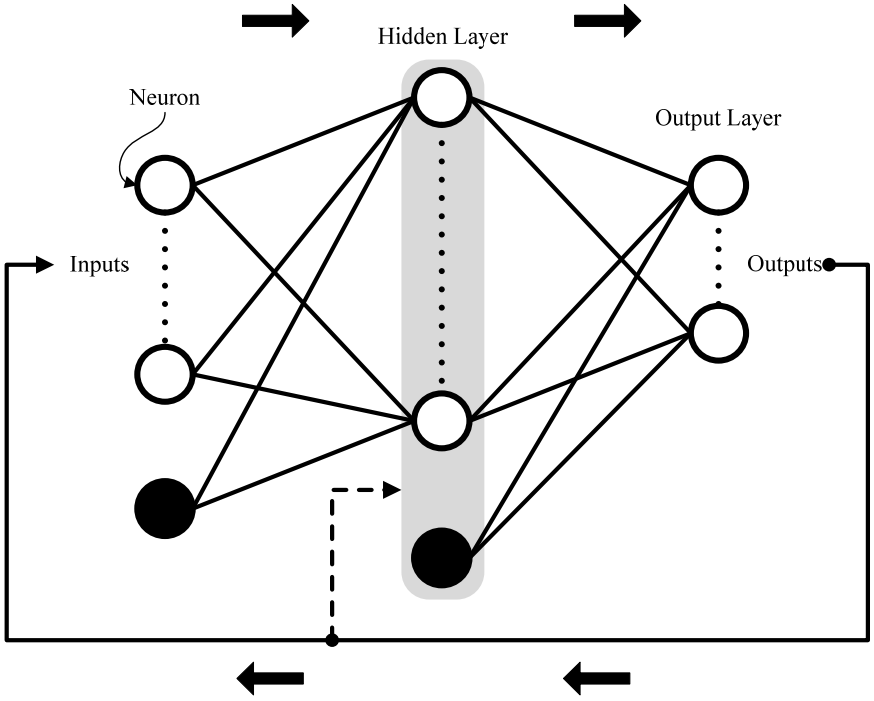

Figure 2-5: Recurrent single-layer network

Elman and Jordan networks surfaced in the late 1980's and are the first examples of RNNs. They also characterize the two most basic ways to introduce feedback into a network. In an Elman network the feedback is introduced by connecting a hidden layer to the inputs and consequently, this network focuses on the sequence of input values. The emphasis in a Jordan network falls on the sequence of output values because the feedback is introduced by connecting the outputs to the inputs [66].

Today's RNNs include but are not limited to competitive networks, Hopfield networks, Boltzmann machines, layer-recurrent networks (LRNs) as well as a number of FNNs which have been modified to include some form of feedback. An LRN is a more complex version of an Elman network because each layer (except the last layer) has a single-delay feedback loop around it [9].

2.3.3 Activation functions

The purpose of a neuron's activation function is to process incoming data and send the result to the next layer. These functions are also referred to as transfer functions and can be found in the hidden and output layers of a neural network. Popular candidates for hidden layer activation functions in multilayer networks are the sigmoidal functions. These functions reduce an infinite input range into a finite output range and are consequently called squashing functions. The choice of the output layer's activation function is relatively simple. It is recommended that a linear type be used for regression problems and sigmoidal types for classification tasks [62].

2.3.4 Training neural networks

Training a neural network is the process in which the network's weights are adjusted using a chosen training algorithm. The algorithm makes use of certain learning rules to adjust the weights according to the training data provided. The training data usually consists of example patterns representing the specific task the network is being trained to perform. The weights can be updated in an iterative manner to improve the performance of the neural network i.e. the network can be trained again and again. The ability to learn without a specified set of rules makes neural networks superior to conventional expert systems [10].

Training a neural network successfully involves understanding the environment in which it is to operate. A model of this environment can then be used to discover what information is available to the neural network. This model is called a learning paradigm. The different learning paradigms are as follow [7], [10]:

- Supervised learning – the network is supplied with example patterns of correct behaviour in the form of input-output pairs and learns to reproduce the relationship present in the data. An explicit teacher is present.
- Unsupervised learning – the desired outputs are not available during training and the network learns through self-organizing behaviour using competitive learning rules or a pre-defined cost function. The network learns to extract features or patterns from input data with no feedback from the environment about desired outputs.
- Reinforcement learning – a computational agent interacts with its environment and learns correct actions from a reinforcement signal grading its performance. The aim is to discover a policy for selecting actions that minimize an estimated cost function.

There are three major practical issues when learning from example patterns: capacity, sample complexity and computational complexity. First of all, the issue of capacity must be addressed. It concerns determining how many examples can be stored and what exactly a specific network is capable of – what functions it can approximate or decision boundaries it can form [10].

The second issue is sample complexity and considers the number of examples necessary to obtain a network which generalizes well. Computational complexity is the third issue and refers to the time taken to properly train the network [10]. Addressing these issues can be traced back directly to the choices in the neural network design process discussed earlier.

2.3.4.1 Training algorithms

A training algorithm or learning rule is a procedure used to adjust the weights of a network i.e. train the network. The purpose of training is to improve the performance of the network as measured by some performance or error function. This procedure can be applied to one pattern at a time or a whole batch of patterns at once. These two training styles are called incremental and batch training respectively [9].

A multitude of training algorithms abound and an obvious problem is choosing the appropriate one for the task at hand. This multitude of training algorithms is due to the fact that original algorithms have been modified, extended or combined into various different forms to achieve whatever tasks they were intended for. Certain algorithms are appropriate only for certain network types and the choice of algorithm becomes simpler once a network architecture has been chosen [9].

Table 2-2 provides a summary of network architectures and which training algorithms or learning rules are commonly used to train each of them. The summary also shows which purpose each network was initially intended for. As with the multitude of algorithms, network architectures have also been modified over time and resulted in various different forms. For this reason, the following summary has been restricted to the original algorithms and architectures.

Note that BP in the summary indicates the backpropagation algorithm – an extremely popular training algorithm. The second stage of the BP algorithm in which the weights are adjusted can be accomplished using a variety of different optimization schemes including conjugate gradient methods as well as evolutionary programming. BP in the summary refers to all of these variations as well as standard gradient descent. Hagan, Demuth and Beale [9] provided the information for this summary.

Table 2-2: Summary of neural networks

Category	Name		Algorithm	Purpose	Behaviour	Paradigm
Feed-forward neural networks	Linear	Perceptron	Perceptron rule	Pattern recognition	Static	Supervised
		Linear layer	Widrow-Hoff rule	Pattern recognition and function approximation	Static	Supervised
		Adaline			Static	Supervised
	Multi-layer perceptron		BP	Pattern recognition and function approximation	Static	Supervised
	Wavelet			Function approximation	Static	Supervised
	Time delay	Linear with tapped delay	Widrow-Hoff rule	Digital signal processing	Dynamic	Supervised
		Focused time-delay	BP	Time-series prediction	Dynamic	Supervised
		Distributed time-delay	BP	Pattern recognition	Dynamic	Supervised
Radial basis function neural networks	Generalized regression		Orthogonal least squares / kernel methods	Function approximation	Static	Supervised
	Probabilistic			Pattern recognition	Static	Supervised
	Simple radial basis			Pattern recognition and function approximation	Static	Supervised
Associative neural networks	Hopfield		Hebb rule	Pattern recognition and optimization	Dynamic	Unsupervised
	Simple recall		Outstar learning	Pattern recall	Static	Unsupervised
	Simple associative		Hebbian learning	Pattern association	Static	Unsupervised
Recurrent neural networks	Layered-recurrent		BP	Filtering and modelling applications	Dynamic	Supervised
	Nonlinear autoregressive with exogenous input			Modelling of nonlinear dynamic systems	Dynamic	Supervised
	Real-time recurrent		BP	Real-time applications	Dynamic	Supervised
	Elman		BP	Recognition and generation of spatial and temporal Patterns	Dynamic	Supervised
Competitive neural networks	Kohonen self-organizing map		Kohonen (instar) learning	Pattern recognition	Static	Unsupervised
	Competitive layer		Kohonen learning	Pattern recognition	Dynamic	Unsupervised
	Hamming				Dynamic	Unsupervised
	Grossberg		Continuous-time instar learning	Clustering	Dynamic	Unsupervised
	Adaptive resonance theory		Instar / outstar learning	Pattern recognition	Dynamic	Unsupervised
	Learning vector quantization		Kohonen (instar) learning	Pattern recognition	Static	Hybrid

2.3.5 Training and testing strategy

Research reveals that the most common training and testing strategy, early stopping, consists of dividing the available data into three separate sets. These sets are called the training, validation and test data sets. The training set is used to train the network while the validation set is used to ensure that the network does not overfit the data. The test set is used to test how well the network generalizes to data it has never seen before [9].

Overfitting occurs when the error on the training data becomes very small but the error on the validation data (new data) is large. The network is very accurate when tested using only training data and cannot generalize when presented with new samples. There are methods to avoid overfitting and thus improve generalization. These methods include using adequately sized networks, using more training data as well as techniques called early stopping and regularization [9].

2.3.6 Parameter selection

When the network architecture and training algorithm have been chosen, appropriate values for their parameters have to be determined. The architecture parameters include the number of layers, the number of neurons in each layer, the type of transfer function in each layer as well as the weights and their initial values. The parameters associated with the training algorithm depend on the algorithm used for training and might include a learning rate or the number of epochs (iterations) used to train the network [67].

2.3.7 Error function

Calculating the error and consequently the error derivatives of a neural network are perhaps the most important part in training. These values are used to adjust the weights in an effort to improve network performance. Generally, if the network is capable and algorithm parameters are chosen correctly, this error will eventually become zero. In pattern recognition problems, this is desirable though function approximation applications require better generalization [9]. It should be clear that the error or performance function is also a way to measure the performance of a network during and after training. There are many candidates for the error function. Care should be taken when choosing an error function and implementing it in the training algorithm.

2.3.8 Neural network control

The field of neural networks is quite extensive. The previous sections only serve as an indication of how many network architectures and training algorithms exist. The number of possible ways to use neural networks in control systems is accordingly, very large. This section provides an overview in the two most fundamental ways a neural network can be used in control systems – providing assistance for an existing controller or being a controller [68].

Assistance is most commonly provided in the form of system identification. The neural network is used to approximate an unknown function, usually the plant to be controlled, or an unknown parameter [69]. Figure 2-6 illustrates function approximation using a neural network.

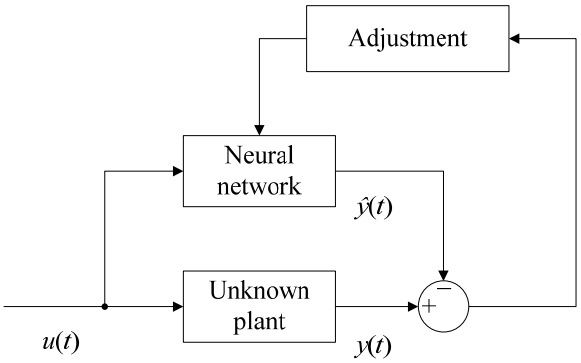

Figure 2-6: Neural network as function approximator

The neural network estimates the plant output, $y(t)$, as $\hat{y}(t)$. The error between the actual output and the estimated output is used to adjust or train the weights of the neural network. When a neural network is used in this manner it is called an observer. The observer weights can initially be adjusted offline to increase the rate of convergence if the desired output data is available [8].

The neural network observer can be used with other neural networks to synthesize a controller. Depending on the type of plant to be controlled and signals available for feedback, various design methods are possible. These design methods include: backstepping [8], sliding-mode [70], NARMA-L2 or feedback linearization [71], multiple model approach [53], adaptive inverse control, internal model control, model predictive control, model reference adaptive control and adaptive critic [55]. Figure 2-7 shows the various ways in which a neural network can be used for assistance in a control system.

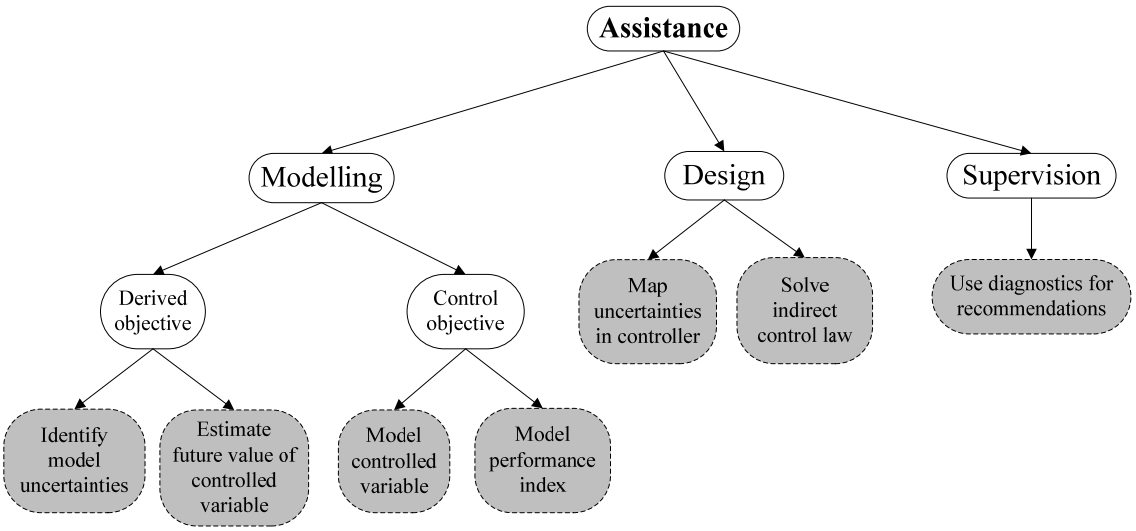

Figure 2-7: Neural networks for assistance

In figure 2-7, a grey block indicates a method and the blocks preceding a grey block, are overlapping features between the different methods. The same applies to figure 2-8 showing the various ways a neural network can be used to control a system.

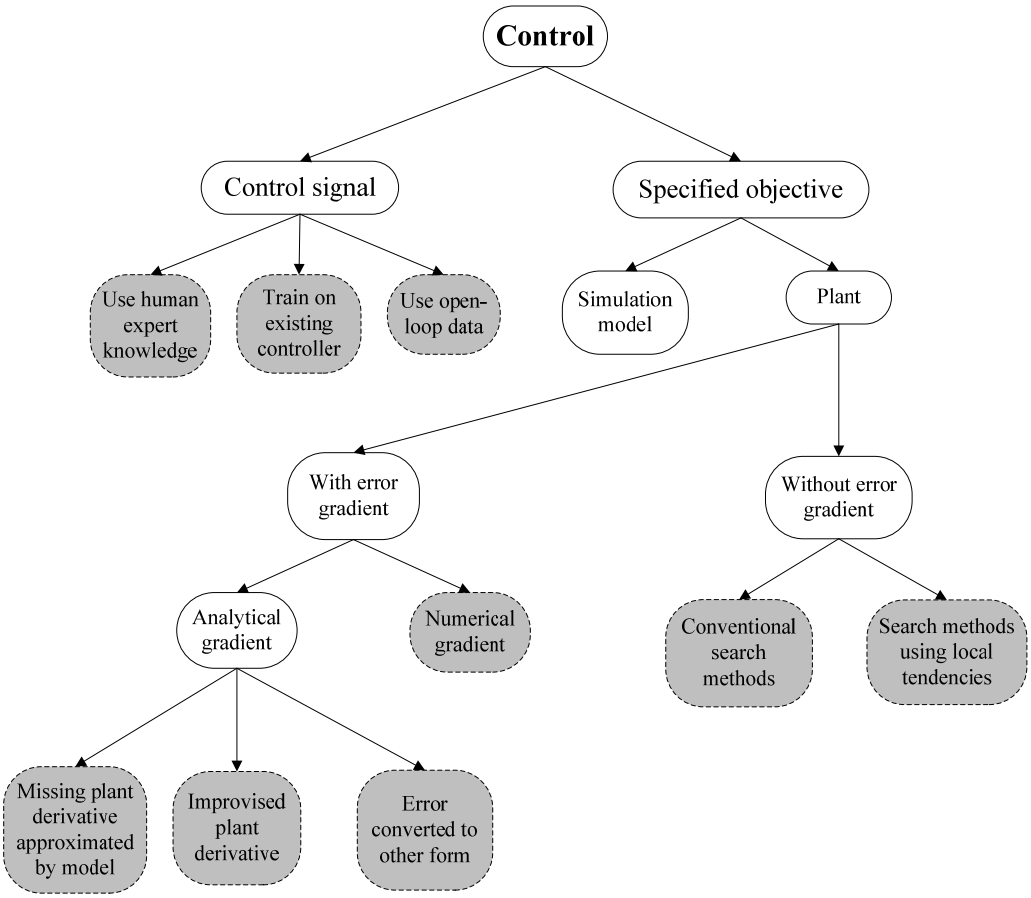

Figure 2-8: Neural networks for control

The control objective needs to be achieved while guaranteeing the overall system's stability. Some of the mechanisms which cause a system to become unstable are parameter drift, controllability and transient behaviour. Large initial parameter approximations cause the system to exhibit poor transient behaviour resulting in modelling errors. Parameter drift is caused by these modelling errors because the neural network does not exactly match the function it is trying to approximate [69].

2.4 Critical overview and conclusions

The research problem clearly indicated that the most important focus areas for this study are AMB systems, adaptive control and neural networks. These focus areas were chosen as the relevant fields of interest to be covered by the literature study. The challenges and considerations associated with the modelling and control of an AMB flywheel system were investigated in this chapter. Adaptive control was also discussed and compared with fixed control methods.

The chapter continued with the selection of an indirect adaptive control law for the adaptive controller. Special attention was given to the system identification loop since it forms an integral part of indirect adaptive control. The selection of neural networks as candidate models for the plant of the AMB flywheel section was also explored. The final section of this chapter provided an overview of neural networks in general. The detailed study on different architectures enables the selection of suitable network architectures for system identification. Different neural network topics such as training, testing and control were also discussed.

Any important literature not found in this chapter will be presented in following chapters as necessary. The following chapter explains how the most suitable neural network architecture to model the plant of the AMB flywheel system was chosen. An offline system identification experiment enables a comparison of different architectures as well as the final architecture selection.

Chapter 3: Architecture selection

This chapter covers the details of an offline system identification experiment comparing different neural network architectures. The estimation error and accuracy of each network are used to perform the comparison. The aim is to obtain the most suitable architecture to model the plant section of the AMB flywheel system. The neural model is then used in the original system as the plant and the frequency response of the closed-loop system is determined. The frequency response of the original system is compared to the frequency response of this new system to validate the neural model.

3.1 Introduction

The objective of the offline system identification experiment is to obtain a black-box model[9] of the plant section[10] of the AMB flywheel system. A simplified discrete-time block-diagram of the AMB system is illustrated in figure 3-1. A neural network will be used to obtain the model of the plant. This new model should be able to replace the actual plant in the original control system without causing considerable changes in the behaviour of the overall system.

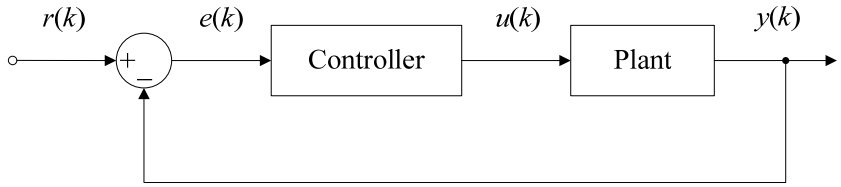

Figure 3-1: Simplified discrete-time block diagram of AMB system

The AMB system is considered in discrete-time because samples of data are used to perform the offline system identification. The plant has all the characteristics of an AMB system described in earlier sections: dynamic, nonlinear, multivariable and unstable. The neural network architectures with the abilities to exhibit the same characteristics must be identified.

3.2 Architecture features comparison

The *Neural Network Toolbox™* found in **MATLAB®** will be used for this phase of the study because it allows for quick implementation and easy comparison between the different architectures. This software package contains many of the architectures listed in table 2-2.

---

[9] Little or no information is known about a black-box system and its internal process – it is viewed in terms of its inputs, outputs and transfer characteristics [2].

[10] The plant section of the AMB flywheel system consists of the power amplifiers, two radial AMBs, flywheel and position sensors [13].

The number of possible architectures for system identification is significantly reduced by noting that only certain architectures allow for dynamic behaviour. Furthermore, only certain architectures are recommended for function approximation and modelling. Table 2-2 is repeated as table 3-1 with the candidate architectures highlighted in grey.

Table 3-1: Candidate architectures

Category	Name		Algorithm	Purpose	Behaviour	Paradigm
Feed-forward neural networks	~~Linear~~	~~Perceptron~~	~~Perceptron rule~~	~~Pattern recognition~~	~~Static~~	~~Supervised~~
		~~Linear layer~~	~~Widrow-Hoff rule~~	~~Pattern recognition and function approximation~~	~~Static~~	~~Supervised~~
		~~Adaline~~			~~Static~~	~~Supervised~~
	~~Multi-layer perceptron~~		~~BP~~	~~Pattern recognition and function approximation~~	~~Static~~	~~Supervised~~
	~~Wavelet~~			~~Function approximation~~	~~Static~~	~~Supervised~~
	Time delay	~~Linear with tapped delay~~	~~Widrow-Hoff rule~~	~~Digital signal processing~~	~~Dynamic~~	~~Supervised~~
		Focused time-delay (FTD)	BP	Time-series prediction	Dynamic	Supervised
		~~Distributed time-delay~~	~~BP~~	~~Pattern recognition~~	~~Dynamic~~	~~Supervised~~
Radial basis function neural networks	~~Generalized regression~~		~~Orthogonal least squares / kernel methods~~	~~Function approximation~~	~~Static~~	~~Supervised~~
	~~Probabilistic~~			~~Pattern recognition~~	~~Static~~	~~Supervised~~
	~~Simple radial basis~~			~~Pattern recognition and function approximation~~	~~Static~~	~~Supervised~~
Associative neural networks	~~Hopfield~~		~~Hebb rule~~	~~Pattern recognition and optimization~~	~~Dynamic~~	~~Unsupervised~~
	~~Simple recall~~		~~Outstar learning~~	~~Pattern recall~~	~~Static~~	~~Unsupervised~~
	~~Simple associative~~		~~Hebbian learning~~	~~Pattern association~~	~~Static~~	~~Unsupervised~~
Recurrent neural networks	Layered-recurrent (LRN)		BP	Filtering and modelling applications	Dynamic	Supervised
	Nonlinear autoregressive with exogenous input (NARX)			Modelling of nonlinear dynamic systems	Dynamic	Supervised
	~~Real-time recurrent~~		~~BP~~	~~Real-time applications~~	~~Dynamic~~	~~Supervised~~
	~~Elman~~		~~BP~~	~~Recognition and generation of spatial and temporal Patterns~~	~~Dynamic~~	~~Supervised~~
Competitive neural networks	~~Kohonen self-organizing map~~		~~Kohonen (instar) learning~~	~~Pattern recognition~~	~~Static~~	~~Unsupervised~~
	~~Competitive layer~~		~~Kohonen learning~~	~~Pattern recognition~~	~~Dynamic~~	~~Unsupervised~~
	~~Hamming~~				~~Dynamic~~	~~Unsupervised~~
	~~Grossberg~~		~~Continuous-time instar learning~~	~~Clustering~~	~~Dynamic~~	~~Unsupervised~~
	~~Adaptive resonance theory~~		~~Instar / outstar learning~~	~~Pattern recognition~~	~~Dynamic~~	~~Unsupervised~~
	~~Learning vector quantization~~		~~Kohonen (instar) learning~~	~~Pattern recognition~~	~~Static~~	~~Hybrid~~

Any architecture in table 2-2 which does not allow for system identification of a nonlinear dynamic system has been eliminated as shown in table 3-1. Selection of the most suitable architecture has now been reduced to a performance comparison between the FTD, LRN and NARX architectures.

3.3 System identification

Offline system identification is used to compare the performance of the three identified architectures when used as modelling tools. The steps outlined in the system identification loop as applied to the modelling of the AMB flywheel system will now be explained.

3.3.1 Prior knowledge

Prior knowledge about the general characteristics of AMB systems already helped to narrow down the number of candidate architectures to only three types. Other parts of significant information known before performing system identification are used in the following sections to make important choices.

3.3.2 Experiment design

The plant of the AMB flywheel system is inherently unstable. This requires the input-output data needed for system identification to be obtained from the system while in closed-loop operation. A technique called direct closed-loop identification will be used because it is regarded as "...the natural approach to closed-loop data analysis." Direct identification is especially useful for this experiment because it has the following features [14]:

- No knowledge about the feedback is required.
- The complexity of the controller is not important.
- Consistent results and optimal accuracy is guaranteed if the model represents the true system.
- As long as the closed loop system and the observer remain stable, an unstable plant poses no problem.

The **SIMULINK**® model of the AMB flywheel system provided by McTronX will be used to obtain the data necessary for system identification. The model encompasses nonlinear effects like saturation and hysteresis as well as modelling eddy currents. The simulation results closely resemble experimental results and the model accurately reproduces the air gap reluctances in the AMB [13].

Two opposing actuators each consisting of a radial AMB and a current-controlled PA are used in the simulation model as the control configuration. A single PD controller is used per AMB axis resulting in four PD controllers since there are four AMB axis directions. Figure 3-2 shows the block diagram of a control loop containing the controller, actuator and rotor for a single AMB axis [13].

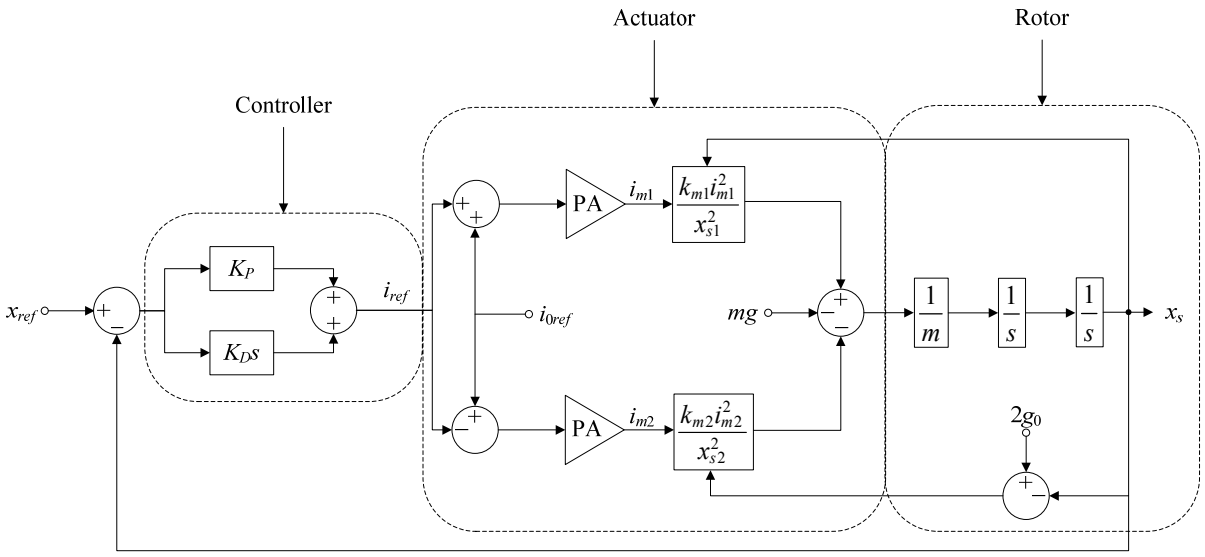

Figure 3-2: Block diagram of single control loop

In figure 3-2, the reference position of the rotor is indicated by x_{ref} and the actual rotor position by x_s. The PD controller constants, K_P and K_D, are used to produce the current reference signal i_{ref}. The bottom PA receives the bias current i_{0ref} minus the current reference signal whereas the top PA uses the bias current plus the current reference signal. The PA outputs are i_{m1} and i_{m2} respectively. Electromagnetic constants are represented by k_{m1} and k_{m2} and the nominal air gap between the AMB and the rotor by g_0. The variables m and g are the rotor mass and gravitational constant respectively and give rise to the gravitational force exerted on the rotor as $F = mg$ [13].

The dependency of the magnetic force in the AMB on the size of the air gap between the electromagnet and the rotor as well as on the current in the coil is given in (3.1). This nonlinear magnetic force (f_1) is inversely proportional to the square of the size of the air gap (x_{s1}) as well as proportional to the square of the current (i_{m1}) in the coil as given in (3.1). The similarity between (2.1) and (3.1) is evident.

$$f_1 = \frac{k_{m1} i_{m1}^2}{x_{s1}^2} \tag{3.1}$$

Equation (3.1) can be seen in the actuator of figure 3-2 right after the top PA. This equation is known as the characteristic equation of the actuator which converts the PA current to a magnetic force. The top and bottom AMB forces are subtracted from each other because they oppose each other. This resulting force is divided by the rotor mass m to obtain the rotor acceleration which is used to determine the rotor position x_s [13].

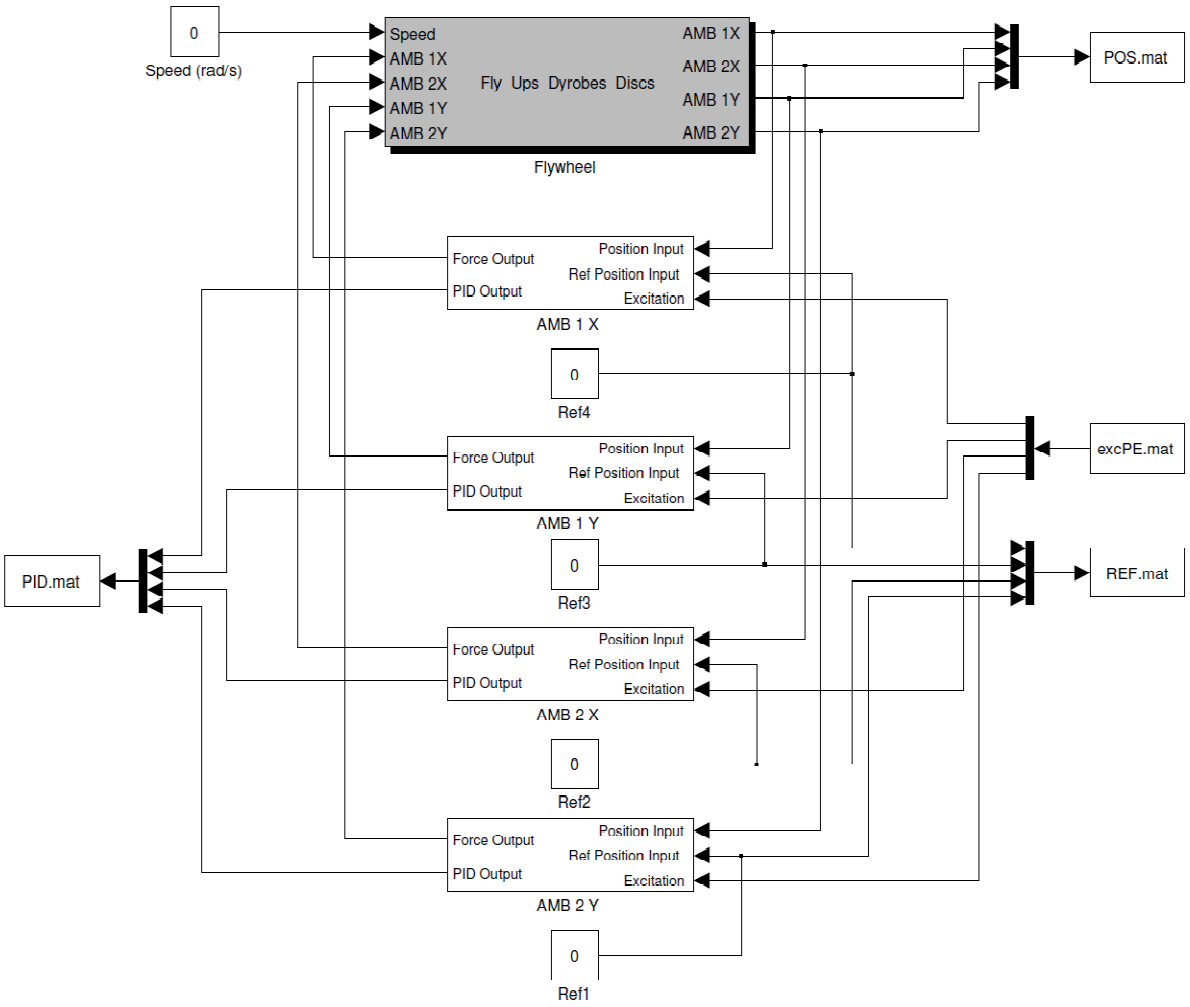

Figure 3-3: Given SIMULINK® control system model

Figure 3-3 shows the given simulation model of the decentralized PD control system in **SIMULINK®**. Notice that the given model has already been modified to allow for the injection of an excitation signal which will be discussed shortly. Four PD control loops for each AMB axis direction and the flywheel model (grey block) can be seen. The flywheel model allows for rotor unbalance to be excluded or included during a simulation. The simulation model also allows for the speed in rad/s and the type of reference signal (step, chirp, multi-sine etc.) to be specified.

