MODELLING AND IDENTIFICATION OF NONLINEAR DC MOTOR DRIVE SYSTEMS USING RECURRENT WAVELET NETWORKS

by

Sarah Hussain Zahidi

A Thesis Presented to the Faculty of the
American University of Sharjah
College of Engineering
in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science in
Electrical Engineering

Sharjah, United Arab Emirates

January 2013

Approval Signatures

We, the undersigned, approve the Master's Thesis of Sarah Hussain Zahidi.

Thesis Title: Modelling and Identification of Nonlinear DC Motor Drive Systems Using Recurrent Wavelet Networks

Signature	**Date of Signature** (dd/mm/yyyy)
_____	_____
Dr. Rached Dhaouadi Professor, Department of Electrical Engineering Thesis Advisor	
_____	_____
Dr. Aydin Yesildirek Associate Professor, Department of Electrical Engineering Thesis Committee Member	
_____	_____
Dr. Mohammad Jaradat Visiting Associate Professor, Department of Mechanical Engineering Thesis Committee Member	
_____	_____
Dr. Mohamed El-Tarhuni Head, Department of Electrical Engineering	
_____	_____
Dr. Hany El Kadi Associate Dean, College of Engineering	
_____	_____
Dr. Yousef Al Assaf Dean, College of Engineering	
_____	_____
Dr. Khaled Assaleh Director of Graduate Studies	

Acknowledgements

I would like to thank my research advisor, Dr. Rached Dhaouadi, for his continued support, encouragement, understanding and patience through the course of my thesis at the American University of Sharjah.

I would like to also extend my thanks to the graduate committee for their time and invaluable suggestions.

Finally, I would like to express my gratitude to my parents, and most especially, my brother, without whose prayers, love and words of wisdom, this thesis would not have been accomplished.

Abstract

The main objective of this research is to study the use of Recurrent Wavelet Networks (RWN) for the modelling and identification of nonlinear dynamic systems. Since the vast majority of physical processes and systems exhibit nonlinearities in their behavior, mathematical models may be difficult to obtain as processes may be affected by external operating conditions and a number of parameters may not be identified. Electromechanical systems are an example of nonlinear systems where parameters such as viscous and coulomb friction, and distributed inertias are often unknown. In such cases, a model is required that will capture the nonlinearities and the dynamics of the system. In this thesis, an online identification method is developed using structured Recurrent Wavelet Networks (RWN) in order to simultaneously identify linear and nonlinear mechanical parameters of an electromechanical system. Network learning is implemented using the gradient descent algorithm. Stability analysis is carried out based on the minimization of a Lyapunov function in order to obtain Adaptive Learning Rates (ALR) for training the network. Simulations are carried out to validate the performance of the proposed adaptive learning rate based modeling and identification technique.

Search Terms: ***Wavelet Networks, Recurrent Wavelet Networks, DC Motor Parameter Identification, Friction Identification, Adaptive Learning Rates***

Table of Contents

Acknowledgements 4
Abstract 5
List of Tables 13
Chapter 1: Introduction 15
1.1 Background 15
1.2 Literature Review 17
1.3 Objectives of Research 24
1.4 Thesis Organization 25
Chapter 2: Wavelet Networks 26
2.1 Wavelets 26
2.2 Conventional Wavelet Networks 28
2.2.1 Architecture. 28
2.2.2 Initialization algorithms. 29
2.2.2 Training algorithms 31
2.3 Nonlinear Function Approximation 35
2.3.1 Offline training. 36
2.3.2 Online training. 49
Chapter 3: Modelling of Dynamic Systems Using Recurrent Wavelet Networks 57
3.1 Architecture 58
3.2 Training Algorithm 59
3.3 Convergence and Stability Analysis 62
3.4 Adaptive Learning Rates 66
3.5 Simulations 76
3.5.1 Nonlinear DC motor discretized using Bilinear transformation. 76
3.5.2 Nonlinear DC motor discretized using Euler Forward transformation. 81
3.5.3 Friction 84
Chapter 4: DC Motor Identification Using Structured Recurrent Wavelet Network 91
4.1 DC Motor Model Derivation 91
4.2 Architecture for DC Motor Parameter Identification 94
4.3 Training Algorithm 96
4.4 Convergence and Stability Analysis 99
4.5 Adaptive Learning Rates 100

Chapter 5: Case Studies and Simulation Analysis 116
5.1 Nonlinear Friction Identification of DC Motor 116
5.2 Linear Parameter Identification of Frictionless DC Motor 124
5.3 Linear Parameter Identification of DC Motor with Viscous Friction 127
5.4 Simultaneous Linear and Nonlinear Parameter Identification of DC Motor 132
Chapter 6: Conclusion and Future Work 140
6.1 Conclusion 140
6.2 Recommendations for Future Work 142
References 143
Appendix A 145
Appendix B 146
Appendix C 149
Appendix D 150
Appendix E 151

List of Figures

Figure 1: Artificial Intelligence Based Modelling and Control Techniques.....15

Figure 2: RWNN-Based Mobile Robot Control Structure 21

Figure 3: RWNN-Based SPC Scheme.....22

Figure 4: RWN Based Dynamic System Identification Architecture.....22

Figure 5: Indirect Adaptive Control Architecture Using RWNs 23

Figure 6: First Order Derivative of Gaussian Function 27

Figure 7: Morlet Wavelet.....27

Figure 8: Conventional Wavelet Network Architecture 28

Figure 9: Dyadic Grid.....31

Figure 10: Levenberg Marquardt Flowchart.....34

Figure 11: Approximation of Static Nonlinear Function.....35

Figure 12: SISO Nonlinear Function 36

Figure 13: MSE for Varying Nw for SISO Static Function Approximation using Heuristic Method.....37

Figure 14: MSE for Varying Nw for SISO Static Function Approximation using Dyadic Grid Method 38

Figure 15: MSE for Varying μ for SISO Static Function Approximation using Heuristic Method.....39

Figure 16: MSE for Varying μ for SISO Static Function Approximation using Dyadic Grid Method.....39

Figure 17: MSE for LM and GD Trained Networks for SISO Static Function Approximation 41

Figure 18: Output of Networks Trained using LM and GD Algorithms 41

Figure 19: MSE for Network Trained using Different Activation Functions.....42

Figure 20: Output of Networks Trained using Different Activation Functions.....43

Figure 21: MISO Nonlinear Function.....43

Figure 22: MSE for Varying Nw for MISO Function Approximation using Heuristic Method 44

Figure 23: MSE for Varying Nw for MISO Function Approximation using Dyadic Grid Method 45

Figure 24: MSE for Varying μ for MISO Function Approximation using Heuristic Method 46

Figure 25: MSE for Varying μ for MISO Function Approximation using Dyadic Grid Method 46

Figure 26: MSE for LM and GD Trained Networks for MISO Function Approximation 47

Figure 27: MSE of Networks Trained using Different Activation Functions for MISO Function Approximation 48

Figure 28: Gaussian Function 49

Figure 29: Online Training of SISO CWN 50

Figure 30: Mean Square Error for Approximation of Gaussian Function 50

Figure 31: Block Diagram for Testing Wavelet Network 51

Figure 32: Output of CWN for Gaussian Function Approximation 51

Figure 33: MSE for Different Activation Functions for Online SISO Function Approximation 52

Figure 34: MISO Function for Online Training 53

Figure 35: Training Signals for MISO Function 53

Figure 36: Online Training of MISO CWN 54

Figure 37: Mean Square Error for Approximation of MISO Function 54

Figure 38: Block Diagram for Testing MISO Wavelet Network 55

Figure 39: Output of CWN for MISO Function Approximation 55

Figure 40: MSE for Different Activation Functions for Online MISO Function Approximation 56

Figure 41: Training Structure for Dynamic Systems 57

Figure 42: RWN Architecture 58

Figure 43: Flowchart for Updating Adaptive Learning Rates 65

Figure 44: Frictional Torque 77

Figure 45: Training RWN for DC Motor, Bilinear Discretization 78

Figure 46: MSE for DC Motor RWN, Bilinear Discretization 78

Figure 47: Learning Rates for Translation and Dilation over First Training Cycle 79

Figure 48: Testing RWN for DC Motor, Bilinear Discretization 80

Figure 49: DC Motor RWN Test Output, Bilinear Discretization 80

Figure 50: Training RWN for DC Motor, Euler Forward Discretization 81

Figure 51: MSE for DC Motor RWN, Euler Forward Discretization 82

Figure 52: Learning Rates for Translation and Dilation over First Training Cycle 82

Figure 53: Testing RWN for DC Motor, Euler Forward Discretization 83

Figure 54: DC Motor RWN Test Output, Euler Forward Discretization 83

Figure 55: Nonlinear Friction Identification 84

Figure 56: Friction Characteristics 85

Figure 57: Generating Training Data for Friction Identification 85

Figure 58: Training Signal for Friction Identification 86

Figure 59: MSE for Varying Nw for Friction Identification 86

Figure 60: Learning Rates for Translation and Dilation over First and Last Training Cycle, Nw=3 87

Figure 61: Learning Rates for Translation and Dilation over First and Last Training Cycle, Nw=7 88

Figure 62: Learning Rates for Translation and Dilation over First and Last Training Cycle, Nw=15 88

Figure 63: Testing Signal for Friction Identification 89

Figure 64: Torque Speed Characteristic for Varying Nw after Testing for Friction Identification 90

Figure 65: Continuous Time Model of DC Motor 91

Figure 66: Discretized Integrator using Euler-forward Method 92

Figure 67: Step 1 of Discretization of DC Motor System 92

Figure 68: Step 2 of Discretization of DC Motor Model 93

Figure 69: Step 3 of Discretization of DC Motor Model 93

Figure 70: Structured Recurrent Wavelet Network Representing DC Motor Model ..94

Figure 71: DC Motor Parameter Identification Training Structure 95

Figure 72: Simplified DC Motor Parameter Identification Training Structure 97

Figure 73: Nonlinear Friction Identification.....116

Figure 74: Training Signal for Nonlinear Frictional Function Identification.....117

Figure 75: MSE for Varying Nw for Nonlinear Frictional Function Approximation118

Figure 76: ALRs for Nonlinear Function Approximation over First and Last Training Cycle, Nw=15 118

Figure 77: Testing Signal for Nonlinear Frictional Function Approximation.....119

Figure 78: Torque Speed Characteristics of DC Motor 120

Figure 79: Training Signal for Nonlinear Frictional Function Identification.....121

Figure 80: MSE for Different Activation Functions for Nonlinear Frictional Function Approximation 122

Figure 81: Testing Signal for Nonlinear Frictional Function Identification.....122

Figure 82: Torque Speed Characteristics.....123

Figure 83: Linear Parameter Identification for Frictionless DC Motor.....124

Figure 84: Training Signal for DC Motor Parameter Training.....124

Figure 85: MSE for Training Linear Parameters of Frictionless DC Motor.....125

Figure 86: ALRs for Linear Parameter Identification of Frictionless DC Motor.....126

Figure 87: Convergence of Kt and J for Parameter Identification of Frictionless DC Motor.....126

Figure 88: Network Learning Structure for DC Motor with Viscous Friction.....127

Figure 89: MSE for Training DC Motor with Viscous Friction 128

Figure 90: ALRs for Linear 2 Parameter Identification of DC Motor with Viscous Friction 129

Figure 91: Convergence of B and J for Parameter Identification of DC Motor with Viscous Friction 129

Figure 92: Network Learning Structure for Linear Parameter Identification.....130

Figure 93: MSE for Training Linear Parameters of DC Motor with Unknown Viscous Friction 130

Figure 94: ALRs for Linear 3 Parameter Identification of DC Motor with Viscous Friction 131

Figure 95: Convergence of B, J and Kt for DC Motor with Viscous Friction 132

Figure 96: Network Learning Structure for Linear and Nonlinear DC Motor Parameter Identification 132

Figure 97: Initialization of RWN for Simultaneous Identification 133

Figure 98: Training Signal for Simultaneous DC Motor Parameter Identification ... 133

Figure 99: Torque Speed Characteristics of DC Motor 134

Figure 100: MSE for Initialization of Friction RWN 134

Figure 101: MSE for Simultaneous Parameter Identification 135

Figure 102: ALRs for Simultaneous Linear and Nonlinear Parameter Identification of DC Motor 136

Figure 103: Convergence of B, J for DC Motor with Nonlinear Friction 136

Figure 104: Estimated Frictional Torque Speed Characteristic 137

Figure 105: Estimated Combined Frictional Characteristic for Simultaneous Identification 139

Figure 106: Comparison of Combined Frictional Characteristic 139

List of Tables

Table 1: MSE for Varying Nw for SISO Static Function Approximation using Heuristic and Dyadic Grid Method 37

Table 2: MSE for Varying μ for SISO Function Approximation using Heuristic and Dyadic Grid Method 38

Table 3: MSE for LM and GD Trained Networks for SISO Static Function Approximation 40

Table 4: MSE for Different Activation Functions for SISO Static Function Approximation 42

Table 5: MSE for Changing Nw for MISO Function Approximation using Heuristic Method 44

Table 6: MSE for Changing μ for MISO Function Approximation using Heuristic Method 45

Table 7: MSE for LM and GD Trained Networks for MISO Function Approximation 48

Table 8: MSE for Different Activation Functions for MISO Function Approximation 49

Table 9: MSE for Different Activation Functions for Online SISO Function Approximation 52

Table 10: MSE for Different Activation Functions for Online MISO Function Approximation 56

Table 11: MSE for Varying Nw for Friction Identification 87

Table 12: MSE for Varying Nw after Testing for Friction Identification 89

Table 13: MSE after Training for Varying Nw for Nonlinear Frictional Function Approximation 117

Table 14: MSE after Testing for Varying Nw for Nonlinear Frictional Function Approximation 119

Table 15:MSE after Training for Different Activation Functions for Nonlinear Function Approximation 121

Table 16: MSE after Testing for Different Activation Functions for Nonlinear Frictional Function Approximation 123

Table 17: Results of Training Linear Parameters of Frictionless DC Motor 125

Table 18: Results of Training Linear Parameters of DC Motor with Known Viscous Friction 128

Table 19: Results of Training Linear Parameters of DC Motor with Unknown Viscous Friction 130

Table 20: Results of Simultaneous Training of Linear and Nonlinear DC Motor Parameters 137

Chapter 1: Introduction

1.1 Background

Traditionally, complex nonlinear systems were modelled using linearization techniques in order to facilitate the design and implementation of PID controllers. In many cases, mathematical models may be difficult to obtain as processes are affected by external operating conditions and a number of process parameters may not be identified. Electromechanical systems are a prime example of nonlinear systems where parameters such as viscous friction, coulomb friction and distributed inertias are often unknown. In such cases, development of an optimized controller requires a more detailed model to capture the nonlinearities and the dynamics of the system.

Over the last few decades, there has been an upsurge in the trend of using Artificial Intelligence (AI) based modelling techniques for modelling nonlinear time varying systems. Such techniques make use of available system input and output data to produce complex mappings and reproduce system models. A number of AI based techniques are available as shown in Figure 1.

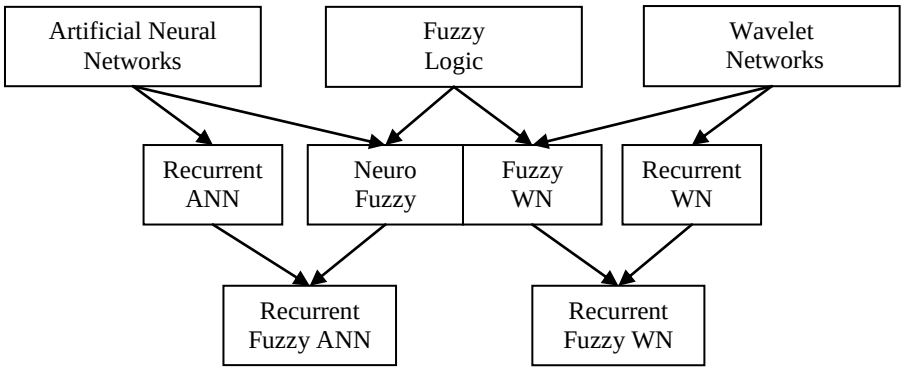


Figure 1: Artificial Intelligence Based Modelling and Control Techniques

Artificial Neural Networks (ANNs) are parallel processing networks that consist of neurons interconnected to form a layered structure in the form of an input layer, hidden layer(s) and an output layer. Typical activation functions used in ANNs

include the sigmoid, the log-sigmoid and the tan-sigmoid functions. ANNs have been widely used in system modelling and identification [1-4] and are able to effectively model system nonlinearities. The distributive nature of the neural network architecture also makes it highly fault tolerant.

ANNs, however, suffer from a number of drawbacks including slow convergence and local minima. Wavelet networks were developed based on ANNs and the wavelet transform theory, as an alternative to feedforward neural networks for the approximation of nonlinear functions, for system modelling, identification and control [5-10]. Using wavelets as activation functions, wavelet networks maintain all the advantages of ANNs while offering a number of additional advantages such as faster convergence as well as providing a much smaller computational overhead due to reduced network sizes. In addition, wavelet networks are capable of handling inputs of higher dimensions.

A number of different modelling techniques exist in literature and the choice of a given technique depends on the requirement from the model as well as the data available. If the structure of the system to be modelled is unknown, simple input output based function mapping may be carried out which is also known as black box modelling. However, in this case, the intricacies of the system are not explicitly identified. More and more emphasis is now being laid on moving from black box modelling towards grey box modelling in which the partially known system structure and approximations are used in developing the network structure and finally to white box modelling in which all system information is available a priori and is incorporated to create an optimum network training structure which will allow for explicit identification of the system parameters [11].

In this research, wavelet networks, both conventional and recurrent, are used for the modelling and identification of a nonlinear electromechanical system. Black box modelling is first carried out and then, based on the a priori knowledge of the system under consideration, a white box approach is adopted in order to identify the linear and nonlinear mechanical parameters that constitute the system.

1.2 Literature Review

Neural networks have been widely used in the area of system modelling and identification for the approximation of continuous nonlinear functions. Wavelet networks were developed based on Artificial Neural Networks (ANN) and the wavelet transform theory, as an alternative to feedforward neural networks for the approximation of nonlinear functions for system modelling and identification. Initial work by Pati and Krishnaprasad in [7], demonstrates how the standard feed forward architecture can be used as a wavelet network given that the activation function satisfies the Morlet-Grossmann admissibility conditions, details of which are provided in Chapter 2. The authors propose the use of an activation function formed through the linear combination of sigmoid functions and highlight the importance of extracting information contained in the training set in order to optimize the network architecture by ensuring the selected activation functions span the spectral range of the given data.

In one of the seminal papers on wavelet networks [5], Zhang and Benenviste were able to successfully develop a wavelet network for black box modelling which not only maintains the universal approximation property inherent in neural networks but also provides an explicit link between the network coefficients and the wavelet transform to allow for better network initialization schemes. In addition, through their work, it is observed that wavelet networks are able to achieve the same approximation quality as traditional feed forward neural networks, with a smaller network size.

In [5-7] and [9], formal initialization procedures were developed in order to improve the efficiency of the wavelet network. In [5], the translation and dilation coefficients are initialized using a regular dyadic grid structure. Grid formation is carried out through division of the input domain into two subintervals by the centre of gravity of the density function of the available data. The translation and dilation coefficients are then selected in each interval and the procedure is repeated within each subinterval until all the translation and dilation coefficients are initialized. In the event that the number of wavelets is not a power of 2, the remaining wavelets are initialized randomly from the finest remaining scale. An alternative initialization method proposed by Zhang in [8], also involves the division of the input domain into a dyadic grid. After grid formation, wavelet selection is carried out based on the least

square error between the observed output and the network output using all wavelets in the grid. The wavelet contributing the least per iteration is eliminated until the number of remaining wavelets equals the number of neurons in the network. In [6], a new correlation based initialization procedure was developed for enhanced network performance. Based on this procedure, a dyadic grid denser in the translation axis is obtained where the number of dilation levels is selected based on the number of neurons in the network. As the wavelet coefficient is a linear correlation coefficient representative of the degree of similarity between the wavelet and the signal to be approximated, once the initialization grid is created, the wavelets with the highest coefficients, and therefore better correlation, are selected. Oussar and Dreyfus provide two other approaches to initializing the translation and dilation parameters in [9]. Using the Heuristic Method, the translation and dilation parameters are selected such that the wavelets extend over the entire input region. The method of Initialization by Selection proposed by Oussar and Dreyfus involves the use of wavelet frames for initialization. A library of wavelets is generated whose dilations are discrete and whose translations lie in the domain of the input vectors given by $[a_k,b_k]$. Taking three successive dilations to ensure the wavelets extend over the entire domain, for each dilation set, the corresponding translation set is selected. Once the library is generated, the direct connection weights are computed using the Least Squares Method and the training sequence for the non-linear part of the model is obtained by subtracting the output of the linear model and the training set. The wavelets are then ranked using the Gram-Schmidt method and selected using the linear model residuals. The remaining network parameters, namely the direct connection weights and the bias weights, are usually initialized to small random values often in the range [0,1] or zeros as in [5] and [9,10,12]. Alternatively, these weights could also be initialized using the Least Squares Method [9].

Training of both neural and wavelet networks used for modelling of static systems is typically carried out using the back propagation type gradient descent algorithm, of which variations are proposed for application to wavelet networks in [5] and [6]. In [5], learning is carried out using the stochastic gradient method. In [6], the training algorithm involves updating the translation and dilation coefficients based on direct minimization techniques while the remaining network parameters are obtained via linear combination. The most important feature of this training procedure involves

the use of a dynamic learning rate which decreases if the error of the current iteration is smaller than the error of the previous iteration and vice versa. Oussar and Dreyfus made used of the Broyden-Fletcher-Golfarb-Shanno gradient algorithm (BFGS) for the training of network parameters [10].

While traditional feedforward neural networks and their conventional wavelet network counterparts discussed above provide a static input/output mapping and have been successfully used for static function approximation as well as system identification and control, the black box modelling of systems with time-varying inputs or outputs is carried out using recurrent networks.

Recurrent wavelet networks (RWN), through internal feedback in the wavelet layer, are able to preserve past network states which allows them to capture the dynamic response of a system with time varying inputs or outputs and adapt quickly to changes in the system [13-16]. The proposed four layer network architecture consists of an input layer, a wavelet layer with self-feedback, a product layer and an output layer. This architecture is a generalized form of the conventional wavelet network since the RWN structure is the same as that of the conventional wavelet network when the self-feedback weights, which represent the rate of information storage, are zero. In [17], Lin et. al propose an alternate four layer architecture. The fundamental difference between the two is the presence of direct weighted connections between the input and the output layer in [13-16] which provide the RWN with the added advantage of improved extrapolation outside the training data and allow for initialization based on process knowledge. An alternate five layer structure is proposed by Lu in [18] in which an adaptive node layer was added called the consequent part of the network. The activation function is selected as the first derivative of the Gaussian function in [13-16], [18] and [16] while the differentiable Mexican hat wavelet was used in [17].

In order to effectively a train the RWN to model a nonlinear dynamic system, it is essential to understand the different models that can be used to represent nonlinear dynamic systems in order to select the best training structure and best training inputs for the network. Four different models were discussed in [11], which are shown in Equations 1.1 to 1.4. The model shown in Equation 1.1 assumes the

system output is a nonlinear function of the input and a linear function of the delayed output.

$$y(n+1) = f\big(u(n), u(n-1) \dots u(n-m+1)\big) + \sum_{i=1}^{k-1} \alpha_i y(n-i) \qquad (1.1)$$

where u represents the input signal, y represents the output, α represents a linear coefficient, f represents an unknown nonlinear function and the discrete time index $m \leq k$.

Similarly, the model represented by Equation 1.2 assumes the output varies linearly with the input and nonlinearly with the delayed output.

$$y(n+1) = f\big(y(n), y(n-1) \dots y(n-k+1)\big) + \sum_{i=1}^{m-1} \alpha_i u(n-i) \qquad (1.2)$$

In the third model described by Equation 1.3, the output depends nonlinearly on both delayed inputs and delayed outputs in a separable manner.

$$y(n+1) = f\big(y(n), y(n-1) \dots y(n-m+1)\big) + g\big(u(n), u(n-1) \dots u(n-k+1)\big) \qquad (1.3)$$

The fourth model given by Equation 1.4 is a generalized model which assumes the output varies nonlinearly with both the delayed inputs and delayed outputs.

$$y(n+1) = f\big(y(n), y(n-1) \dots y(n-m+1), u(n), u(n-1) \dots u(n-k+1)\big) \qquad (1.4)$$

Training of recurrent networks can be done using a number of different training algorithms, the most popular of which are the backpropagation type gradient descent algorithm and real time recurrent learning (RTRL) [1-4,19]. While these methods typically make use of static learning rates, it was observed in [1] and [20] that such static learning rates were impractical since not only is the selection of the

optimal leaning rate a trial and error based process but also because very large learning rates lead to system instability and small learning rates lead to slow network training. As a result, in order to ensure network stability and to guarantee convergence, adaptive learning rates were developed based on the Lyapunov Stability Theory, for use in recurrent neural networks [20-21]. This method was then adopted for use in RWN [13-17], however recurrent terms in the network equations were assumed negligible when solving for the adaptive learning rates.

In [13], two RWNs were used to generate two control inputs, namely the translational and rotational displacements, for the stable path tracking of a mobile robot as shown in Figure 2. Here, the authors made use of ALRs for training of the network based on the gradient descent algorithm.

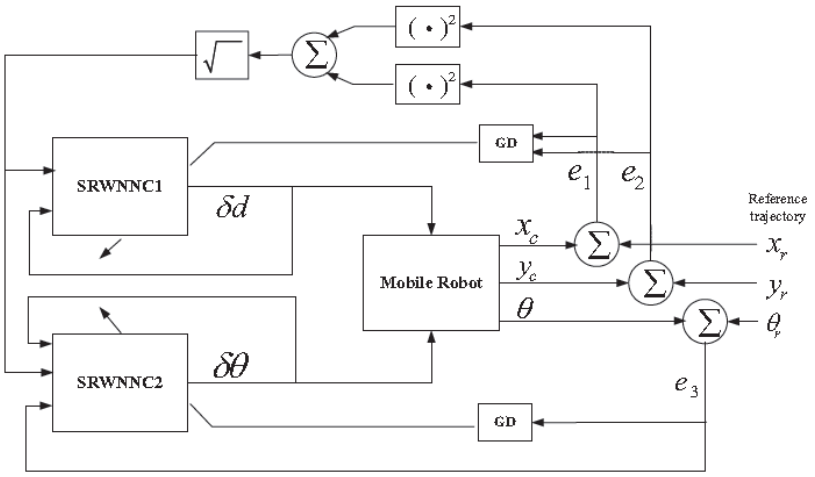


Figure 2: RWNN-Based Mobile Robot Control Structure [13]

In [18], Lu developed a stable predictive control (SPC) scheme based on the RWN as shown in Figure 3. The nonlinear modelling of the system is carried out using a RWN where the gradient descent algorithm with ALR is employed for the precondition part of the network for updating the translation and dilation coefficients and the feedback weights while the consequent parameters are identified using the recursive least squares method. As in [13], network convergence is guaranteed using Lyapunov's Stability Theorem which ensures the learning rate remains within the stable region.

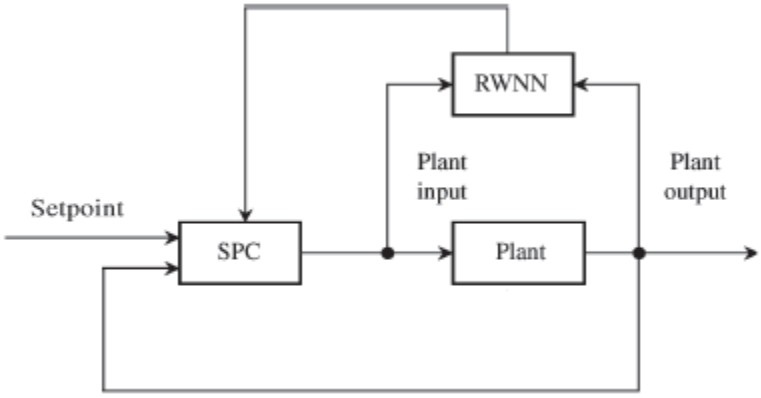


Figure 3: RWNN-Based SPC Scheme [18]

In [14] and [16], dynamic system identification is carried out using the series parallel method as shown in Figure 4 where the inputs of the RWN identifier are the current input and the previous output of the dynamic system.

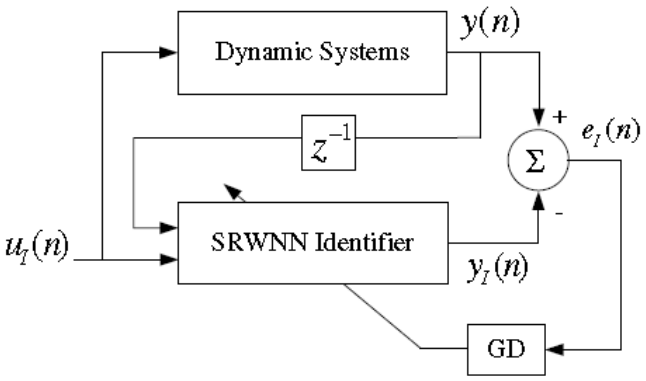


Figure 4: RWN Based Dynamic System Identification Architecture [14]

Training of the dynamic system identifier is carried out offline using ALR based gradient descent method while an indirect online adaptive control technique for a RWN is used to control the system [14]. The RWN controller inputs consist of the reference signal and the last plant output as shown in Figure 5. The proposed control system is applied to the Duffing and water bath system and the performance of the

RWN controller with ALR is compared to that of a traditional wavelet network and an RWN with static learning rates. It is seen that for a fixed number of iterations, the RWN with ALR outperforms the other two controllers showing faster convergence and lowest mean square error. The ability of the RWN controller to recover from disturbances is also tested and is seen to have a fast rejection capability.

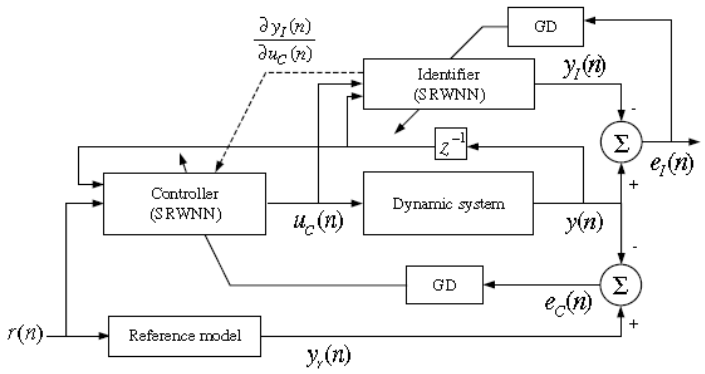


Figure 5: Indirect Adaptive Control Architecture Using RWNs [14]

In the previous papers, emphasis was laid on black box modelling and using RWNs in order to control highly nonlinear systems. However, since black box models are unable to provide the intrinsic details of the system being modelled [11], research is now looking towards the application of neural and wavelet networks for parameter identification of nonlinear systems.

In [22], the authors make use of a partially recurrent network known as an Elman Recurrent Network in order to identify the mechanical and electrical parameters of a linear DC motor without nonlinear friction. The structure of the network is selected such that it is equivalent to the state space equations of the DC motor. In this case, the back propagation algorithm is not used; rather, the authors favour the use of Genetic Algorithms (GA) for updating the network parameters. All parameters are estimated except the resistance of the DC motor which is assumed known and the network is seen to produce satisfactory results.

In [23-24], the authors provide some of the leading work done in the use of structured recurrent neural networks in order to identify the parameters of nonlinear

systems. Full use is made of a priori knowledge of the system at hand in order to carry out intelligent modelling and identification. In [23], simultaneous identification of the linear and nonlinear parts of the system is accomplished by first discretizing the system in order to create a structured recurrent neural network where training is carried out using the gradient descent algorithm. The nonlinearity in the system is learnt by a Radial Basis Function Network (RBFN) or Multi-Layer Perceptron Network (MLP). Simulations are carried out on a two-body system coupled by a damped elastic spring in order to identify the linear parameters and the friction torque which acts as the nonlinearity in the system. In [23], no stability analysis is carried out but the range of the unknown system parameters are limited to ensure overall system stability. The work done in [24] follows the same principle of creating a structured network in order to model and identify the parameters of a system with an isolated nonlinearity. The authors mention, however, that the structure is only capable of correctly identifying the system parameters provided the system being modelled is unique. In [24], identification is carried out for a multi stand rolling system. The parameters are all correctly identified, however, due to the complexity and size of the system, convergence time is very large. In addition static learning rates are used which may contribute to the slow speed of convergence.

In [25], a structured recurrent network is developed for a nonlinear two mass system which is trained using the Levenberg-Marquardt (LM) algorithm which involves computation of the Jacobian using Real Time Recurrent Learning (RTRL). The training is carried out in a quasi-online fashion in order to be able to use the LM optimization algorithm; however, it is observed that this kind of training and optimization might lead to getting trapped in local minima thereby giving incorrect desired values.

1.3 Objectives of Research

The aim of this thesis is to build and expand on the work done in [13] and [23] in order to develop a structured Recurrent Wavelet Network (RWN) which will not only provide an accurate representation of the real plant but will also be able to identify linear mechanical parameters as well as static or time-varying nonlinearities of the system. The first part of the thesis is dedicated to developing a RWN model as

well as a structured RWN based on the system under investigation. Mathematical modelling of the system is carried out and stability analysis is carried out based on the Lyapunov theory in order to derive the adaptive learning rates (ALR). The second part of the thesis involves carrying out online identification of the system and observing the effects of changing RWN parameters on the learning capabilities of the network. The original contributions of this thesis include developing a structured RWN for simultaneous linear and nonlinear mechanical parameter identification for a DC Motor and the derivation and application of ALRs using the Lyapunov stability theory.

1.4 Thesis Organization

This work is organized as follows. Chapter 1 gives a general overview of the topics under consideration and provides a comprehensive literature review addressing the work that has been accomplished in this field. Chapter 2 introduces wavelets, conventional Wavelet Networks (WN) and the various initialization and training algorithms available. The use of WN for the modelling of static nonlinear systems is studied. Chapter 3 presents the Recurrent Wavelet Network (RWN) together with a complete study on stability analysis. The derivation of adaptive learning rates to ensure network stability is also carried out for the black box modelling of a DC Motor. Chapter 4 builds on the RWN and involves the design and construction of structured RWN for a DC Motor. The effectiveness of the structured RWN in identifying the linear and nonlinear mechanical parameters of the system is detailed in Chapter 5. The conclusion of this research and recommendations for future work are provided in Chapter 6.

Chapter 2: Wavelet Networks

2.1 Wavelets

In order for a function to be considered a mother wavelet in the Morlet-Grossmann sense, certain admissibility conditions, given below, must be satisfied [5]. In essence, mother wavelets should be band pass signals.

- Zero mean
- Oscillatory
- Fast decay to zero
- $C_{\psi_s} = \int_0^{\infty} \frac{|\psi_s(\omega)|^2}{\omega} d\omega < \infty$

A set of daughter wavelets is constructed through the translation and dilation of the mother wavelet *h(t)* as given in Equation 2.1 where *m* and *d* are the translation and dilation coefficients respectively. Using a set of the daughter wavelets, it is possible to approximate a signal.

$$h_{m,d}(t) = h\left(\frac{t-m}{d}\right) \tag{2.1}$$

One of the popularly used wavelets is the first derivative of the Gaussian function given by Equation 2.2 and shown in Figure 6.

$$\phi(x) = -xe^{-\frac{x^2}{2}} \tag{2.2}$$

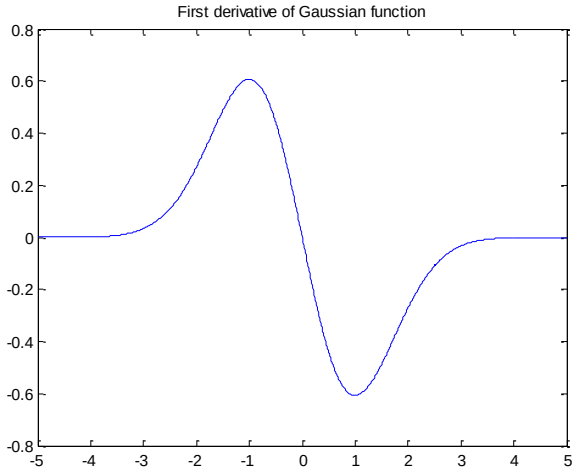


Figure 6: First Order Derivative of Gaussian Function

The Morlet wavelet, also known as the Cos-Gaussian function, is given by Equation 2.3 and shown in Figure 7.

$$\phi(x) = e^{-\frac{x^2}{2}}\cos(\omega_o x) \tag{2.3}$$

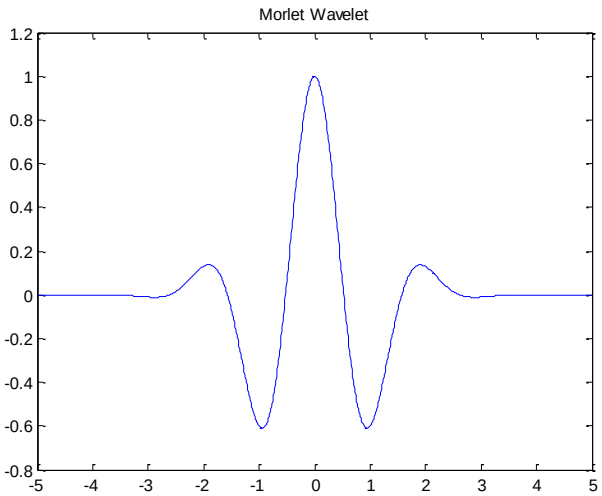


Figure 7: Morlet Wavelet

A number of other mother wavelets are commonly used including the Mexican Hat wavelet, the Haar wavelet, the Meyer wavelet and the Daubechies wavelet.

2.2 Conventional Wavelet Networks

The conventional wavelet network is used for static modelling of a nonlinear system. The architecture along with the initialization and training algorithms are provided in the following sections.

2.2.1 Architecture. The architecture of a conventional MIMO wavelet network consisting of N_i inputs, N_w wavelets in the hidden layer and $N_o$outputs is shown below in Figure 8.

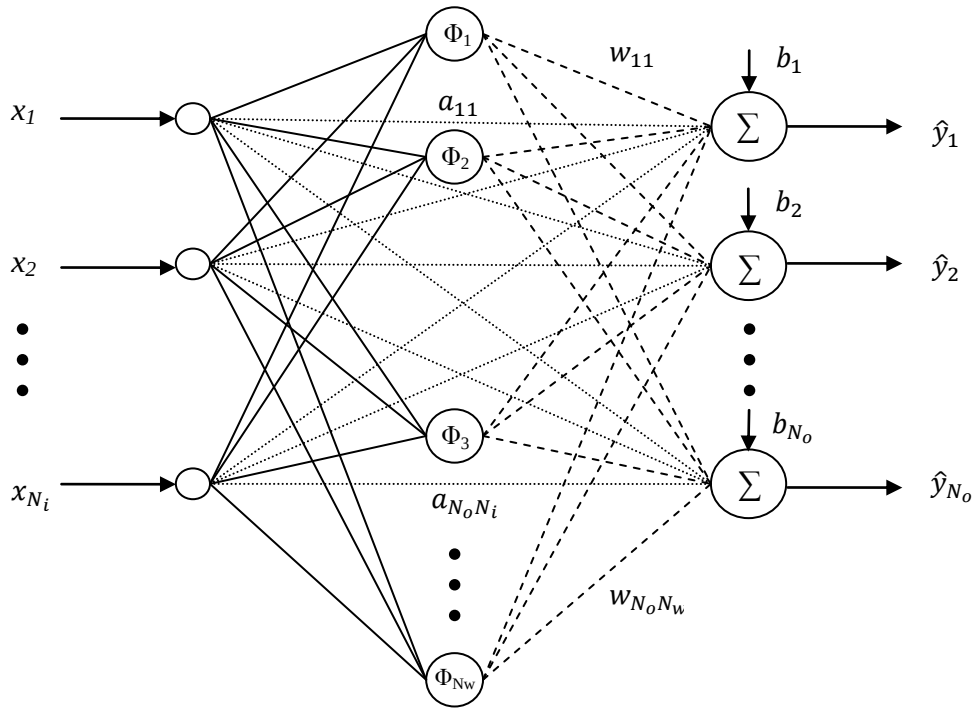


Figure 8: Conventional Wavelet Network Architecture

Each output of the wavelet network is given by Equation 2.4,

$$\hat{y}_k = b_k + \sum_{j=1}^{N_w} w_{kj}\Phi_j + \sum_{i=1}^{N_i} a_{ki}x_i \tag{2.4}$$

where a_{ki} represents the weights of the direct connections between the outputs and the inputs, w_{kj} represents the weight of the outputs of the neurons and b_k represents

the weight of the output bias. Φ_j is a multidimensional wavelet given as the product of N_i scalar wavelets as shown in Equation 2.5,

$$\Phi_j = \prod_{i=1}^{N_i} \varphi\left(\frac{x_i - m_{ji}}{d_{ji}}\right) = \prod_{i=1}^{N_i} \varphi(z_{ji}) \tag{2.5}$$

where m_{ji} is the translation coefficient and d_{ji} is the dilation coefficient.