3.3.3 Obtaining data

The input-output data for system identification is obtained by introducing a suitable excitation signal[11] into the system and measuring the plant's inputs and outputs. As shown in figure 3-4, the excitation signal can be injected at point 1, 2, or 3 in the block diagram of a control system [72]. The excitation signal will be injected at point 2 since it is closest to the component of interest – the plant [73].

[11] The aim of an excitation signal is to excite the system under consideration and force it to reveal its characteristics [14].

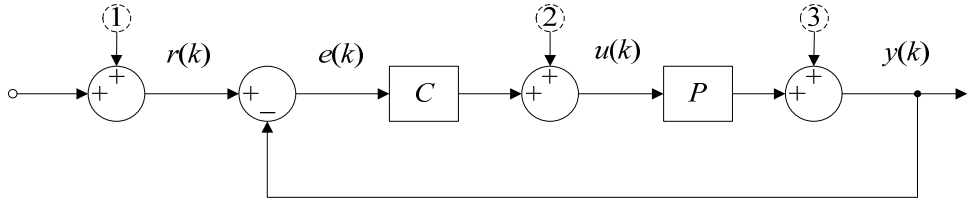

Figure 3-4: Excitation points

The signals $u(k)$ and $y(k)$ will be measured to obtain the input-output data necessary for system identification. The type of excitation signal injected also determines how informative the data from the experiment will be. An excitation signal that generates input-output data rich in information about the system's behaviour is called a persistently exciting signal [28]. For the majority of system identification problems, the condition of persistent excitation is compulsory [14].

The following signals are commonly selected as excitation signals: Gaussian white noise, random binary signals, pseudo-random binary signals, chirp signals and multi-sines. A multi-sine signal is a sum of sinusoids and each sinusoid differs in frequency and phase. It is considered a natural choice of excitation. This makes sense considering that Ljung [14] describes a persistently exciting signal as one which "...contains sufficiently many distinct frequencies." Identification of higher order models would therefore require excitation signals with more frequencies [14].

Following recommendations from Ljung [14] and van Vuuren [73], a multi-sine excitation signal will be injected at point 2 in figure 3-4. The AMB flywheel system currently operates at a maximum speed of 8 000 r/min which roughly translates to 133 Hz. Incorporating more prior knowledge about the system, the multi-sine excitation signal will contain frequencies up to a maximum of 133 Hz.

3.3.4 Choosing a model set

This step in the system identification loop consists of specifying the collection of candidate models to be used in the search for an optimal solution. Though three architectures (FTD, LRN and NARX) have been selected, the model set is not yet completely specified. Choices regarding the number of hidden layers, number of neurons in each layer and the type of activation function still have to be made. Other important considerations include how the weights will be initialized, the learning rate and the time the network takes to converge to the smallest error possible [62].

Each of the architectures will now be presented in a little more detail. The FTD architecture provided by the *Neural Network Toolbox*™ is recommended for time-series prediction. It consists of a static FNN with a tapped delay line at the input layer. The FTD architecture shown in figure 3-5 is considered the simplest dynamic network because dynamics are only present at the input, $p(t)$ [74].

The delay is represented by **TDL** with **b1** and **b2** referring to the biases of each layer. The input layer weights and the output layer weights are referred to by **IW** and **LW** respectively. The activation function for each layer is represented by f_i with i = 1,2. The output from layer 1 to layer 2 is represented by $a_1(t)$ and the final output by $a_2(t)$.

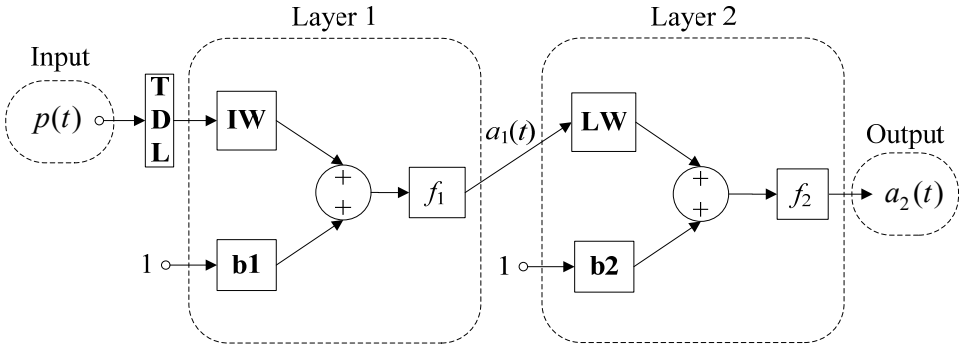

Figure 3-5: FTD architecture

The NARX network architecture is commonly used for time-series prediction, nonlinear filtering and most significantly, modelling of nonlinear dynamic systems. A feedback connection from the output to the input makes the network recurrent. Figure 3-6 illustrates the NARX architecture [74].

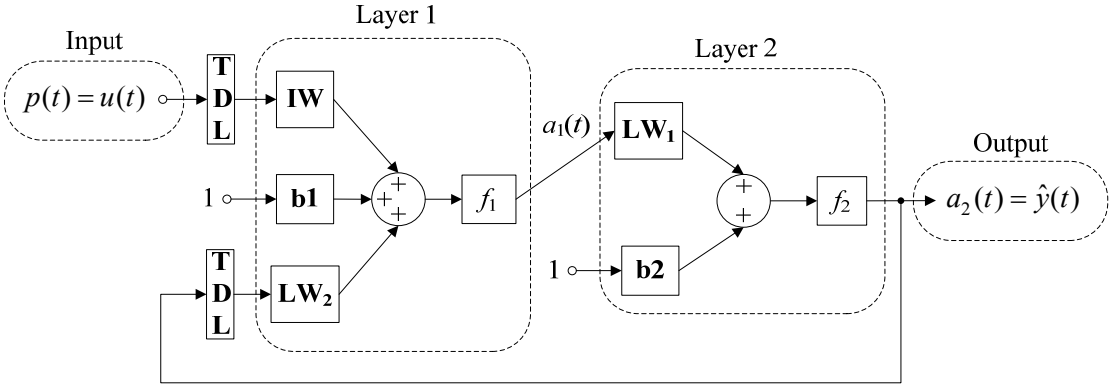

Figure 3-6: NARX architecture

The difference equation in (3.2) describes the NARX architecture [8].

$$y(k) = f(y(k-1), y(k-2), ..., y(k-n), u(k-1), u(k-2), ..., u(k-m)) + d(k) \tag{3.2}$$

Equation (3.2) clearly shows that the next value of the output $y(k)$ depends on the previous values of the output as well as previous values of the independent input signal $u(k)$. The network is used to approximate the unknown nonlinear function $f(\cdot)$ and calculate the estimate $\hat{y}(k)$ of the output $y(k)$. The disturbance $d(k)$ is an unknown vector acting on the system at time instant k [8].

The LRN architecture is a more complex version of the Elman architecture and has been applied successfully to filtering and modelling applications. This architecture has a single-delay feedback connection around each layer except the output layer. Figure 3-7 illustrates the LRN architecture [74].

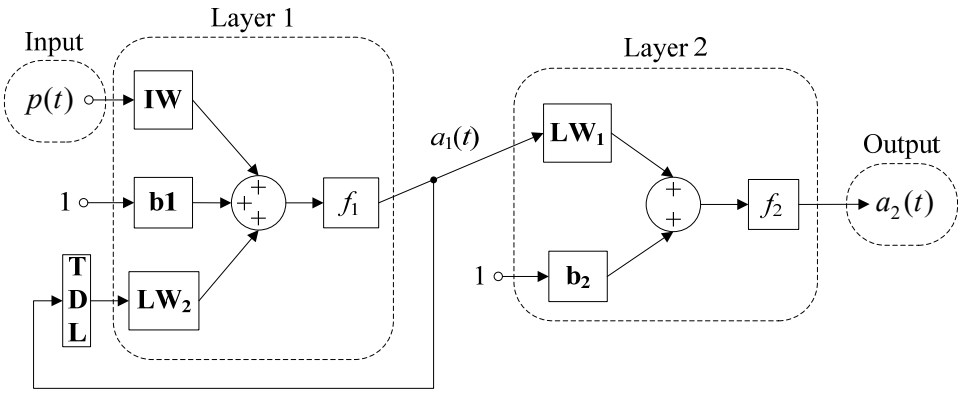

Figure 3-7: LRN architecture

Armed with a little more knowledge about the candidate architectures, the selection of architecture parameters becomes a little easier. Most significantly are the number of neurons in each layer and the number of layers. While it is true that using more neurons requires more computation, it also enables the network to approximate more complex functions. Using more layers provides the network with the ability to approximate complex functions more efficiently. The key lies in choosing the number of layers and the number of neurons which will produce a balanced neural network – not too simple and not too complex. Networks which are too simple will underfit data and networks which are too complex will overfit data [9].

Keeping to the principle of trying simpler designs first, a constructive approach will be used to determine the optimal number of layers and their neurons. The number of hidden layers and neurons will each start at a small number. The network is trained until the error converges and then the number of neurons is increased by a specified integer. The number of layers will be increased in the same way but only one layer at a time. This process continues until the performance of the network stops improving on both training and validation data. The number of layers and the number of neurons which resulted in the smallest training and validation error are then considered to be optimal [75].

The approach described above is completely exhaustive – it tries every possible combination of number of neurons and number of layers. It can be modified and made more effective by searching a range of possible network sizes, called a window, and focussing only on the best networks. Consider figure 3-8 describing a window approach.

Each bubble is a network with the number of neurons written on it. Start with the top 8 networks, train them and determine each network's performance. Say, for example, the 3 networks in grey in the first row performed the best. This indicates that the optimal number of neurons lies somewhere between 20 and 30 neurons. The initial range is expanded using a finer window as indicated by the second row.

The second row of networks is trained and each network's performance determined. In this example, the networks with 24 and 26 neurons in the second row performed significantly better than the rest. The range of the second row is expanded resulting in the third row of networks. Notice that no more integers exist between the numbers of neurons on each bubble in the third row. The third row networks are trained and in this example, the network with 25 neurons performed better than the other two networks. It can be concluded that 25 neurons is the optimal number of neurons.

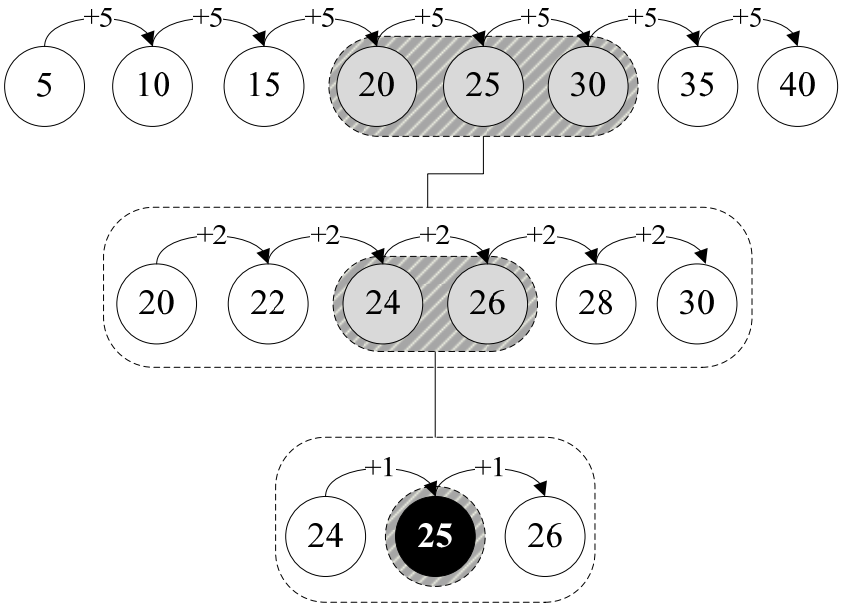

Figure 3-8: Window approach

If a larger range of neurons is used at the start, this window approach will take longer but the results might also be different. The time it takes to determine the optimal network is also affected by the increase in neurons used between one network and the next. If the increase was small and the range large, many different networks would be trained and the approach reverts back to being exhaustive. The initial range should be as large as hardware limitations allow to ensure the best results.

This window approach allows for an effective comparison between the different candidate architectures because only the best networks from each search will be compared. A way to determine the optimal network size has now been improvised, but the choice of activation function still remains.

The choices of network initialization and learning rate will be deferred to the model calculation section because they are usually specific to the chosen training algorithm. Many training algorithms require the derivative of the error function to be available. This derivative is only possible if the activation function used by the neural network is differentiable. Only differentiable activation functions will be considered for this experiment [74].

Sigmoid activation functions are differentiable and are recommended in function approximation problems [9]. The *Neural Network Toolbox*™ provides the following two types of sigmoid functions: logistical sigmoid (logsig) and hyperbolic tangent sigmoid (tansig). These activation functions are illustrated in figure 3-9. The tansig activation function squashes the input values to the range [-1, 1] whereas the logsig activation function squashes it to [0, 1].

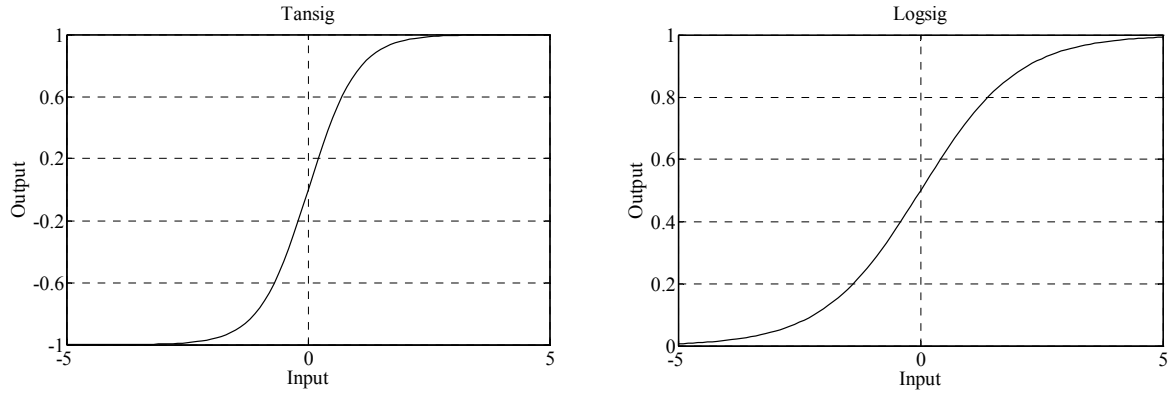

Figure 3-9: Logsig and tansig activation functions

The tansig activation function will be used for the different network designs because the data used during training has been normalized to the range [-1, 1] by *Neural Network Toolbox*™ data pre-processing functions. The purpose of this experiment is function approximation implying that a pure linear activation function should be used for the output layer [9]. This ensures that the outputs of the network can take on any value and will not be limited to only a small range of values. When neural network training is under consideration, normalization of a data vector is defined as rescaling by the minimum and range of the vector.

Normalization is used to ensure that all the values lie within a specified range such as [-1, 1]. Since some data vectors have ranges much larger than others data vectors, say [0, 1000] as opposed to [0, 1], the neural network might never converge. This problem occurs because some inputs are given more importance than others due to the relatively large size of their values. Normalizing the data to [-1, 1] ensures that the variability of a vector, not the size of its values, reflects its importance [76]. All inputs and output are normalized to [-1, 1].

3.3.5 Choosing a condition of fit

This section discusses the method used to assess a candidate network's performance using the available data sets. The error between the actual output and the neural network output, called the estimation error, should always be as small as possible. Prior knowledge about the system output data indicates that there will usually be positive and negative values. The *Neural Network Toolbox*™ compensates for this data characteristic by using the following error functions as the estimation error: mean squared error, mean absolute error and sum squared error.

The mean squared error (*mse*) is given in (3.3) and will be used as the estimation error in this experiment [74]. The variable $y(j)$ is the desired output value and $\hat{y}(j)$ is simply the actual output value obtained by the neural network for input sample j. There are n samples in the data set.

$$mse = \frac{1}{n}\left(\sum_{j=1}^{n} \left(y(j) - \hat{y}(j) \right)^2 \right) \tag{3.3}$$

A popular method used to avoid overfitting is called early stopping. The available data is divided into three sets: training set, validation set and test set. The training set is used to iteratively adjust the network weights and minimize the estimation error on the training set. After a specified number of iterations, the estimation error on the validation set is calculated to determine how the network performs on new data. If this validation error increases significantly, the network training is stopped and the values of the weights are reverted to the values at which they produced the smallest validation error [9].

The validation error produced by different networks provides a way of comparing the different designs and then choosing the best one. The estimation error on the third set, the test set, would be a way to test how well the network generalizes to new data. When the validation and test errors present more or less the same characteristics and the final estimation error on the training data is small, it usually indicates that a network has been trained well [9].

The available data will be divided into three different sets using specified ratios and therefore the number of samples in each set will not always be the same. Since *mse* uses the mean of the squared error data, it allows for a fair performance comparison of the network on the different sets regardless of the number of samples. The optimal size of the data sets used to train, test and validate the network will differ from model set to model set. Some network designs require more samples to train properly than others. Again, *mse* provides a way of comparing how different networks perform on data sets of different sizes. It is important to note that the number of samples in the different sets should be kept constant during an iteration of the system identification loop but may be varied from one iteration to the next [14].

When training a neural network, the chosen training algorithm can easily get trapped in the local minima of the error surface. A way to avoid this is by using multiple (10 to 100) random restarts – reinitializing the same network weights again and again. Obviously, this implies that the same network could initially perform very poorly but when training arrives at a global minimum, the results are more than satisfactory [7]. Using *mse* as a performance measure gives absolutely no indication of the effect of random initializations in some training algorithms.

It seems that a better method of determining network performance, other than *mse*, is required. Zemouri, Gouriveau and Zerhouni [77] provide performance metrics for neural networks which enable objective comparisons between different networks. Equation (3.4) defines the accuracy of a neural network [77].

$$\text{Accuracy} = \frac{1}{\text{Timeliness} + \text{Precision} + \text{Repeatability}} \tag{3.4}$$

The probability of a neural network to have an estimate close to the actual value is called the timeliness of the network. Equation (3.5) defines the timeliness of the network [77].

$$\text{Timeliness} = \overline{E} = \frac{1}{M}\sum_{i=1}^{M} E(i) \tag{3.5}$$

where M is the total number of times the training algorithm was executed (number of runs) and $E(i)$ is the mean estimation error of the i^{th} execution producing neural model i.

The precision of a neural network is the probability of the network to produce grouped estimates i.e. estimates that are close together. Equation (3.6) is used to calculate the precision of the network [77].

$$\text{Precision} = \overline{std} = \frac{1}{M}\sum_{i=1}^{M} std(i) \tag{3.6}$$

where $std(i)$ is the standard deviation of the i^{th} execution of the training algorithm. Equation (3.7) is used to calculate the standard deviation [77].

$$std(i) = \sqrt{\frac{1}{n}\sum_{j=1}^{n}\left(E(i) - y(j)\right)^2} \tag{3.7}$$

where n is the number of data samples and $y(j)$ is the j^{th} system output.

The repeatability of a neural network reveals how reliable the training process is. If the repeatability is small, training can be done repeatedly and the network will perform more or less the same every time. The effect of random initializations in some training algorithms can also be seen in the repeatability parameter. Equation (3.8) defines the repeatability of a neural network [77].

$$\text{Repeatability} = \frac{\sigma(std) + \sigma(E)}{2} \tag{3.8}$$

where $\sigma(std)$ and $\sigma(E)$ represent the standard deviation of all the $std(i)$ and $E(i)$ values as shown in (3.9) and (3.10).

$$\sigma(std) = \sqrt{\frac{1}{M}\sum_{i=1}^{M}\left(\overline{std} - std(i)\right)^2} \tag{3.9}$$

$$\sigma(E) = \sqrt{\frac{1}{M}\sum_{i=1}^{M}\left(\overline{E} - E(i)\right)^2} \tag{3.10}$$

If a neural network has small values for its timeliness, precision and repeatability, it will have high accuracy according to (3.4) and high confidence in the output estimates. The test data should be used to determine these performance metrics [77]. The accuracy will be used instead of the *mse* to determine a network's performance since it provides more information about the network and the training process. Note that the *mse* is still used by the *Neural Network Toolbox*™ for early stopping.

3.3.6 Calculating the model

Calculating the model when the candidate is a neural network consists of training the network. Batch training and incremental training are two commonly used training styles. The network weights are only updated once all the data samples have been presented when training in batch style. Incremental training consists of updating the weights after each data sample has been presented. Batch training cannot be used in an online training scheme because all the data is not readily available. Offline training can be done using either style though it is mostly done in batch style [9].

Since this system identification experiment is conducted offline, batch training will be used to train the FTD, NARX and LRN networks. The batch training algorithms in the *Neural Network Toolbox*™ consist of error backpropagation methods and all its variations. Standard backpropagation consists of two distinct steps – propagating the error derivatives back through the network and using the values to adjust the weights according to the gradient descent method [9].

Gradient descent in the second step can include heuristic modifications like a momentum term and a variable learning rate to improve network performance. These gradient descent backpropagation algorithms are very slow and if the algorithm parameters are not chosen properly, the network will perform poorly. The *Neural Network Toolbox*™ provides other backpropagation algorithms which use numerical optimization techniques in the second step of the algorithm [9].

The backpropagation algorithms using numerical optimization techniques converge significantly faster to a solution even though they require more computational effort. These algorithms include the following [9]:

- Conjugate gradient backpropagation
- Quasi-Newton backpropagation
- Levenberg-Marquardt backpropagation

Any of the backpropagation algorithms can be used but in an effort to remain practical, the slow-converging algorithms will not be considered. General requirements for an optimal algorithm are a fast convergence rate whilst not making unreasonable demands in terms of computer storage and processing power. It should be obvious that larger and more complicated network designs also increase computational demands and thus care should be taken to make a balanced decision between network design and training algorithm [74].

Table 3-2 shows a comparison of the backpropagation (BP) algorithms under consideration. The function column contains the name of the *m-file* typed in **MATLAB**® to use the specified training algorithm. The algorithms are compared by their convergence rate, optimal network size, intended application, computational effort and storage requirements.

Table 3-2: Comparison of BP algorithms

Function	Type	Convergence rate	Network Size	Application	Computational effort	Storage requirements
trainbfg	Quasi-Newton	Fast	Small	Nonlinear optimization	High	Moderate
trainbfgc						
trainlm	Levenberg-Marquardt	Very fast	Small	Function approximation	Very high	High
trainbr						
traincgb	Conjugate gradient	Fast	Large	Function approximation and pattern recognition	Moderate	Moderate
traincgf						
traincgp						
trainscg						

Literature strongly recommends the conjugate gradient and Levenberg-Marquardt backpropagation algorithms for function approximation problems such as modelling a nonlinear dynamic system [9]. The *Neural Network Toolbox*™ documentation [74] also suggests using Levenberg-Marquardt backpropagation for function approximation since it is the fastest algorithm in the toolbox, but only when training small-sized networks. Scaled conjugate gradient backpropagation is recommended for function approximation when training larger networks [74].

The main focus of this offline system identification experiment is architecture selection and consequently, no in-depth comparison will be performed between the various training algorithms. Following the recommendations in the relevant literature, Levenberg-Marquardt backpropagation (**trainlm**) will be used for networks with 300 weights or less and scaled conjugate gradient backpropagation (**trainscg**) for larger networks [74].

Selection of the initial weights and learning rate will be left to the chosen training algorithm. The values selected by the algorithm are considered to be appropriate for a large range of problems and do not require modification [74]. The NARX architecture allows for something very special with regard to training. The network is used in a parallel configuration but trained offline in a series-parallel configuration. Both configurations can be seen in figure 3-10.

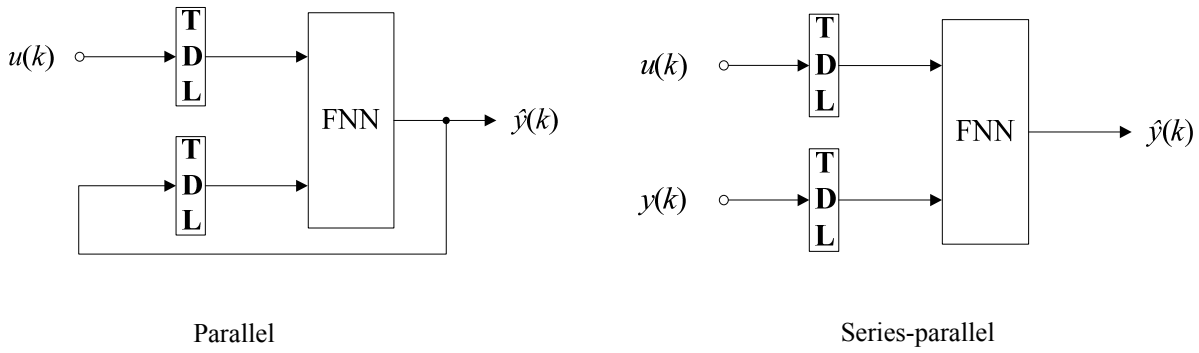

Figure 3-10: NARX configurations

Using supervised learning, the actual output $y(k)$ of the system is available during training. The series-parallel configuration enables the network to train using the available actual output instead of the estimated output $\hat{y}(k)$. The result is a feed-forward architecture which allows for the use of less complex training algorithms. The input to the neural network is significantly more accurate in the series-parallel configuration and helps the network weights to converge faster [74].

3.3.7 Model validation

The new model has to be validated to see how it affects the original system's behaviour. The conformity of the new model will only be proven if it can replace the original plant without causing considerable changes in the behaviour of the closed-loop control system. A suitable way to compare the two system behaviours is to simulate them in closed-loop and determine the frequency response of each system. The frequency response of the AMB control system under consideration is obtained in accordance with ISO CD 14839-3 [78]. Consider figure 3-11 which shows a simplified block diagram of a control system.

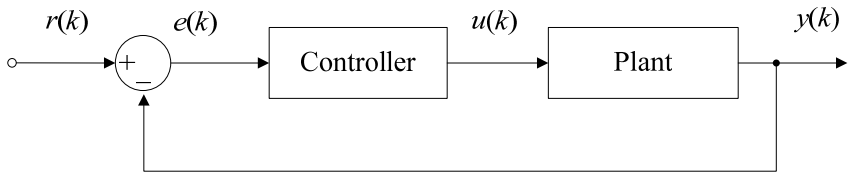

Figure 3-11: Simplified block diagram

The reference input $r(k)$ to the system should be a sinusoidal disturbance with varying frequency. The required data should be measured at rotor standstill and/or nominal operating speed but over the maximum frequency range. The maximum frequency range, f_{max}, starts at 0 Hz and ends at the maximum of either three times the rated speed or 2 kHz [78]. Since the maximum speed is 500 Hz, three times the maximum speed is 1.5 kHz. The maximum frequency range is thus: $0 < f_{max} < 2$ kHz.

It is worth the time and effort to investigate the dynamic behaviour of the original system at various frequencies. The *Shannon theorem* states that harmonic signals are only recreated correctly if sampled at a rate at least twice the signal frequency. The discrete-time frequency response is thus "...only uniquely defined up to a frequency which corresponds to half the sampling rate..." [1]. The sampling time of the original system is 100 μs which is equal to a sampling rate of 10 kHz. Half the sampling rate – the Nyquist frequency – is 5 kHz. A chirp reference input ranging from 0 Hz to 5 kHz over 10 s is therefore applied to AMB 1(x)[12] and the frequency response of the original system determined.

Figure 3-12 shows the frequency response but only up to 3 kHz since little or nothing happens after this frequency. The current maximum operating speed, 133 Hz, and the rated maximum speed, 500 Hz, can also be seen. Figure 3-12 also shows different peaks or critical frequencies corresponding to the rigid and bending modes of the rotor. The first two critical frequencies, 35 Hz and 65 Hz, are the two rigid modes of the rotor. The third critical frequency, 665 Hz, is the first bending mode.

[12] From this point forward, AMB 1(x) will refer to the x-axis of radial AMB 1. The same applies to the other AMB and axis directions. AMB 1 represents the upper AMB and AMB 2 represents the bottom AMB.

The original AMB flywheel system was designed to operate just below the first bending mode. In order to keep as close as possible to the practical system, only the first two critical frequencies will be considered. Since the frequency response between the 133 Hz and 500 Hz is neglibly small, the current maximum operating speed of 133 Hz will be considered the upper frequency of interest. The reference input in figure 3-11 will therefore be a multi-sine signal with frequencies ranging from 0 Hz to 133 Hz.