In this thesis, the first derivative of the Gaussian function is chosen as the mother wavelet as given in Equation 2.6.

$$\varphi(z_{ji}) = -z_{ji} e^{-\frac{{z_{ji}}^2}{2}} \tag{2.6}$$

The complete set of network parameters is given by the vector $\theta = \{b_k, a_{ki}, w_{kj}, m_{ji}, d_{ji}\}$.

2.2.2 Initialization algorithms. The network parameters can be initialized using a number of different techniques as in [5], [6] and [9]. Typically, the weights of the direct connections, the weights from the neurons to the outputs and the output bias weights are initialized to small random values. The Least Squares Method can also be used to initialize the direct connection and bias weights a_{ki} and b_k respectively. Based on the Least Squares Method, the weights b_k and a_{ki} will be determined using Equation 2.7.

$$\theta_{init} = (x^T x)^{-1} x^T Y \tag{2.7}$$

where *x* is the matrix of inputs and *Y* is the matrix of the desired outputs.

Several different methods to initialize the translation and dilation coefficients have been developed such as the heuristic method, initialization by selection, as well as several variations of the dyadic grid method.

In the heuristic method, proposed by Oussar and Dreyfus in [9], the maximum and minimum of each input are determined as a_i and b_i respectively. The translation and dilation coefficients are then selected using Equations 2.8 and 2.9, to ensure the wavelets extend over the entire input domain.

$$m_{ji} = 0.5(a_i + b_i) \tag{2.8}$$

$$d_{ji} = 0.2(b_i - a_i) \tag{2.9}$$

The dyadic grid method, introduced by Zhang [5] and elaborated on in [6], involves selection of the translation and dilation coefficients based on the division of the input domain into a dyadic grid. Grid formation is carried out through the division of the input domain, given by [a,b], into two subintervals by *p*, the centre of gravity of the density function of the available data. The translation and dilation coefficients are selected using Equations 2.10 and 2.11.

$$m_1 = p \tag{2.10}$$

$$d_1 = \zeta(b - a), \zeta = 0.5 \tag{2.11}$$

The procedure is repeated within each subinterval until all the translation and dilation coefficients are initialized. In the event that the number of wavelets is not a power of 2, the remaining wavelets are initialized randomly from the finest remaining scale. The resulting dyadic grid formation from which the wavelets are selected is shown in Figure 9.

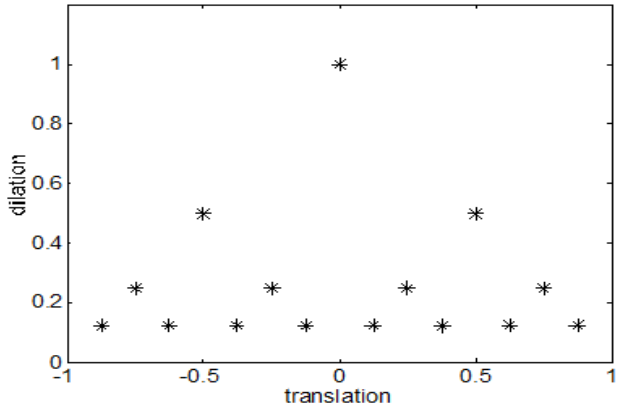


Figure 9: Dyadic Grid [5]

2.2.2 Training algorithms

2.2.3.1 Gradient descent algorithm. Training a conventional wavelet network is carried out using the gradient descent algorithm which involves adjusting the network parameters, θ, to ensure the minimization of a cost function given by Equation 2.12.

$$J(\theta) = \frac{1}{2}\sum_{k=1}^{N_o}\sum_{p=1}^{N_p}\left(y_{kp} - \hat{y}_{kp}\right)^2 = \frac{1}{2}\sum_{k=1}^{N_o}\sum_{p=1}^{N_p}\left(e_{kp}\right)^2 \tag{2.12}$$

where e_{kp} is the error between the desired output and the output of the wavelet network for a pattern, p.

Using Equation 2.4, the partial derivative of the cost function with respect to the network parameters is given in Equation 2.13.

$$\frac{\partial J}{\partial \theta} = -\sum_{p=1}^{N_p} e_{kp}\frac{\partial \hat{y}_{kp}}{\partial \theta} \tag{2.13}$$

The partial derivative of the network output $\hat{y}_{kp}$ with respect to the network parameters is given in Equations 2.14 to 2.18.

$$\frac{\partial \hat{y}_{kp}}{\partial b_k} = 1 \tag{2.14}$$

$$\frac{\partial \hat{y}_{kp}}{\partial a_{ki}} = x_i \tag{2.15}$$

$$\frac{\partial \hat{y}_{kp}}{\partial w_{kj}} = \Phi_j \tag{2.16}$$

$$\frac{\partial \hat{y}_{kp}}{\partial m_{ji}} = \frac{\partial \hat{y}_{kp}}{\partial \Phi_j} . \frac{\partial \Phi_j}{\partial z_{ji}} . \frac{\partial z_j}{\partial m_{ji}} = - \frac{w_{kj}}{d_{ji}} . \frac{\partial \Phi_j}{\partial z_{ji}} \tag{2.17}$$

$$\frac{\partial \hat{y}_{kp}}{\partial d_{ji}} = \frac{\partial \hat{y}_{kp}}{\partial \Phi_j} . \frac{\partial \Phi_j}{\partial z_{ji}} . \frac{\partial z_j}{\partial d_{ji}} = -w_{kj} . \frac{x_i - m_{ji}}{{d_{ji}}^2} . \frac{\partial \Phi_j}{\partial z_{ji}} = - \frac{w_{kj}}{d_{ji}} . z_{ji} . \frac{\partial \Phi_j}{\partial z_{ji}} \tag{2.18}$$

The partial derivative of the multidimensional wavelet function with respect to z_{ji} is given by Equation 2.19.

$$\frac{\partial \Phi_j}{\partial z_{ji}} = \varphi(z_{j1}) . \varphi(z_{j2}) . \ldots \ . \varphi'(z_{ji}) . \ldots \ . \varphi(z_{jN_i}) \tag{2.19}$$

Each parameter is then updated using Equation 2.20, where μ is the learning rate, γ is the momentum coefficient and *n* is the iteration index.

$$\Delta\theta(n) = -\mu \frac{\partial J}{\partial \theta} + \gamma \Delta\theta(n-1) \tag{2.20}$$

The parameters will continue to be updated until the mean squared error, computed as in Equation 2.21, reaches a particular desired value.

$$MSE = \frac{1}{N_p} \sum_{p=1}^{N_p} (y_p - \hat{y}_p)^2 \tag{2.21}$$

2.2.3.2 Levenberg-Marquardt algorithm. A conventional wavelet network can also be trained using the Levenberg-Marquardt (LM) algorithm which is a combination of the gradient descent method and the Gauss Newton method [26]. Training involves adjusting the network parameters, θ, to ensure the minimization of a cost function given by Equation 2.22.

$$V(\theta) = \sum_{k=1}^{N_o} \sum_{p=1}^{N_p} \left(y_{kp} - \hat{y}_{kp}\right)^2 = \sum_{k=1}^{N_o} \sum_{p=1}^{N_p} \left(e_{kp}\right)^2 \tag{2.22}$$

where e_{kp} is the error between the desired output and the output of the wavelet network for a pattern, *p*, and is a function of the network parameters θ.

Using Equation 2.4, the partial derivatives of the network output with respect to the network parameters are computed in order to form the Jacobian matrix as given in Equation 2.23.

$$J = \begin{bmatrix} \frac{\partial \hat{y}_{k,1}}{\partial b} & \frac{\partial \hat{y}_{k,1}}{\partial a_{ki}} & \frac{\partial \hat{y}_{k,1}}{\partial w_{kj}} & \frac{\partial \hat{y}_{k,1}}{\partial m_{ji}} & \frac{\partial \hat{y}_{k,1}}{\partial d_{ji}} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \frac{\partial \hat{y}_{k,Np}}{\partial b} & \frac{\partial \hat{y}_{k,Np}}{\partial a_{ki}} & \frac{\partial \hat{y}_{k,Np}}{\partial w_{kj}} & \frac{\partial \hat{y}_{k,Np}}{\partial m_{ji}} & \frac{\partial \hat{y}_{k,Np}}{\partial d_{ji}} \end{bmatrix} \tag{2.23}$$

Each parameter is then updated using Equation 2.24, where μ is the learning rate, *J* is the Jacobian matrix, *I* is the identity matrix and *Y* and $\hat{Y}$ are the matrices of the desired network output and the actual network outputs respectively.

$$\Delta\theta = (J^T J + \mu I)^{-1} J^T (Y - \hat{Y}) \tag{2.24}$$

The network output is then calculated for the new values of the network parameters and the cost function given in Equation 2.25, $V(\theta + \Delta\theta)$, is recomputed. If the sum of the square of errors is seen to be reduced from the initial calculation, the learning rate μ is reduced by a factor of β, $\theta_{new} = \theta_{old} + \Delta\theta$ and the process then repeated. If the sum of square of errors is not reduced then the learning rate μ is increased by a

factor of β, $\Delta\theta$ is recomputed and $V(\theta + \Delta\theta)$ calculated again. The algorithm is said to converge when the sum of the squares of the error has reduced to the target value or when a certain number of iterations has been reached. In this way the LM algorithm can be used for offline batch training of a CWN. The LM algorithm can be summarized in the form of a flowchart as shown in Figure 10.

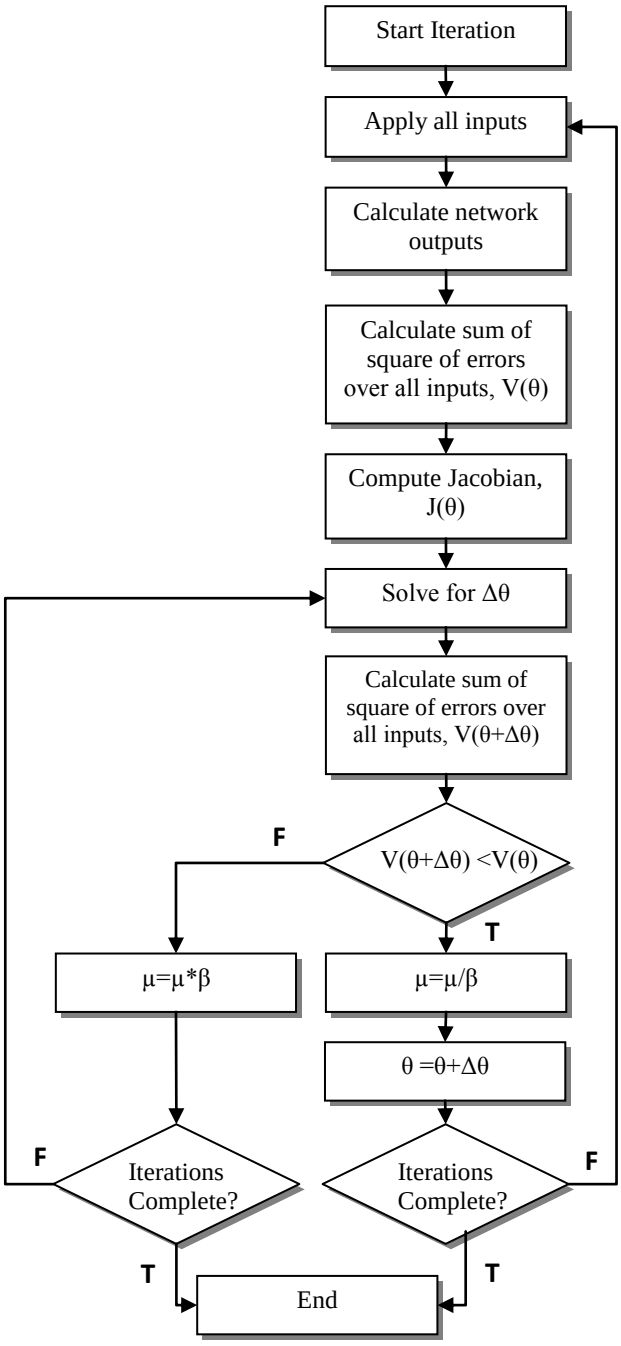


Figure 10: Levenberg Marquardt Flowchart

2.3 Nonlinear Function Approximation

Figure 11 shows a block diagram of the training structure for a wavelet network used for the approximation of static nonlinear functions.

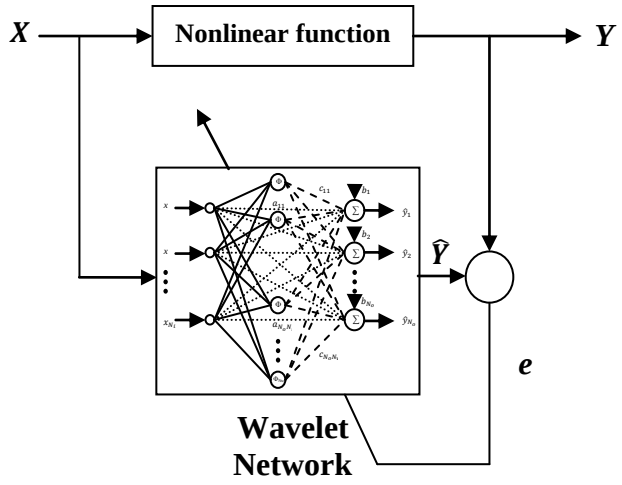


Figure 11: Approximation of Static Nonlinear Function

This is a black box model in which the inputs are presented to the wavelet network which then trains so as to minimize the error between the network output and the output of the original system. MATLAB and Simulink were used to simulate the training of the network for a single input single output (SISO) and a multi input single output (MISO) nonlinear function. Both functions to be approximated were taken from seminal papers [8,9] on wavelet modelling. The default activation function used is the first derivative of the Gaussian.

For both cases, offline training was carried out using wavelet networks initialized using the heuristic method and the dyadic grid method in order to compare the initialization techniques in terms of speed of convergence and mean squared error (MSE). Further, the effect of changing the number of neurons in the hidden layer was investigated. The effect of changing the learning rates was also studied. In each case, the system performance was evaluated based on the mean square error obtained after a fixed number of iterations. The next step involved comparing the performance of the systems when using different activation functions. Finally, the gradient descent

method for training was compared with the Levenberg Marquardt (LM) algorithm in order to determine which method is more efficient for offline training.

Online training was then carried out for a SISO Gaussian function and a MISO system using different activation functions to compare the network efficiency.

2.3.1 Offline training.

2.3.1.1 SISO system. The nonlinear function to be approximated is given by Equation 2.25 and shown in Figure 12.

$$f(x) = \begin{cases} -2.186x - 12.864 & x \in [-10, -2) \\ 4.246x & x \in [-2, 0) \\ 10e^{-0.05x-0.5}\sin(0.03x^2 + 0.7x) & x \in [0, 10) \end{cases} \quad (2.25)$$

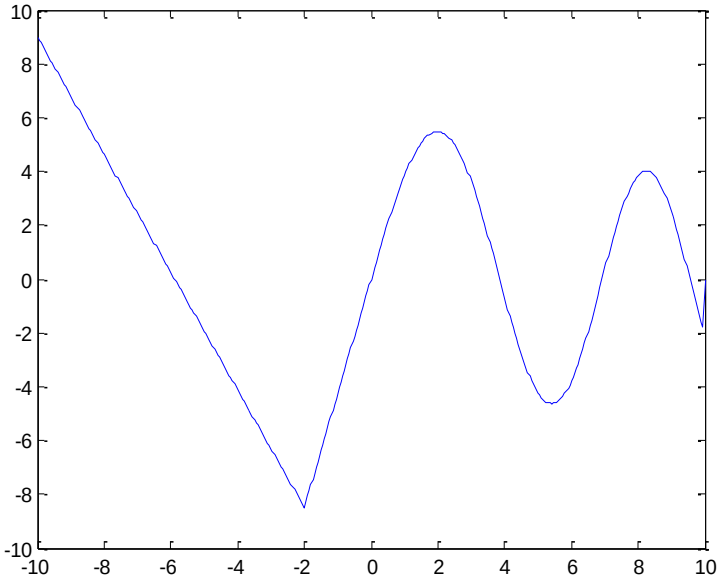


Figure 12: SISO Nonlinear Function

The translation and dilation coefficients were initialized using the heuristic method while the weights of the direct connections, hidden layer to output weights and the output bias weights were initialized using the least squares method. The number of iterations was fixed to 10000. In order to determine the effect of changing the number of neurons in the hidden layer on the MSE, the learning rate and

momentum rate were set to 0.01 and 0.4 respectively and the number of neurons was varied. The simulation was the repeated, initializing the translation and dilation coefficients using the dyadic grid method, keeping all other conditions identical. The results are tabulated in Table 1 and shown graphically in Figures 13 and 14.

Table 1: MSE for Varying Nw for SISO Static Function Approximation using Heuristic and Dyadic Grid Method

Iterations	Nw	Heuristic Method MSE	Dyadic Grid Method MSE
10000	7	0.6208	0.3019
	15	1.0511	0.0828
	31	1.1262	0.1292

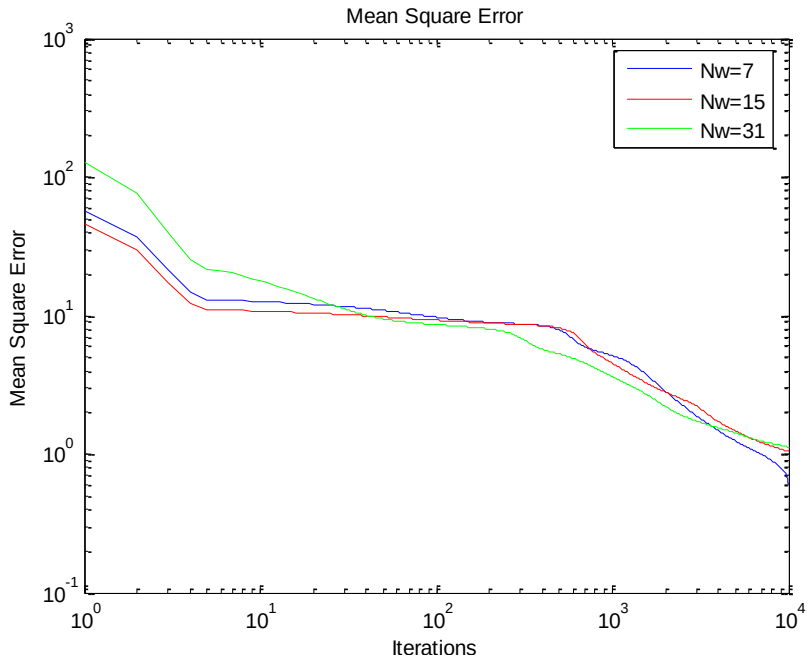


Figure 13: MSE for Varying Nw for SISO Static Function Approximation using Heuristic Method

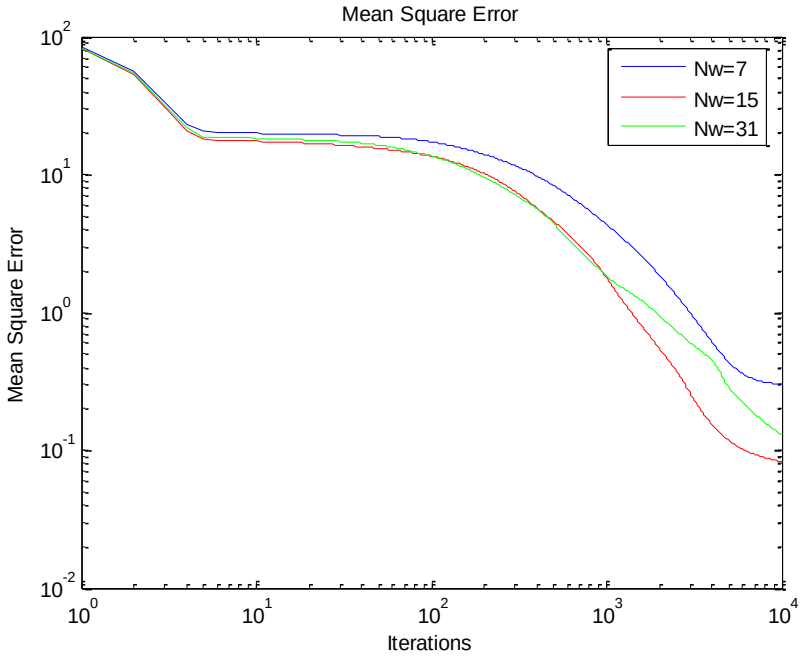


Figure 14: MSE for Varying Nw for SISO Static Function Approximation using Dyadic Grid Method

The number of neurons was then fixed at 15 and the momentum rate fixed at 0.6. The effect of altering the learning rate was then investigated and the MSE after 10000 iterations was determined for both cases when the translation and dilation coefficients are initialized using the heuristic method and the dyadic grid method. The results are tabulated in Table 2 and a graphical representation is shown in Figures 15 and 16.