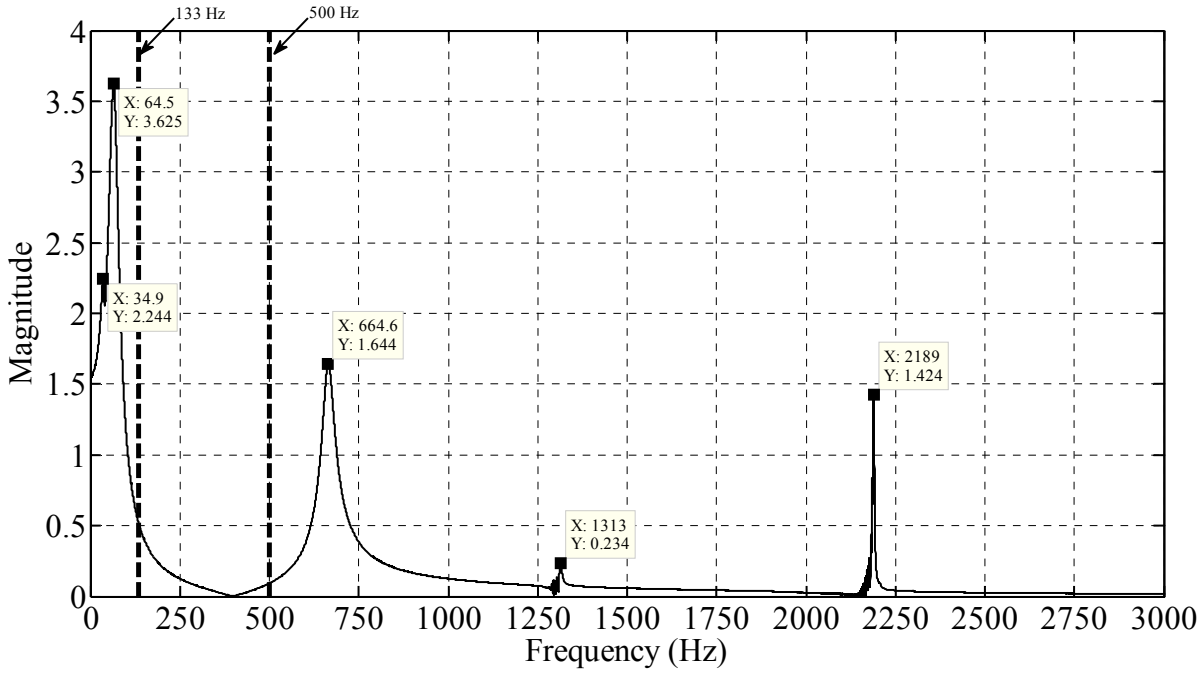

Figure 3-12: Frequency response up to 3 kHz

The measured output $y(k)$ in figure 3-11 is in the time domain and it is transformed to the frequency domain using the Fourier transform. This frequency domain data can then be considered the frequency response of the system. To obtain the frequency response of the system with the neural model as the plant, the system can be modified as illustrated in figure 3-13. The new system is compared with the original system by comparing their frequency responses.

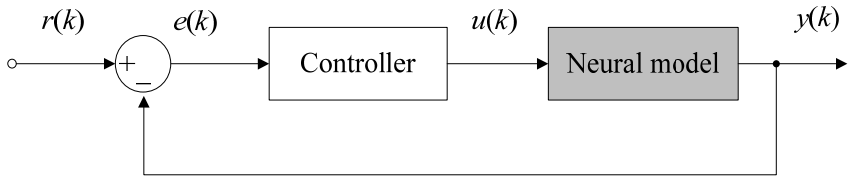

Figure 3-13: New system with neural model as plant

3.3.8 Model revision

When a model is deemed to be a poor representation of the system under consideration, it should be revised. The following changes can be made to the neural network model in order to improve its performance [74]:

- Re-initialize the weights and train the network again.
- Increase the number of hidden layer neurons or layers.
- Use larger sets of training data i.e. more training patterns.
- Try other training algorithms.

The window approach discussed earlier makes use of the first two changes to improve model performance. It should be noted that when increasing the number of hidden layer neurons it should be done gradually. If the network design is too complex the problem can become under-characterized and the network will have too many parameters to optimize. Also, more training patterns help the network to improve its generalization capability [9].

If there is a limited supply of data to train the network, there are other techniques to improve generalization other than early stopping. One such method is to use a training algorithm which uses regularization – a modification of the performance or error function. Regularization ensures that the weight values are smaller and the network response much smoother [9].

3.3.9 Implementation

The offline system identification experiment is implemented in **MATLAB®** making use of built-in functions as well as the *Neural Network Toolbox™*. Figure 3-14 illustrates the injection of an excitation signal into the control loop of AMB 1(x). The other excitation signals are injected in the same way. Remember that there are four control loops – one for each AMB axis direction.

The *Neural Network Toolbox™* does not allow the use of early stopping when training the NARX and FTD architectures. The goal with these architectures is to achieve an estimation error as small as possible using the training data. Using early stopping would prevent the estimation error on the training set from converging to its minimum. However, early stopping is used by the toolbox when training the LRN architecture [74].

Even though the three architectures cannot be trained in the same way, they are evaluated using the same model fit. The performance metrics (accuracy, timeliness, precision, repeatability) are all calculated using the test data set and provide objective comparisons between the architectures [77].

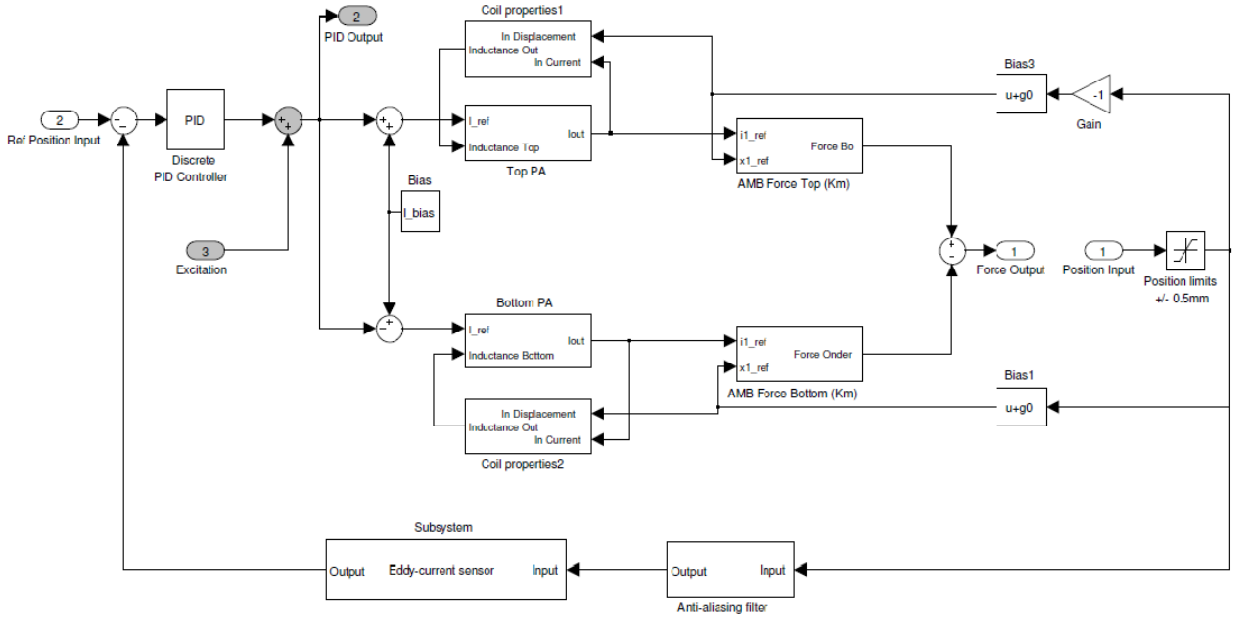

Figure 3-14: Excitation of AMB 1(x)

A high-level diagram is shown in figure 3-15 which explains the execution of the software loop. The software consists of functions from the *Neural Network Toolbox*™ as well as custom-coded functions indicated by bold text. Before the execution starts, the data is divided into three different sets for early stopping and normalized to the range [-1, 1] which makes training easier. The loop starts by initializing a model set according to a specified range of neurons and the number of layers.

The first network is then created using the chosen number of neurons and activation function as well as initializing the weights. Training of the specified network commences and after each epoch[13], the mean squared error on the validation set is calculated. In the case of an LRN network, the validation error is tested against the previous epoch's result. If the validation error increases for more than 6 consecutive epochs, training stops and the results are saved. The FTD and NARX networks are trained for a fixed number of epochs since early stopping is not used with these two architectures.

The next network goes through the same process until all the candidate models have been trained and tested for a specified number of runs. The candidate models are selected using the window approach described earlier to avoid exhausting all the possible combinations of neurons and layers. The final element of the system identification loop, model revision, is inherent to the window approach because models are revised by using multiple restarts and increasing the number of neurons and layers.

[13]An epoch, or one iteration of the training algorithm, is the term used to refer to the point in time when all the data samples have been presented to the network for training [74].

Small networks are trained using Levenberg-Marquardt backpropagation and large networks are trained using scaled conjugate gradient backpropagation. The most suitable networks for the modelling of the AMB flywheel system are determined by evaluating the results from the software loop described above. Only these networks are validated according to their frequency response as discussed in the model validation section.

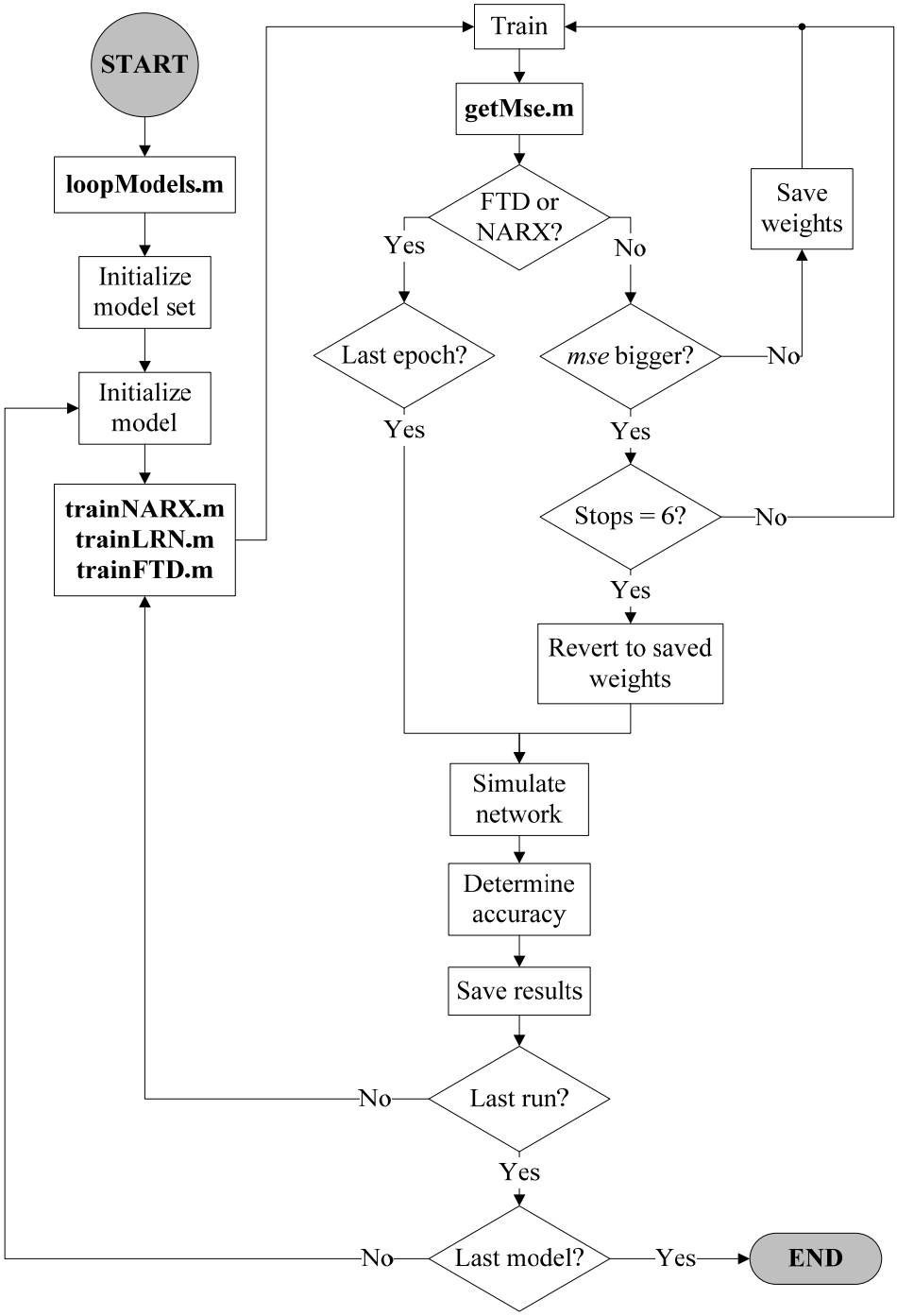

Figure 3-15: Architecture selection software execution loop

3.3.10 Results[14]

The horizontal and vertical directions of an AMB must be excited separately. This is achieved by applying two multi-sine signals with a 90° phase difference to the AMB [73]. Figure 3-16 shows the two excitation signals for AMB 1 – the solid line is for the x-axis and the dashed line is for the y-axis. Notice that the amplitude of an excitation signal is limited to the range [-1, 1] A. Excitation signals with amplitudes outside this range cause the rotor to move too close to the retainer bearings.

The existing AMB flywheel system uses a sampling rate of 100 μs. The same sampling rate is used throughout all other simulations. The total simulation time is 0.5 s which is equal to 5 000 samples @ 100 μs. The maximum frequency of the multi-sine signal is limited to 133 Hz and it contains 20 different frequencies. The same type of excitation signal is used for AMB 2(x) and AMB 2(y).

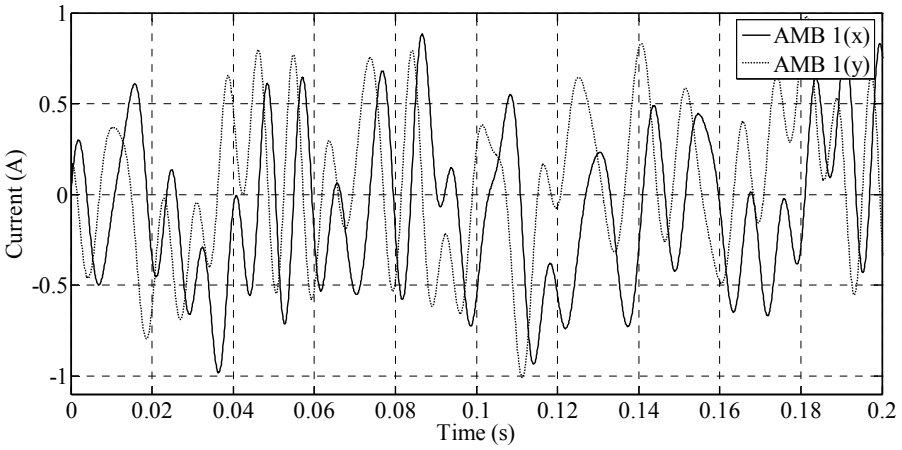

Figure 3-16: Excitation signal for AMB 1

Figure 3-17 shows the plant input measured after the injection of the excitation signal for AMB 1. The signal consists of the excitation signal summed with the PD controller output.

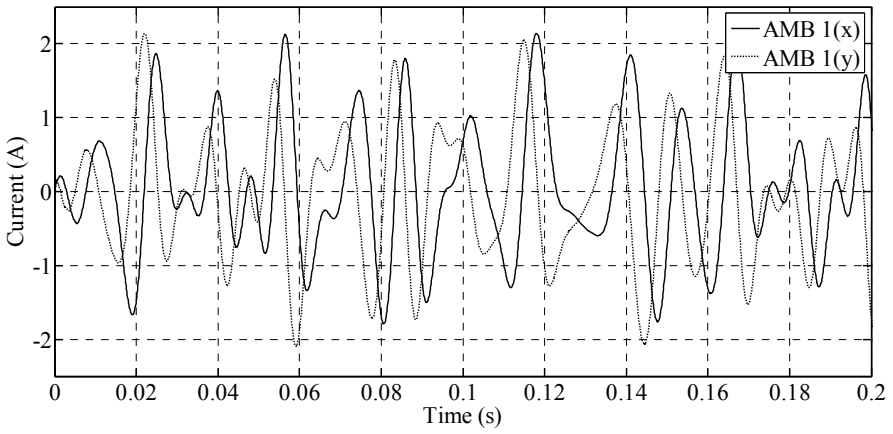

Figure 3-17: Plant input for AMB 1

[14] Not all the results are presented here because there are simply too many. Only the best results will be shown.

The plant input is used as the input data for neural network training. Figure 3-18 displays the plant output or rotor position in response to the multi-sine excitation signal. Notice that it displays two signals – a solid line for the x-axis of AMB 1 and a dashed line for the y-axis of AMB 1. These signals are used as output data for neural network training. The excitation signals, plant input and output for AMB 2 are very similar to those of AMB 1.

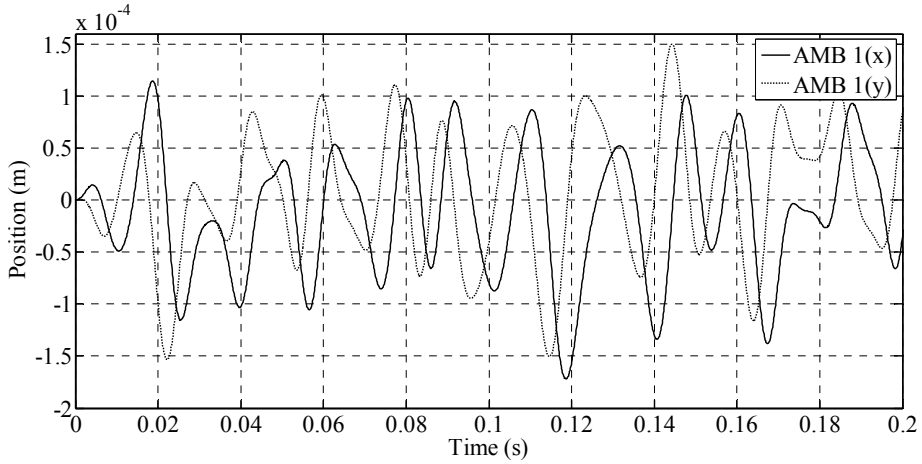

Figure 3-18: Plant output for AMB 1

Figure 3-19 shows the rotor's movement for each AMB in both the x- and y-axis directions. Notice the dashed circles representing the retainer bearings at 250 μm from the centre [0, 0]. The centre is the nominal reference point in all simulations conducted during this study.

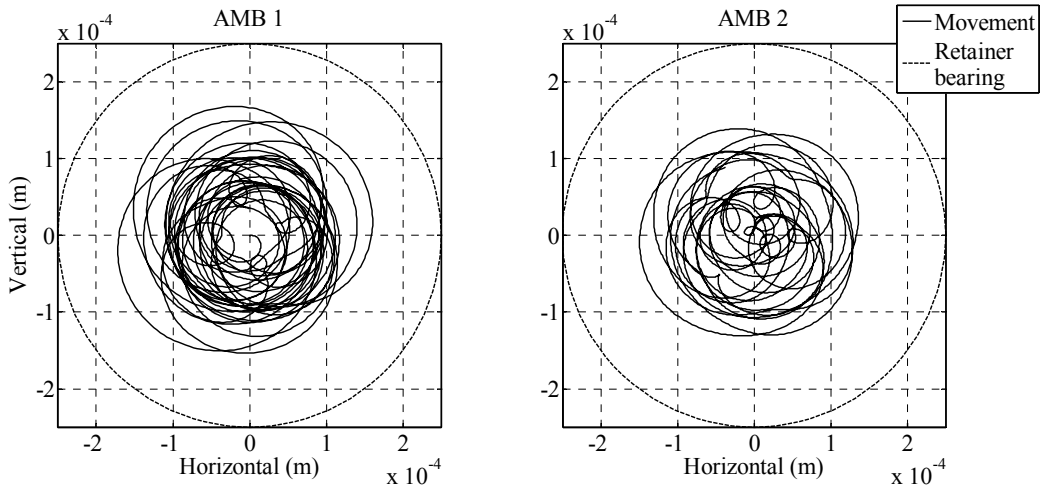

Figure 3-19: Rotor movement

The input-output data is divided into three sets (training, validation and test) for early stopping in the training of LRN networks. Figure 3-20 shows how the available input data for AMB 1(x) is divided into training, validation and test sets according to the ratio 0.7:0.2:0.1. The data for AMB 1(y), AMB 2(x) and AMB 2(y) is divided in the same way.

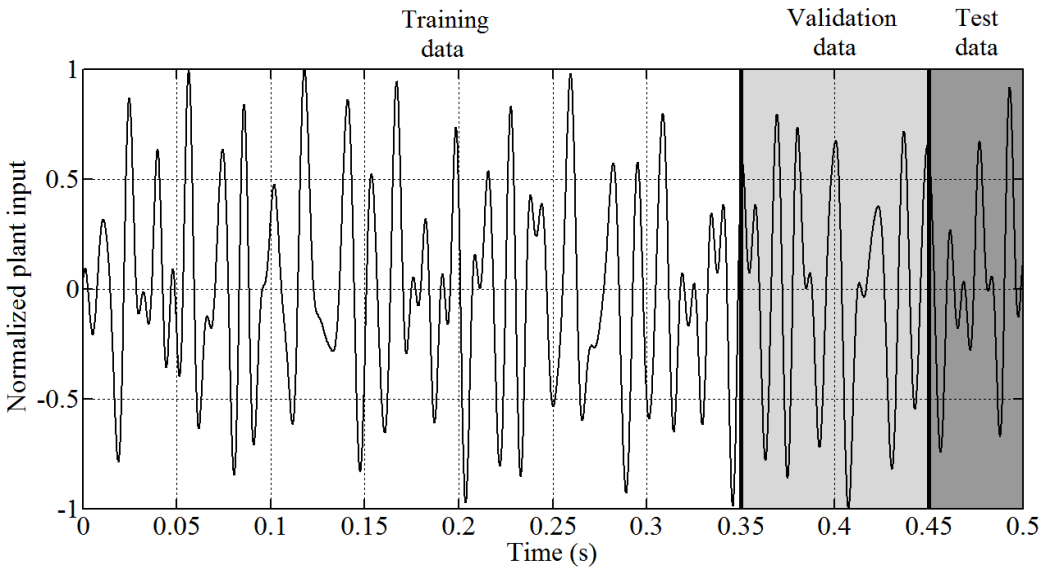

Figure 3-20: Data division

The test data set is used to determine the accuracy, timeliness, precision and repeatability of all three architectures. Even though the NARX and FTD architectures do not use the validation data set for early stopping, they cannot use this set in their training. The amount of training data should be the same for each network to allow for a fair comparison.

The results from the window approach will now be presented. All the networks use a 4-dimensional input and a 4-dimensional output because there are four AMB axis directions. Figure 3-21 illustrates the training results for the FTD architecture using 1 hidden layer of neurons – the first model set.

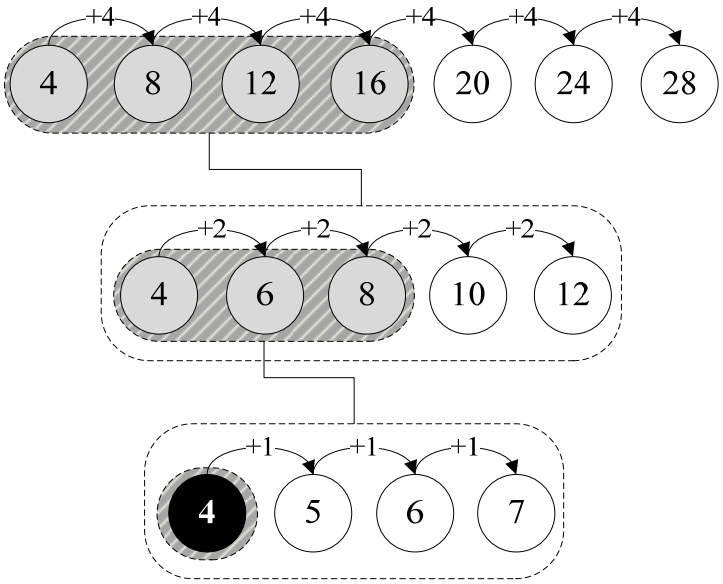

Figure 3-21: FTD training results

The same network is trained for a number of $M = 10$ runs[15] before moving on to the next network. The result from each run is saved to be used in the accuracy calculations of a candidate network. As can be seen in figure 3-21, the initial range of neurons starts at 4 and increases to a maximum[16] of 28 using 4 neurons at each increase. Figure 3-22 shows the accuracy of each FTD network from the first row in figure 3-21.

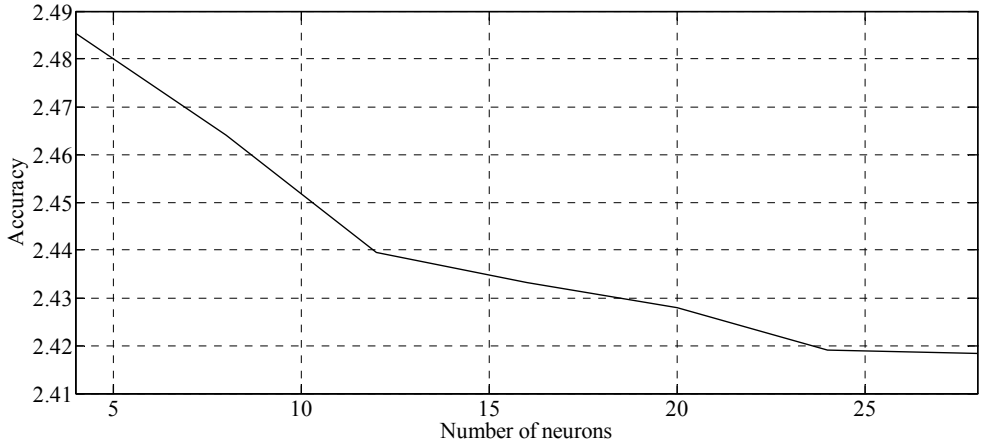

Figure 3-22: Accuracy of first row FTD networks

The curve indicates that the optimal number of neurons is somewhere between 4 and 12 neurons. This range is expanded in the second row of figure 3-21 using an increase of 2 neurons from network to network. The accuracy of the second row of FTD networks can be seen in figure 3-23. The optimal number of neurons seems to be somewhere between 4 and 8 neurons. The final row of FTD networks consequently has a range of 4 to 8 neurons with an increase of 1 from network to network.

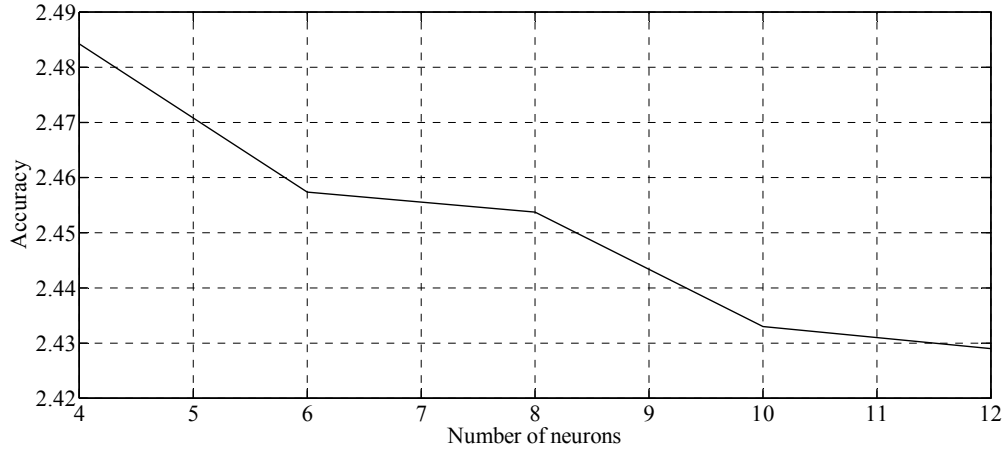

Figure 3-23: Accuracy of second row FTD networks

[15] The number of training runs or executions of the training algorithm is not to be confused with the number of epochs.
[16] The maximum number of neurons is chosen as 28 because the LRN architecture displays an out-of-memory error during training when using more than this number.

Figure 3-24 shows the accuracy of the final row of FTD networks. The highest accuracy occurs at 4 neurons and since this is the last row of networks, 4 neurons is considered the optimal number in this experiment for an FTD network.

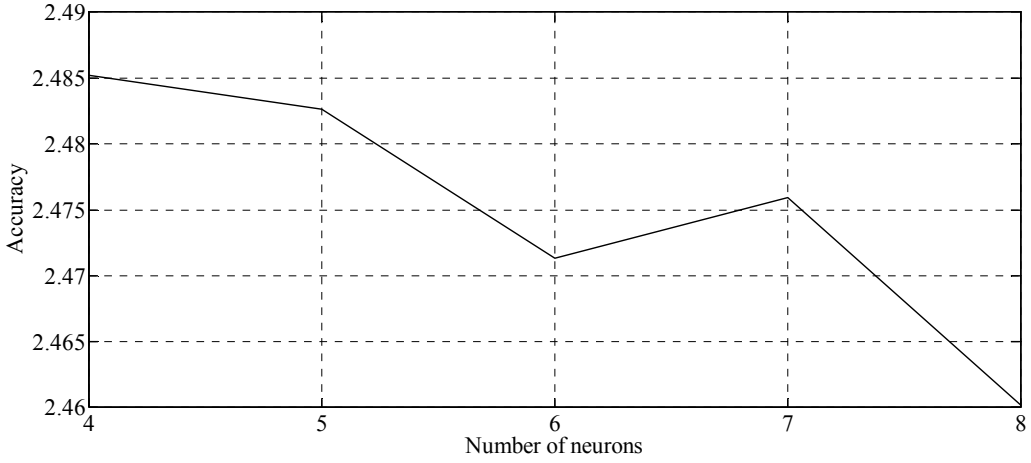

Figure 3-24: Accuracy of final row FTD networks

The FTD network using 4 neurons and 1 hidden layer is simulated in open-loop using the validation data set. Figure 3-25 shows the output of the FTD network against the original system output for AMB 1 (x). Notice that the output data is dimensionless since it is still normalized to [-1, 1]. The results for AMB 1(y), AMB 2(x) and AMB 2(y) are similar. The overall *mse* achieved by the FTD network is 1.16×10^{-3}

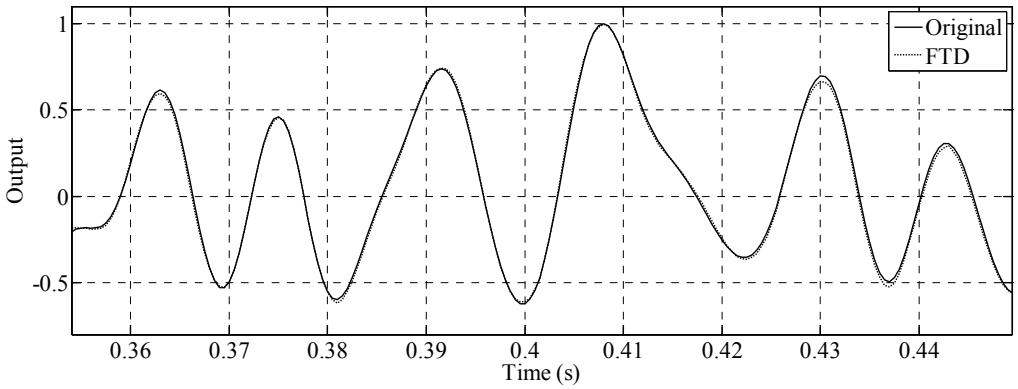

Figure 3-25: FTD output

The training results of the second model set are illustrated in figure 3-26. The NARX architecture with 1 hidden layer of neurons is used. The same initial range of the FTD architecture is used for the NARX architecture. The number of runs is also $M = 10$.

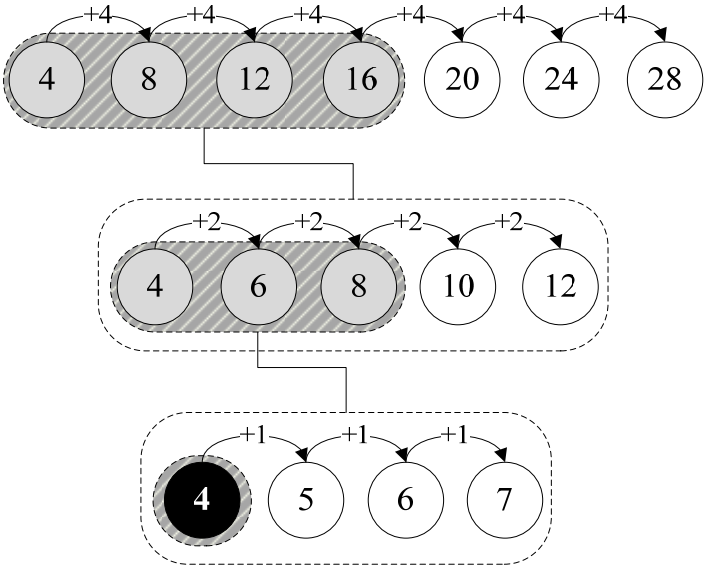

Figure 3-26: NARX training results

Figure 3-27 shows the accuracy of each NARX network from the first row in figure 3-26. The curve indicates that the optimal number of neurons is somewhere between 4 and 16 neurons. This range is expanded in the second row of figure 3-26 using an increase of 2 neurons from network to network.

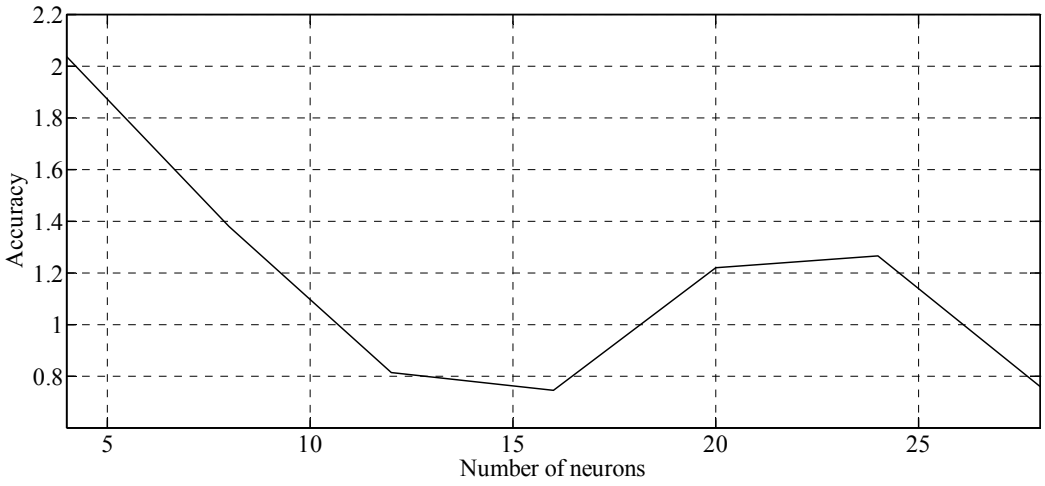

Figure 3-27: Accuracy of first row NARX networks

The accuracy of the second row of NARX networks can be seen in figure 3-28. The optimal number of neurons seems to be somewhere between 4 and 8 neurons. The final row of NARX networks consequently has a range of 4 to 8 neurons with an increase of 1 from network to network.

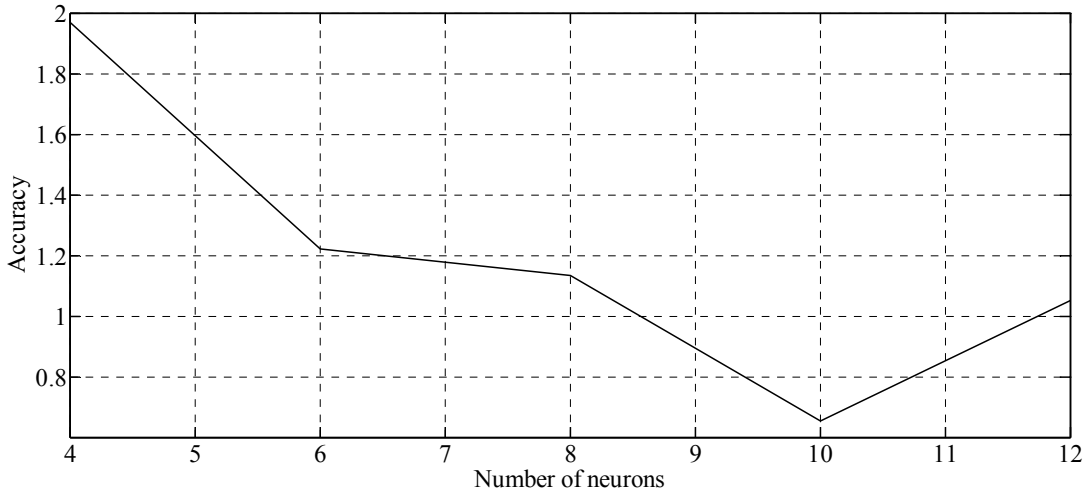

Figure 3-28: Accuracy of second row NARX networks

Figure 3-29 shows the accuracy of the final row of NARX networks. The highest accuracy occurs at 4 neurons and since this is the last row of networks, 4 neurons is considered the optimal number in this experiment for a NARX network.

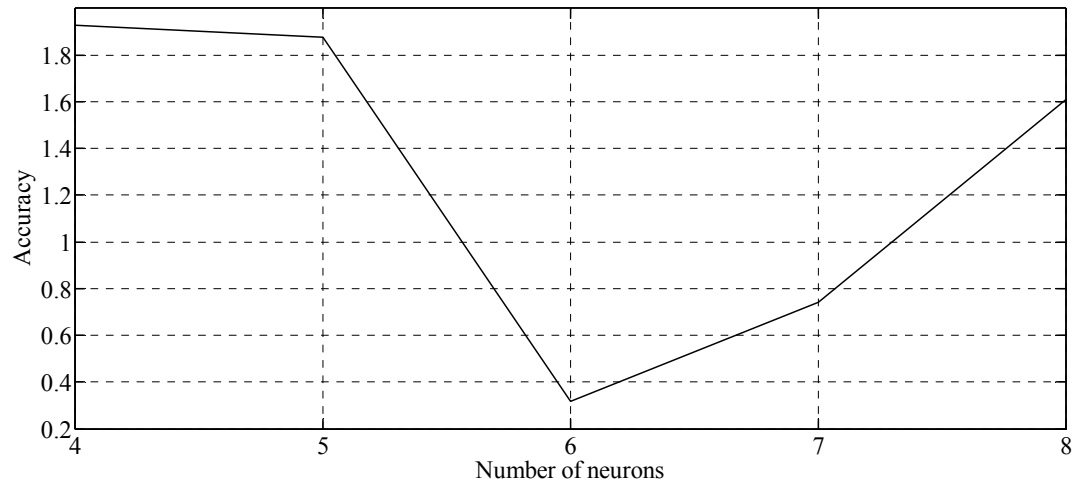

Figure 3-29: Accuracy of final row NARX networks

The NARX network using 4 neurons and 1 hidden layer is simulated in open-loop using the validation data set. Figure 3-30 shows the output of the NARX network against the original system output for AMB 1(x). The results for AMB 1(y), AMB 2(x) and AMB 2(y) are similar. The overall *mse* achieved by the NARX network is 1.74×10^{-1}.

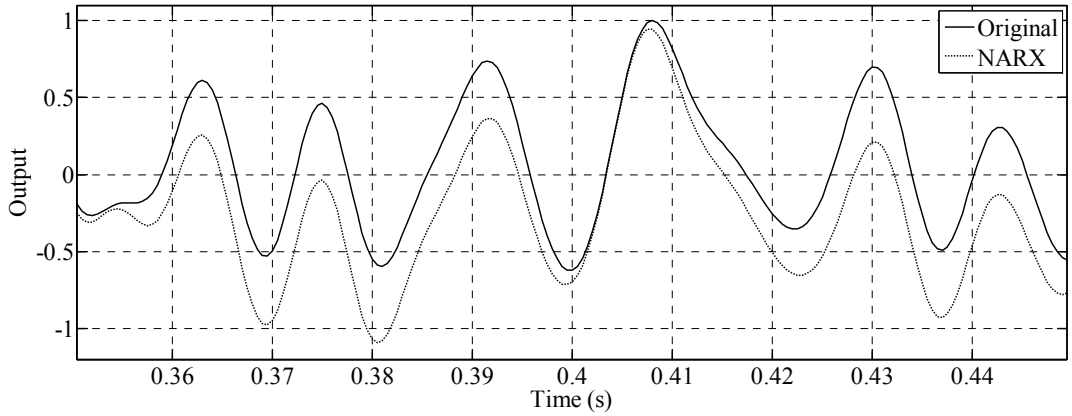

Figure 3-30: NARX output

Figure 3-31 illustrates the training results for the LRN architecture using 1 hidden layer of neurons – the third and final model set. The same initial range of the FTD architecture is used for the LRN architecture. The number of runs is also $M = 10$.

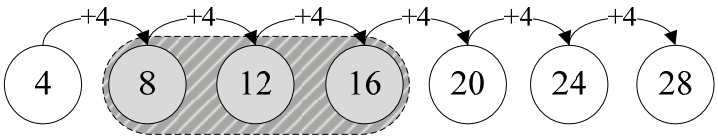

Figure 3-31: LRN training results

Figure 3-32 shows the accuracy of each LRN network from the first row in figure 3-31. The curve indicates that the optimal amount of neurons is somewhere between 8 and 16 neurons.

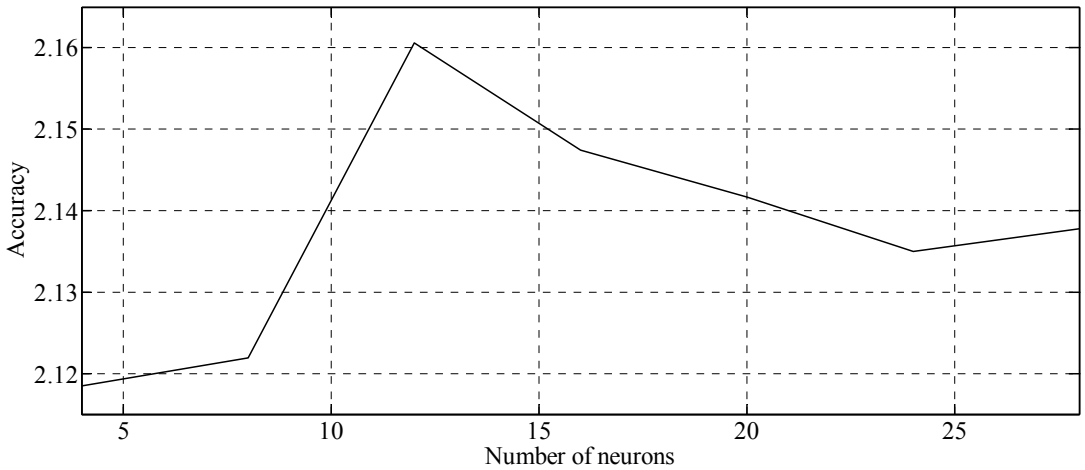

Figure 3-32: Accuracy of first row LRN networks

Training the first row of LRN networks takes considerably longer than training the first row of the other architectures and warrants an investigation. The excess training time of the LRN networks cannot be attributed to the training algorithm since the same algorithm is used for all three architectures. The next logical step is to take a look at the network parameters i.e. the weights and biases. Table 3-3 shows each network architecture and the number of network parameters for a specified number of neurons.

Table 3-3: Network parameters

Neurons	Parameters for FTD	Parameters for NARX	Parameters for LRN
10	94	134	194
20	184	264	584
30	274	394	1174
40	364	524	1964

The LRN architecture requires significantly more weights for the same number of neurons than the other two architectures. This is the reason why training of an LRN network takes so much longer than training of a NARX or FTD network. The LRN architecture is excluded from the rest of the experiment because of its high level of computational demand.

Even though the optimal number of neurons have been found for both the FTD and NARX architectures, it is useful to evaluate the *mse* on the validation set on last row of networks. Table 3-4 contains the final FTD and NARX networks and their corresponding *mse* on the validation set.

Table 3-4: Network *mse* values on validation set

Neurons	*mse* for FTD	*mse* for NARX
4	1.16×10^{-3}	1.74×10^{-1}
5	7.88×10^{-4}	1.38×10^{-1}
6	9.57×10^{-4}	1.39×10^{-1}
7	7.33×10^{-4}	1.05×10^{-1}
8	1.14×10^{-3}	1.07×10^{-1}

When moving on to 2 hidden layers of neurons, the maximum number of neurons has to be decreased to 20 neurons due to the memory limitations of the personal computer used. Figure 3-33 displays the accuracy of all three architectures ranging from 4 to 20 neurons using an increase of 4 neurons from network to network. The window approach is not completed using 2 hidden layers since no significant increase in accuracy can be seen by adding another hidden layer. For this reason, it is decided to stop increasing the number of layers.

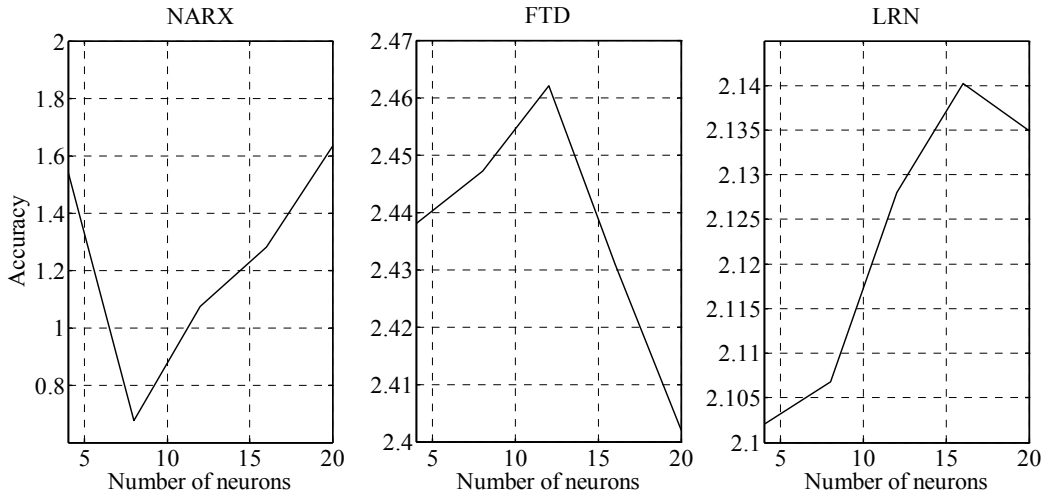

Figure 3-33: Accuracy for 2 hidden layers

The optimal FTD and optimal NARX network are now each simulated in the original closed-loop control system as the plant. Fresh, previously unseen data is presented to each network. The networks are tested by applying a step reference input at time 0.1 s with an amplitude of 10 μm to AMB 1(x) and comparing their responses with the original system response. The original system step response for AMB 1(x) can be seen in figure 3-34.

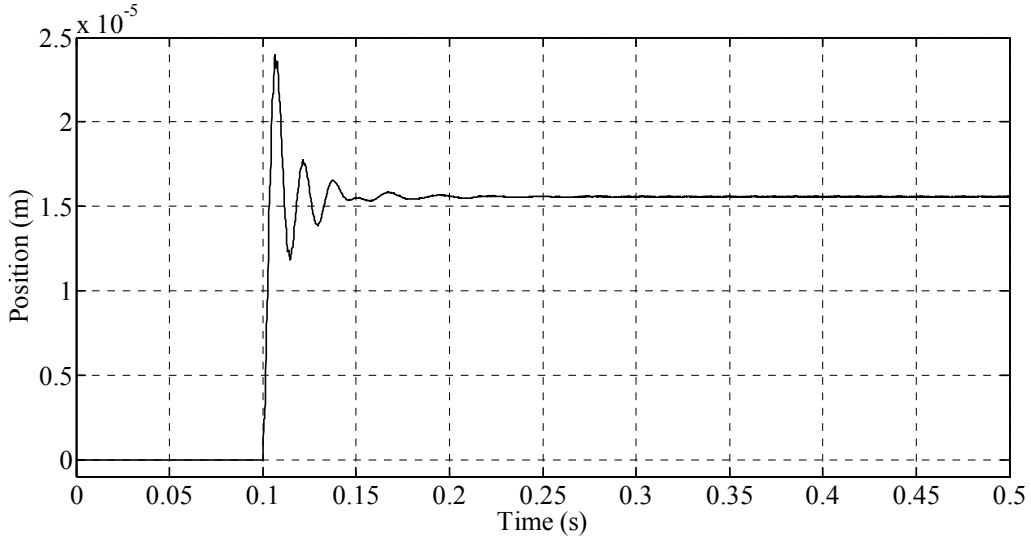

Figure 3-34: Original system step response

Figure 3-35 shows the movement of the rotor in both the x- and y-axis directions. The step response of the system using the FTD network as plant against the original system step response can be seen in figure 3-36. Notice that the response for the system containing the FTD network is considerably different from the response of the original system.

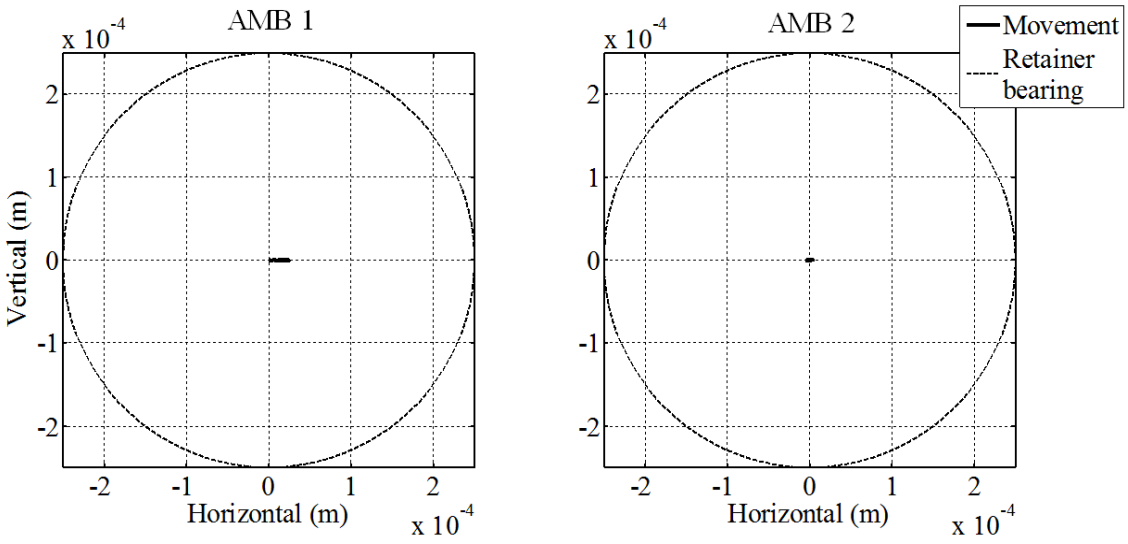

Figure 3-35: Rotor movement for original system with step input

Consider figure 3-37 which shows the rotor movement for the system containing the FTD network as the plant. The rotor clearly touches the retainer bearings which is not the case for the original system. Comparing figure 3-37 with figure 3-35 it is clear that the FTD network does not represent the true plant. The FTD network will not enter the model validation phase since it is now deemed invalid. The step response of the system containing the NARX network as the plant will now be considered. Applying the same step reference input to AMB 1(x), the step response using the NARX network against the step response of the original system can be seen in figure 3-38. Notice the significant steady-state error produced by the NARX network.

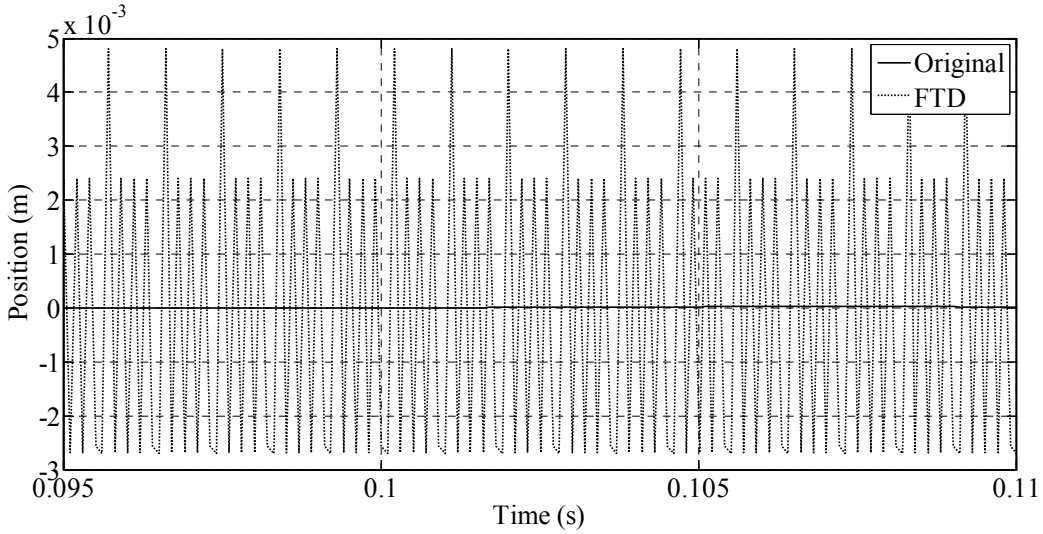

Figure 3-36: FTD step response

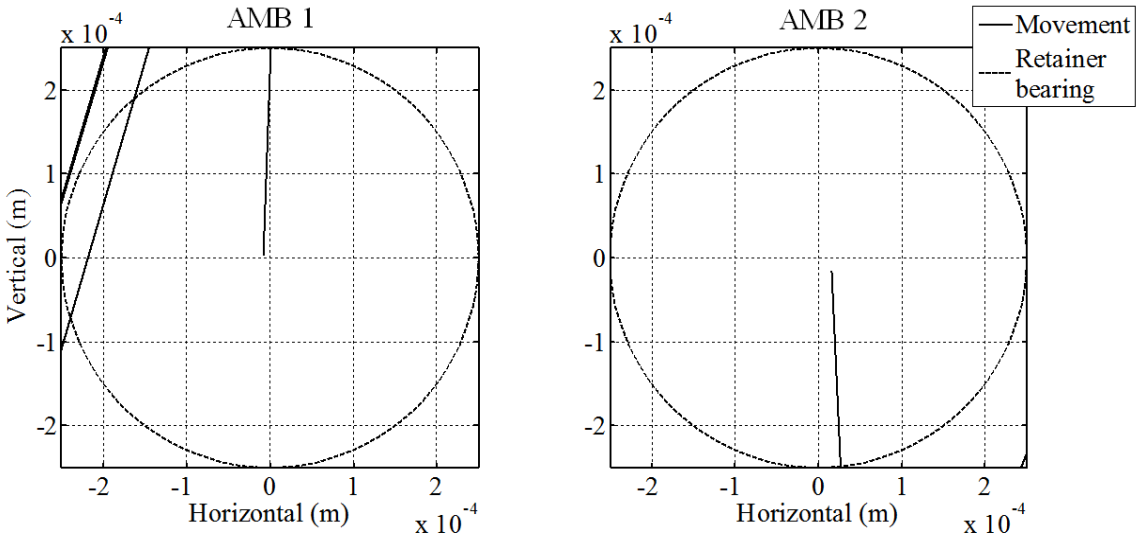

Figure 3-37: Rotor movement for FTD with step input

The movement of the rotor in both the x- and y-axis directions is illustrated in figure 3-39. Comparing figure 3-39 with figure 3-35 it is clear that the NARX network produces a response similar to the original plant. The NARX network will now enter the model validation phase. The data required to determine the frequency response of the system is obtained using multi-sine excitation signals at the reference input and measuring the system output. The multi-sine signals contain 20 frequencies ranging from 0 Hz to 133 Hz. The same multi-sine signals used for AMB 1(x) and AMB 2(x) are shifted in phase by 90° and applied to AMB 1(y) and AMB 2(y) respectively. The amplitude of all the multi-sine signals is 10 µm.

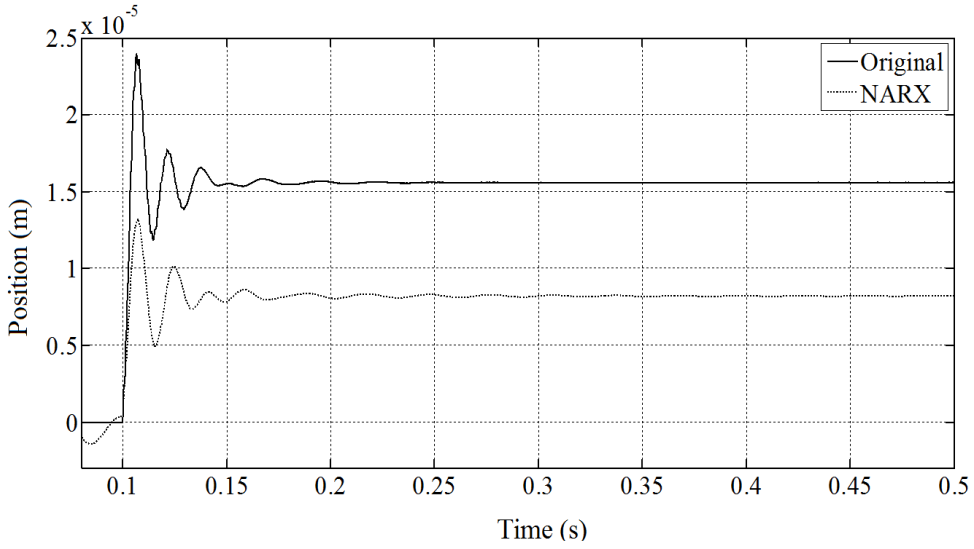

Figure 3-38: NARX step response

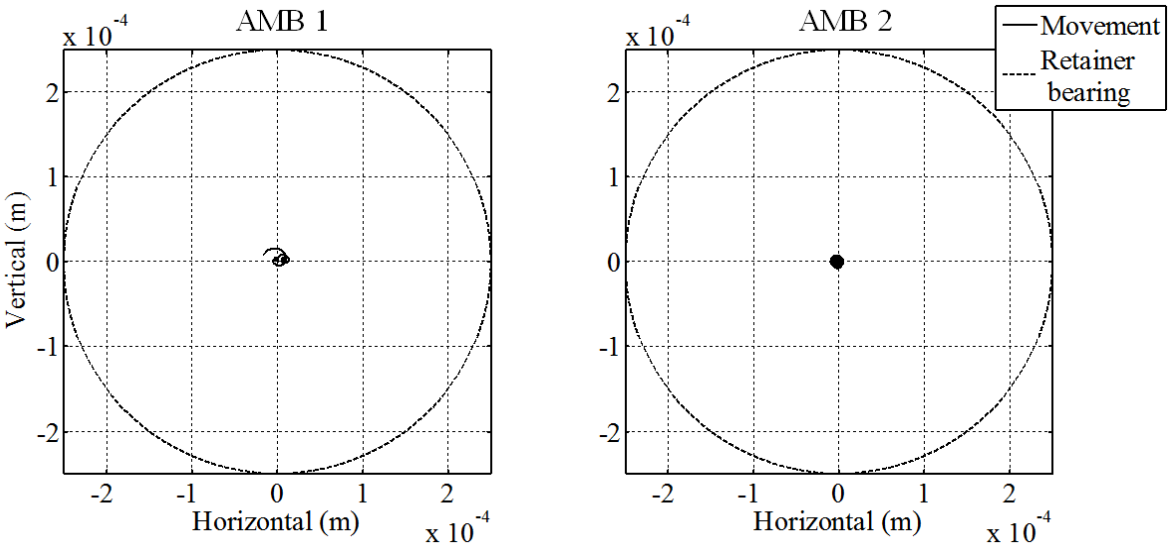

Figure 3-39: Rotor movement for NARX with step input

Figure 3-40 displays the frequency response of the original system against the frequency response of the system containing the NARX network as the plant. Figure 3-41 also displays the frequency response of the original system against the frequency response of the system containing the NARX network as the plant but is has been zoomed in to the frequency range [0, 150] Hz. The frequency range of interest, 0 Hz to 133 Hz, can now be observed better. The frequency response of the NARX system for AMB 1(x) and the frequency response of the original system for AMB 1(x) are clearly similar.

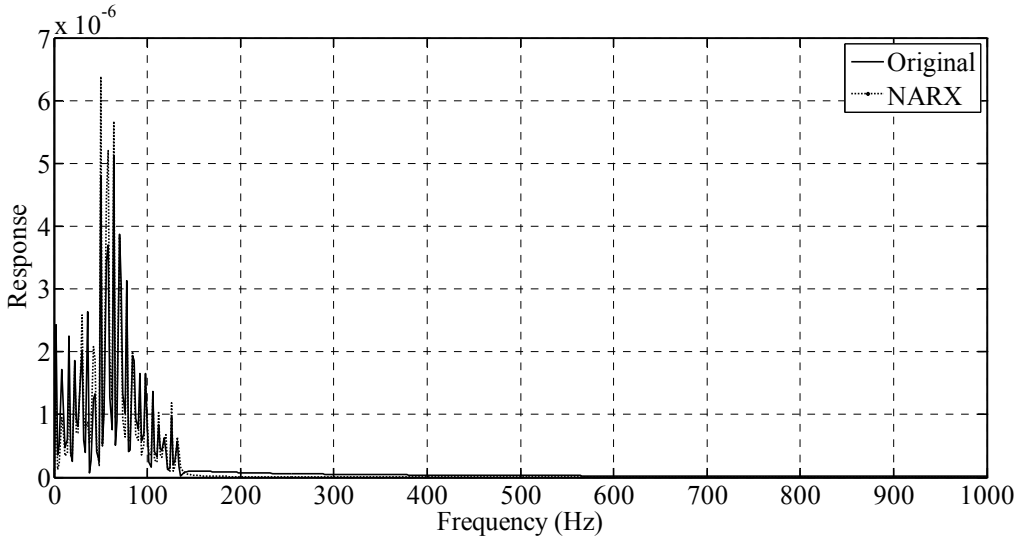

Figure 3-40: NARX frequency response

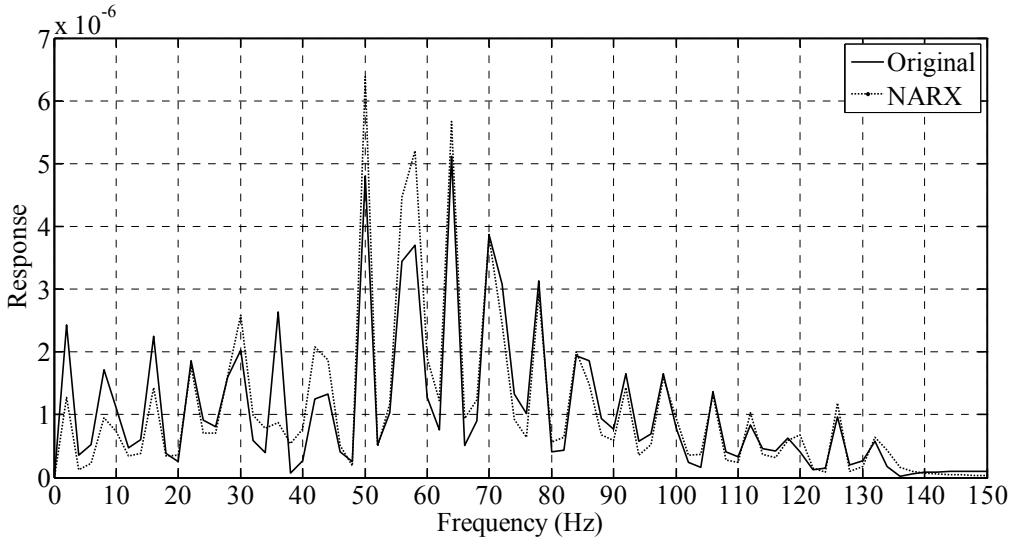

Figure 3-41: NARX frequency response (zoomed)

The frequency responses for AMB 1(y), AMB 2(x) and AMB 2(y) will not be shown. The correlation coefficients of the NARX system responses with the original responses are presented in table 3-5 instead. A coefficient of 1 implies perfect positive correlation between the response of the NARX system and the response of the original system. No correlation is indicated by a coefficient value of 0. The NARX system response clearly exhibits the same characteristics of the original system since all of the coefficients are close to 1 [79].