Table 2: MSE for Varying μ for SISO Function Approximation using Heuristic and Dyadic Grid Method

Iterations	**μ**	**Heuristic Method MSE**	**Dyadic Grid Method MSE**
10000	0.01	0.7853	0.0232
	0.02	0.2841	0.0190
	0.04	0.0983	0.0170
	0.08	0.0198	0.0103

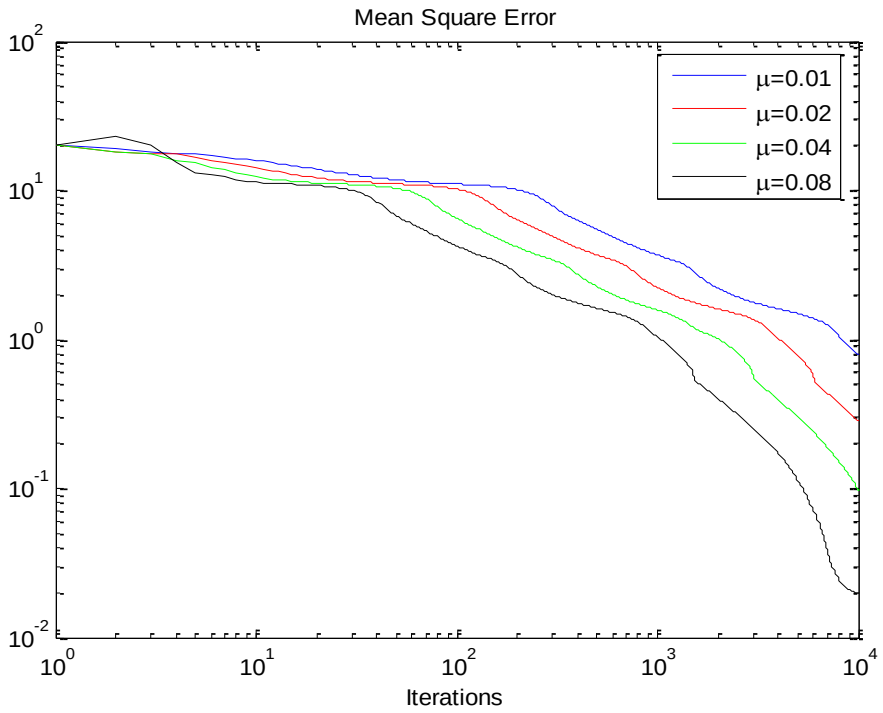


Figure 15: MSE for Varying μ for SISO Static Function Approximation using Heuristic Method

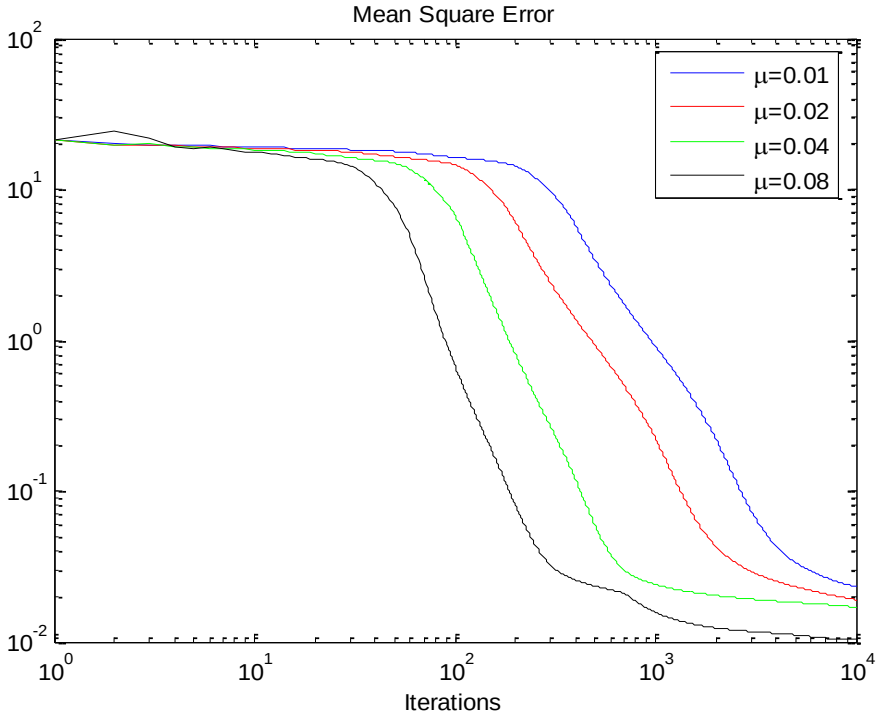


Figure 16: MSE for Varying μ for SISO Static Function Approximation using Dyadic Grid Method

From the results obtained it was observed that the network initialized using the dyadic grid method performed better that the one initialized using the heuristic method. Faster convergence and lower MSE were observed. It was also seen that for this function, smaller network sizes performed better than larger networks. Keeping the momentum rate fixed, an increase in the learning rate also served to reduce the MSE. An important observation was that the selection of the learning and the momentum rates was arbitrary being done through trial and error in order to determine the ranges in which the network would be able to model the system.

Next, a comparison was drawn between the gradient descent algorithm and the LM algorithm. The network trained using the LM algorithm consists of a hidden layer with 15 neurons with a learning rate of 0.1 and a β of 2. The performance of this network was compared to a network trained using the gradient descent algorithm with a learning rate of 0.08 and momentum rate of 0.6 which is initialized using the dyadic grid method. Training was carried out for 100 iterations.

Table 3 and Figure 17 show the MSE of the network trained using each of the two algorithms. Figure 18, which compares the network outputs for both cases, shows that the LM algorithm provides much faster convergence and better response.

Table 3: MSE for LM and GD Trained Networks for SISO Static Function Approximation

	Mean Square Error	
Iterations	LM Algorithm	Gradient Descent Algorithm
100	0.01081	0.6716

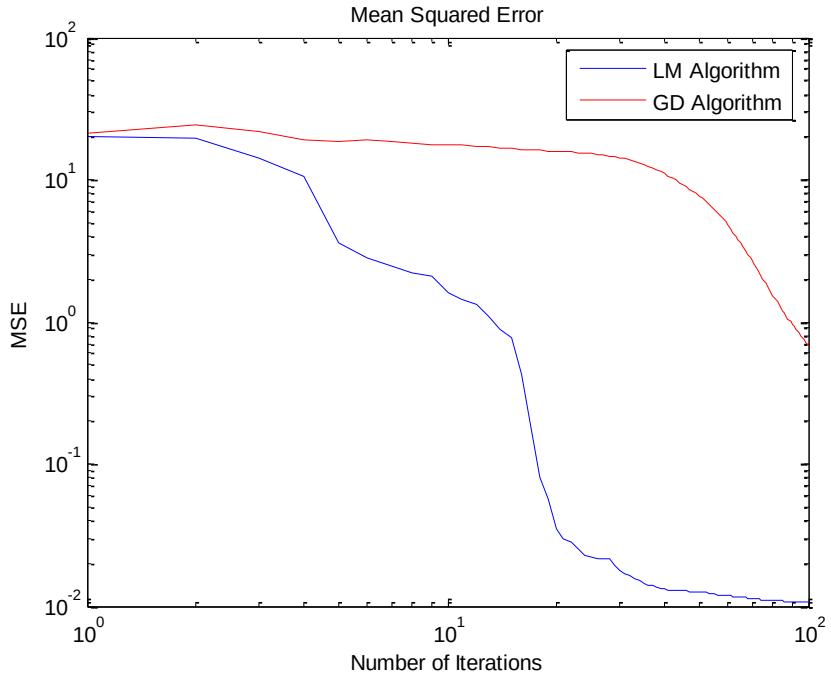


Figure 17: MSE for LM and GD Trained Networks for SISO Static Function Approximation

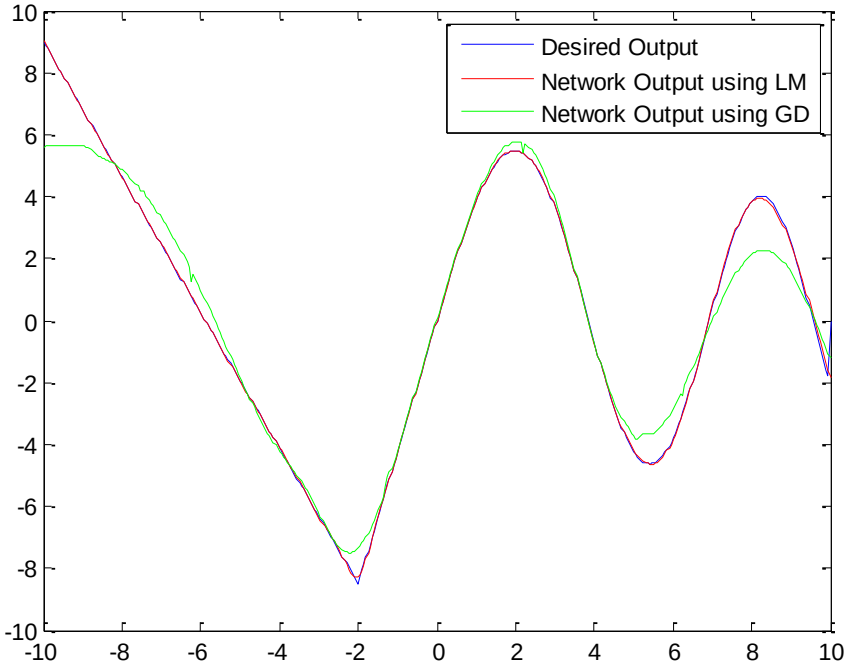


Figure 18: Output of Networks Trained using LM and GD Algorithms

The effect of changing the activation function on the network performance was tested for a wavelet network with 15 neurons in the hidden layer, with learning and momentum rates set to 0.08 and 0.6 respectively. Initialization of the translation and dilation parameters was done using the dyadic grid method. The gradient descent algorithm was used for training. Table 4 shows the MSE after 1000 iterations for the network which was tested using the first derivative of the Gaussian function, the Mexican Hat wavelet and the Morlet wavelet as activation functions. It is seen that the first derivative of the Gaussian function provides the best network performance.

Table 4: MSE for Different Activation Functions for SISO Static Function Approximation

	Mean Square Error		
Iterations	1st Derivative of Gaussian	Mexican Hat	Morlet
1000	0.0158	0.0301	0.0430

Figure 19 shows the comparison of the MSE for the three different activation functions and Figure 20 shows the comparison in the network output after 1000 iterations.

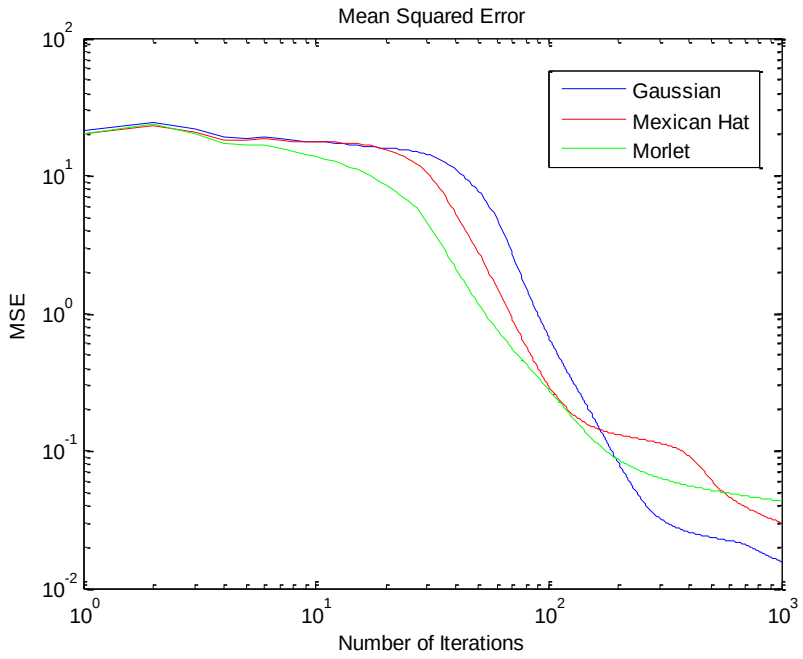


Figure 19: MSE for Network Trained using Different Activation Functions

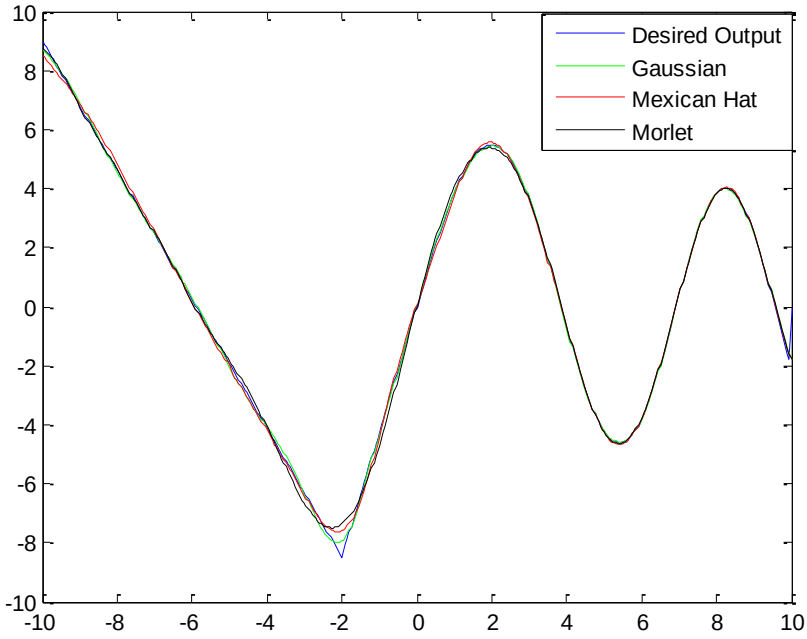


Figure 20: Output of Networks Trained using Different Activation Functions

2.3.1.2 MISO system. The nonlinear function to be approximated is given by Equation 2.26 and shown in Figure 21.

$$f(x_1, x_2) = 1.335(1.5 - 1.5x_1) + e^{2x_1 - 1} \sin(3\pi(x_1 - 0.6)^2) + e^{3x_2 - 1.5} \sin(4\pi(x_2 - 0.9)^2) \quad (2.26)$$

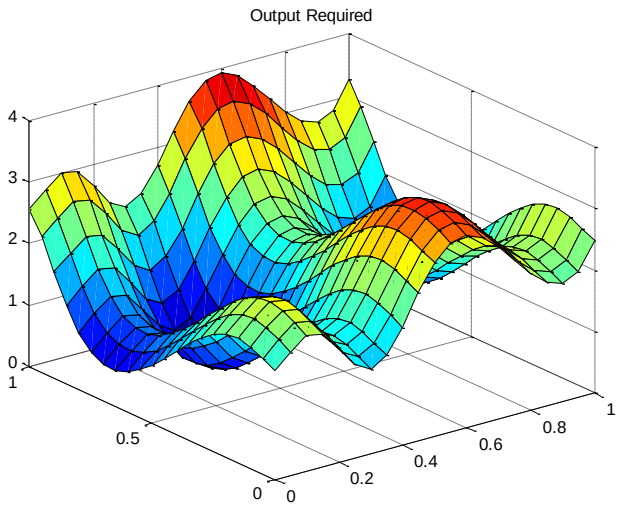


Figure 21: MISO Nonlinear Function

The translation and dilation coefficients were initialized using the heuristic method while the weights of the direct connections, hidden layer to output weights and the output bias weights were initialized to small random values. The simulation was run for 10000 iterations. In order to study the effect of changing the number of neurons in the hidden layer the learning rate and momentum rate were fixed to 0.1 and 0.6 respectively. The simulation was the repeated, this time initializing the translation and dilation coefficients using the dyadic grid method, keeping all other conditions identical. The results are tabulated in Table 5 and shown graphically in Figures 22 and 23.

Table 5: MSE for Changing Nw for MISO Function Approximation using Heuristic Method

Iterations	Nw	Heuristic Method MSE	Dyadic Grid Method MSE
10000	7	0.0136	0.0104
	15	5.0251e-4	4.4011e-4
	31	7.6238e-4	2.3128e-4

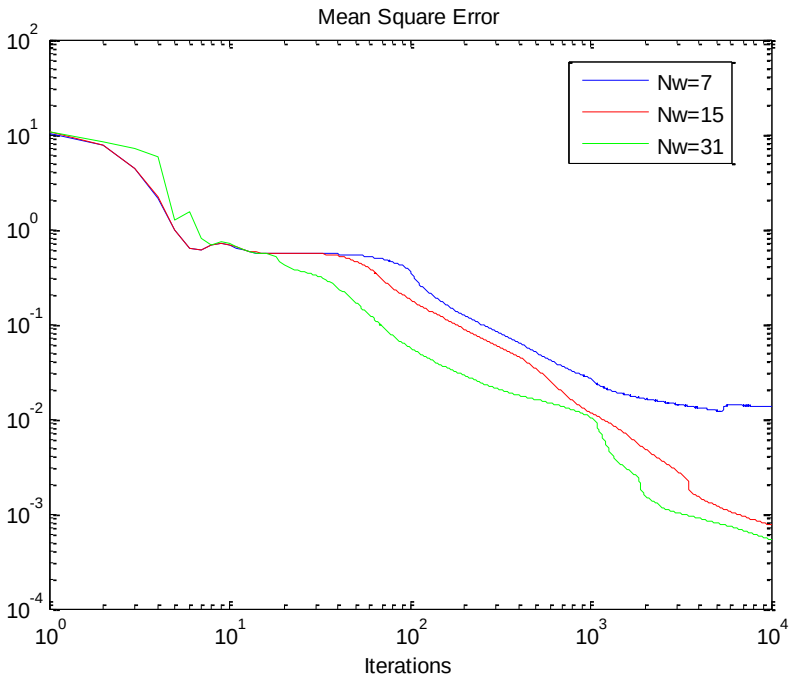


Figure 22: MSE for Varying Nw for MISO Function Approximation using Heuristic Method

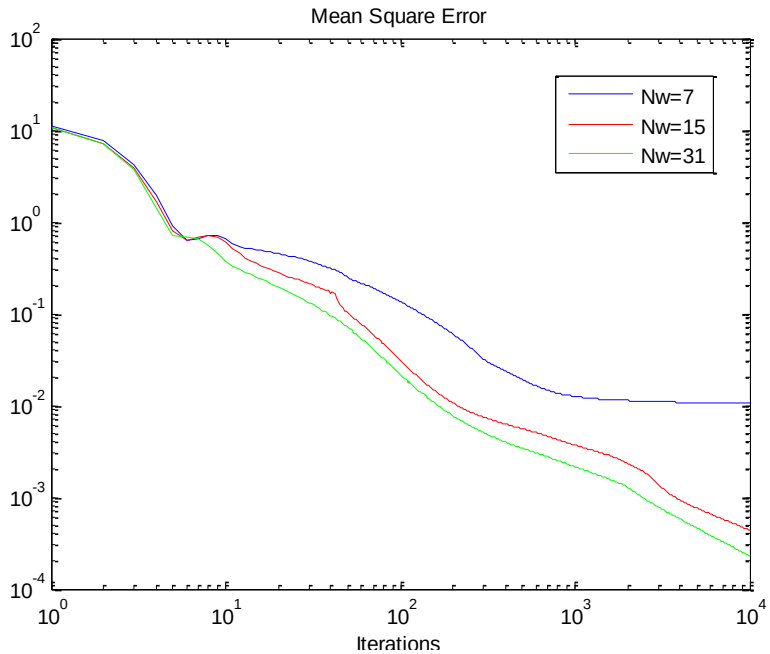


Figure 23: MSE for Varying Nw for MISO Function Approximation using Dyadic Grid Method

The number of neurons was then fixed at 31 and the momentum rate fixed at 0.6. The effect of altering the learning rate was then investigated and the MSE after 10000 iterations was determined for both cases when the translation and dilation coefficients are initialized using the heuristic method and the dyadic grid method. The results are tabulated in Table 6 and a graphical representation is shown in Figures 24 and 25.