Table 3-5: Response correlation

AMB axis	Correlation coefficient
AMB 1(x)	0.958
AMB 1(y)	0.965
AMB 2(x)	0.921
AMB 2(y)	0.916

3.4 Conclusions

Prior knowledge about the problem and application area assisted significantly in narrowing the choice of network architecture. More than 20 architectures and their corresponding training algorithms were reduced to 3 architectures and 2 training algorithms. Methods of determining network performance and model validity were also developed using prior knowledge.

The LRN architecture was trained using early stopping whereas the FTD and NARX architectures did not make use of early stopping for training. This makes sense considering that different network architectures require training techniques according to their inherent features.

A matrix called the Jacobian, which contains the first error derivatives with respect to the weights and biases of a network, is calculated and stored for use in Levenberg-Marquardt backpropagation. The Jacobian matrix becomes quite large for networks using many neurons [74]. This explains why storage requirements and computational effort increased significantly when using Levenberg-Marquardt backpropagation on large networks.

The scaled conjugate gradient backpropagation algorithm does not require a computationally expensive line search at each epoch like all the other conjugate gradient algorithms [74]. This proved useful since training large networks proved to be time-consuming with Levenberg-Marquardt backpropagation. Scaled conjugate gradient backpropagation was an effective alternative for training large networks.

With careful inspection of the figures in the previous section and table 3-4, it can be seen that the *mse* was definitely not the best indication of a network's accuracy. The network which achieved the smallest *mse* on the validation data set was not the network which achieved the highest accuracy. It is important to achieve as small an *mse* as possible, but not at the cost of network accuracy (timeliness, precision and repeatability).

Moving on to the next chapter, online system identification and adaptive control will be the main focus. The chosen architecture should train quickly to adapt to changes in its environment [8]. Since the LRN requires a high level of computational effort, it was dismissed as a candidate architecture. Even though the optimal FTD network achieved a higher accuracy and a smaller *mse* than the optimal NARX network, it did not work in a closed-loop configuration.

Careful consideration of the inputs and outputs of an AMB system provides some insight. A 1-DOF AMB control system with PD control is under consideration. The transfer function of the plant $G_p(s)$ using a linearized model of the system is defined as the ratio of the output to the input in the Laplace domain in (3.11).

$$G_p(s) = \frac{Y(s)}{U(s)} = \frac{2k_i}{ms^2 - 2k_s} \tag{3.11}$$

where m is the mass of the rotor and k_i and k_s are electromagnetic constants. All the variables are given the value of unity, including the sampling time T, to simplify calculations. With the help of **MATLAB**®, the variables are substituted with their values and the transfer function becomes (3.12).

$$\frac{Y(s)}{U(s)} = \frac{2}{s^2 - 2} \tag{3.12}$$

Using inverse Laplace transformation on (3.12), the transfer function in the time-domain is obtained. The discrete-time transfer function is obtained by substituting $t = kT = k$ in the time-domain transfer function. The z-transform of the discrete-time transfer function is transformed into a difference equation using the real translation property of the z-transform. The final difference equation is given in (3.13).

$$y(n) = 1.2u(n-1) - 1.2u(n-2) - 4.4y(n-1) - y(n-2) \tag{3.13}$$

Equation (3.13) indicates that the output $y(n)$ depends on previous values of both the input $u(n)$ and the output. Since the NARX architecture makes use of previous values of the input and the output, it is able to model the plant of the AMB flywheel system successfully. The FTD architecture only makes use of previous values of the input and cannot model the plant of the AMB flywheel system successfully.

The inherent instability of the AMB flywheel system did not pose a problem to the NARX architecture. According to Ljung [14], unstable systems can be identified under the condition that both the closed-loop system and observer are stable. Models ideal for this type of problem are the ARX and ARMAX type as well as their nonlinear versions, NARX and NARMAX [14].

The NARX architecture's ability to model the plant of the AMB flywheel system has been well validated even though it produced a significant steady-state error. This steady-state error can be reduced by using superior training algorithms. The training algorithms used in this chapter are batch style algorithms meaning they are applied offline. An online observer will require an incremental style algorithm allowing for fast updates of the neural network. The NARX architecture will be used in the following chapters as an online observer for the plant of the AMB flywheel system [8].

A final recommendation needs to be made at this point. When training a neural network, the tendency to get stuck in a local minimum of the error function is significant. The window approach used in this chapter to determine the optimal network parameters may have been successful, but it still required a considerable amount of effort. A global optimization method such as a genetic algorithm could be used instead to determine the optimal network parameters and avoid local error minima [7].

The system identification loop has also proven to be quite effective in obtaining an optimal neural model for the plant of the AMB flywheel system. The next chapter will discuss the various steps taken to select a suitable adaptive control design for the AMB flywheel system and integration of the validated NARX architecture into the chosen design.

Chapter 4: Adaptive control design

This chapter starts by describing the system under consideration and reviewing indirect adaptive control using neural networks. The use of reinforcement learning in an online adaptive control system is motivated. The focus then shifts to the details of an adaptive critic controller. A discussion on the chosen adaptive control design follows and finally, optimization of the controller parameters using genetic algorithms is explained.

4.1 Introduction

The aim of this chapter is to obtain an adaptive controller for the AMB flywheel system. The first step is to identify the nonlinear, dynamic, multivariable and inherently unstable plant of the AMB flywheel system online. The discrete time Brunovsky canonical form (4.1) can be used to describe the nonlinear dynamics of a SISO plant with *n* system states [8].

$$\begin{aligned} x_1(k+1) &= x_2(k) \\ x_2(k+1) &= x_3(k) \\ &\vdots \\ x_n(k+1) &= f(\mathbf{x}(k)) + g(\mathbf{x}(k))u(k) \\ y(k) &= h(\mathbf{x}(k)) \end{aligned} \tag{4.1}$$

The vector $\mathbf{x}(k) \in R^n$ is called the state vector and $u(k)$ is the input to the system, also called the control signal. Functions $f(\cdot)$ and $g(\cdot)$ are unknown nonlinear functions. The output $y(k)$ can be a function of the system states or have a particular form as given in (4.2) [8].

$$y(k) = h(x_1(k)) \tag{4.2}$$

Equation (4.1) can also be written in MIMO form by defining each system state as a vector where *m* indicates the number of inputs as well as the number of outputs in (4.3).

$$\mathbf{x}_i(k) \in R^m \tag{4.3}$$

Consequently, the state vector becomes a state matrix in (4.4).

$$\mathbf{X}(k) \in R^{n \times m} \tag{4.4}$$

The MIMO representation of (4.1) now becomes (4.5).

$$\begin{aligned} \mathbf{x}_1(k+1) &= \mathbf{x}_2(k) \\ \mathbf{x}_2(k+1) &= \mathbf{x}_3(k) \\ &\vdots \\ \mathbf{x}_n(k+1) &= \mathbf{f}(\mathbf{X}(k)) + \mathbf{G}(\mathbf{X}(k))\mathbf{u}(k) \\ \mathbf{y}(k) &= \mathbf{h}(\mathbf{X}(k)) \end{aligned} \tag{4.5}$$

The control signal is now a vector $\mathbf{u}(k) \in R^m$. The unknown nonlinear functions become $\mathbf{f}(\cdot) \in R^m$ an unknown nonlinear function vector and $\mathbf{G}(\cdot) \in R^{m \times m}$ a matrix of unknown nonlinear functions. Equation (4.5) is an appropriate system description of the plant of the AMB flywheel system and will be used to represent it in the rest of this chapter [8]. It has already been concluded in the previous chapter that the NARX architecture will be used to model the plant. The key lies in combining (4.5) with a NARX neural network and using it in an indirect adaptive control system.

Other choices regarding controller design still have to be made. The choices of controller structure and error system are among these. Equally important is the development of adaptive laws. Analysis of stability and robustness and evaluation of system performance are also included in the controller design choices [52].

Adaptive control design is based on the certainty equivalence principle. It focuses on using approximations of unknown parameters as if they were the correct values of those parameters. The adaptive law is driven by a system performance function to generate the parameter approximations. These approximations are also used in a design equation to construct an adaptive controller which maps the plant approximations to controller parameters. For this principle to be effective, the approximations of the parameters should satisfy the following: small estimation errors, bounded parameter estimates and slow parameter variations [52].

4.2 Indirect adaptive control

In an indirect adaptive control (IAC) design the controller parameters are calculated using model approximations. These approximations are obtained from a model of the plant using system identification. The adaptive law uses an approximation error representing the mismatch between the plant output $\mathbf{y}(k) \in R^m$ and the online model approximations from the observer [52]. Figure 4-1 illustrates an IAC system in MIMO form.

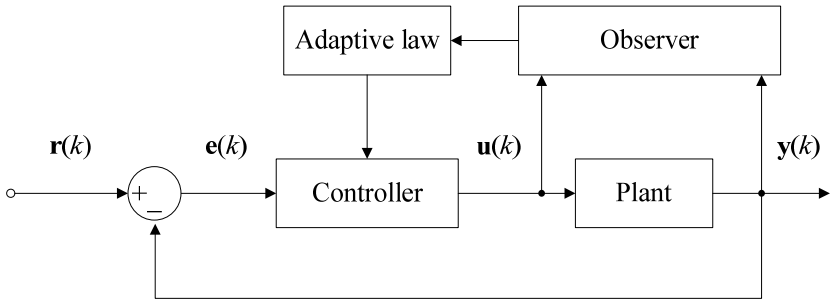

Figure 4-1: Indirect adaptive control system (MIMO)

4.2.1 Using neural networks

The NARX architecture is most suitable to model the plant of the AMB flywheel system and it will be used as an online observer. A NARX neural network can be used as an online observer as illustrated in figure 4-2. The block diagram is in discrete-time because it represents a simulation model using sampled data. The controller structure is also chosen as a neural network because this study's aim is to incorporate neural networks into an adaptive controller for the AMB flywheel system.

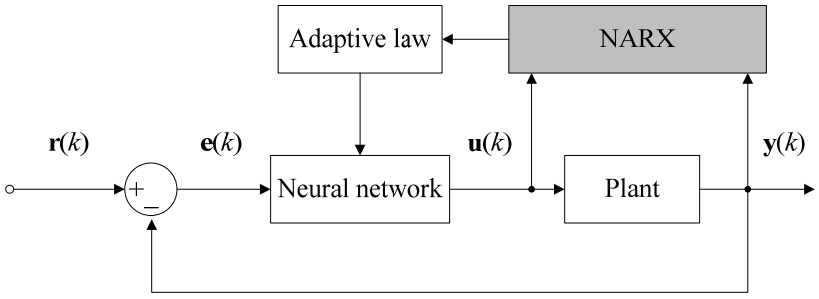

Figure 4-2: Discrete-time IAC with neural network (MIMO)

When using neural networks in an online manner, a significant issue arises. Offline training is made easy by supervised learning because there is usually a desired output signal to be matched. This is not the case for an online neural network. Since the nonlinear functions $\mathbf{f}(\cdot)$ and $\mathbf{g}(\cdot)$ in (4.5) are unknown in many online problems, it is difficult to provide desired outputs to the network for training [8].

A neural controller can be trained online to mimic an existing controller which provides the desired output signal, but the whole point is to improve system performance, not match it. Many neural control methods require online training when no desired output signal is available. For these reasons, the error system will make use of reinforcement learning [8].

4.2.2 Reinforcement learning

"If an action is followed by a satisfactory state of affairs, or an improvement in the state of affairs, then the tendency to produce that action is strengthened or reinforced (rewarded). Otherwise, that tendency is weakened or inhibited (penalized)." This is the basic notion behind reinforcement learning according to Jang, Sun and Mizutani [7]. When the action selections depend upon state information, the idea is extended to feedback control [8].

A reinforcement signal is used to evaluate the performance of a computational agent interacting with its environment. The aim is to discover a policy for selecting actions that minimize a specified performance index [7]. Reinforcement learning methods are considered potentially useful to feedback control systems since these methods do not require comprehensive information about the system or its behaviour [8].

During supervised learning with a neural network as the computational agent, the weights are adjusted in proportion to the output error of the neurons. Something different occurs when using reinforcement learning – the weights are adjusted according to the reinforcement signal [8]. This is especially useful since no desired output signals are available online to train a neural controller. The general idea of reinforcement learning is extended to neural networks using an adaptive critic model [7].

4.3 Adaptive critic model

The adaptive critic model consists of two main components: the critic and the actor. The critic generates a reinforcement signal estimated from state information and environment inputs – it evaluates the system performance. The actor (computational agent) uses this reinforcement signal from the critic to learn correct actions for optimal control [7]. In this study, both components of the adaptive critic model are neural networks.

The critic neural network uses the standard Bellman equation or a simple weighted sum of the tracking errors as a performance index. The weights of the critic neural network are updated in such a way as to minimize this index. At the same time, the critic sends the reinforcement signal to the actor. The actor neural network is trained using the reinforcement signal and information about the tracking error to keep the tracking error as small as possible [8].

The reinforcement signal generally carries much less information than the training signal used in supervised learning. Still, the adaptive critic is very successful in controlling complex systems. Analytical stability and performance proofs are lacking for many neural control designs. Adaptive critic neural control does not suffer from this drawback – not even when applied to nonlinear discrete time systems [8]. Figure 4-3 illustrates the adaptive critic model in MIMO form.

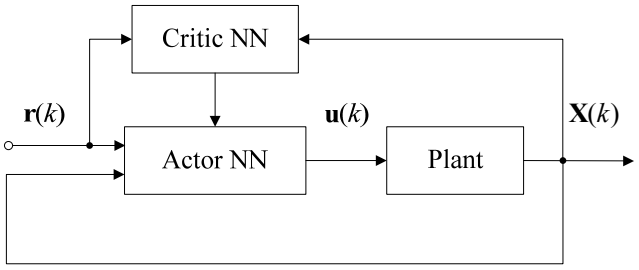

Figure 4-3: Adaptive critic model (MIMO)

The aim of this study is an adaptive controller using output feedback since many practical AMB systems cannot provide state feedback. Adaptive critic neural control has many useful features but it requires that state information be sent to the critic. One solution is to make use of system identification and employ an online state observer. This observer can estimate the plant states and provide the necessary information the critic requires [2]. Figure 4-4 illustrates the proposed solution using a NARX neural network as the online observer.

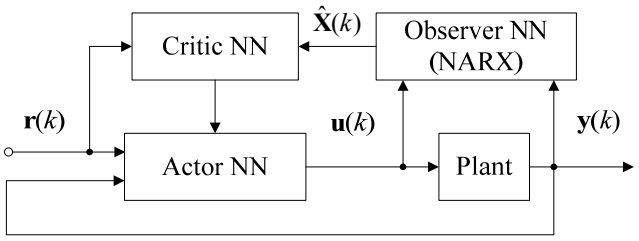

Figure 4-4: Adaptive critic model with observer (MIMO)

4.4 Control system design

In the previous section, the NARX architecture has been integrated into an indirect adaptive control system as a state observer. This control system employs an adaptive critic model using neural networks. Figure 4-5 illustrates the proposed adaptive critic neural control (ACNC) system. It has not yet been decided how the NARX neural network will estimate the states or how all the neural networks will be updated. These two issues are the focus of this section.

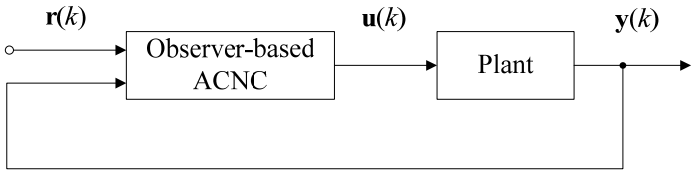

Figure 4-5: Temporary control system design

4.4.1 Lyapunov stability analysis methods

Determining the stability of a linear system consists of simply locating the poles of the system transfer function [3]. Stability analysis of nonlinear dynamic systems is much more involved and no generally applicable method exists. Lyapunov stability analysis methods can be used in the design of adaptive controllers and their parameter update laws for linear and nonlinear systems [8].

The Lyapunov approach makes use of passivity notions and provides a way of studying the stability of complex systems and designing stable controllers. Selecting the correct Lyapunov function is the key to the approach. This function may be difficult to find when considering nonlinear dynamic systems since the function has to depend on time. The function may not even exist at all. Lyapunov proofs are also much more complex for discrete time systems than continuous time systems. Feedback controllers for nonlinear systems of the type (4.6) can be designed using the Lyapunov approach [8].

$$\mathbf{x}(k+1) = f(\mathbf{x}(k)) + g(\mathbf{x}(k))u(k) \tag{4.6}$$

The output feedback control design for nonlinear MIMO discrete-time systems by Jagannathan [8] will be used in this study. It makes use of the adaptive critic model and a neural network as an online state observer. The plant to be controlled is assumed to be of the form (4.5) and consequently, (4.6). The chosen design has a number of useful features for practical applications [8].

First of all, network parameters are tuned online in the chosen control design. Analytical proof of closed-loop stability even in the presence of unknown but bounded disturbances and neural network estimation errors using the Lyapunov approach is the second feature of interest. Finally, the design takes actuator magnitude constraints due to saturation of certain components in the AMB flywheel system into account [8].

The chosen design makes use of Lyapunov stability analysis methods to prove the closed-loop stability of the controller. Conventional discrete-time adaptive control presents a number of challenges. The PE (persistent excitation), LIP (linearity-in-the-parameters) and CE (certainty equivalence) conditions are always assumed. Making sure that these conditions are satisfied is not always possible. By making use of a Lyapunov function which contains the system identification and controller errors, this problem is avoided and the conditions of PE, LIP and CE are relaxed [8].

Using a single Lyapunov function has the potential to guarantee both stable identification and stable tracking at the same time, but the analytical proofs are much more complex. This design ensures closed-loop stability in the presence of neural network estimation errors and unknown but bounded disturbances [8].

Previous work on adaptive critic neural control only ensures error convergence in ideal circumstances which is hardly practical. The boundedness of all closed-loop signals is ensured by using the Lyapunov approach even though many nonlinear effects may be present. A strategic utility function is estimated by the critic neural network which is considered a long-term measure of system performance. By using several neural networks at the same time, multiple nonlinear effects are taken into consideration [8].

4.4.2 Observer design

The aim of this section is to combine (4.5) with a NARX neural network and design an online observer for the plant of the AMB flywheel system. To achieve this aim, (4.5) will be used in the form of (4.7) [8]. The variable n represents the number of system states.

$$\begin{aligned} \mathbf{x}_1(k+1) &= \mathbf{x}_2(k) \\ &\vdots \\ \mathbf{x}_n(k+1) &= \mathbf{f}(\mathbf{x}(k)) + \mathbf{G}(\mathbf{x}(k))\mathbf{u}(k) \\ \mathbf{y}(k) &= \mathbf{x}_1(k) \end{aligned} \tag{4.7}$$

The state matrix of (4.4) has been changed back into a state vector to serve as input to a single-layer neural network. It is not the same as the state vector from (4.1) of a SISO system – it now has the form of (4.8) for a MIMO system [8].

$$\mathbf{x}(k) = [\mathbf{x}_1^{\mathrm{T}}(k), \mathbf{x}_2^{\mathrm{T}}(k), ..., \mathbf{x}_n^{\mathrm{T}}(k)]^{\mathrm{T}} \in R^{nm} \tag{4.8}$$

with each $\mathbf{x}_i(k) \in R^m, i = 1,...,n$ representing a system state and m the number of inputs and the number of outputs. The output $\mathbf{y}(k) \in R^m$ is available at the k^{th} time step but the states $\mathbf{x}_i(k)$, $i = 2,...,n$ are unavailable. Using (4.7) the state observer in (4.9) can be used to estimate the state vector $\mathbf{x}(k)$ [8].

$$\begin{aligned} \hat{\mathbf{x}}_1(k) &= \hat{\mathbf{x}}_2(k-1) \\ &\vdots \\ \mathbf{x}_n(k) &= \hat{\mathbf{W}}_1^{\mathrm{T}}(k-1)\,\varphi_1(\hat{\mathbf{V}}_1^{\mathrm{T}}\hat{\mathbf{z}}_1(k-1)) \end{aligned} \tag{4.9}$$

with $\hat{\mathbf{z}}_1(k-1) = [\hat{\mathbf{x}}_1^{\mathrm{T}}(k-1), ..., \hat{\mathbf{x}}_n^{\mathrm{T}}(k-1), \mathbf{u}^{\mathrm{T}}(k-1)]^{\mathrm{T}} \in R^{(n+1)m}$ the input vector at time instant $(k-1)$ and each $\hat{\mathbf{x}}_i(k) \in R^m$ the estimate of $\mathbf{x}_i(k) \in R^m$, $i = 1,...,n$. The variable φ_1 refers to the activation function of the observer throughout this design. The hidden layer weight matrix of the observer is defined in (4.10).

$$\hat{\mathbf{V}}_1^{\mathrm{T}}(k-1) \in R^{(n+1)m \times n_1} \tag{4.10}$$

with n_1 the number of hidden layer neurons used by the observer.

The output layer weight matrix of the observer is defined in as:

$$\hat{\mathbf{W}}_1^{\mathrm{T}}(k-1) \in R^{n_1 \times m} \tag{4.11}$$

The question still remains as to how this observer design can be used with a NARX neural network. Consider figure 4-6 illustrating a NARX network using the observer design.

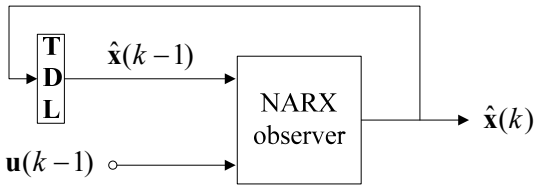

Figure 4-6: NARX observer

For a neural network to have a NARX architecture, the current output should depend on previous values of the output as well as previous values of the input [8]. The figure clearly shows that this is the case for the chosen observer design where the output consists of state information and the input is simply the control signal at time instant $(k-1)$.

4.4.3 Actor design

The aim of this section is to choose a design for the adaptive critic controller such that it (1) adheres to magnitude constraints due to saturation; (2) all the signals in the closed-loop system are UUB (uniformly ultimately bounded); (3) the state $\mathbf{x}(k)$ follows a reference trajectory $\mathbf{r}_v(k)$; and (4) a long-term system performance index is minimized [8]. Note that the reference trajectory $\mathbf{r}_v(k)$ has the same form as the state vector $\mathbf{x}(k)$ as given in (4.12).

$$\mathbf{r}_v(k) = [\mathbf{r}^{\mathrm{T}}(k),...,\mathbf{r}^{\mathrm{T}}(k+n-1)]^{\mathrm{T}} \in R^{nm} \tag{4.12}$$

The tracking error between the actual output and the reference is defined as:

$$\mathbf{e}_i(k) = \mathbf{x}_i(k) - \mathbf{r}_v(k+i-1),\ i = 1,...,n \tag{4.13}$$

Since all the system states are not available at the k^{th} time instant, the modified tracking error is used as an estimate of the tracking error in (4.14).

$$\hat{\mathbf{e}}_i(k) = \hat{\mathbf{x}}_i(k) - \mathbf{r}_v(k+i-1),\ i = 1,...,n \tag{4.14}$$

The control signal produced by the actor network is given by the following equation:

$$\mathbf{u}(k) = \begin{cases} \mathbf{v}(k), & \text{if } \|\mathbf{v}(k)\| \le u_{\max} \\ u_{\max}\,\text{sgn}(\mathbf{v}(k)), & \text{if } \|\mathbf{v}(k)\| > u_{\max} \end{cases} \tag{4.15}$$

where $u_{\max}$ is the upper limit defined by the actuator of the AMB flywheel system. The Frobenius norm $\|\cdot\|$ is used throughout this design. The auxiliary control signal $\mathbf{v}(k)$ is given by:

$$\mathbf{v}(k) = \hat{\mathbf{W}}_2^{\mathrm{T}}(k)\varphi_2(\hat{\mathbf{V}}_2^{\mathrm{T}}\hat{\mathbf{s}}(k)) = \hat{\mathbf{W}}_2^{\mathrm{T}}(k)\varphi_2(\hat{\mathbf{s}}(k)) \tag{4.16}$$

where $\hat{\mathbf{s}}(k)$ is called the actor neural network's input. This input is given in (4.17).

$$\hat{\mathbf{s}}(k) = [\hat{\mathbf{x}}_1{}^{\mathrm{T}}(k),\ldots,\hat{\mathbf{x}}_n{}^{\mathrm{T}}(k),\hat{\mathbf{e}}_n^{\mathrm{T}}(k)]^{\mathrm{T}} \in R^{(n+1)m} \tag{4.17}$$

The variable φ_2 refers to the activation function of the actor throughout this design. The hidden layer weight matrix of the actor is defined in (4.18).

$$\hat{\mathbf{V}}_2^{\mathrm{T}}(k) \in R^{(n+1)m\times n_2} \tag{4.18}$$

with n_2 the number of hidden layer neurons used by the actor. The output layer weight matrix of the actor is defined as:

$$\hat{\mathbf{W}}_2^{\mathrm{T}}(k) \in R^{n_2\times m} \tag{4.19}$$

Figure 4-7 illustrates the actor neural network according to this design. It can clearly be seen that the current value of the output depends only on current values of the inputs. A simple feed-forward neural network, the multi-layer perceptron, with a single hidden layer is used as the actor network in this design [8].

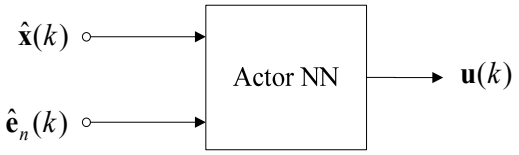

Figure 4-7: Actor neural network

4.4.4 Critic design

The critic neural network signal is given simply by (4.20).

$$\hat{\mathbf{Q}}(k) = \hat{\mathbf{W}}_3^{\mathrm{T}}(k)\varphi_3(\hat{\mathbf{V}}_3^{\mathrm{T}}\hat{\mathbf{x}}(k)) \tag{4.20}$$

where $\hat{\mathbf{Q}}(k) \in R^m$ is the estimate of the strategic utility function $\mathbf{Q}(k) \in R^m$ at time instant k. The strategic utility function is actually a sum of future performance indices and is thus considered an indication of the system's long-term performance [8]. The variable φ_3 refers to the activation function of the critic throughout this design. The hidden layer weight matrix of the critic is defined in (4.21).

$$\hat{\mathbf{V}}_3^{\mathrm{T}}(k) \in R^{nm \times n_3} \tag{4.21}$$

with n_3 the number of hidden layer neurons used by the critic. The output layer weight matrix of the critic is defined as:

$$\hat{\mathbf{W}}_3^{\mathrm{T}}(k) \in R^{n_3 \times m} \tag{4.22}$$

Figure 4-8 illustrates the critic neural network according to the chosen design. It can clearly be seen that the current value of the output depends only on the current value of the input. A simple feed-forward neural network, the multi-layer perceptron, with a single hidden layer is used as the critic network in this design [8].

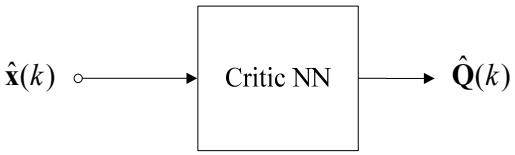

Figure 4-8: Critic neural network

4.4.5 Observer-based ACNC

The observer-based adaptive critic neural controller (ACNC) from figure 4-5 simply consists of the observer, actor and critic neural networks from the previous sections as illustrated in figure 4-9. The observer-based ACNC is expanded to show its three different components and their respective inputs and outputs. A clear picture of the proposed design can now be formed.

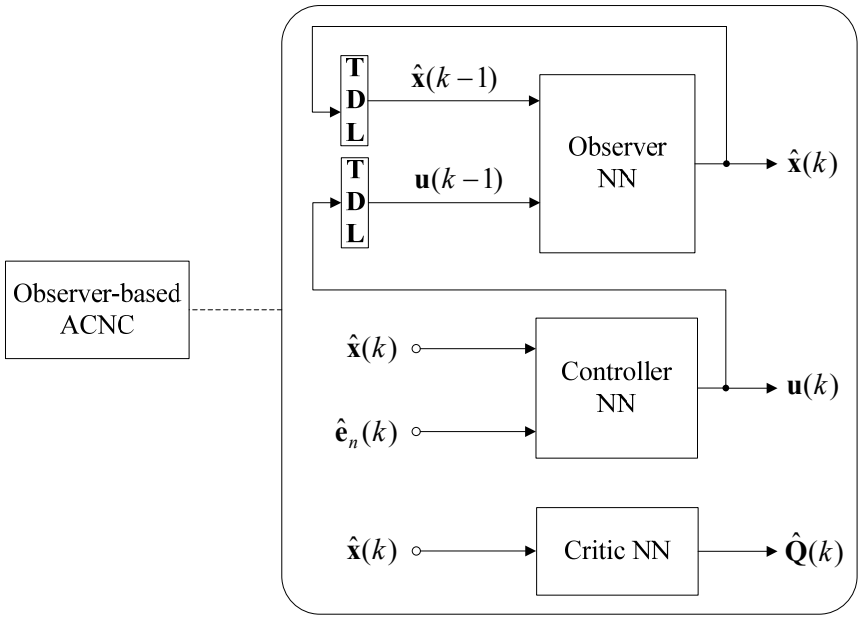

Figure 4-9: Observer-based ACNC

4.4.6 Adaptive laws

The adaptive laws are derived during the Lyapunov stability proofs by Jagannathan [8] which ensure stable update algorithms for the neural network parameters. Note that these update algorithms are used online and no offline training phase is required even though it is performed in many adaptive critic designs. The adaptive law for the observer is given by [8]:

$$\hat{\mathbf{W}}_1(k+1) = \hat{\mathbf{W}}_1(k) - \alpha_1 \varphi_1(\hat{\mathbf{z}}_1(k))(\hat{\mathbf{W}}_1^T(k)\varphi_1(\hat{\mathbf{z}}_1(k)) + \mathbf{j}_1\tilde{\mathbf{x}}_1(k))^T \tag{4.23}$$

The observer input $\hat{\mathbf{z}}_1(k)$ is defined as:

$$\hat{\mathbf{z}}_1(k) = [\hat{\mathbf{x}}_1^{\mathrm{T}}(k),...,\hat{\mathbf{x}}_n^{\mathrm{T}}(k),\mathbf{u}^{\mathrm{T}}(k)]^{\mathrm{T}} \in R^{(n+1)m} \tag{4.24}$$

The observer estimation error $\tilde{\mathbf{x}}_1(k)$ is given by:

$$\tilde{\mathbf{x}}_1(k) = \hat{\mathbf{x}}_1(k) - \mathbf{x}_1(k) \in R^m \tag{4.25}$$

The variable $\mathbf{j}_1(k) \in R^{m\times m}$ is a design matrix and $\alpha_1 \in R^+$ is the learning rate of the observer. Equation (4.26) defines the adaptive law for the actor [8].

$$\hat{\mathbf{W}}_2(k+1) = \hat{\mathbf{W}}_2(k) - \alpha_2 \varphi_2(\hat{\mathbf{s}}(k))(\hat{\mathbf{e}}_n(k+1) - \mathbf{j}_2\hat{\mathbf{e}}_n(k) + \hat{\mathbf{Q}}(k))^T \tag{4.26}$$

The modified tracking error at time instant $(k+1)$ is given by:

$$\hat{\mathbf{e}}_i(k+1) = \hat{\mathbf{x}}_i(k+1) - \mathbf{r}_v(k+i),\ i = 1,...,n \tag{4.27}$$

The variable $\mathbf{j}_2(k) \in R^{m \times m}$ is a design matrix and $\alpha_2 \in R^+$ is the learning rate of the actor. The adaptive law for the critic is given by [8]:

$$\hat{\mathbf{W}}_3(k+1) = \hat{\mathbf{W}}_3(k) - \alpha_3 \varphi_3(\hat{\mathbf{x}}(k))(\hat{\mathbf{Q}}(k) + \alpha^{N+1}\mathbf{p}(k) - \alpha\hat{\mathbf{Q}}(k-1))^T \tag{4.28}$$

The variable α is a design parameter and $\alpha_3 \in R^+$ is the learning rate of the critic. The final time is indicated by N in seconds. The utility function $\mathbf{p}(k)$ is given by:

$$p_i(k) = \begin{cases} 0, & \text{if abs}(\hat{\mathbf{e}}_n^i(k)) \le c \\ 1, & \text{otherwise} \end{cases} \qquad i = 1,...,m \tag{4.29}$$

The variable $c \in R^+$ is a pre-defined threshold for the modified tracking error. Table 4-1 summarizes the suggested ranges for the values of each of the design parameters.

Table 4-1: Suggested ranges for design parameters

Parameter	Suggested range
α_1	$0 < \alpha_1 < \frac{1}{n_1}$
α_2	$0 < \alpha_2 < \frac{1}{n_2}$
α_3	$0 < \alpha_3 < \frac{1}{n_3}$
α	$0 < \alpha < \frac{\sqrt{2}}{2} \approx 0.7$

Unfortunately, there is no suggestion for selecting the number of neurons of each network. No more information is given about the values of $\mathbf{j}_1$ and $\mathbf{j}_2$ either. It is quite clear that a number of control design parameters require optimization. As pointed out in the previous chapter, a global optimization method can be used to avoid the local minima of an error surface and significantly decrease design effort. Genetic algorithms will be used to optimize the design parameters of the ACNC system.

4.5 Derivative-free optimization

The most significant challenge when using gradient-based optimization techniques is to find the global minimum of the error surface. These deterministic methods guarantee convergence to the nearest local minimum when used properly, but many local minima may exist. This means that the nearest local minimum for different initial positions may not be the same minimum and consequently the algorithm may produce different results [7].

Choosing the initial positions well is clearly an important step but also a difficult one because the choice is almost never obvious. To avoid using the same initial positions and search in a space which covers a variety of choices, some type of random initialization method needs to be implemented. Apart from a proper initialization method, another challenge might be the calculation of the gradient in some deterministic method. It is often a time-consuming and difficult calculation [80].

Derivative-free optimization methods such as the genetic algorithm have been used to circumvent this problem though it should be mentioned that these stochastic methods also have their own set of challenges. Stochastic derivative-free optimization methods have to converge to minima in the error function without any gradient information and consequently require many function evaluations. This implies that these derivative-free methods sometimes require more computations than the deterministic type derivative-based methods to produce satisfactory results [7].

Derivative-free methods have some desirable characteristics and these include [7]:

- No derivative calculations.
- The principles used are of an intuitive nature.
- The objective function does not need to be differentiable and can be as complex as required by the application under consideration.
- The methods are considered global optimization methods because of their random nature.

4.5.1 Genetic algorithms

Genetic algorithms (GAs) are based on the ideas of natural selection and evolution as proposed by Darwin and are one of the branches of evolutionary programming. They provide a systematic way of randomly searching in a population of solutions for the best-performing individual. More efficient search techniques such as GAs are used when the space of candidate solutions is simply too large for an exhaustive search. It should be noted though that the final solution is usually less than optimum. GAs are considered general purpose optimization tools and offer freedom and flexibility in complex optimization problems because of their derivative-freeness [7].

GAs also offer the following useful features [7]:

- Implementation on parallel-processing machines is possible because GAs are parallel-search procedures. Consequently, the time to converge to a final solution can be drastically reduced.
- Both continuous and discrete optimization problems present no challenge for GAs.
- The ever-present problem of getting trapped in a local minimum can be avoided because GAs have a random nature.
- GAs can be used for both structure identification and parameter estimation in complicated models such as neural networks and fuzzy systems.

4.5.2 Optimization process

The following steps describe a simple genetic algorithm used for optimization [7]:

1. Randomly initialize a population of solutions and calculate the fitness of each candidate solution or individual.
2. Use the best individuals (selection) in the population to produce the next generation:
    a. Apply elitism.
    b. Apply crossover.
    c. Apply mutation.
3. Evaluate the fitness of each individual in the new generation.
4. Repeat steps 2 and 3 until a stopping condition is satisfied.

The concepts mentioned in the steps above will each be given a brief overview but first it is important to note the meaning of the following terms [7]:

- Each individual or chromosome in the population is just a value or a set of values for the parameter or parameters which have to be optimized.
- Each chromosome has a fitness value which is usually the inverse of the chosen error function evaluated using the specific chromosome.
- The population is repeatedly evolved or modified to better overall fitness or reduce total error.
- The new generation is created (reproduction) using crossover and mutation schemes.
- The chromosomes in the current generation with the best fitness values are used as parents in the next generation to produce new offspring or children.

4.5.3 Initialization and selection

Before the initial population can be generated, the points in the parameter space have to be transformed into chromosome form i.e. into sets of strings or genes. This is called encoding. After the population has been generated and each individual's fitness evaluated, selection has to be performed. Selection consists of choosing the parents which will be used in reproduction [80].

The parents with the best fitness values are selected to ensure that as the population evolves, chromosomes with above-average fitness will survive. A certain number of the best chromosomes from the current generation are kept unchanged into the next generation. This "survival of the fittest" concept is called elitism [80].

4.5.4 Crossover and mutation

Crossover and mutation entails the generation of new chromosomes i.e. children from parent chromosomes. Crossover ensures that these children will have the best genes from the previous generation and consequently exploits the current population's potential. Sometimes the population does not contain the necessary information to produce an acceptable solution regardless of how extensively genes are combined and recombined. The genetic algorithm is said to stagnate when continuous reproduction using crossover does not deliver an acceptable solution [80].

To avoid stagnation, the concept of mutation is employed. Mutation ensures that random new chromosomes, impossible using crossover, can be generated when necessary. The possibility of large mutations is kept small to ensure that good chromosomes from the crossover step do not go lost [80].

4.6 System identification loop

The various steps of the system identification loop as presented in previous chapters are addressed in this section. Experiment design and obtaining data are addressed in the next chapter since these steps are concerned with implementation. The model sets will consist of NARX networks in the form of online observers with the number of state variables n varying from set to set. The condition of fit for the observer will be as small an estimation error as possible. Each neural model will be calculated using the weight update algorithms outlined in the adaptive control design of Jagannathan [8].

The best observer from each model set will be determined using a genetic algorithm. A model will be validated only if it can provide the corresponding actor and critic with enough state information to ensure stable control. If the model in the form of an online observer fails validation, it is deemed inadequate and the steps of the system identification loop are revised. Revision will occur by increasing the number of state variables n used by the observer.

4.7 Performance verification

The performance of both the new control system and the original PD control system has to be evaluated. This section should not be confused with the model validation step of the system identification loop. Performance verification will only be performed on the final adaptive controller and will enable a comparison between the new and original controllers. Performance verification consists of three parts: step response, robustness analysis and stability analysis.

4.7.1 Step response

The first test will consist of a step disturbance at the reference input to determine the step response of each system. Standard performance measures defined in terms of the step response of a system include: percentage overshoot (*P.O.*) and settling time (T_s). These two measures provide an indication of the similarity between the actual response and the step input. The settling time is defined as the time taken by the system to settle within 2% of the final value. For a second-order system, the percentage overshoot and settling time are given in (4.30) and (4.31) respectively [3].

$$P.O. = \frac{M_p - fv}{fv} \times 100\% \tag{4.30}$$

where M_p is the peak value of the step response and *fv* is the final value of the step response.

$$T_s = 4\tau = \frac{4}{\zeta\omega_n} \tag{4.31}$$

The variables τ, ζ and ω_n represent the time constant, damping ratio and natural frequency of the system respectively.

4.7.2 Robustness analysis

The aim of this section is to determine how the system responds to parameter variations and if it remains stable despite changes in its environment. This measure of robustness will consist of determining how sensitive the system is to a chirp input disturbance. Evaluation of the step response and the sensitivity function ensures that both time and frequency domain analysis are performed.

The sensitivity function in accordance with ISO CD 14839-3 of the AMB control system under consideration in one axis direction is defined as the ratio of two signals in the frequency domain – the error signal, $E(s)$, and the reference input signal, $R(s)$, as given by [78]:

$$G_s(s) = 20\log\left(\frac{E(s)}{R(s)}\right) \text{ in dB} \tag{4.32}$$

The data required for sensitivity is simply the signals $r(k)$ and $e(k)$ as specified in figure 4-10. The measured data is used to determine the sensitivity of the new control system and compare it with the sensitivity of the original PD control system.

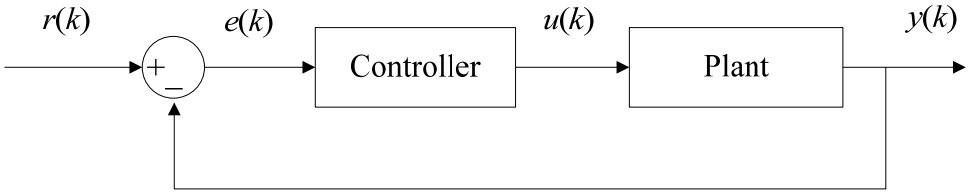

Figure 4-10: Measurement of sensitivity data

The reference input to the system should be a sinusoidal disturbance with varying frequency. The sensitivity data should be measured at rotor standstill and/or nominal operating speed but over the maximum frequency range. The maximum frequency range, $f_{\max}$, starts at 0 Hz and ends at the maximum of either three times the rated speed or 2 kHz [78].

The ISO CD 14839-3 standard also specifies that the sensitivity of each control loop should be determined. The overall system sensitivity is taken as the worst of all the measured sensitivities [78]. Since the measured data is in the time domain, it is transformed to the frequency domain using the Fourier transform. This frequency domain data is used in (4.32) to evaluate the sensitivity function. The sensitivity function can be visualized using a frequency graph and by examining the peaks, different conclusions can be drawn about the system's robustness and performance.

4.7.3 Stability analysis

The stability of each control system will be evaluated by calculating the gain and phase margin of the system. These margins are based on the Nyquist stability criterion and provide an indication of the system's stability in the presence of disturbances. The gain margin g_m is defined in (4.33) as the minimum amount by which the open-loop gain can increase before the closed-loop system becomes unstable [23].

$$g_m = \frac{1}{\left|L(i\omega_{pc})\right|} \tag{4.33}$$

The variable ω_{pc} is the phase crossover frequency. This frequency is simply the minimum frequency where the phase of the loop transfer function $L(s)$ is 180° (or -180°). The phase margin is defined in (4.34) as the amount of phase lag needed before the closed-loop system becomes unstable [23].

$$\varphi_m = \pi + \arg L(i\omega_{gc}) \tag{4.34}$$

The variable ω_{gc} represents the gain crossover frequency. This frequency is simply the minimum frequency where the loop transfer function $L(s)$ has a magnitude of 1 [23]. A gain margin larger than 6 dB and a phase margin larger than 30° will indicate an exceptionally robust system.

4.8 Conclusions

The observer-based ACNC makes use of online system identification to obtain the state information required by the adaptive critic controller. The NARX observer uses only input-output data from the plant of the AMB flywheel system to achieve online system identification. The stability of the proposed design is proven analytically by Jagannathan [8] using Lyapunov stability analysis methods.

Finally, the design is applicable to the control of multivariable, nonlinear, dynamic and inherently unstable systems [8]. The proposed adaptive control design clearly conforms to all the requirements of a controller for the AMB flywheel system. The steps of the system identification loop were also identified in the application of the chosen adaptive control design.

The stability and performance measures presented in this chapter will be used on the linear PD control system as well as the non-linear adaptive control system. Though this may seem incorrect, it is common practice in AMB literature [13]. The next chapter consists of implementation of the observer-based ACNC in the AMB flywheel system and its performance verification.

Chapter 5: Simulation results

This chapter contains the implementation and simulation results of the chosen adaptive controller. The observer optimization results are presented first. Its response at various rotor speeds is shown and compared with the original system output. The step response of the new control system is compared with the step response of the original PD control system. Performance verification continues with robustness and stability analysis of each control system.

5.1 Implementation

This section addresses the steps of experiment design and obtaining data in the system identification loop. The chosen control design is implemented in **MATLAB®** making use of built-in functions as well as the *Genetic Algorithm Toolbox*™. The neural networks are coded in level-2 *m*-file s-functions in **MATLAB®** and not using the *Neural Network Toolbox*™ as in the previous chapter. The *Neural Network Toolbox*™ simply does not provide all the functionality required by the chosen design. The s-functions are used in **SIMULINK®** to complete a simulation model of the new control system. Since the original AMB flywheel system uses a sampling time of 100 μs, the new adaptive control system in **SIMULINK®** also uses a sampling time of 100 μs.

5.1.1 Optimization using genetic algorithms

Since the observer network uses a different error signal than the critic and actor networks, it can be optimized separately using a genetic algorithm. The estimation error of the observer, $\tilde{\mathbf{x}}_1(k)$, is used in the genetic algorithm as the function to be minimized. Figure 5-1 shows how the observer is connected to the original system and simulated in order to determine an optimal set of values for the parameters, $\mathbf{j}_1$ and n_1, of the observer. Notice that the observer optimization can be performed in an open-loop manner since it does not affect the closed-loop system. Performing the optimization offline at rotor standstill saves time but to ensure the observer's accuracy, it is still tested in the closed-loop system at various rotor speeds.

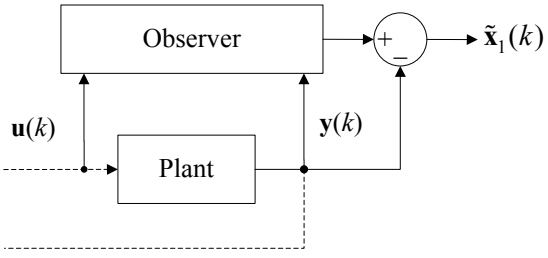

Figure 5-1: Optimization of observer parameters

The observer optimization is achieved by injecting multi-sine excitation signals with frequencies ranging from 0 Hz to 133 Hz as described in the previous chapter to ensure persistent excitation (PE). Remember that PE is not required for the complete design – it is just used to determine optimal values for the parameters of the observer. The resulting input and output signals are the same as those used in the previous chapter for offline system identification.

The difference between the observer in this chapter and the observer in the offline system identification experiment of the previous chapter can be seen in the network parameters. The offline observer's weights were trained up to certain point and then the network was deemed sufficient for use in the closed-loop system. The design parameters of the online observer in this chapter are optimized offline but the network weights are always re-initialized and continuously trained when used in the closed-loop system. The online observer is termed a self-tuning network since adaptation never stops which makes it ideal for time-varying plants [5].

The next step is the optimization of the critic and actor parameters. The values for the observer parameters are kept constant since they have already been optimized. The tracking error, $\mathbf{e}(k)$, is used in the form of a performance index as the function to be minimized by the genetic algorithm. A performance index is simply a quantitative measure of a system's performance. This index should be minimized to obtain an optimal control system. The ITAE (integral of time multiplied by absolute error) performance index reduces the contribution of large transient-state errors and emphasizes errors in the steady-state response. The ITAE performance index is defined in (5.1) [3].

$$\text{ITAE} = \int_0^T t\left|e(t)\right|dt \tag{5.1}$$

The critic and actor parameters to be optimized are $\mathbf{j}_2$, n_2, n_3 and α. Note that the learning rate of each network is chosen as shown in (5.2). The tracking error threshold is selected as $c = 1\times10^{-20}$ m since position signals can be as small as 1×10^{-35} m.

$$\alpha_i = \frac{1}{n_i}, \quad i = 1,...,3 \tag{5.2}$$

The final parameter which requires selection is the number of system states, n. Equation (5.3) describes the simple relationship between the state space system order n_{SP} and the number of mechanical degrees of freedom (DOF). Each DOF has two associated state variables, usually position and velocity [1].

$$n_{\text{SP}} = 2n_{\text{DOF}} \tag{5.3}$$

Since the AMB flywheel system has 4 DOF, it requires at least 8 state variables for successful system identification. The state vector of the observer-based ACNC in (4.8) now becomes:

$$\mathbf{x}(k) = [\mathbf{x}_1^{\mathrm{T}}(k), \mathbf{x}_2^{\mathrm{T}}(k)]^{\mathrm{T}} \in R^8 \tag{5.4}$$

The variables $\mathbf{x}_1(k)$ and $\mathbf{x}_2(k)$ are defined as:

$$\mathbf{x}_1(k) = \begin{bmatrix} x_{11}(k) \\ x_{12}(k) \\ x_{13}(k) \\ x_{14}(k) \end{bmatrix} \text{ and } \mathbf{x}_2(k) = \begin{bmatrix} \dot{x}_{21}(k) \\ \dot{x}_{22}(k) \\ \dot{x}_{23}(k) \\ \dot{x}_{24}(k) \end{bmatrix} \tag{5.5}$$

Equation (5.5) clearly indicates that $n = 2$ in the ACNC design with each DOF represented by two state variables, x_{ij} and $\dot{x}_{ij}$ for $i = 1,2$ and $j = 1,...,4$. Note that (5.3) is only an indication of the minimum number of state variables required – more can be used if necessary.

The number of inputs and the number of outputs of the system are both $m = 4$. The genetic algorithm for both optimization processes uses an initial population of 40 real-coded chromosomes. The algorithm then continues optimization until the number of 20 generations is reached. The best chromosome of the final population is selected as the best set of parameters.

The population ranges for each of the parameters are selected as shown in table 5-1. These ranges were selected using a trial-and-error approach. Any parameter values outside these ranges cause unstable updates to the network weights and consequently, the closed-loop system becomes unstable. Note that the variable $\mathbf{I} \in R^{m \times m}$ represents the identity matrix.

Table 5-1: Selected ranges for parameters

Parameter	Selected range
$\mathbf{j}_1$	$\mathbf{j}_1 = j_1\mathbf{I}$ where $0 < j_1 < 50$
$\mathbf{j}_2$	$\mathbf{j}_2 = j_2\mathbf{I}$ where $0 < j_2 < 20$
n_1, n_2, n_3	$4 < n_i < 100$ for $i = 1,2,3$
α	$0.001 < \alpha < 0.1$

5.1.2 Initialization of network weights

According to the design in Jagannathan [8], all of the output layer weights are initialized to zero and trained online. The hidden layer weights are initialized randomly to a specified range and kept constant. It is believed that the observer error can be made arbitrarily small by randomly initializing the hidden layer weights and choosing n_1 sufficiently large [8]. The range to which the output layer weights are initialized is selected by the genetic algorithm along with all the other parameters.

5.2 Observer verification

The observer output obtained using an optimal parameter set as determined by the genetic algorithm is under consideration. This open-loop output is compared with the original plant output of the AMB flywheel system at 0 r/min with PE in figure 5-2. The observer with $n = 2$ used $j_1 = 20$ and $n_1 = 40$. The hidden layer weights were initialized to [-0.0001, 0.0001]. The final *mse* obtained by the observer with $n = 2$ was 7.446×10^{-12}.

Notice in figure 5-2 how the observer output is almost identical to the original output. This is to be expected since the condition of PE is satisfied and the observer is only being tested in an open-loop configuration. Closed-loop tests will soon follow and provide a better method of verifying the observer's performance.

Since the genetic algorithm initializes random networks to enable a global search, it does not allow for repeated initialization of the same network. This implies that the performance metrics used in the previous chapter – accuracy, timeliness, precision and repeatability – cannot be used here since they require initialization of the same network again and again [77].

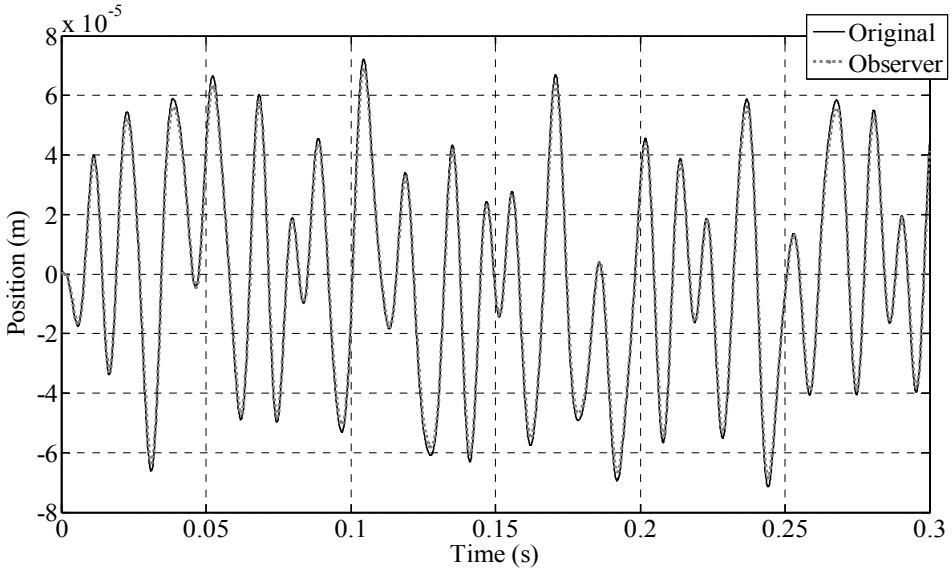

Figure 5-2: AMB 1(x) observer output at 0 r/min and $n = 2$ with PE

The NARX observer with $n = 2$ is simulated in the original closed-loop system at each of the rigid mode frequencies as determined in the previous chapter: 2,200 r/min (35 Hz) and 4,000 r/min (65 Hz). Figure 5-3 and figure 5-4 show the observer output at each of these rotor speeds without PE. A zero reference input was applied to each AMB axis direction. Results for the other AMB axis directions are very similar.

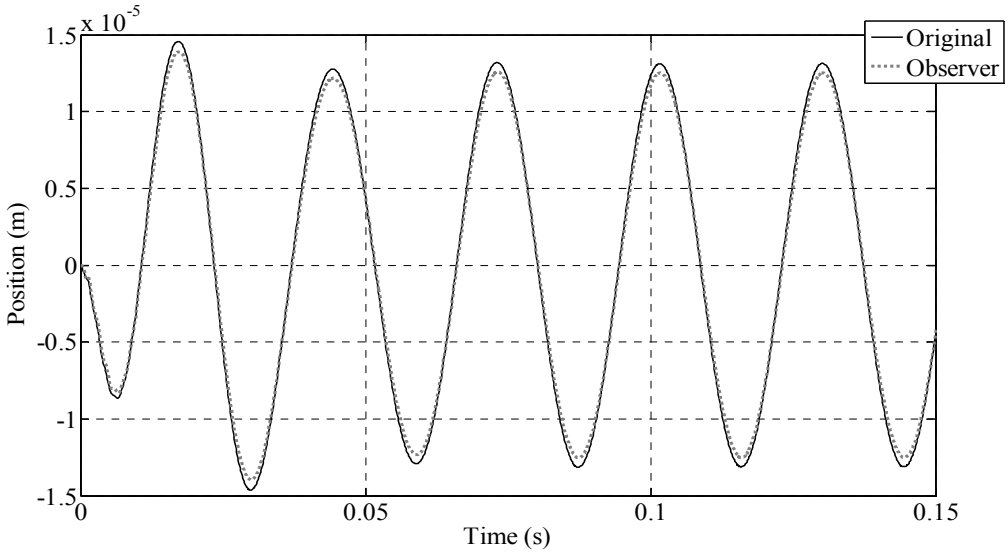

Figure 5-3: AMB 1(x) observer output at 2,200 r/min and $n = 2$ without PE

Notice the difference in the time scale of the plant dynamics when using a higher rotor speed as seen by comparing figure 5-3 and figure 5-4. Figure 5-5 shows the observer output with $n = 2$ at the current maximum operating speed of 8,300 r/min (133 Hz), also without PE.

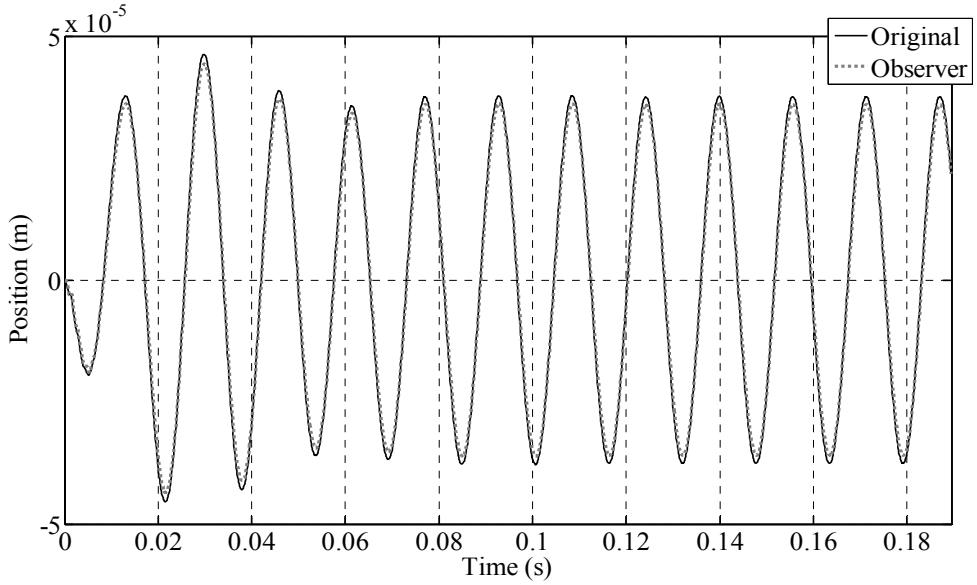

Figure 5-4: AMB 1(x) observer output at 4,000 r/min and $n = 2$ without PE

Figure 5-5 clearly shows that the observer produced a very small estimation error at the current maximum operating speed of 8,300 r/min (133 Hz). Figure 5-6 shows the observer output with $n = 2$ at the rated maximum operating speed of 31,400 r/min (500 Hz).

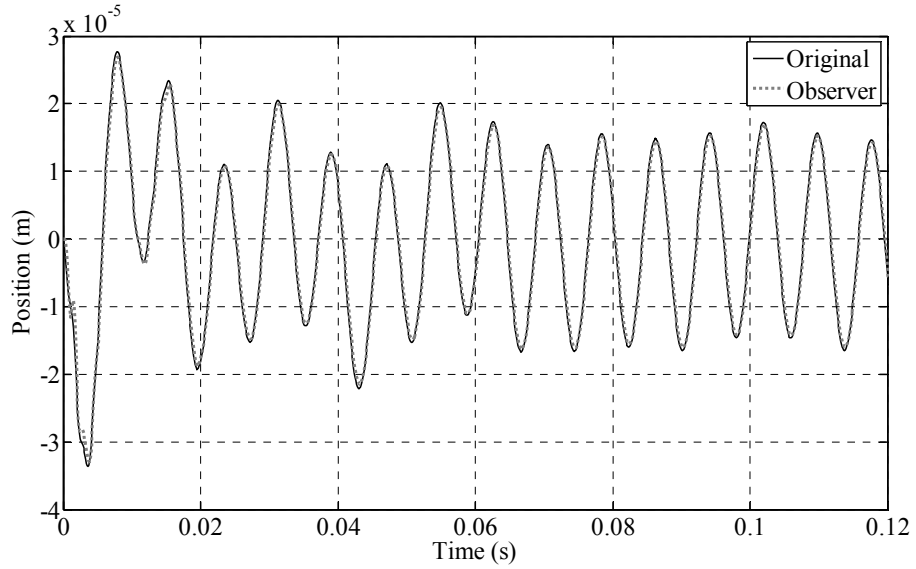

Figure 5-5: AMB 1(x) observer output at 8,300 r/min and $n = 2$ without PE

The NARX observer produced a much larger estimation error at very high rotor speeds as shown in figure 5-6. Notice also that it was only possible to simulate the original control system using decentralized PD control for 0.035 s at a rotor speed of 31,400 r/min before it became unstable. Results for the other AMB axis directions are very similar.

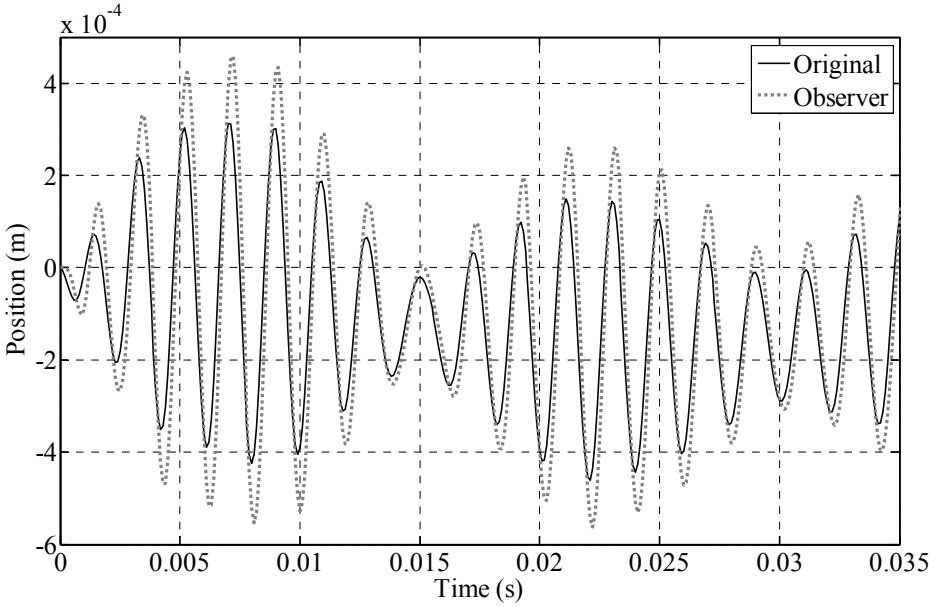

Figure 5-6: AMB 1(x) observer output at 31,400 r/min Hz and $n = 2$ without PE

Recall that the order used by the observer is the same as the order used by the controller. Adaptive controllers with $n < 10$ were found to be inadequate since they could not stabilize the system. The rotor delevitates with lower order adaptive controllers. Consequently, the order of the observer had to be increased until the corresponding controller could provide feedback stabilization. The final adaptive controller uses $n = 20$ and the corresponding observer output with PE at rotor standstill can be seen in figure 5-7. Notice the time delay in the observer output when using a higher order.