Table 6: MSE for Changing μ for MISO Function Approximation using Heuristic Method

Iterations	μ	Heuristic Method MSE	Dyadic Grid Method MSE
10000	0.02	0.0028	7.4482e-4
	0.05	0.0013	3.3013e-4
	0.08	7.5450e-4	4.0521e-4

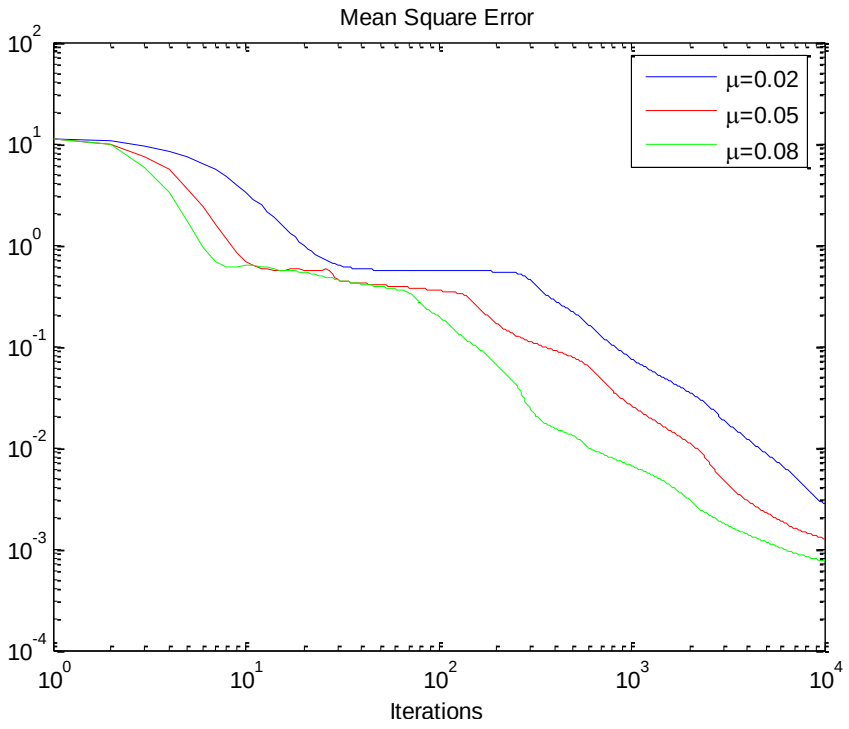


Figure 24: MSE for Varying μ for MISO Function Approximation using Heuristic Method

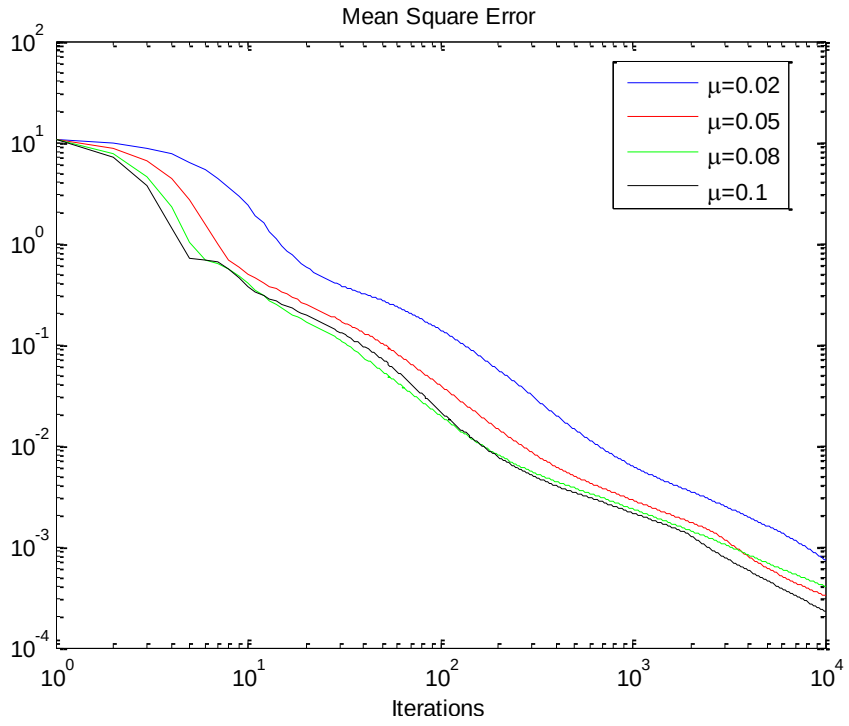


Figure 25: MSE for Varying μ for MISO Function Approximation using Dyadic Grid Method

From the results, it is seen that for this system, the network performs better when the number of neurons are increased. In addition, the dyadic grid initialization method provided better results with the network converging faster and producing a smaller MSE. Trial and error was used again in the selection of the momentum rate and the learning rate. Increasing the learning rate up to 0.1 for a fixed momentum rate was seen to improve the network training.

Next, a comparison was drawn between the gradient descent algorithm and the LM algorithm. The network trained using the LM algorithm consists of a hidden layer with 31 neurons with a learning rate of 0.5 and a β of 8. The performance of this network was compared to a network trained using the gradient descent algorithm with a learning rate of 0.1 and momentum rate of 0.6 which is initialized using the dyadic grid method. Training was carried out for 100 iterations.

Figure 26 and Table 7 show the MSE of the network trained using each of the two algorithms.

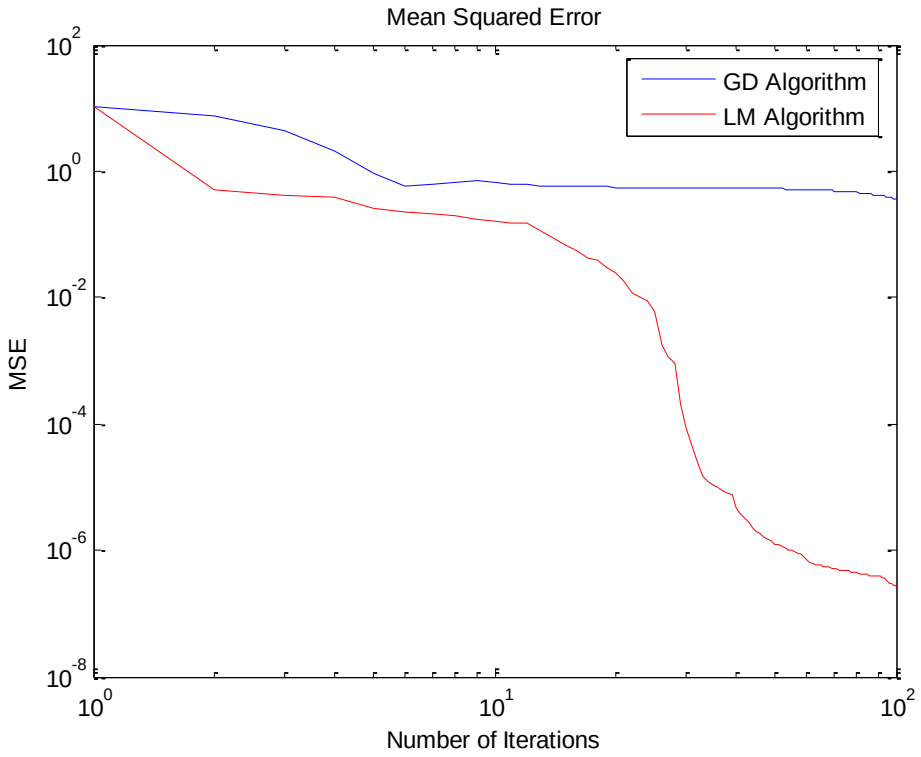


Figure 26: MSE for LM and GD Trained Networks for MISO Function Approximation

Table 7: MSE for LM and GD Trained Networks for MISO Function Approximation

	Mean Square Error	
Iterations	LM Algorithm	Gradient Descent Algorithm
100	2.628e-7	0.361

The effect of changing the activation function on the network performance was tested for a wavelet network with 31 neurons in the hidden layer, with learning and momentum rates set to 0.1 and 0.6 respectively. Initialization of the translation and dilation parameters was done using the dyadic grid method. The gradient descent algorithm was used for training. Table 8 shows the MSE after 1000 iterations for the network which was tested using the first derivative of the Gaussian function, the Mexican Hat wavelet and the Morlet wavelet as activation functions. It is seen that the first derivative of the Gaussian function provides the best network performance. Figure 27 shows the comparison of the MSE for the three different activation functions.

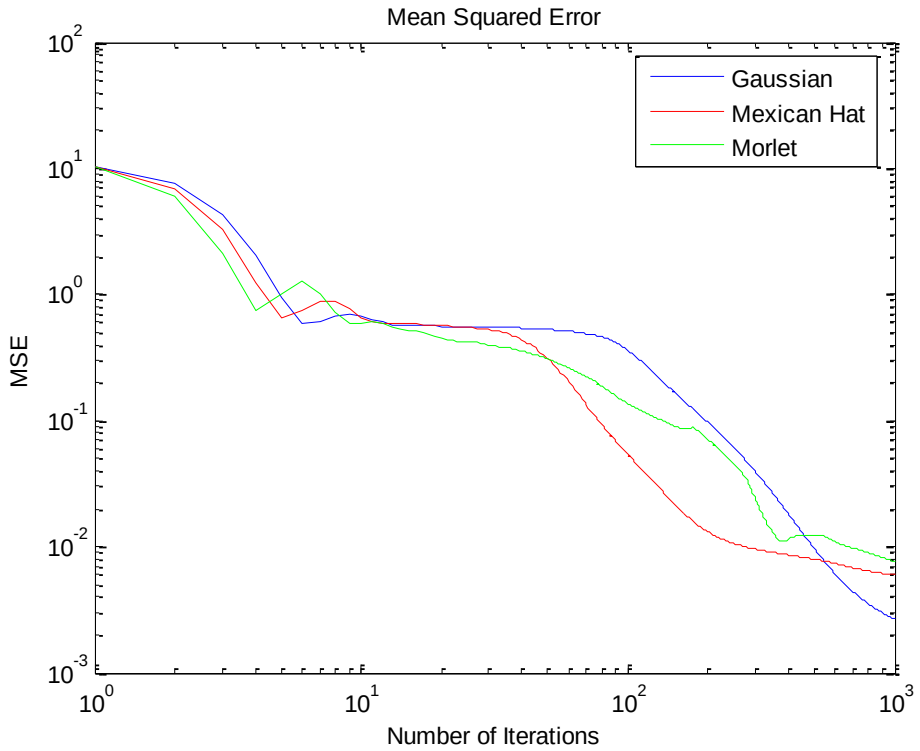


Figure 27: MSE of Networks Trained using Different Activation Functions for MISO Function Approximation

Table 8: MSE for Different Activation Functions for MISO Function Approximation

	Mean Square Error		
Iterations	1st Derivative of Gaussian	Mexican Hat	Morlet
1000	0.0027	0.0060	0.0077

2.3.2 Online training.

2.3.2.1 SISO system. The nonlinear Gaussian function to be approximated is given by Equation 2.27 and shown in Figure 28.

$$f(x) = e^{-10{x_1}^2} \tag{2.27}$$

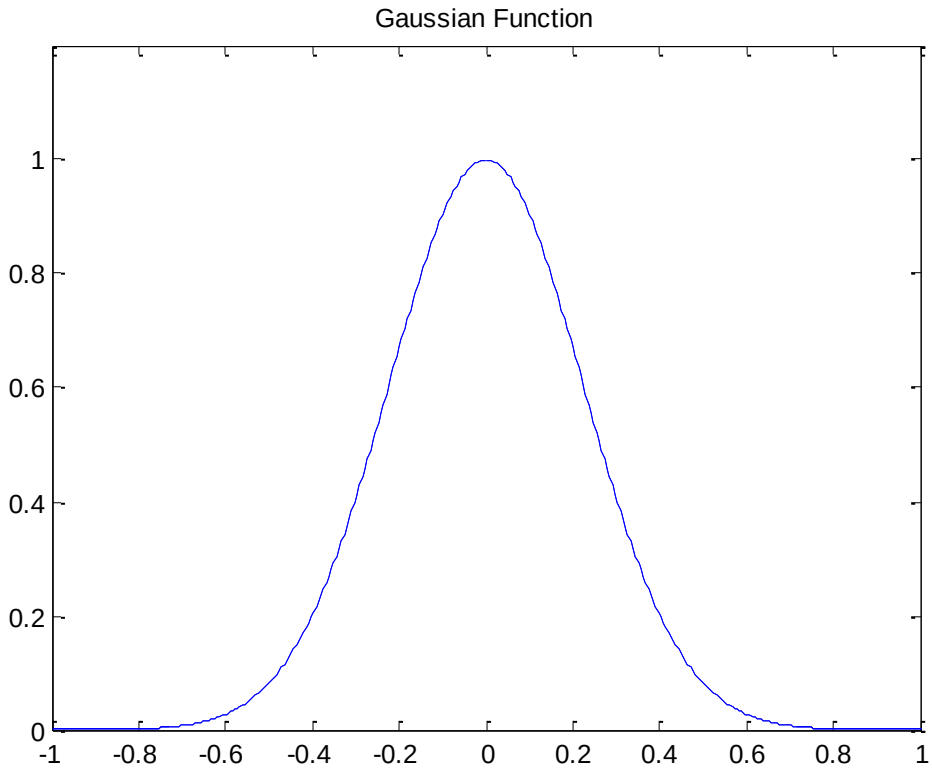


Figure 28: Gaussian Function

Training was carried out using a chirp signal and the simulation was run for 10000 iterations for a wavelet network with seven neurons in the hidden layer and the

learning and momentum rates set to 0.1 and 0.9 respectively. The Simulink block diagram for training the network online is shown in Figure 29 and the MSE is provided in Figure 30. After training for 10000 iterations, the final value of the MSE was found to be 4.6294e-8.

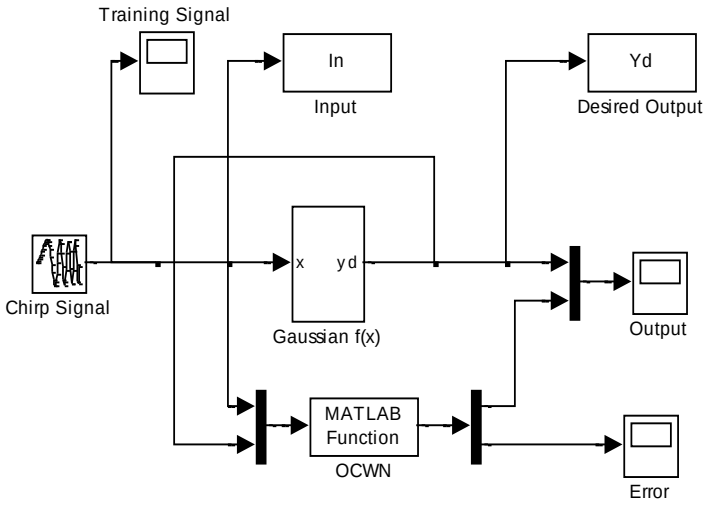


Figure 29: Online Training of SISO CWN

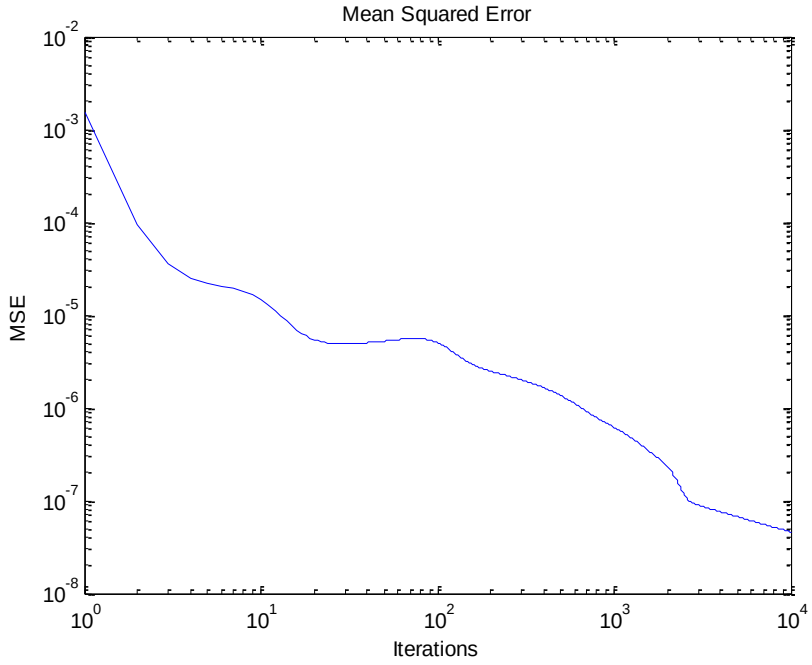


Figure 30: Mean Square Error for Approximation of Gaussian Function

After training the network, the network was then tested using sinusoids and triangular waves as the input signals. The block diagram for testing of the network is shown in Figure 31.

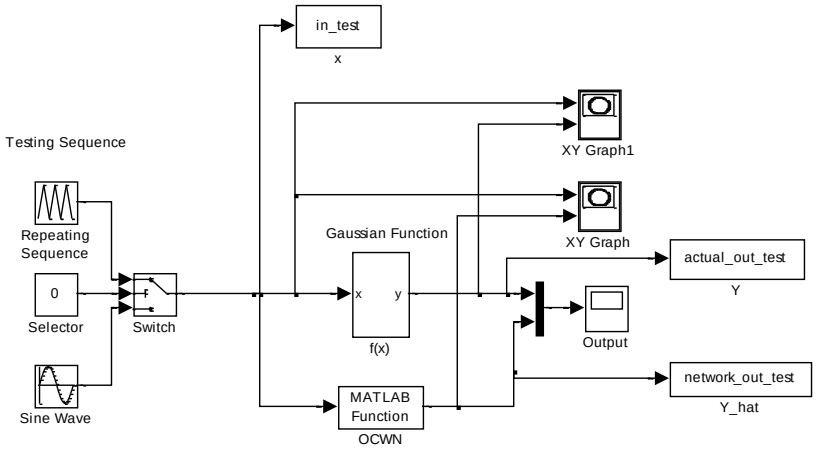


Figure 31: Block Diagram for Testing Wavelet Network

A graph of the network output versus the input was plotted and is shown in Figure 32. From the graph it can be seen that the network was trained to successfully approximate the desired Gaussian function. After testing, the mean square error was found to be 2.7517e-5.

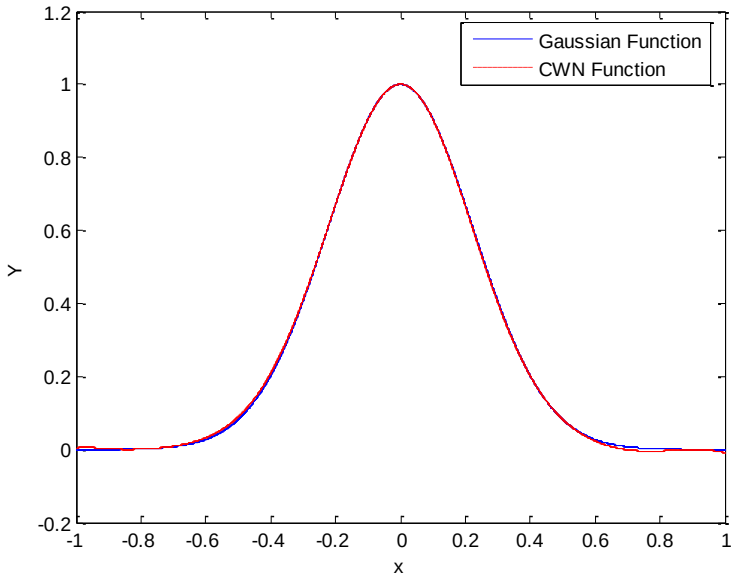


Figure 32: Output of CWN for Gaussian Function Approximation

The effect of changing the activation function for online training of the SISO system was then tested. A network with 7 neurons in the hidden layer was selected with a learning rate of 0.1 and a momentum rate of 0.9. Three different activation functions were tested: first derivative of the Gaussian, Morlet and Mexican Hat wavelets. Table 9 and Figure 33 show the MSE of each network after 1000 iterations. It was seen that selecting the Morlet wavelet as the activation function allowed for faster network convergence.