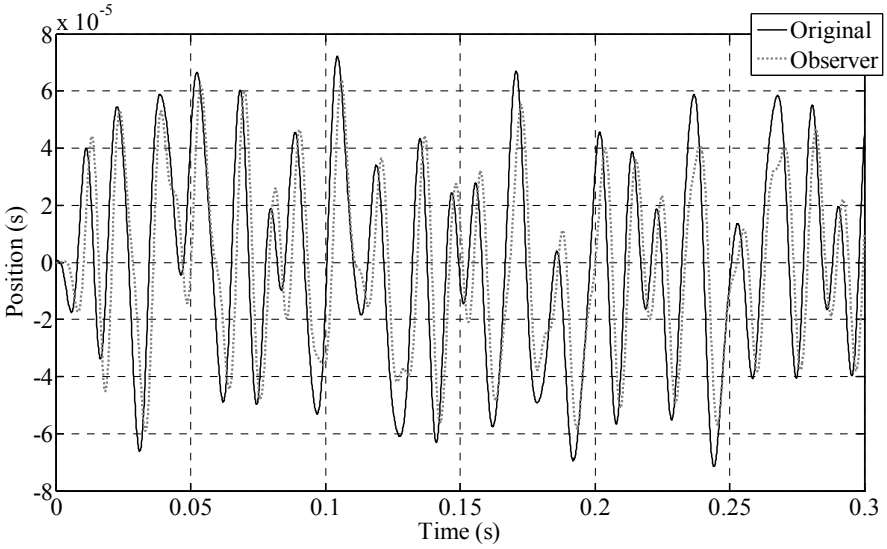

Figure 5-7: AMB 1(x) observer output at 0 r/min and $n = 20$ with PE

Simulating the observer in closed-loop configuration without PE at the current maximum rotor speed of 8,300 r/min (133 Hz) can be seen in figure 5-8. Note that the low order observers do satisfy the condition of fit, namely a very small estimation error, but they fail model validation and are not considered further.

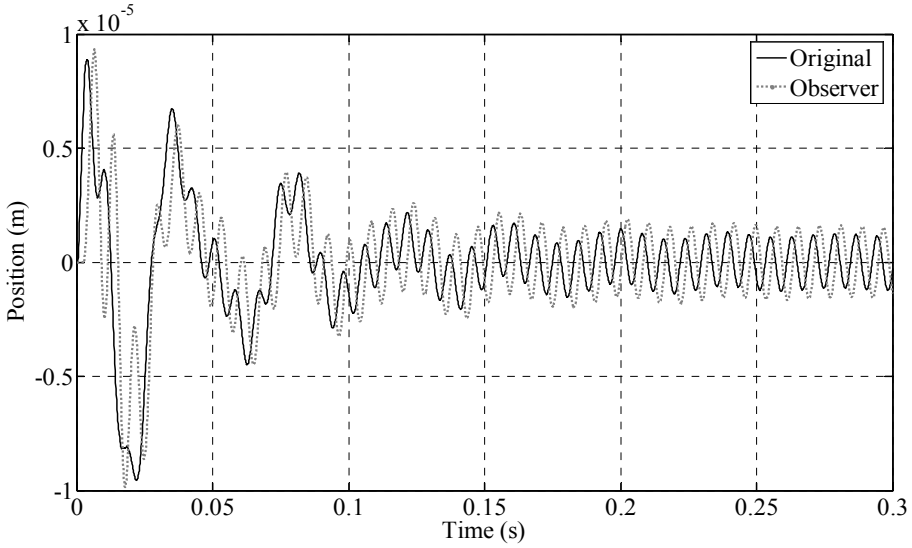

Figure 5-8: AMB 1(x) observer output at 8,300 r/min and $n = 20$ without PE

5.3 Adaptive controller verification

This section presents the adaptive controller results in the form of step response, robustness and stability analysis. The first results are the step responses of three different adaptive controllers. The adaptive controllers use $n = 6$, $n = 12$ and $n = 20$ respectively. Various adaptive controllers with n ranging from 2 as recommended by (5.3) to 30 were tested, but it was found that higher order controllers provided better performance.

The results of the adaptive controllers with $n = 6$ and $n = 12$ are only shown to provide a performance comparison with the adaptive controller using $n = 20$. This is to emphasize the improvement in performance when using more state variables. The step response results are followed by robustness and stability analysis results of the controller with $n = 20$.

Once again, only a few results are presented in this section since there are simply too many. Table 5-2 shows the optimal values as determined by the genetic algorithms for each ACNC. The observer hidden layer weights were initialized to [-0.03, 0.03], the actor hidden layer weights to [-9, 9] and the critic hidden layer weights to [-0.5, 0.5]. All of the output layer weights were initialized to zero.

Table 5-2: Optimal values for ACNC parameters

Parameter	$n = 6$	$n = 12$	$n = 20$
j_1	7.8	9.9	2.1
j_2	1	1	1
n_1	40	100	30
n_2	4	4	4
n_3	126	82	145
N	5	5	5
α	0.01	0.01	0.01

5.3.1 Step response

A step input at 0.1 s with an amplitude of 1 μm is applied to AMB 1(x). A larger amplitude for the step input can be used, but the original PD control system destabilizes at some point. The original PD control system runs longer before destabilizing by using an amplitude of 1 μm for the step input and allows for a better comparison with the ACNC. The response of the original system and the ACNC with $n = 6$ can be seen in figure 5-9.

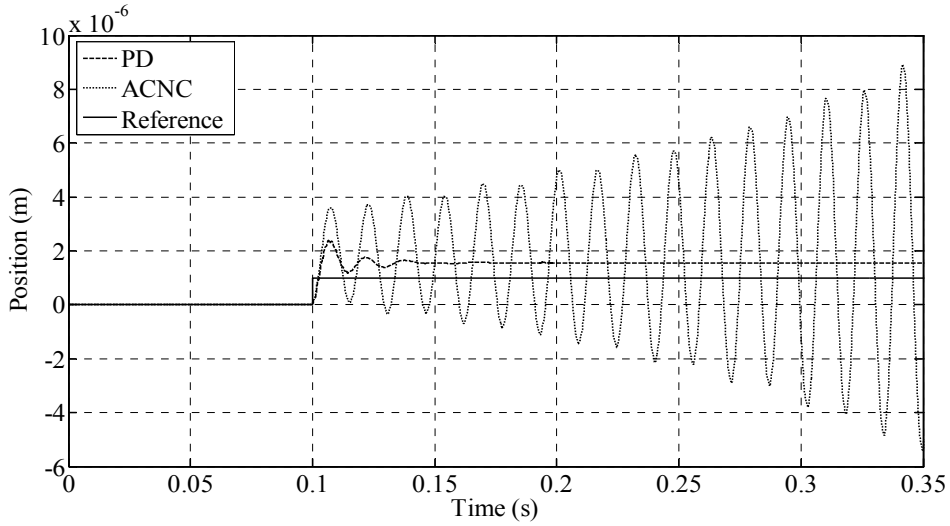

Figure 5-9: Step response of PD controller and ACNC with $n = 6$

Notice that the amplitude of the step response using an ACNC with $n = 6$ only gets larger and larger with time as shown in figure 5-9. The rotor destabilizes at some point and accordingly, the ACNC with $n = 6$ is no longer considered. An adaptive controller with $n = 12$ is compared with the original decentralized PD control system in figure 5-10.

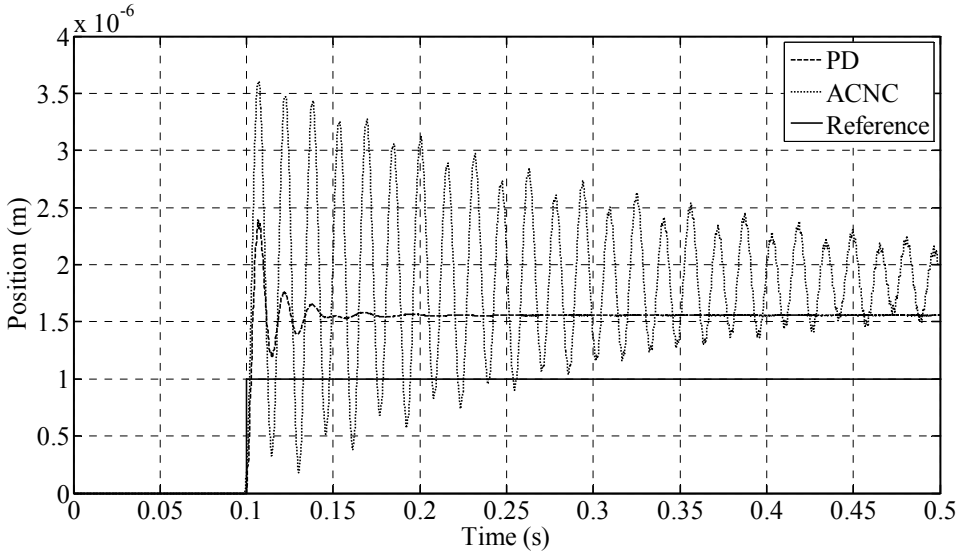

Figure 5-10: Step response of PD controller and ACNC with $n = 12$

When studying figure 5-10, it becomes obvious that the settling time for the ACNC with $n = 12$ is much larger than the original PD control system. This ACNC is also disregarded since it does not provide a level of performance even close to the PD control system's performance. The performance of the ACNC with $n = 20$ will be verified in the following sections. The step response of the final ACNC with $n = 20$ can be seen in figure 5-11.

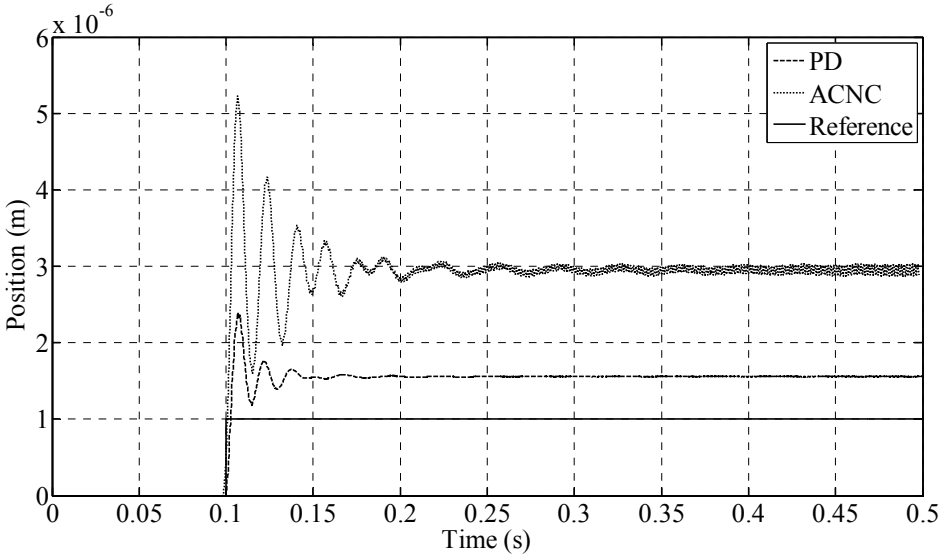

Figure 5-11: Step response of PD controller and ACNC with n = 20

The percentage overshoot (*P.O.*) and settling time (T_s) can be calculated from the step response data in figure 5-11. The original system has a settling time of T_s = 0.14 s and a *P.O.* as determined in the following:

$$P.O. = \frac{M_p - fv}{fv} \times 100\% = \frac{2.4 \times 10^{-6} - 1.6 \times 10^{-6}}{1.6 \times 10^{-6}} \times 100\% = 50\%$$

The adaptive system has a settling time of T_s = 0.24 s and a *P.O.* as determined in the following:

$$P.O. = \frac{M_p - fv}{fv} \times 100\% = \frac{5.2 \times 10^{-6} - 2.8 \times 10^{-6}}{2.8 \times 10^{-6}} \times 100\% = 86\%$$

Analysis of the step response results would clearly indicate that the ACNC does not measure up to the original decentralized PD control system. The advantages of the ACNC can only be seen by allowing the AMB system to run a little longer after the step input has been applied.

Figure 5-12 shows how the rotor destabilizes[17] after 3 s when using decentralized PD control but remains stable using the ACNC with n = 20. The system response using the ACNC with n = 20 is so small, it can almost not be seen in figure 5-12. The reason behind this phenomenon will have to be investigated.

[17] Rotor destabilization refers to the rotor levitating far from its reference point and touching the retainer bearings.

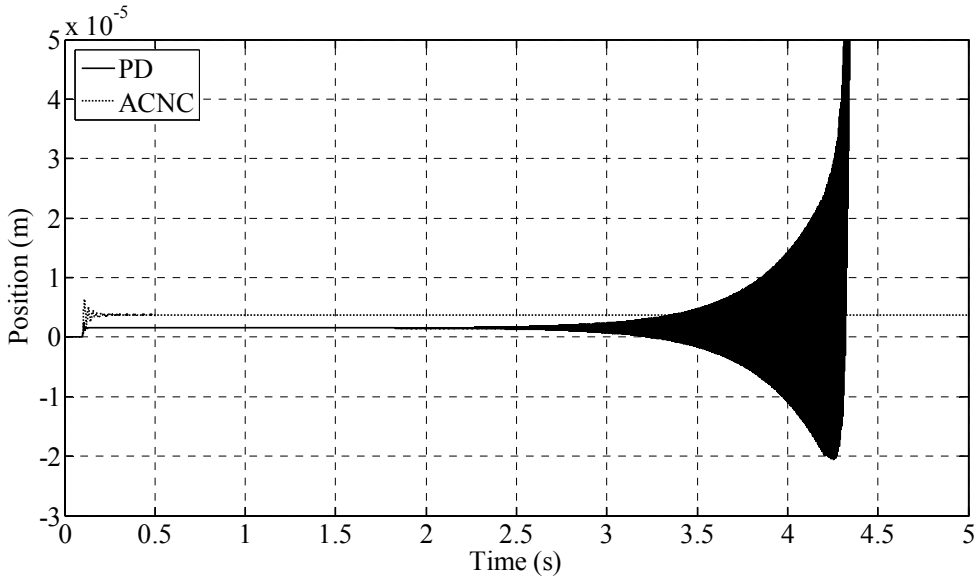

Figure 5-12: Step response of PD controller and ACNC with *n* = 20 for 5 s

5.3.2 Robustness analysis

A chirp disturbance at the reference input with amplitude 0.001 μm ranging from 0 Hz to 2 kHz is applied to AMB 1(x) over a period of 40 s. Once again, the very small input amplitude is necessitated because the rotor destabilizes using decentralized PD control with larger reference input amplitudes.

Figure 5-13 shows the worst-case sensitivity of each control system as well as magnitude limits of 8 dB, 12 dB and 14 dB. These limits indicate different operating zones of an AMB system as defined in the ISO CD 14839-3 standard. These zones are defined as [78]:

- Zone A – the region below 8 dB. Newly commissioned machines usually fall within this operating zone.
- Zone B – the region between 8 dB and 12 dB. Machines in this zone are normally accepted for long-term operation.
- Zone C – the region between 12 dB and 14 dB. Machines in this zone cannot be operated for long periods of time. Corrective action is required.
- Zone D – the region after 14 dB. Machines in this zone will be damaged severely and cannot be operated.

The two control systems, one with decentralized PD control and the other with an ACNC, have differing sensitivities at the critical frequencies identified previously as shown in figure 5-13. The adaptive control system falls within zone A whereas the decentralized PD system falls within zone D.

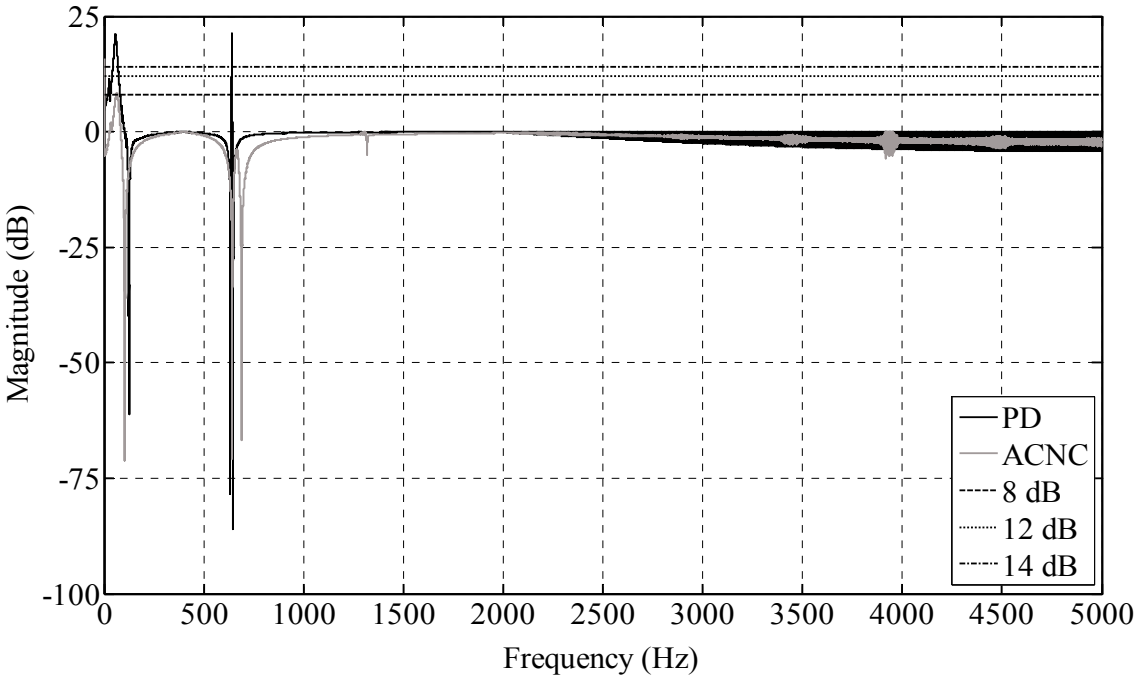

Figure 5-13: Worst-case sensitivity of PD controller and ACNC with small chirp signal

At this point, the adaptive controller has not yet been thoroughly tested for stability against external disturbances. This is clearly visible in figure 5-13 since the sensitivity function of the adaptive control system never goes past zone A (higher than 8 dB). A larger disturbance amplitude is required but this means that the decentralized PD control system cannot be tested again. Figure 5-14 shows the worst-case sensitivity of the observer-based ACNC with a chirp disturbance of amplitude 1 μm.

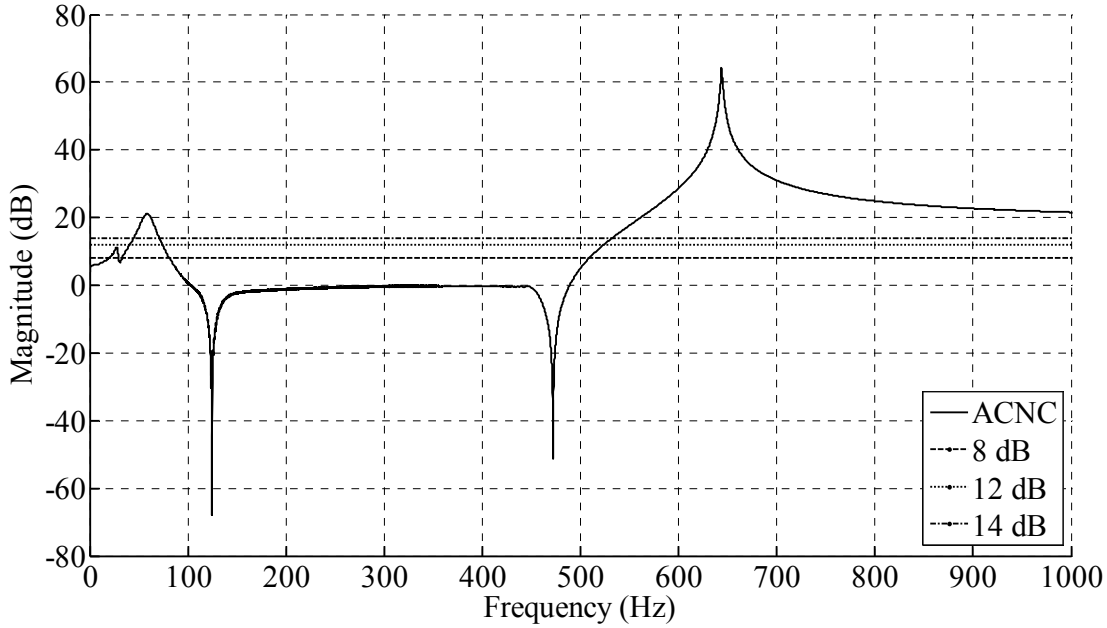

Figure 5-14: Worst-case sensitivity ACNC with large chirp signal

The worst-case sensitivity of the adaptive control system with the larger chirp disturbance applied at the reference input puts the adaptive system in zone D. Note that the sensitivity function is not calculated across 2 kHz this time. The NARX observer is known to be relatively accurate up to an operating speed of 31,400 r/min (500 Hz) and consequently, the larger chirp disturbance ranges from 0 Hz only to 500 Hz. The chirp signal is applied to AMB 1(x) over a period of 10 s.

5.3.3 Stability analysis

The gain and phase margins are determined using the data obtained during the first robustness test in the previous section. This is done to enable a comparison between the original control system and the adaptive control system. Figure 5-15 shows the gain and phase of both the decentralized PD control system and the adaptive control system. The gain crossover frequency (ω_{gc}) and phase crossover frequency (ω_{pc}) are indicated on the figure by GCF and PCF respectively.

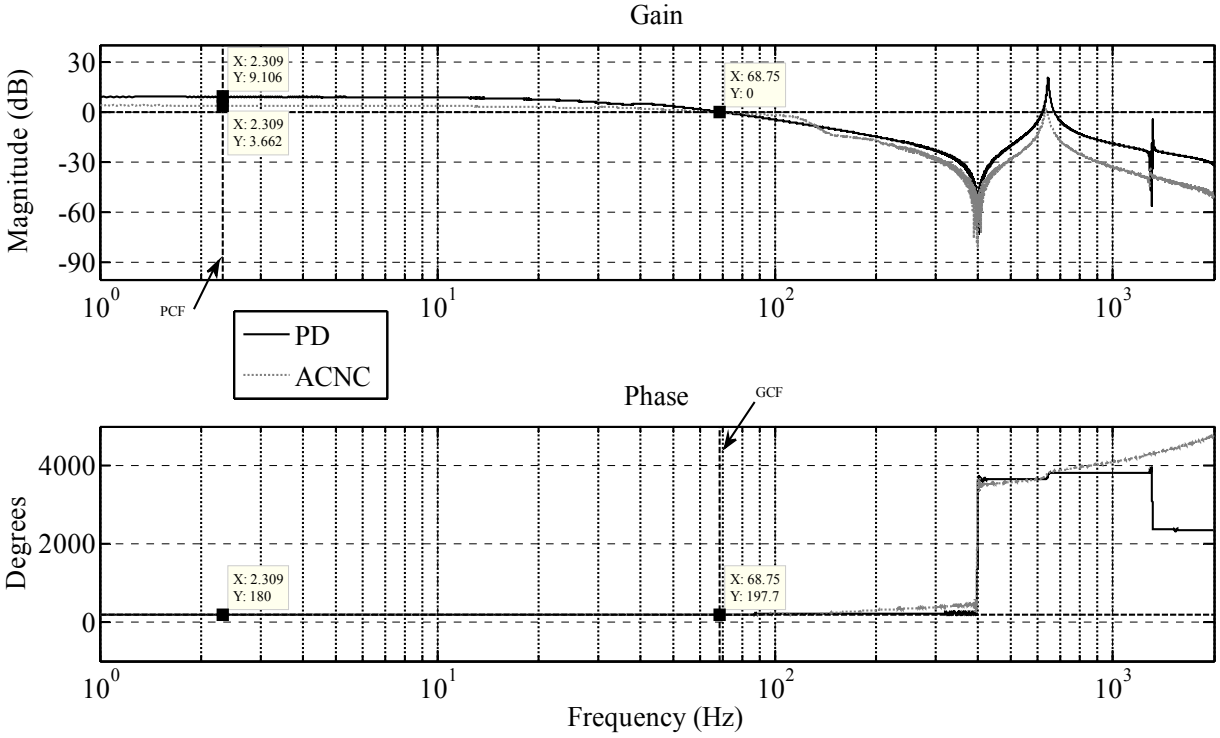

Figure 5-15: Worst-case gain and phase margins of PD controller and ACNC with *n* = 20

The worst-case gain margin achieved by the decentralized PD control system is 9.1 dB and the worst-case phase margin is 17.7°. The worst-case gain margin achieved by the adaptive control system is 3.7 dB and the worst-case phase margin is also 17.7°.

5.4 Results assessment

This section aims to evaluate the results of the step response, robustness analysis and stability analysis of the decentralized PD controller and the observer-based ACNC presented in the previous section.

5.4.1 Step response

The step response of the adaptive control system was clearly not as good as the step response of the decentralized PD control system. The ACNC settled much slower and overshot the final value significantly. However, a small settling time and low overshoot mean absolutely nothing if the system cannot remain stable. Such is the case of the PD control system. It warrants further investigation.

Upon closer inspection of the other AMB axis directions, an interesting occurrence was observed in AMB 1(y). The position output for AMB 1(y) of the decentralized PD control system can be seen in figure 5-16. The corresponding position output of the adaptive control system is shown in figure 5-17. The occurrence of chatter is clearly visible in both figures [1]. Notice that the chatter gradually disappears using the ACNC but in the decentralized PD control system, the chatter becomes uncontrollable.

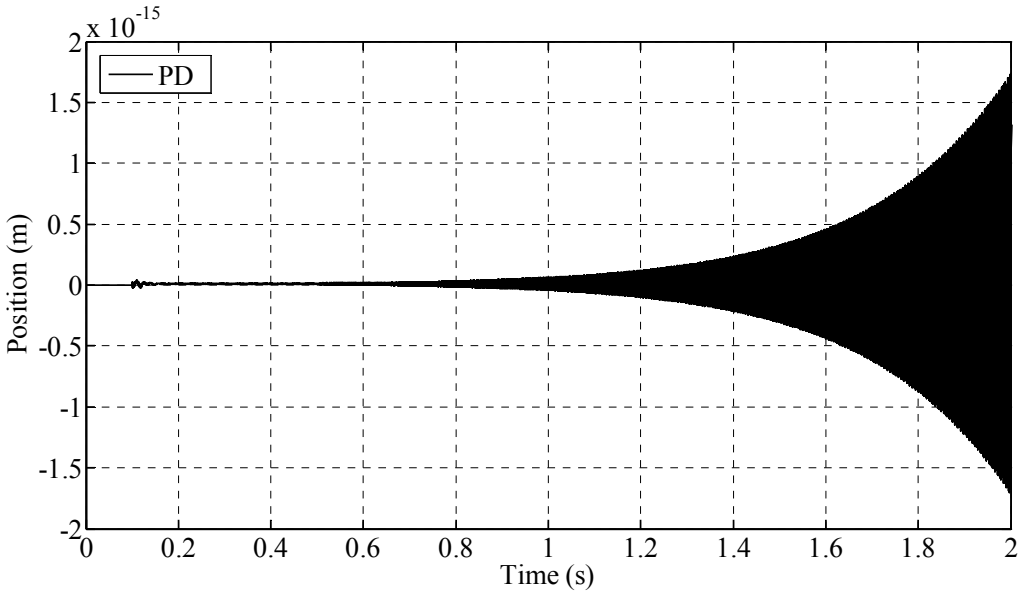

Figure 5-16: AMB 1(y) output for decentralized PD control

The chatter on the AMB 1(y) axis starts out very small, but as time progresses, it becomes large enough to finally destabilize the rotor along AMB 1(x). This is due to cross-coupling[18] effects between the x- and y-axis directions of AMB 1.

[18] Cross-coupling refers to the interaction in terms of coupling between the x- and y-axis directions of an AMB and between AMB 1(x) and AMB 2(x) as well as AMB 1(y) and AMB 2(y) [1].

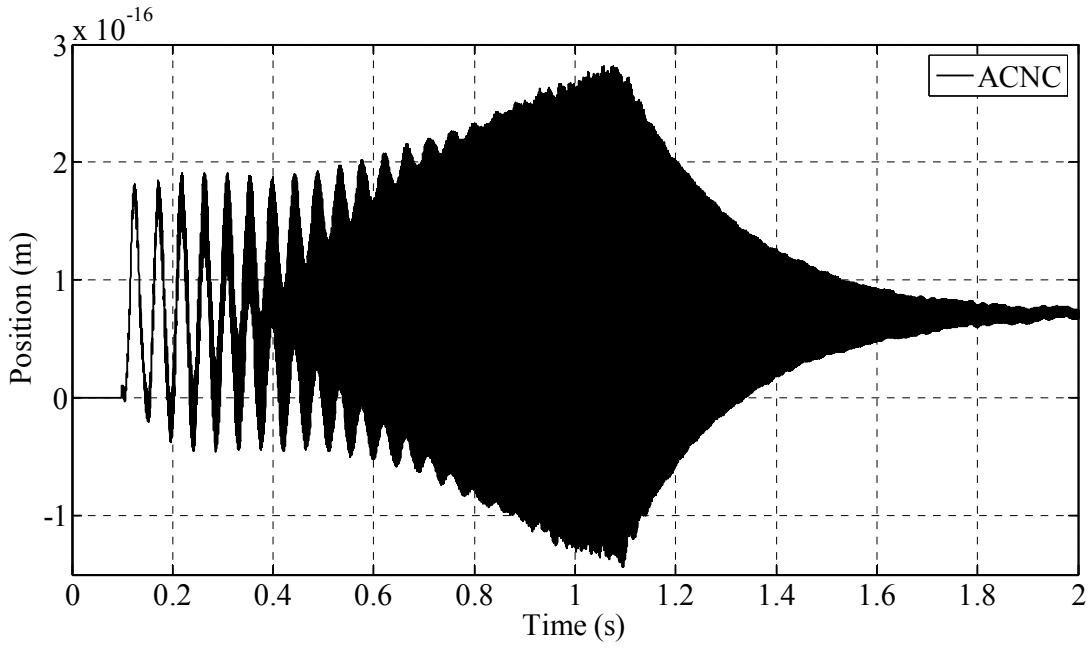

Figure 5-17: AMB 1(y) output for ACNC

5.4.2 Robustness analysis

Evaluation of the sensitivity function of each control system verifies that the adaptive control system is more robust against external disturbances than the decentralized PD control system. This level of robustness is also exhibited by newly commissioned systems as described in the ISO CD 14839-3 standard [78]. When significantly larger disturbances are applied, the adaptive control system proves to be only somewhat robust.

5.4.3 Stability analysis

The gain and phase margins achieved by the decentralized PD control system verify a robust system. The same holds true for the adaptive control system. More significantly, the gain and phase margins for both systems guarantee closed-loop stability [3].

5.4.4 Other comments

Optimizing the ACNC parameters using genetic algorithms did not deliver consistent results. Consider figure 5-18 showing the results from a genetic algorithm using 40 chromosomes and 10 generations to optimize the number of neurons for the critic and the number of neurons for the actor. The z-axis shows the performance achieved by each chromosome during each generation as the inverse of the ITAE performance index.

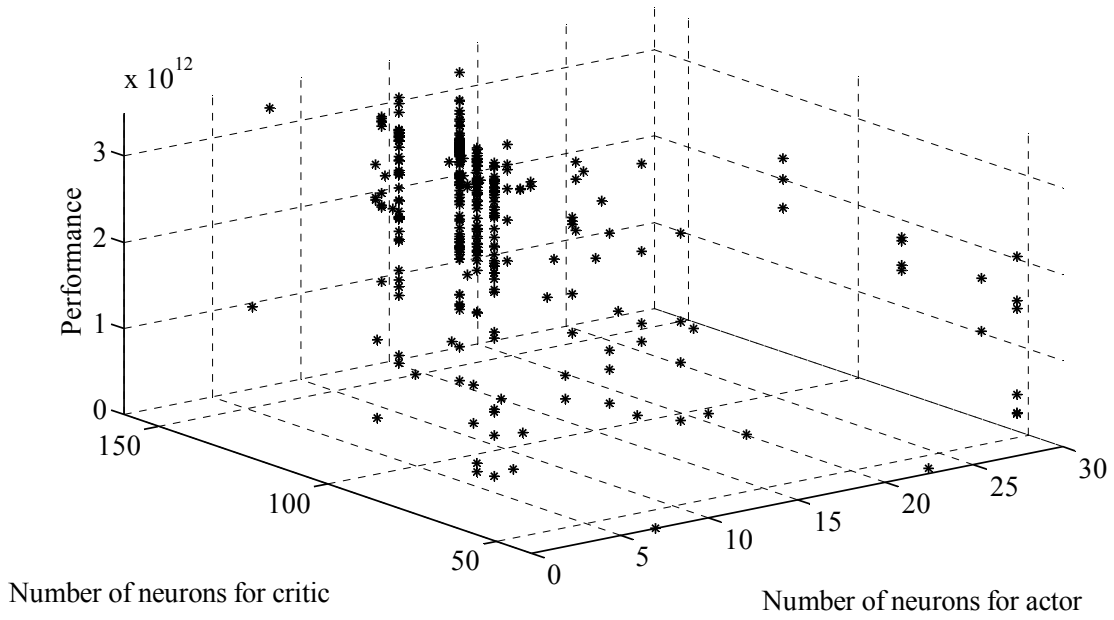

Figure 5-18: Genetic algorithm results for 40 generations

The genetic algorithm results clearly indicate that similar chromosomes did not achieve the same performance level at each initialization. The only random factor which came into play was the initialization of the hidden layer weights for each network. A set of network weights for a certain chromosome was different every time the chromosome was tested. This resulted in significant performance variations by similar chromosomes.

5.5 Conclusions

This chapter presented the implementation and simulation results of the observer-based ACNC applied to the AMB flywheel system. The NARX observer was optimized separately and then integrated into the adaptive controller. Results indicate that significantly more neurons are required for online neural networks than offline neural networks. This is largely due to the way the online networks of the observer-based ACNC are trained – quickly and continuously [5].

The adaptive controller was thoroughly tested by means of external disturbances. Robustness and stability analysis were also performed which verified that the adaptive control system is indeed robust and closed-loop stable. The final ACNC was of order 80 ($n \times m = 20 \times 4 = 80$), which is much higher than the decentralized PD control system. Since the PD controllers are decoupled and the adaptive controller uses cross-coupling, the two control approaches cannot be fairly compared. Only the implementation of each control approach can be used in a comparison. In the next chapter, various conclusions are drawn regarding the adaptive controller and the decentralized PD control scheme using the results from this chapter.

Chapter 6: Conclusions and recommendations

This chapter starts by drawing conclusions using results from the previous chapters. Important issues addressed during this study are highlighted and the outcome of proposed methodologies discussed. Concluding remarks continue by discussing the difference in performance between the decentralized PD control system and the adaptive control system. The chapter closes with recommendations for future work aimed at improving the adaptive controller as applied to the AMB flywheel system.

6.1 Conclusions

The objective of this study was to obtain an adaptive controller for the Fly-UPS system using online system identification, adaptive control design methods and neural networks. The neural networks had to continually adapt an AMB plant model online and acquire an adaptive controller for the 4-DOF AMB flywheel system. Overall system control in terms of stability and robustness had to be improved.

An indirect adaptive control design using observer, actor and critic neural networks was selected for the adaptive controller. This controller was called an observer-based ACNC. The chosen adaptive design boasted analytical proofs using Lyapunov stability analysis methods which guarantee stable online system identification and control. The use of state information by the adaptive controller was the main drawback since the required number of state variables had to be estimated. The complexity of the neural networks was also limited because the update algorithms of the chosen adaptive design only allowed for the use of single-layer networks.

Online system identification was achieved using a NARX neural network as the observer. The NARX observer had to encompass all the effects of the plant of the AMB flywheel system. These effects included the speed-dependent dynamic behaviour of the flexible rotor in the form of unbalance effects and flexible modes. The nonlinear magnetic force as well as saturation and hysteresis of the ferromagnetic material in the magnets are also included. Frequency response analysis provided the means for validation of the NARX architecture as the most suitable architecture to model the plant of the AMB flywheel system. The resulting neural model successfully represented the inherently unstable, nonlinear, dynamic and multivariable plant of the AMB flywheel system.

The differing time scales of various processes in the AMB flywheel system proved to be a problem when rotor speeds past the current maximum operating speed were considered. The NARX state observer produced larger and larger estimation errors as the rotor speed increased. The adaptive control system's sensitivity to external disturbances increased significantly as the observer's accuracy decreased.

The adaptive controller was successfully implemented in **SIMULINK®** and its parameters optimized with the help of genetic algorithms. The genetic algorithms proved to deliver inconsistent results for various reasons. One of these reasons was the random initialization of neural network weights. By improving the use of genetic algorithms, even better results are bound to surface.

The issue of performance verification was addressed in the form of step response, robustness and stability analysis. The decentralized PD control scheme proved insufficient when faced with severe cross-coupling effects. The adaptive controller kept the system stable under the same conditions. This is due to the fact that the decentralized PD control scheme consists of 4 separate controllers which do not share any kind of information with each other. The PD controllers are purely SISO even though they are implemented in parallel. The MIMO adaptive controller however, allowed for the multivariable nature of the AMB flywheel system and kept the system stable in the presence of severe cross-coupling effects.

A high bandwidth is required by the high closed-loop rigid body modes of the AMB flywheel system. The occurrence of chatter in the AMB flywheel system is due to unattainable high bandwidths of certain system components. Chatter can destabilize even a marginally stable closed-loop AMB system because of the many nonlinear effects in the system [1]. The decentralized PD control system could clearly not control the chatter and the rotor delevitated. The adaptive controller proved superior to the decentralized PD control scheme by containing the chatter along the AMB axis directions and keeping the rotor suspended.

In comparison with the decentralized PD control scheme, the observer-based ACNC had a relatively high order. Even so, the high computational resources usually associated with adaptive control were not evident in the observer-based ACNC [28]. The majority of computational effort was caused by the given simulation model of the plant of the AMB flywheel system. In contrast, the complexity of the chosen adaptive control design became apparent in the selection of optimal values for the controller parameters. Tuning of PD and PID controllers have become relatively straight-forward [23].

Comparison of a coupled PD controller with the observer-based ACNC would obviously have provided more insight into the performance and stability of each non-linear control approach. The only AMB flywheel system available at the time of this study was the linear decentralized PD control system. Fortunately, the implementation of each control approach – decentralized PD control and observer-based ACNC – could be properly compared with regard to the AMB flywheel system under consideration.

It should be clear at this point that the research problem and important issues were addressed successfully. The adaptive controller displayed an improvement in performance over the decentralized PD control scheme, even though only in a simulation environment. The observer-based ACNC adapted to external disturbances and allowed for the AMB flywheel system's complex behaviour. This study may have justified further research into adaptive control using artificial intelligence techniques as applied to the AMB flywheel system.

6.2 Recommendations for future work

This section presents recommendations for future work aimed at improving the adaptive controller as applied to the AMB flywheel system as well as a few general suggestions.

6.2.1 Eliminate need for state information

A significant drawback posed by the online NARX state observer was selecting the number of required state variables n. The number of state variables were approximated based on the system's DOF. When flexibility is considered, the number of DOF becomes much larger than the 4 DOF of the AMB flywheel system simulation model. This would imply using more state variables than indicated by (5.3) to perform state estimation successfully [1].

The question as to how many state variables are required still remains. Future work could include eliminating the need for state information by using a neural input-output model instead of a neural state observer. This would necessitate a control design method using output feedback and not state feedback as required by the adaptive critic model [8].

6.2.2 Increase hidden layers

Only a single hidden layer was used by each network in the chosen adaptive control design. The approximation capability of a network can be improved by adding another hidden layer. With more hidden layers, complex functions can be mapped more efficiently. The key lies in choosing the number of hidden layers which provides an optimal balance between computational complexity and accuracy [9]. Future work could include modifying the chosen adaptive control design to allow the use of more than a single hidden layer in each network.

6.2.3 Higher sampling rate for observer

At high rotor speeds such as 31,400 r/min, it became clear that the observer could not keep up. Online system identification fell behind because the time scales of underlying closed-loop dynamics and parameter variations became smaller and smaller. The only way to ensure that the identification occurs faster and avoid instability is to increase the sampling rate of the observer [15].

Unfortunately, increasing the sampling rate poses a problem to the practical AMB flywheel system since it is already set to the highest sampling rate allowed by the hardware. Future work could include improving the hardware to allow for higher sampling rates and consequently allow faster observer updates.

6.2.4 Improve network weights initialization

The hidden layer weights of the observer, actor and critic were initialized to random numbers uniformly distributed between selected ranges. These weights were kept constant while the output layer weights which were initialized to zero, were updated using the algorithm in the chosen design. It is believed that the estimation error can be made as small as desired by initializing the weights in this way and choosing the number of hidden layer weights sufficiently large [8].

It was shown in the previous chapter that random initialization of the hidden layer weights proved a serious problem. The same ACNC did not produce consistent results when re-initialized and simulated. Future work could include improving the way the hidden layer weights are initialized. The following are suggestions to find optimal values for the hidden layer weights:

- Just as the optimal values for the observer were determined offline, optimal hidden layer weights could also be determined offline. The method proposed by Yam and Chow could be used if desired output data were available or could be generated accurately [81].
- The hidden layer weights do not have to be kept constant – they could also be trained online like the output layer weights and in doing so, eliminate the need to improve the weights initialization method [82]. This would unfortunately also add to the computational effort required by the ACNC.
- Training the hidden layer weights online requires an extremely fast convergence rate since the AMB system is inherently unstable. Decentralized PD control could be used to initially stabilize the rotor while the neural network controller learns. A switching algorithm can be used to determine when to switch to the neural controller [53].

All of these recommendations to avoid the problems caused by randomly initializing the hidden layer weights obviously require more effort than indicated by the chosen control design. This extra effort is necessitated by the complexity of the AMB flywheel system.

6.2.5 Enhance use of genetic algorithms

If computer memory and time were not as limited as in this study, the genetic algorithm parameters could be modified to allow for a better search. The search space could be enlarged to allow for more possible chromosomes or the population size could be increased to test more chromosomes. The number of generations could also be increased to allow the algorithm to search longer for a solution.

The relationships between the various parameters of the chosen adaptive control design were more often than not, ambiguous. The effect on control system performance of varying a certain parameter was also repeatedly unclear. This necessitated the use of an iterative optimization method for the controller parameters since tuning by hand would have taken much too long.

As explained previously, the use of an iterative optimization method presented its own problems. The genetic algorithm results clearly indicated that the same adaptive controller did not always deliver the same level of performance. Future work could include the computation of the relationships between parameters as well as the effects of different parameters on system performance. This would enable improved parameter selection for the adaptive controller. A multi-objective genetic algorithm could even be used since it allows for more than one error function to be minimized [42].

6.2.6 Proven method to select number of neurons

Global optimization of the number of hidden layer neurons using genetic algorithms became very time-consuming especially when the search space was large. Some executions of the genetic algorithm took up to 30 hours on a high-performance personal computer. To avoid this problem in future work, a proven method to select the number of neurons in the hidden layer of a neural network is required. Ways to address the minimum number of required neurons do exist but these boil down to educated guesses [8].

No generally applicable method exists to explicitly decide the size of a neural network i.e. the number of hidden layers and the number of neurons [8]. Future work could include the analytical derivation of a method to determine, not just iteratively optimize, the number of hidden layer neurons in a neural network.

6.2.7 Adaptive controller order reduction

Clearly, a controller of order 80 will not suffice for practical applications. The physical implementation platform of the AMB flywheel system only allows for a maximum order of 19 at the current sampling time of 100 μs. For the adaptive control design of this study to be practically feasible, a way of reducing its order should definitely be investigated in future work.

Since the random initialization of hidden layer weights produced inconsistent results, the full power of the GAs in terms of optimization could not be realized. It is hoped that future work to improve the way the weights are initialized will lead to better optimization results by the GAs and consequently, a low-order adaptive controller. Controller order reduction would enable physical implementation of an observer-based ACNC and allow the potential of adaptive control using artificial intelligence techniques to be fully realized.

Appendix A: Project CD

The project CD contains all the files necessary to reproduce the results found in this document. Some of these files were given in which case it is indicated and others were programmed. The folders on the CD are as follow:

A.1 **MATLAB®** files:
   a. Architecture selection
   b. Adaptive control design
   c. Performance verification

A.2 **SIMULINK®** files:
   a. Architecture selection
   b. Adaptive control design
   c. Performance verification

A.3 Nonlinear model (given)

A.4 Documentation
   a. Dissertation
   b. Important literature
   c. Proposal

References

[1] G. Schweitzer and E. H. Maslen, *Magnetic Bearings: Theory, Design and Application to Rotating Machinery*. Heidelberg, Germany: Springer, 2009.

[2] J. T. Spooner, M. Maggiore, R. Ordonez, and K. M. Passino, *Stable Adaptive Control and Estimation for Nonlinear Systems: Neural and Fuzzy Approximator Techniques*. New York, USA: John Wiley and Sons, 2002.

[3] R. C. Dorf and R. H. Bishop, *Modern Control Systems*, 10th ed. Upper Saddle River, New Jersey: Pearson Prentice Hall, 2005.

[4] S. Sastry and M. Bodson, *Adaptive Control: Stability, Convergence and Robustness*, T. Kailath, Ed. Englewood Cliffs, New Jersey: Prentice Hall, 1989.

[5] G. A. Dumont and M. Huzmezan, "Concepts, Methods and Techniques in Adaptive Control," in *American Control Conference*, Vancouver, Canada, 2002, pp. 1137-1150.

[6] R. Ordonez and K. M. Passino, "Indirect Adaptive Control for a Class of Non-linear Systems with Time-varying Structure," *International Journal of Control*, vol. 74, no. 7, pp. 701-717, 2001.

[7] J.-S. R. Jang, C.-T. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing*. Upper Saddle River, New Jersey: Prentice-Hall, 1997.

[8] S. Jagannathan, *Neural Network Control of Nonlinear Discrete-Time Systems*. Boca Raton, Florida: Taylor and Francis Group, 2006.

[9] M. T. Hagan, H. B. Demuth, and M. Beale, *Neural Network Design*. Boston, USA: PWS Publishing Company, 1996.

[10] A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial Neural Networks: A Tutorial," *IEEE Computer*, pp. 31-44, Mar. 1996.

[11] Y. Diao and K. M. Passino, "Intelligent Fault-tolerant Control using Adaptive and Learning Methods," *Control Engineering Practice*, no. 10, pp. 801-817, 2002.

[12] S. Zerkaoui, F. Druaux, E. Leclercq, and D. Lefebvre, "Stable Adaptive Control with Recurrent Neural Networks for Square MIMO Non-linear Systems," *Engineering Applications of Artificial Intelligence*, no. 22, pp. 702-717, 2009.

[13] S. Myburgh, "The development of a fully suspended AMB system for a high-speed flywheel application," North-West University Dissertation, 2007.

[14] L. Ljung, *System Identification - Theory for the User*. Upper Sadle River, New Jersey: Prentice Hall, 1999.

[15] B. D. O. Anderson, "Failures of Adaptive Control Theory and their Resolution,"

Communications in Information and Systems, vol. 5, no. 1, pp. 1-20, 2005.

[16] G. Li, "Robust Stabilization of Rotor Active Magnetic Bearings Systems," PhD Dissertation, University of Virginia, 2007.

[17] J. Ritonja, B. Polajzer, D. Dolinar, B. Grcar, and P. Cafuta, "Active Magnetic Bearings Control," in *29th Chinese Control Conference*, Beijing, China, 2010, pp. 5604-5609.

[18] P. F. Ribeiro, B. K. Johnson, M. L. Crow, A. Arsoy, and Y. Liu, "Energy Storage Systems for Advanced Power Applications," *Proceedings of the IEEE*, vol. 89, no. 12, pp. 1744-1756, Dec. 2001.

[19] H. Ibrahim, A. Ilincaa, and J. Perronb, "Energy storage systems - Characteristics and comparisons," *Renewable and Sustainable Energy Reviews*, no. 12, pp. 1221-1250, 2008.

[20] B. Bolund, H. Bernhoff, and M. Leijon, "Flywheel energy and power storage systems," *Renewable and Sustainable Energy Reviews*, no. 11, pp. 235-258, 2007.

[21] E. O. Ranft, "The Development of a Flexible Rotor Active Magnetic Bearing System," North-West University Dissertation, 2005.

[22] J. Ritonja, B. Polajzer, and D. Dolinar, "Decentralized Control for Active Magnetic Bearings," in *Power Electronics and Motion Control Conference*, Portoroz, 2006, pp. 1413-1418.

[23] K. J. Astrom and R. M. Murray, *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton, New Jersey, USA: Princeton University Press, 2008.

[24] R. P. Jastrzebski and R. Pollanen, "Centralized Optimal Position Control for Active Magnetic Bearings: Comparison with Decentralized Control," *Electrical Engineering*, vol. 91, no. 2, pp. 101-114, Aug. 2009.

[25] H.-C. Chen, "OPTIMAL FUZZY PID CONTROLLER DESIGN OF AN ACTIVE MAGNETIC BEARING SYSTEM BASED ON ADAPTIVE GENETIC ALGORITHMS," in *Seventh International Conference on Machine Learning and Cybernetics*, Kunming, 2008, pp. 2054-2060.

[26] C.-T. Lin and C.-P. Jou, "GA-Based Fuzzy Reinforcement Learning for Control of a Magnetic Bearing System," in *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS - PART B: CYBERNETICS*, 2000, pp. 276-289.

[27] M. Golob and B. Tovornik, "Modeling and control of the magnetic suspension system," in *ISA Transactions*, 2003, pp. 89-100.

[28] K. J. Astrom and B. Wittenmark, *Adaptive Control*, 2nd ed. New York, USA: Prentice Hall, 2008.

[29] N.-C. Tsai, C. W. Chiang, and C. H. Kuo, "Robust Sliding Mode Control for Axial AMB Systems," in *5th Asian Control Conference*, 2004, pp. 64-69.

[30] N. S. Gibson, "HINF CONTROL OF ACTIVE MAGNETIC BEARINGS: AN INTELLIGENT UNCERTAINTY MODELING APPROACH," PhD Dissertation, North Carolina State University, Raleigh, 2004.

[31] R. P. Jastrzebski, "Signal-based Hinf optimal control for AMB system based on genetic algorithm," in *IEEE International Conference on Control and Automation*, Christchurch, New Zealand, 2009, pp. 715-721.

[32] P. Lundstrom, S. Skogestad, and Z.-Q. Wang, "UNCERTAINTY WEIGHT SELECTION FOR H-INFINITY AND MU-CONTROL METHODS," in *Conference on Decision and Control*, Brighton, England, 1991, pp. 1537-1542.

[33] Y. Uchimura and H. Kazerooni, "A mu-synthesis based control for compliant manoeuvres," in *IEEE International Conference on Systems, Man, and Cybernetics*, 1999, pp. 1014-1019.

[34] P. V. Kokotovic, "The joy of feedback: nonlinear and adaptive," *IEEE Control Systems Magazine*, vol. 12, no. 3, pp. 7-17, 1992.

[35] P. Tsiotras, "Zero- and Low-Bias Control Designs for Active Magnetic Bearings," *IEEE Transactions on Control Systems Technology*, vol. 11, no. 6, pp. 889-904, Nov. 2003.

[36] C. L. Phillips and H. T. Nagle, *Digital Control System Analysis and Design*, 3rd ed. Saddle River, New Jersey, USA: Prentice Hall, 1995.

[37] A. M. Beizema, J. M. Echeverria, M. Martinez-Iturralde, I. Egana, and L. Fontan, "Comparison between Pole-Placement Control and Sliding Mode Control for 3-Pole Radial Magnetic Bearings," in *International Symposium on Power Electronics, Electrical Drives, Automation and Motion*, 2008, pp. 1315-1320.

[38] B. Shafai, S. Beale, P. La Rocca, and E. Cusson, "Magnetic Bearing Control Systems and Adaptive Forced Balancing," *IEEE Control Systems*, pp. 4-13, Apr. 1994.

[39] X. Chen, L. Ji, and K. Liu, "A BP Neural Network Controller for Magnetic Suspended Flywheel System," in *IEEE International Conference on Computer Science and Information Technology*, 2010, pp. 448-452.

[40] T. W. S. Chow and S.-Y. Cho, *Neural Networks and Computing: Learning Algorithms and Applications*, W.-K. Chen, Ed. London, UK: Imperial College Press, 2007.

[41] S.-S. Yu, S.-J. Wu, and T.-T. Lee, "Optimal Fuzzy Control of Radial Active Magnetic Bearing Systems," in *IEEE International Symposium on Computational Intelligence in Robotics and Automation*, Kobe, Japan, 2003, pp. 1393-1398.

[42] P. Schroder, A. J. Chipperfield, P. J. Fleming, and N. Grum, "Multi-Objective Optimisation of Distrubuted Active Magnetic Bearing Controllers," in *Genetic Algorithms in Engineering Systems: Innovations and Applications*, 1997, pp. 13-18.

[43] G. Song and R. Mukherjee, "Integrated Adaptive Robust Control of Active Magnetic Bearings,"

in *IEEE International Conference on Systems, Man, and Cybernetics*, 1996, pp. 1784-1790.

[44] A. R. Husain, M. N. Ahmad, and A. H. Mohd. Yatim, "Sliding Mode Control with Linear Quadratic Hyperplane Desing: An application to an Active Magnetic Bearing System," in *The 5th Student Conference on Research and Development*, 2007, pp. 1-6.

[45] S. Sivrioglu and K. Nonami, "Sliding Mode Control With Time-Varying Hyperplane for AMB Systems," *IEEE/ASME Transactions on Mechatronics*, vol. 3, no. 1, pp. 51-59, Mar. 1998.

[46] X. Yu and O. Kaynak, "Sliding-Mode Control With Soft Computing: A Survey," *IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS*, vol. 56, no. 9, pp. 3275-3285, Sep. 2009.

[47] S. Sivrioglu, "Adaptive backstepping for switching control magnetic bearing system with vibrating base," *IET Control Theory Applications*, vol. 1, no. 4, pp. 1054-1059, 2007.

[48] S. Sivrioglu and K. Nonami, "Adaptive Output Backstepping Control of a Flywheel Zero-Bias AMB System with Parameter Uncertainty," in *42nd IEEE Conference on Decision and Control*, Maui, Hawaii, 2003, pp. 3942-3947.

[49] S. You, "ADAPTIVE BACKSTEPPING CONTROL OF ACTIVE MAGNETIC BEARINGS," Cleveland State University Dissertation, 2007.

[50] J. A. Aseltine, A. R. Mancini, and C. W. Sarture, "A Survey of Adaptive Control Systems," *IRE Transactions on Automatic Control*, vol. 6, no. 1, pp. 102-108, Dec. 1958.

[51] C. Li and L.-L. Xie, "On Robust Stability of Discrete-Time Adaptive Nonlinear Control," *Systems and Control Letters*, vol. 55, no. 6, pp. 452-458, Jun. 2006.

[52] G. Tao, *Adaptive Control Design and Analysis*. Hoboken, New Jersey: John Wiley and Sons, 2003.

[53] K. S. Narendra, M. J. Feiler, and Z. Tian, "Control of Complex Systems Using Neural Networks," in *Modeling and Control of Complex Systems*. Boca Raton, USA: CRC Press, 2008, ch. 2, pp. 13-98.

[54] K. J. Astrom, "Adaptive Feedback Control," *Proceedings of the IEEE*, vol. 75, no. 2, pp. 185-217, Feb. 1987.

[55] M. T. Hagan and H. B. Demuth, "Neural Networks for Control," in *American Control Conference*, 1999, pp. 1642-1656.

[56] F. Gang and L. Rogelio, *Adaptive Control Systems*. Oxford, UK: Newnes, 1999.

[57] K. S. Narendra and J. Balakrishnan, "Adaptive Control using Multiple Models," *IEEE Transactions on Automatic Control*, vol. 42, no. 2, pp. 171-187, Feb. 1997.

[58] S. N. Sivanandam, S. Sumathi, and S. N. Deepa, *Introduction to Fuzzy Logic using MATLAB*. Berlin, Germany: Springer, 2007.

[59] K. M. Passino and S. Yurkovich, *Fuzzy Control*. Menlo Park, California, USA: Addison Wesley

Longman, 1998.

[60] T. J. Ross, *Fuzzy Logic with Engineering Applications*, 2nd ed. Chichester, England: John Wiley and Sons, 2004.

[61] H. Choi, G. Buckner, and N. Gibson, "Neural Robust Control of a High-Speed Flexible Rotor Supported on Active Magnetic Bearings," in *American Control Conference*, Minneapolis, Minnesota, USA, 2006, pp. 3679-3684.

[62] V. Kecman, *Learning and Soft Computing*. Cambridge, Massachusetts: The MIT Press, 2001.

[63] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford, UK: Clarendon Press, 1995.

[64] S. J. Yoo, J. B. Park, and Y. H. Choi, "Indirect Adaptive Control of Non-linear Dynamic Systems using Self-recurrent Wavelet Neural Networks via Adaptive Learning Rates," *Information Sciences: An International Journal*, no. 177, pp. 3074-3098, 2007.

[65] M. Neser, "Neuro-Fuzzy Techniques for Intelligent Control," Masters Dissertation, North-West University, Potchefstroom, 2006.

[66] L. R. Medsker and L. C. Jain, *Recurrent Neural Networks: Design and Applications*. Boca Raton, USA: CRC Press, 2001.

[67] A. Senyard, E. Kazmierczak, and L. Sterling, "Software Engineering Methods for Neural Networks," in *Proceedings of the Tenth Asia-Pacific Software Engineering Conference*, 2003, pp. 468-477.

[68] M. Agarwal, "A Systematic Classification of Neural-Network-Based Control," in *Third IEEE Conference on Control Applications*, Glasgow, UK, 1997, pp. 75-93.

[69] M. M. Polycarpou and P. A. Ioannou, "Modelling, Identification and Stable Adaptive Control of Continuous-Time Nonlinear Dynamical Systems Using Neural Networks," in *American Control Conference*, Los Angeles, 1992, pp. 36-40.

[70] R. Safaric, K. Jezernik, and M. Pec, "Implementation of neural network sliding-mode controller for DD robot," in *IEEE International Conference on Intelligent Engineering Systems*, 1997, pp. 83-88.

[71] M. T. Hagan, H. B. Demuth, and O. De Jesus, "An introduction to the use of neural networks in control systems," *International Journal of Robust and Nonlinear Control*, vol. 12, no. 11, pp. 959-985, Sep. 2002.

[72] U. Forssell and L. Ljung, "Closed-loop Identification Revisited," *Automatica*, vol. 35, no. 7, pp. 1215-1241, Apr. 1999.

[73] P. A. van Vuuren, "Robustness estimation of self-sensing active magnetic bearings via system identification," PhD Thesis, North-West University, Potchefstroom, 2009.

[74] H. Demuth, M. Beale, and M. Hagan, *Neural Network Toolbox TM 6 User's Guide*. Natick, MA:

The MathWorks, Inc., 2009.

[75] C. C. Klimasauskas, "Applying Neural Networks, Part III: Training a Neural Network," *PC AI*, vol. 5, no. 3, pp. 20-24, May 1991.

[76] W. Sarle. (2010, Jun.) Neural Network FAQs. [Online]. http://www.faqs.org/faqs/ai-faq/neural-nets/

[77] R. Zemouri, R. Gouriveau, and N. Zerhouni, "Defining and applying prediction perfromance metrics on a recurrent NARX time series model," *Neurocomputing*, Jun. 2010.

[78] ISO-TC108/SC2/WG7, "Active magnetic bearing - Evaluation of stability margin," Patent ISO CD148393-3, Oct. 7, 2003.

[79] **MATLAB®** (2010) - The Language of Technical Computing, The MathWorks, Inc. (DVD)

[80] O. Cordon, F. Herrera, F. Hoffmann, and L. Magdalena, *Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases*. River Edge, New Jersey: World Scientific Publishing, 2001.

[81] J. Y. F. Yam and T. W. S. Chow, "A weight initialization method for improving training speed in feed forward neural network," *Neurocomputing*, no. 30, pp. 219-232, Mar. 2000.

[82] F. L. Lewis, S. Jagannathan, and A. Yesildirek, "Neural Network Control of Robot Arms and Nonlinear Systems," in *Neural Systems for Control*. Academic Press, 1997, pp. 157-206.

[83] L. Chen and K. S. Narendra, "Non-linear Adaptive Control using Neural Networks and Multiple Models," *Automatica*, no. 37, pp. 1245-1255, 2001.

[84] Y. Fu and T. Chai, "Non-linear Multi-variable Adaptive Control using Multiple Models and Neural Networks," *Automatica*, no. 43, pp. 1101-1110, 2007.

[85] G. P. Liu, V. Kadirkamanathan, and S. A. Billings, "Variable Neural Networks for Adaptive Control of Non-linear Systems," *IEEE Transactions on Systems, Man and Cybernetics - Part C: Applications and Reviews*, vol. 29, no. 1, pp. 34-43, Feb. 1999.

[86] T. Hayakawa, W. M. Haddad, and N. Hovakimyan, "Neural Network Adaptive Control for a Class of Non-linear Uncertain Dynamical Systems with Asymptotic Stability Guarantees," *IEEE Transactions on Neural Networks*, vol. 19, no. 1, pp. 80-89, Jan. 2008.