Table 9: MSE for Different Activation Functions for Online SISO Function Approximation

	Mean Square Error		
Iterations	1st Derivative of Gaussian	Mexican Hat	Morlet
1000	1.819e-7	3.38e-8	1.359e-9

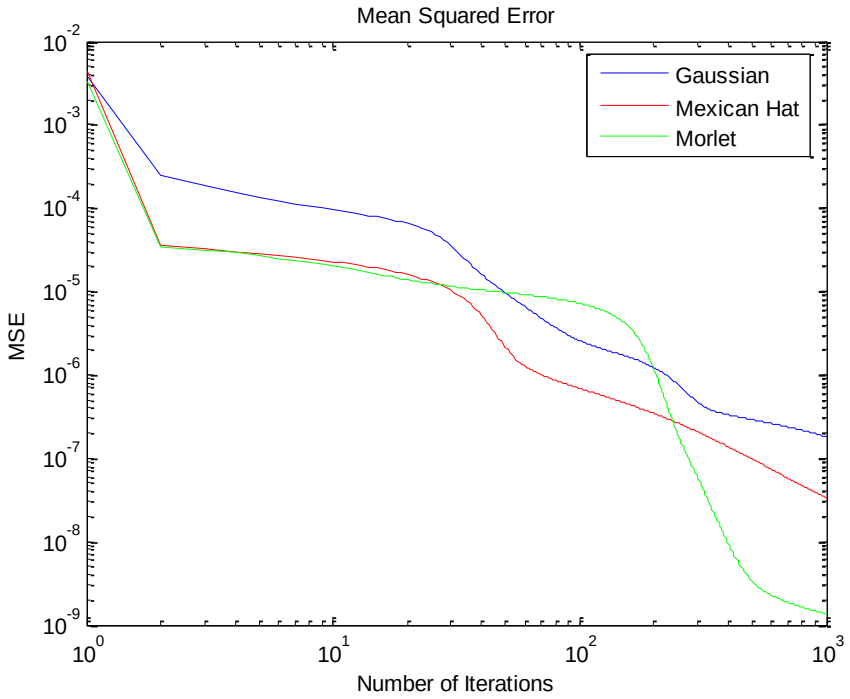


Figure 33: MSE for Different Activation Functions for Online SISO Function Approximation

2.3.2.2 MISO system. The nonlinear function to be approximated, for the input range of [-0.5, 0.5], is given by Equation 2.28 and shown in Figure 34.

$$f(x_1, x_2) = 1.335(1.5 - 1.5x_1) + e^{2x_1 - 1} \sin(3\pi(x_1 - 0.6)^2) + e^{3x_2 - 1.5} \sin(4\pi(x_2 - 0.9)^2) \quad (2.28)$$

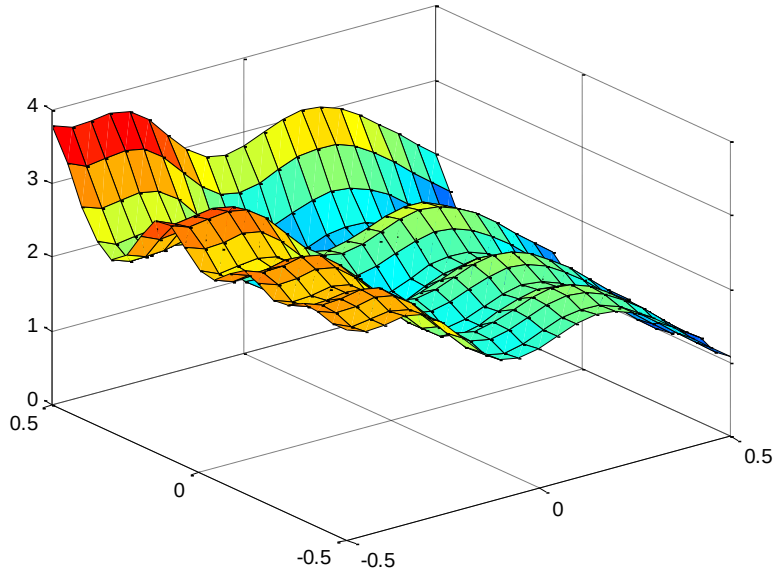


Figure 34: MISO Function for Online Training

In order to train the network over the entire input domain, the training signals were chosen so as to form a dense spiral over the desired input domain as shown in Figure 35.

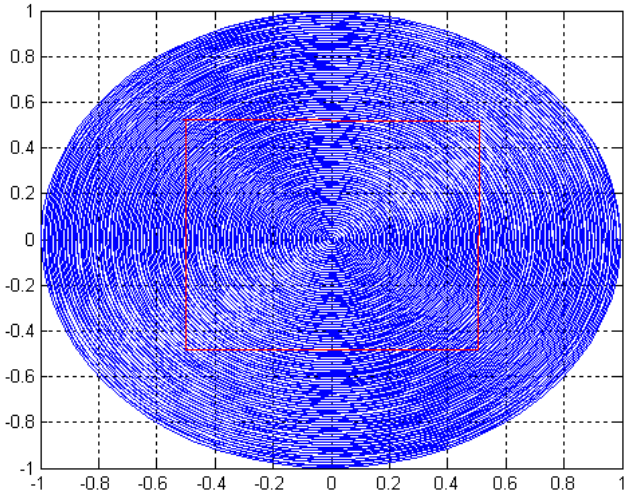


Figure 35: Training Signals for MISO Function

The Simulink block diagram for training the network online is shown in Figure 36. The simulation was run for 1600 iterations for a wavelet network with 31 neurons in the hidden layer. The momentum rates for the translation and dilation coefficients were selected to be different from those used for the direct connection and bias weights. The momentum coefficients were set to 0.8 and 0.6 respectively and the learning rate was selected as 0.001.

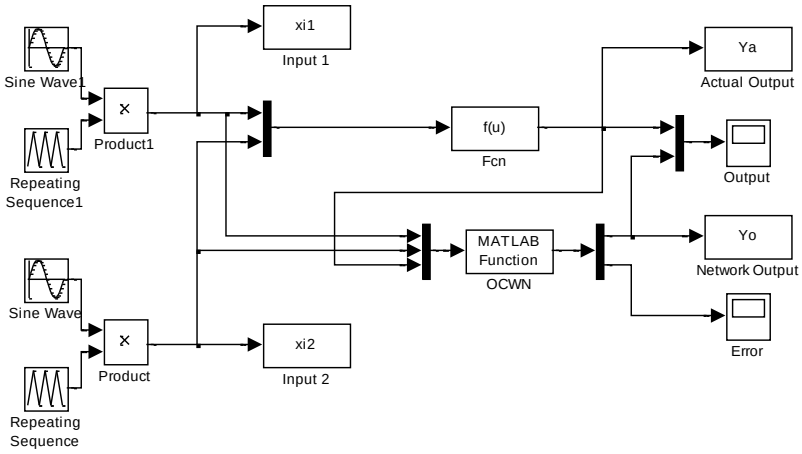


Figure 36: Online Training of MISO CWN

A graph of the MSE per iteration is shown in Figure 37. At the end of 1600 iterations the error was found to be 0.0013.

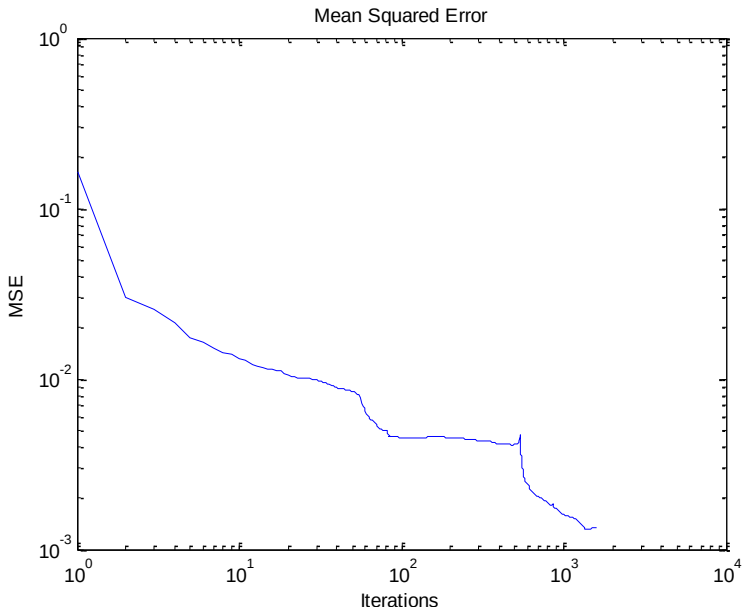


Figure 37: Mean Square Error for Approximation of MISO Function

The network was then tested for an input mesh between -0.5 and 0.5 as shown in Figure 38.

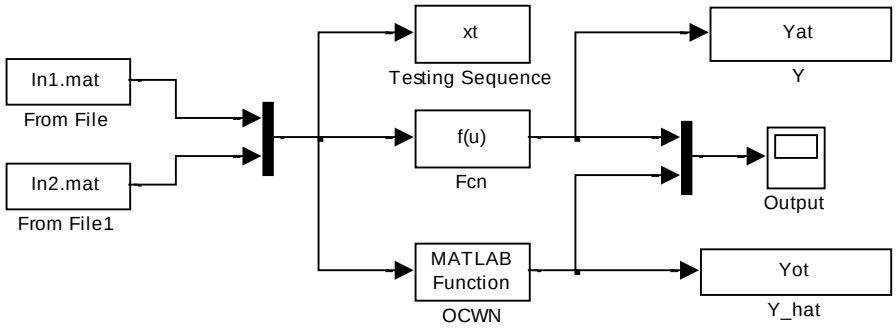


Figure 38: Block Diagram for Testing MISO Wavelet Network

From Figure 39 it can be see that the network output closely follows the desired output signal. The MSE upon testing was found to be 0.0094. With further network training, this error can be further reduced to provide a more accurate representation.

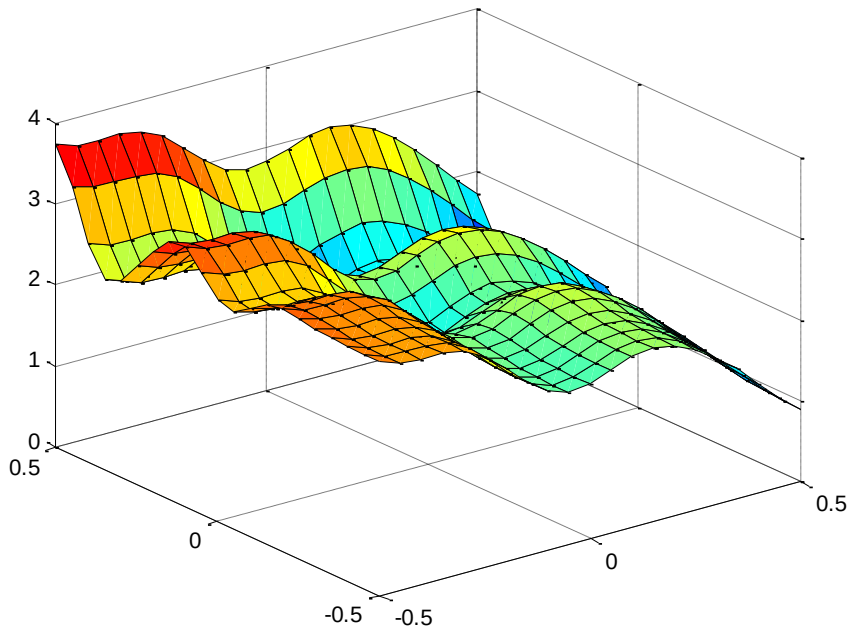


Figure 39: Output of CWN for MISO Function Approximation

The effect of changing the activation function for online training of the MISO system was then tested. A network with 31 neurons in the hidden layer was selected with a learning rate of 0.001 and momentum rates of 0.8 and 0.6 were selected for the translation and dilation coefficients and the remaining network coefficients respectively. Three different activation functions were tested: first derivative of the Gaussian function, Morlet and Mexican Hat wavelets. Table 10 and Figure 40 show the MSE of each network after 1500 iterations. In this case the Mexican Hat wavelet gave the smallest MSE after 1500 iterations.

Table 10: MSE for Different Activation Functions for Online MISO Function Approximation

	Mean Square Error		
Iterations	1st Derivative of Gaussian	Mexican Hat	Morlet
1500	0.0013380	0.00083426	0.0013868

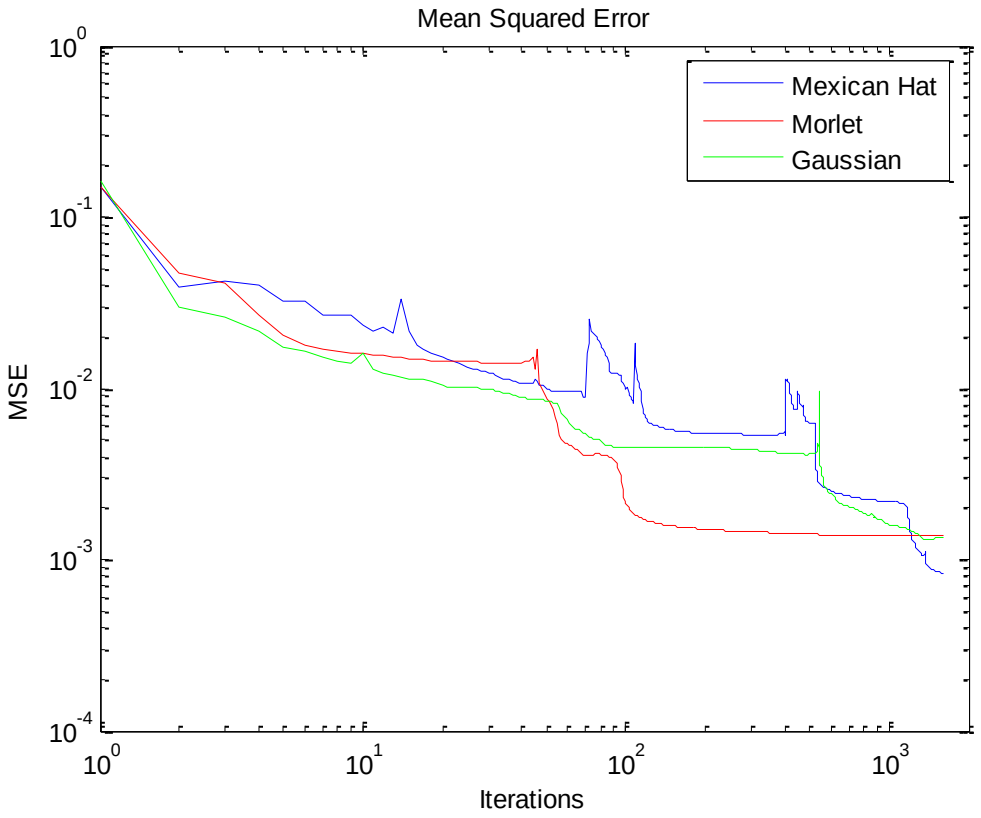


Figure 40: MSE for Different Activation Functions for Online MISO Function Approximation

Chapter 3: Modelling of Dynamic Systems Using Recurrent Wavelet Networks

Unlike the conventional wavelet networks which are used to create static mappings, Recurrent Wavelet Networks (RWN) are used for the modelling of dynamic systems with time varying inputs or outputs as shown in Figure 41.

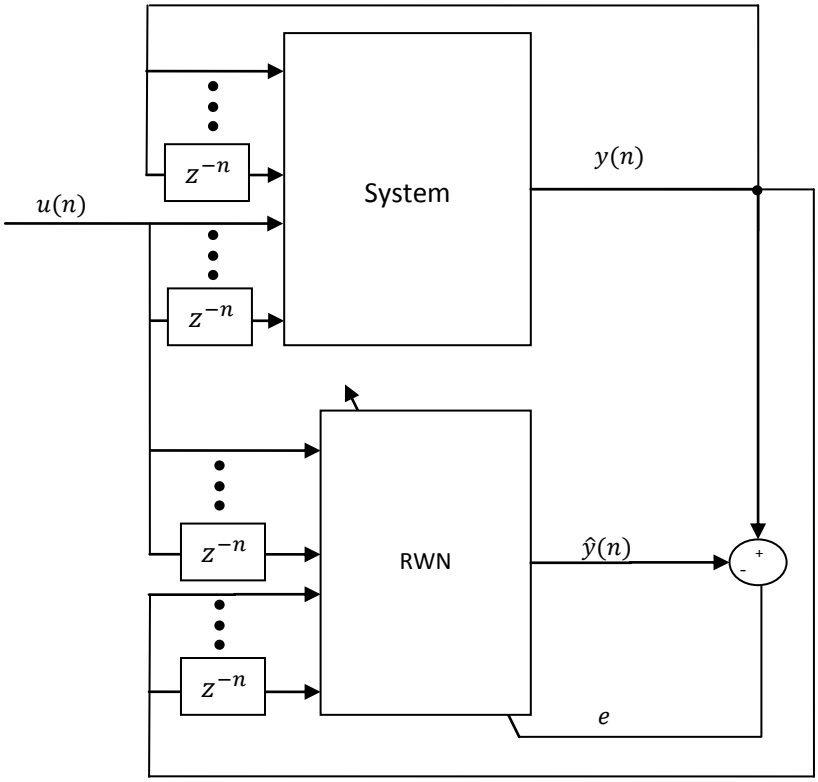


Figure 41: Training Structure for Dynamic Systems

3.1 Architecture

Modelling of dynamic systems is made possible using RWN through the use of feedback in the wavelet layer of the network. The most popular RWN architecture is a four layer structure with self-feedback in the wavelet layer as shown in Figure 42 [13-16]. A generalization of the WNN structure, the RWN is equivalent to the WNN when the feedback weights are set to zero.

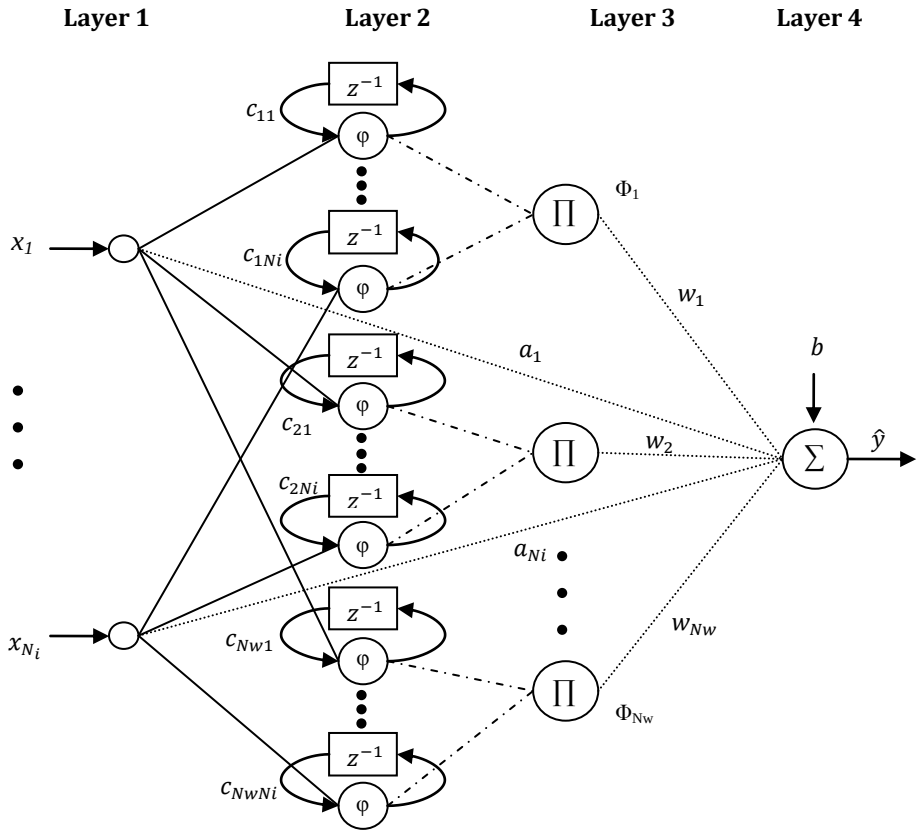


Figure 42: RWN Architecture

The output of the recurrent wavelet network is given by Equation 3.1,

$$\hat{y}(n) = b + \sum_{j=1}^{N_w} w_j \Phi_j(n) + \sum_{i=1}^{N_i} a_i x_i(n) \tag{3.1}$$

where a_i represents the weights of the direct connections between the output and the input, b represents the bias weight and w_j represents the weight between the product nodes of Layer 3 and the output node. Φ_j is a multidimensional wavelet given as the product of N_i scalar wavelets given by Equation 3.2.

$$\Phi_j(n) = \prod_{i=1}^{N_i} \varphi\left(z_{ji}(n)\right) = \prod_{i=1}^{N_i} \varphi\left(\frac{u_{ji}(n) - m_{ji}}{d_{ji}}\right) \tag{3.2}$$

where m_{ji} is the translation coefficient and d_{ji} is the dilation coefficient and u_{ji} represents the input to Layer 2 and is given by Equation 3.3.

$$u_{ji}(n) = x_i(n) + c_{ji}\varphi(z_{ji}(n-1)) \tag{3.3}$$

The mother wavelet is selected to be the first derivative of the Gaussian function, therefore Equation 3.2 can be re-written as shown in Equation 3.4.

$$\Phi_j(n) = \prod_{i=1}^{N_i} \varphi_{ji}(n) = \prod_{i=1}^{N_i} -z_{ji}(n)\, e^{-\frac{{z_{ji}}^2(n)}{2}} \tag{3.4}$$

The complete set of network parameters is given by the weighting vector
$\theta = [b \quad a_i \quad w_j \quad m_{ji} \quad d_{ji} \quad c_{ji}]^T$.

3.2 Training Algorithm

Training of the recurrent wavelet network is carried out using the gradient descent algorithm which involves adjusting the network parameters, θ, to ensure the minimization of a cost function given by Equation 3.5,

$$J(\theta) = \frac{1}{2}(y(n) - \hat{y}(n))^2 = \frac{1}{2}(e(n))^2 \tag{3.5}$$

where e is the error between the desired output and the output of the wavelet network.

The partial derivative of the cost function with respect to the network parameters is given in Equation 3.6.

$$\frac{\partial J}{\partial \theta} = -e(n)\frac{\partial \hat{y}(n)}{\partial \theta} \tag{3.6}$$

The partial derivative of the network output $\hat{y}$ with respect to each of the network parameters is given in Equations 3.7 to 3.12.

$$\frac{\partial \hat{y}(n)}{\partial b} = 1 \tag{3.7}$$

$$\frac{\partial \hat{y}(n)}{\partial a_i} = x_i(n) \tag{3.8}$$

$$\frac{\partial \hat{y}(n)}{\partial w_j} = \Phi_j(n) = \prod_{i=1}^{N_i} \varphi(z_{ji}(n)) \tag{3.9}$$

$$\frac{\partial \hat{y}(n)}{\partial m_{ji}} = \frac{\partial \hat{y}(n)}{\partial \Phi_j(n)} \cdot \left(\prod_{\substack{p=1 \\ p \neq i}}^{N_i} \varphi\left(z_{jp}(n)\right) \right) \cdot \frac{\partial \varphi\left(z_{ji}(n)\right)}{\partial m_{ji}} = w_j \cdot \left(\prod_{\substack{p=1 \\ p \neq i}}^{N_i} \varphi\left(z_{jp}(n)\right) \right) \cdot \frac{\partial \varphi(z_{ji}(n))}{\partial m_{ji}} \tag{3.10}$$

$$\frac{\partial \hat{y}(n)}{\partial d_{ji}} = \frac{\partial \hat{y}(n)}{\partial \Phi_j(n)} \cdot \left(\prod_{\substack{p=1 \\ p \neq i}}^{N_i} \varphi(z_{jp}(n)) \right) \cdot \frac{\partial \varphi(z_{ji}(n))}{\partial d_{ji}} = w_j \cdot \left(\prod_{\substack{p=1 \\ p \neq i}}^{N_i} \varphi(z_{jp}(n)) \right) \cdot \frac{\partial \varphi(z_{ji}(n))}{\partial d_{ji}} \tag{3.11}$$

$$\frac{\partial \hat{y}(n)}{\partial c_{ji}} = \frac{\partial \hat{y}(n)}{\partial \Phi_j(n)} \cdot \left(\prod_{\substack{p=1 \\ p \neq i}}^{N_i} \varphi(z_{jp}(n)) \right) \cdot \frac{\partial \varphi(z_{ji}(n))}{\partial c_{ji}} = w_j \cdot \left(\prod_{\substack{p=1 \\ p \neq i}}^{N_i} \varphi(z_{jp}(n)) \right) \cdot \frac{\partial \varphi(z_{ji}(n))}{\partial c_{ji}} \tag{3.12}$$

Equations 3.10 to 3.12 can be rewritten for ease as given in Equations 3.13 to 3.15, where

$$V_{1,ji}(n) = \frac{\partial \varphi\left(z_{ji}(n)\right)}{\partial m_{ji}}, V_{2,ji}(n) = \frac{\partial \varphi\left(z_{ji}(n)\right)}{\partial d_{ji}} \text{ and } V_{3,ji}(n) = \frac{\partial \varphi\left(z_{ji}(n)\right)}{\partial c_{ji}}.$$

$$\frac{\partial \hat{y}(n)}{\partial m_{ji}} = w_j . \left(\prod_{\substack{p=1 \\ p \neq i}}^{N_i} \varphi\left(z_{jp}(n)\right) \right) . V_{1,ji}(n) \tag{3.13}$$

$$\frac{\partial \hat{y}(n)}{\partial d_{ji}} = w_j . \left(\prod_{\substack{p=1 \\ p \neq i}}^{N_i} \varphi\left(z_{jp}(n)\right) \right) . V_{2,ji}(n) \tag{3.14}$$

$$\frac{\partial \hat{y}(n)}{\partial c_{ji}} = w_j . \left(\prod_{\substack{p=1 \\ p \neq i}}^{N_i} \varphi\left(z_{jp}(n)\right) \right) . V_{3,ji}(n) \tag{3.15}$$

Recursive equations are then obtained as shown in Equation 3.16 to 3.18.

$$\frac{\partial \varphi(z_{ji}(n))}{\partial m_{ji}} = \frac{\partial \varphi(z_{ji}(n))}{\partial z_{ji}(n)} . \frac{\partial z_{ji}(n)}{\partial m_{ji}}$$

$$\frac{\partial \varphi(z_{ji}(n))}{\partial m_{ji}} = \frac{\partial \varphi(z_{ji}(n))}{\partial z_{ji}(n)} . \frac{1}{d_{ji}} \left(c_{ji} . \frac{\partial \varphi(z_{ji}(n-1))}{\partial m_{ji}} - 1 \right)$$

$$V_{1,ji}(n) = \frac{\partial \varphi(z_{ji}(n))}{\partial z_{ji}(n)} . \frac{1}{d_{ji}} \left(c_{ji} . V_{1,ji}(n-1) - 1 \right) \tag{3.16}$$

$$\frac{\partial \varphi(z_{ji}(n))}{\partial d_{ji}} = \frac{\partial \varphi(z_{ji}(n))}{\partial z_{ji}(n)} . \frac{\partial z_{ji}(n)}{\partial d_{ji}}$$

$$\frac{\partial \varphi(z_{ji}(n))}{\partial d_{ji}} = \frac{\partial \varphi(z_{ji}(n))}{\partial z_{ji}(n)} . \frac{1}{d_{ji}} \left(-z_{ji}(n) + c_{ji} . \frac{\partial \varphi\left(z_{ji}(n-1)\right)}{\partial d_{ji}} \right)$$

$$V_{2,ji}(n) = \frac{\partial \varphi(z_{ji}(n))}{\partial z_{ji}(n)} . \frac{1}{d_{ji}} \left(-z_{ji}(n) + c_{ji} . V_{2,ji}(n-1) \right) \tag{3.17}$$

$$\frac{\partial\varphi(z_{ji}(n))}{\partial c_{ji}} = \frac{\partial\varphi(z_{ji}(n))}{\partial z_{ji}(n)} . \frac{\partial z_{ji}(n)}{\partial c_{ji}}$$

$$\frac{\partial\varphi(z_{ji}(n))}{\partial c_{ji}} = \frac{\partial\varphi(z_{ji}(n))}{\partial z_{ji}(n)} . \frac{1}{d_{ji}} \left(\varphi\big(z_{ji}(n-1)\big) + c_{ji} \frac{\partial\varphi(z_{ji}(n-1))}{\partial c_{ji}} \right)$$

$$V_{3,ji}(n) = \frac{\partial\varphi(z_{ji}(n))}{\partial z_{ji}(n)} . \frac{1}{d_{ji}} \big(\varphi\big(z_{ji}(n-1)\big) + c_{ji} V_{3,ji}(n-1) \big) \tag{3.18}$$

where $\frac{\partial\varphi(z_{ji}(n))}{\partial z_{ji}(n)} = ({z_{ji}}^2(n) - 1) e^{-\frac{{z_{ji}}^2(n)}{2}}$

Each parameter is then updated using Equation 3.19, where μ(n) represents the adaptive learning rate and *n* represents a data point in the training cycle.

$$\theta(n+1) = \theta(n) + \Delta\theta(n) = \theta(n) - \mu(n) \frac{\partial J}{\partial \theta} \tag{3.19}$$

3.3 Convergence and Stability Analysis

In order to ensure the convergence of the training algorithm for the structure represented in Figure 41, adaptive learning rates are derived from the discrete Lyapunov stability theorem. The discrete Lyapunov function is defined as in Equation 3.20 [13, 20-21].

$$V(n) = \frac{1}{2} e^2(n) \tag{3.20}$$

The change in the Lyapunov function is given in Equation 3.21.

$$\Delta V(n) = V(n+1) - V(n) = \frac{1}{2} (e^2(n+1) - e^2(n)) \tag{3.21}$$

The error term, $e(n+1)$, can be approximated as shown in Equation 3.22.

$$e(n+1) = e(n) + \Delta e(n) \cong e(n) + \left[\frac{\partial e(n)}{\partial \theta^i} \right]^T . \Delta\theta^i \tag{3.22}$$

where θ, the weighting vector, is

$$\theta = [b \quad a_i \quad w_j \quad m_{ji} \quad d_{ji} \quad c_{ji}]^T$$

and

$$\left[\frac{\partial e(n)}{\partial \theta^i}\right]^T = \left[\frac{\partial e(n)}{\partial b} \quad \frac{\partial e(n)}{\partial a} \quad \frac{\partial e(n)}{\partial w} \quad \frac{\partial e(n)}{\partial m} \quad \frac{\partial e(n)}{\partial d} \quad \frac{\partial e(n)}{\partial c}\right]$$

Substituting Equation 3.22 in Equation 3.21, the change in the Lyapunov function can be written as shown in Equation 3.23.

$$\Delta V(n) = \frac{1}{2}\left(2e(n).\Delta e(n) + \Delta e^2(n)\right) = \Delta e(n)\left(e(n) + \frac{1}{2}\Delta e(n)\right) \tag{3.23}$$

The change in each of the weights, $\Delta\theta^i$, is computed as in Equation 3.24.

$$\Delta\theta^i = -\mu_i(n)\frac{\partial J(n)}{\partial \theta^i} = \mu_i(n)e(n)\frac{\partial \hat{y}(n)}{\partial \theta^i} \tag{3.24}$$

Using Equations 3.22 and 3.24, the change in the Lyapunov function can be written as shown in Equation 3.25.

$$\Delta V(n) = \mu_i(n)e(n).\left[\frac{\partial e(n)}{\partial \theta^i}\right]^T.\frac{\partial \hat{y}(n)}{\partial \theta^i}\left(e(n) + \frac{1}{2}\mu_i(n)e(n).\left[\frac{\partial e(n)}{\partial \theta^i}\right]^T.\frac{\partial \hat{y}(n)}{\partial \theta^i}\right) \tag{3.25}$$

where

$$\frac{\partial e(n)}{\partial \theta^i} = -\frac{\partial \hat{y}(n)}{\partial \theta^i}$$

Therefore, $\Delta V(n)$ can be written as shown in Equation 3.26.

$$\Delta V(n) = -\mu_i(n)e(n).\left\|\frac{\partial \hat{y}(n)}{\partial \theta^i}\right\|^2\left(e(n) - \frac{1}{2}\mu_i(n)e(n).\left\|\frac{\partial \hat{y}(n)}{\partial \theta^i}\right\|^2\right) \tag{3.26}$$

Equation 3.26 is rewritten as Equation 3.27.

$$\Delta V(n) = -e^2(n)\left(\mu_i(n)\left\|\frac{\partial \hat{y}(n)}{\partial \theta^i}\right\|^2 - \frac{1}{2}\mu_i{}^2(n)\left\|\frac{\partial \hat{y}(n)}{\partial \theta^i}\right\|^4\right) = -\gamma e^2(n) \qquad (3.27)$$

where

$$\gamma = \mu_i(n)\left\|\frac{\partial \hat{y}(n)}{\partial \theta^i}\right\|^2\left(1 - \frac{1}{2}\mu_i(n)\left\|\frac{\partial \hat{y}(n)}{\partial \theta^i}\right\|^2\right)$$

γ can be rewritten in terms of η^i and μ^i as shown in Equation 3.28, where $\eta_i = \mu_i(n)\left(\max_n \left\|\frac{\partial \hat{y}(n)}{\partial \theta^i}\right\|^2\right)$.

$$\gamma = \frac{1}{2}\mu_i(n)\left\|\frac{\partial \hat{y}(n)}{\partial \theta^i}\right\|^2\left(2 - \eta_i\frac{\left\|\frac{\partial \hat{y}(n)}{\partial \theta^i}\right\|^2}{\max_n\left\|\frac{\partial \hat{y}(n)}{\partial \theta^i}\right\|^2}\right) \geq \frac{1}{2}\mu_i(n)\left\|\frac{\partial \hat{y}(n)}{\partial \theta^i}\right\|^2(2 - \eta_i) \qquad (3.28)$$

since

$$\frac{\left\|\frac{\partial \hat{y}(n)}{\partial \theta^i}\right\|^2}{\max_n\left\|\frac{\partial \hat{y}(n)}{\partial \theta^i}\right\|^2} \leq 1$$

To ensure convergence, $\Delta V(n) < 0$. In order to meet this condition, $\gamma > 0$. This is possible by selecting the learning rate as shown in Equation 3.29.

$$\mu_i(n)(2 - \eta_i) > 0 \qquad (3.29)$$

Substituting for η_i leads to Equation 3.30,

$$\mu_i(n)\left(2 - \mu_i(n)\left(\max_n\left\|\frac{\partial \hat{y}(n)}{\partial \theta^i}\right\|^2\right)\right) > 0 \qquad (3.30)$$

From Equation 3.30, the learning rates for each of the weights should be selected as shown in Equation 3.31, in order to guarantee convergence.

$$0 < \mu_i(n) < \frac{2}{\max_n\left\|\frac{\partial \hat{y}(n)}{\partial \theta^i}\right\|^2} \qquad (3.31)$$

Based on Equation 3.31, the method by which the adaptive learning rates are obtained in each iteration is shown in the flowchart in Figure 43, where *N* represents the total number of data points per iteration.

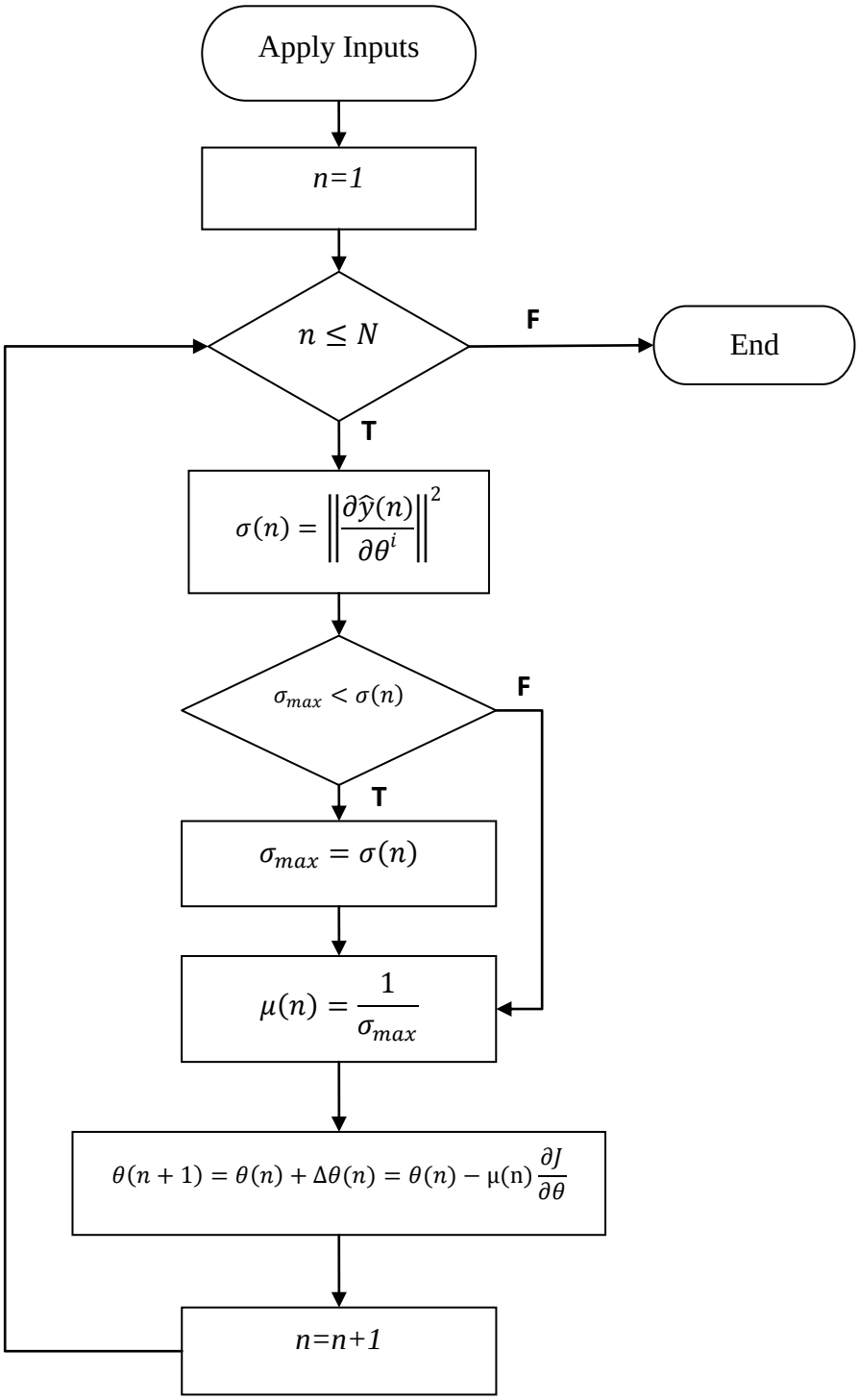


Figure 43: Flowchart for Updating Adaptive Learning Rates

3.4 Adaptive Learning Rates

From the results of the stability analysis, adaptive learning rates were derived for each of the network parameters that make up the weighting vector θ. Details on computation of matrix norms can be found in Appendix A. Appendix B provides the details on the solution of the Recursive Equations 3.16-3.18 which will be required to solve for the adaptive learning rates.

- μ_b

For the bias weight at the output layer, the learning rate, $\mu_b(n)$, is selected as shown in Equation 3.32.

$$0 < \mu_b(n) < \frac{2}{\max_n \left\| \frac{\partial \hat{y}(n)}{\partial b} \right\|^2} \tag{3.32}$$

From Equation 3.7, $\frac{\partial \hat{y}(n)}{\partial b} = 1$. Therefore the norm is given by Equation 3.33,

$$\left\| \frac{\partial \hat{y}(n)}{\partial b} \right\| = 1 \tag{3.33}$$

From Equation 3.33, the adaptive learning rate is selected as shown in Equation 3.34.

$$0 < \mu_b(n) < 2 \tag{3.34}$$

- μ_a

For the direct connections between the input layer and the output layer, the learning rate, $\mu_a(n)$, is selected as shown in Equation 3.35.

$$0 < \mu_a(n) < \frac{2}{\max_n \left\| \frac{\partial \hat{y}(n)}{\partial \boldsymbol{a}} \right\|^2} \tag{3.35}$$

From Equation 3.8, $\frac{\partial \hat{y}(n)}{\partial a_i} = x_i(n)$. Therefore the norm is given by Equation 3.36,

$$\left\| \frac{\partial \hat{y}(n)}{\partial \boldsymbol{a}} \right\| = \sqrt{{x_1}^2(n) + \quad . \quad . \quad + {x_{N_i}}^2(n)} \tag{3.36}$$

The maximum of the norm at time index *n* is given by Equation 3.37.

$$\max_i \left\| \frac{\partial \hat{y}(n)}{\partial \boldsymbol{a}} \right\| = \sqrt{N_i {x_{max}}^2(n)} \tag{3.37}$$

where

$$x_{max}(n) = \max_i \left| x_{i,max}(n) \right|$$

From Equation 3.37 the maximum norm can be computed as shown in Equation 3.38,

$$\max_n \left\| \frac{\partial \hat{y}(n)}{\partial \boldsymbol{a}} \right\| = \sqrt{N_i x_{max}^2} \tag{3.38}$$

where

$$x_{max} = \max_n |x_{max}(n)|$$

Substituting Equation 3.38 in Equation 3.35, the adaptive learning rate is selected as shown in Equation 3.39.

$$0 < \mu_a(n) < \frac{2}{N_i x_{max}^2} \tag{3.39}$$

- μ_w

The learning rate, $\mu_w(n)$, for the weights connecting the hidden layer to the output layer is selected so as to satisfy Equation 3.40.

$$0 < \mu_w(n) < \frac{2}{\max_n \left\| \frac{\partial \hat{y}(n)}{\partial \boldsymbol{w}} \right\|^2} \tag{3.40}$$

From Equation 3.9, $\frac{\partial \hat{y}(n)}{\partial w_j} = \Phi_j$. Therefore the norm is given by Equation 3.41,

$$\left\| \frac{\partial \hat{y}(n)}{\partial \boldsymbol{w}} \right\| = \sqrt{{\Phi_1}^2(n) + \quad . \quad . \quad + {\Phi_{N_w}}^2(n)} \tag{3.41}$$

The maximum of the norm at time index *n* is given by Equation 3.42.

$$\max_j \left\| \frac{\partial \hat{y}(n)}{\partial \boldsymbol{w}} \right\| = \sqrt{N_w {\Phi_{\max}}^2(n)} \tag{3.42}$$

where

$$\Phi_{max}(n) = \max_j \left| \Phi_{j,max}(n) \right|$$

From Equation 3.42, the maximum norm is given by Equation 3.43,

$$\max_n \left\| \frac{\partial \hat{y}(n)}{\partial \boldsymbol{w}} \right\| = \sqrt{N_w \Phi_{max}^2} \tag{3.43}$$

where

$$\Phi_{max} = \max_n |\Phi_{max}(n)|$$

Since the selected activation function is the first derivative of a Gaussian, then $\left|\Phi_j(n)\right| \leq 1$. The maximum norm can be computed as shown in Equation 3.44.

$$\max_n \left\| \frac{\partial \hat{y}(n)}{\partial \boldsymbol{w}} \right\| = \sqrt{N_w} \tag{3.44}$$

Substituting Equation 3.44 into Equation 3.40, the adaptive learning rate is selected as shown in Equation 3.45.

$$0 < \mu_w(n) < \frac{2}{N_w} \tag{3.45}$$

- μ_m

The learning rate, $\mu_m(n)$, for the translation coefficients is selected so as to satisfy Equation 3.46.

$$0 < \mu_m(n) < \frac{2}{\max_n \left\| \frac{\partial \hat{y}(n)}{\partial \boldsymbol{m}} \right\|^2} \tag{3.46}$$

In order to solve for the norm of the matrix $\frac{\partial \hat{y}(n)}{\partial \boldsymbol{m}}$, first $\left|\frac{\partial \hat{y}(n)}{\partial m_{ji}}\right|$ must be computed. Equations 3.13 and 3.16 are rewritten here for convenience,

$$\frac{\partial \hat{y}(n)}{\partial m_{ji}} = w_j . \left(\prod_{\substack{p=1 \\ p \neq i}}^{N_i} \varphi\left(z_{jp}(n)\right) \right) \frac{\partial \varphi\left(z_{ji}(n)\right)}{\partial m_{ji}} = w_j . \left(\prod_{\substack{p=1 \\ p \neq i}}^{N_i} \varphi\left(z_{jp}(n)\right) \right) . V_{1,ji}(n) \tag{3.47}$$

where

$$V_{1,ji}(n) = \frac{\partial \varphi\left(z_{ji}(n)\right)}{\partial z_{ji}(n)} . \frac{1}{d_{ji}} \left(c_{ji} . V_{1,ji}(n-1) - 1 \right) \tag{3.48}$$

Since $0 < \left|\varphi_{ji}(n)\right| < 1$, Equation 3.47 can be written as Equation 3.49.

$$\left| \frac{\partial \hat{y}(n)}{\partial m_{ji}} \right| < |w_j| |V_{1,ji}(n)| \tag{3.49}$$

The solution of the recursive equation, given by Equation 3.48, is given in Equation 3.50,

$$V_{1,ji}(n) = \left(\prod_{p=0}^{n-1} \varphi'_{ji}(p+1) \right) \left(V_{1,ji}(0) \left(\frac{c_{ji}}{d_{ji}} \right)^n \right) - \frac{1}{d_{ji}} \sum_{p=0}^{n-1} \left(\left(\frac{c_{ji}}{d_{ji}} \right)^{n-p-1} \prod_{r=0}^{n-p-1} \varphi'_{ji}(n-r) \right) \tag{3.50}$$

Since $V_{1,ji}(0)=0$, Equation 3.50 can be simplified to Equation 3.51.

$$V_{1,ji}(n) = -\frac{1}{d_{ji}} \sum_{p=0}^{n-1} \left(\left(\frac{c_{ji}}{d_{ji}} \right)^{n-p-1} \prod_{r=0}^{n-p-1} \varphi'_{ji}(n-r) \right) \tag{3.51}$$

Since $0 \leq \left|\varphi'_{ji}\right| \leq 1$, Equation 3.51 can be written as shown in Equation 3.52.

$$\left|V_{1,ji}(n)\right| \leq \left| \frac{1}{d_{ji}} \right| \sum_{p=0}^{n-1} \left| \frac{c_{ji}}{d_{ji}} \right|^{n-p-1} \tag{3.52}$$

Assuming $\left|\frac{c_{ji}}{d_{ji}}\right| < 1$, Equation 3.52 can be rewritten as shown in Equation 3.53 which represents a finite geometric series.

$$\left|V_{1,ji}(n)\right| \leq \left|\frac{1}{d_{ji}}\right| \sum_{r=0}^{n-1} \left|\frac{c_{ji}}{d_{ji}}\right|^{r} \tag{3.53}$$

where

$$r = n - p - 1,$$

The sum of the finite geometric series will therefore always be less than or equal to the infinite series as shown in Equation 3.54.

$$\left|V_{1,ji}(n)\right| \leq \left|\frac{1}{d_{ji}}\right| \sum_{r=0}^{\infty} \left|\frac{c_{ji}}{d_{ji}}\right|^{r} \tag{3.54}$$

The solution of the infinite geometric series is given by Equation 3.55.

$$\left|V_{1,ji}(n)\right| \leq \left|\frac{1}{d_{ji}}\right| \frac{1}{1 - \left|c_{ji}/d_{ji}\right|} \tag{3.55}$$

Substituting Equation 3.55 in Equation 3.49, Equation 3.56 is obtained.

$$\left|\frac{\partial \hat{y}(n)}{\partial m_{ji}}\right| < \left|\frac{w_j}{d_{ji}}\right| \frac{1}{1 - \left|c_{ji}/d_{ji}\right|} \tag{3.56}$$

For a given point in time *n*, let $L_{1,max}(n) = \max_{i,j} \left|\frac{\partial \hat{y}(n)}{\partial m_{ji}}\right|$. The maximum norm is therefore given by Equation 3.57.

$$\max_{n} \left\|\frac{\partial \hat{y}(n)}{\partial \boldsymbol{m}}\right\| = \sqrt{N_w N_i}\, L_{1,max} \tag{3.57}$$

where

$$L_{1,max} = \max_{n}\big(\, L_{1,max}(n)\big)$$

Substituting Equation 3.57 in Equation 3.46, the adaptive learning rate is selected as shown in Equation 3.58.

$$0 < \mu_m(n) < \frac{2}{N_w N_i L_{1,max}^2} \tag{3.58}$$

- μ_d

The learning rate, $\mu_d(n)$, for the dilation coefficients is selected so as to satisfy Equation 3.59.

$$0 < \mu_d(n) < \frac{2}{\max_n \left\| \frac{\partial \hat{y}(n)}{\partial \boldsymbol{d}} \right\|^2} \tag{3.59}$$

In order to solve for the norm of the matrix $\frac{\partial \hat{y}(n)}{\partial \boldsymbol{d}}$, first $\left| \frac{\partial \hat{y}(n)}{\partial d_{ji}} \right|$ must be computed. Equations 3.14 and 3.17 are rewritten here for convenience,

$$\frac{\partial \hat{y}(n)}{\partial d_{ji}} = w_j . \left(\prod_{\substack{p=1 \\ p \neq i}}^{N_i} \varphi\left(z_{jp}(n)\right) \right) \frac{\partial \varphi\left(z_{ji}(n)\right)}{\partial d_{ji}} = w_j . \left(\prod_{\substack{p=1 \\ p \neq i}}^{N_i} \varphi\left(z_{jp}(n)\right) \right) . V_{2,ji}(n) \tag{3.60}$$

where

$$V_{2,ji}(n) = \frac{\partial \varphi\left(z_{ji}(n)\right)}{\partial z_{ji}(n)} . \frac{1}{d_{ji}} \left(-z_{ji}(n) + c_{ji} . V_{2,ji}(n-1) \right) \tag{3.61}$$

Since $0 < \left| \varphi_{ji}(n) \right| < 1$, Equation 3.60 can be written as Equation 3.62.

$$\left| \frac{\partial \hat{y}(n)}{\partial d_{ji}} \right| < |w_j| |V_{2,ji}(n)| \tag{3.62}$$

The solution of the recursive equation, given by 3.61, is given in Equation 3.63,

$$V_{2,ji}(n) = \left(\prod_{p=0}^{n-1} \varphi'_{ji}(p+1) \right) \left(V_{2,ji}(0) \left(\frac{c_{ji}}{d_{ji}} \right)^n \right) - \frac{1}{d_{ji}} \sum_{p=0}^{n-1} \left(z_{ji}(p+1) \left(\frac{c_{ji}}{d_{ji}} \right)^{n-p-1} \prod_{r=0}^{n-p-1} \varphi'_{ji}(n-r) \right) \tag{3.63}$$

Since $V_{2,ji}(0)=0$, Equation 3.63 can be simplified to Equation 3.64.

$$V_{2,ji}(n) = -\frac{1}{d_{ji}} \sum_{p=0}^{n-1} \left(z_{ji}(p+1) \left(\frac{c_{ji}}{d_{ji}} \right)^{n-p-1} \prod_{r=0}^{n-p-1} \varphi'_{ji}(n-r) \right) \tag{3.64}$$

Since $0 \leq \left| \varphi'_{ji} \right| \leq 1$, Equation 3.64 can be written as shown in Equation 3.65.

$$\left| V_{2,ji}(n) \right| \leq \left| \frac{1}{d_{ji}} \right| \sum_{p=0}^{n-1} \left| z_{ji}(p+1) \left(\frac{c_{ji}}{d_{ji}} \right)^{n-p-1} \right| \tag{3.65}$$

Assuming $\left|\frac{c_{ji}}{d_{ji}}\right| < 1$, Equation 3.65 can be rewritten as shown in Equation 3.66,

$$\left|V_{2,ji}(n)\right| \leq \left|\frac{1}{d_{ji}}\right| \sum_{q=1}^{n} \left|z_{ji}(q)\right| \left|\frac{c_{ji}}{d_{ji}}\right|^{n-q} \tag{3.66}$$

where

$$q = p + 1,$$

Equation 3.66 can then be written as shown in Equation 3.67.

$$\left|V_{2,ji}(n)\right| \leq \left|\frac{1}{d_{ji}}\right| \sum_{q=1}^{n} \left|\frac{x_i(q) + c_{ji}\varphi\left(z_{ji}(q-1)\right) - m_{ji}}{d_{ji}}\right| \left|\frac{c_{ji}}{d_{ji}}\right|^{n-q} \tag{3.67}$$

From the inequality, $|a + b| \leq |a| + |b|$, Equation 3.67 can be rewritten as Equatio 3.68.

$$\left|V_{2,ji}(n)\right| \leq \left|\frac{1}{{d_{ji}}^2}\right| \sum_{q=1}^{n} \left(|x_i(q)| + \left|c_{ji}\varphi\left(z_{ji}(q-1)\right)\right| + \left|m_{ji}\right|\right) \left|\frac{c_{ji}}{d_{ji}}\right|^{n-q} \tag{3.68}$$

Since, $0 < \left|\varphi_{ji}(n)\right| < 1$, Equation 3.68 can be further simplified as shown in Equation 3.69.

$$\left|V_{2,ji}(n)\right| < \left|\frac{1}{{d_{ji}}^2}\right| \sum_{q=1}^{n} \left[|x_i(q)| \left|\frac{c_{ji}}{d_{ji}}\right|^{n-q} + \left|c_{ji}\right| \left|\frac{c_{ji}}{d_{ji}}\right|^{n-q} + \left|m_{ji}\right| \left|\frac{c_{ji}}{d_{ji}}\right|^{n-q}\right] \tag{3.69}$$

Equation 3.69 can be written as shown in Equation 3.70,

$$\left|V_{2,ji}(n)\right| < \left|\frac{1}{{d_{ji}}^2}\right| \sum_{q=1}^{n} \left[\left|x_{i,max}\right| \left|\frac{c_{ji}}{d_{ji}}\right|^{n-q} + \left|c_{ji}\right| \left|\frac{c_{ji}}{d_{ji}}\right|^{n-q} + \left|m_{ji}\right| \left|\frac{c_{ji}}{d_{ji}}\right|^{n-q}\right] \tag{3.70}$$

where

$$x_{i,max} = \max_{q} |x_i(q)|$$

Equation 3.70 is rewritten as shown in Equation 3.71 which represents a finite geometric series.

$$\left|V_{2,ji}(n)\right| < \left|\frac{1}{{d_{ji}}^2}\right| \sum_{r=0}^{n-1} \left[\left|x_{i,max}\right| \left|\frac{c_{ji}}{d_{ji}}\right|^r + \left|c_{ji}\right| \left|\frac{c_{ji}}{d_{ji}}\right|^r + \left|m_{ji}\right| \left|\frac{c_{ji}}{d_{ji}}\right|^r \right] \tag{3.71}$$

where

$$r = n - q,$$

Equation 3.71 represents the sum of finite geometric series which will always be less than or equal to the infinite sum of the series as shown in Equation 3.72.

$$\left|V_{2,ji}(n)\right| < \left|\frac{1}{{d_{ji}}^2}\right| \sum_{r=0}^{\infty} \left[\left|x_{i,max}\right| \left|\frac{c_{ji}}{d_{ji}}\right|^r + \left|c_{ji}\right| \left|\frac{c_{ji}}{d_{ji}}\right|^r + \left|m_{ji}\right| \left|\frac{c_{ji}}{d_{ji}}\right|^r \right] \tag{3.72}$$

Solving the infinite geometric series leads to Equation 3.73.

$$\left|V_{2,ji}(n)\right| < \left|\frac{1}{{d_{ji}}^2}\right| \left(\frac{\left|x_{i,max}\right| + \left|c_{ji}\right| + \left|m_{ji}\right|}{1 - \left|c_{ji}/d_{ji}\right|} \right) \tag{3.73}$$

Substituting Equation 3.73 in Equation 3.62 leads to Equation 3.74.

$$\left|\frac{\partial \hat{y}(n)}{\partial d_{ji}}\right| < \left|\frac{w_j}{{d_{ji}}^2}\right| \left(\frac{\left|x_{i,max}\right| + \left|c_{ji}\right| + \left|m_{ji}\right|}{1 - \left|c_{ji}/d_{ji}\right|} \right) \tag{3.74}$$

Let, $L_{2,max}(n) = \max_{i,j} \left|\frac{\partial \hat{y}(n)}{\partial d_{ji}}\right|$. The maximum norm is therefore given by Equation 3.75.

$$\max_n \left\| \frac{\partial \hat{y}(n)}{\partial \boldsymbol{d}} \right\| = \sqrt{N_w N_i}\, L_{2,max} \tag{3.75}$$

where

$$L_{2,max} = \max_n \big(L_{2,max}(n) \big)$$

Substituting Equation 3.75 in Equation 3.59, the adaptive learning rate is selected as shown in Equation 3.76.

$$0 < \mu_d(n) < \frac{2}{N_w N_i L_{2,max}^2} \tag{3.76}$$

- μ_c

The learning rate, $\mu_c(n)$, for the feedback coefficients is selected so as to satisfy Equation 3.77.

$$0 < \mu_c(n) < \frac{2}{\max_n \left\| \frac{\partial \hat{y}(n)}{\partial \boldsymbol{c}} \right\|^2} \tag{3.77}$$

In order to solve for the norm of the matrix $\frac{\partial \hat{y}(n)}{\partial \boldsymbol{c}}$, first $\left| \frac{\partial \hat{y}(n)}{\partial c_{ji}} \right|$ must be computed. Equations 3.15 and 3.18 are rewritten here for convenience,

$$\frac{\partial \hat{y}(n)}{\partial c_{ji}} = w_j . \left(\prod_{\substack{p=1 \\ p \neq i}}^{N_i} \varphi\left(z_{jp}(n)\right) \right) \frac{\partial \varphi\left(z_{ji}(n)\right)}{\partial c_{ji}} = w_j . \left(\prod_{\substack{p=1 \\ p \neq i}}^{N_i} \varphi\left(z_{jp}(n)\right) \right) . V_{3,ji}(n) \tag{3.78}$$

where

$$V_{3,ji}(n) = \frac{\partial \varphi(z_{ji}(n))}{\partial z_{ji}(n)} . \frac{1}{d_{ji}} \left(\varphi\big(z_{ji}(n-1)\big) + c_{ji} V_{3,ji}(n-1) \right) \tag{3.79}$$

Since $0 < \left|\varphi_{ji}(n)\right| < 1$, Equation 3.78 can be written as Equation 80.

$$\left| \frac{\partial \hat{y}(n)}{\partial c_{ji}} \right| < |w_j| |V_{3,ji}(n)| \tag{3.80}$$

The solution of the recursive equation, given by Equation 3.79, is given in Equation 3.81,

$$V_{3,ji}(n) = \left(\prod_{p=0}^{n-1} \varphi'_{ji}(p+1) \right) \left(V_{3,ji}(0) \left(\frac{c_{ji}}{d_{ji}} \right)^n \right) + \frac{1}{d_{ji}} \sum_{p=0}^{n-1} \left(\varphi_{ji}(p) \left(\frac{c_{ji}}{d_{ji}} \right)^{n-1-p} \prod_{r=0}^{n-p-1} \varphi'_{ji}(n-r) \right) \tag{3.81}$$

Since $V_{3,ji}(0)=0$, Equation 3.81 can be simplified to Equation 3.82.

$$V_{3,ji}(n) = \frac{1}{d_{ji}} \sum_{p=0}^{n-1} \left(\varphi_{ji}(p) \left(\frac{c_{ji}}{d_{ji}} \right)^{n-1-p} \prod_{r=0}^{n-p-1} \varphi'_{ji}(n-r) \right) \tag{3.82}$$

Since $0 \leq \left|\varphi'_{ji}\right| \leq 1$, and $0 < \left|\varphi_{ji}(n)\right| < 1$, Equation 3.82 can be written as shown in Equation 3.83.

$$\left|V_{3,ji}(n)\right| < \left| \frac{1}{d_{ji}} \right| \sum_{p=0}^{n-1} \left| \frac{c_{ji}}{d_{ji}} \right|^{n-p-1} \tag{3.83}$$

Assuming $\left|\frac{c_{ji}}{d_{ji}}\right| < 1$, Equation 3.83 can be rewritten as shown in Equation 3.84 which represents a finite geometric series.

$$\left|V_{3,ji}(n)\right| \leq \left|\frac{1}{d_{ji}}\right| \sum_{r=0}^{n-1} \left|\frac{c_{ji}}{d_{ji}}\right|^{r} \tag{3.84}$$

where

$$r = n - p - 1,$$

The sum of the finite geometric series is therefore always less than or equal to the infinite sum as shown in Equation 3.85.

$$\left|V_{3,ji}(n)\right| \leq \left|\frac{1}{d_{ji}}\right| \sum_{r=0}^{\infty} \left|\frac{c_{ji}}{d_{ji}}\right|^{r} \tag{3.85}$$

Solving the infinite geometric series leads to Equation 3.86.

$$\left|V_{3,ji}(n)\right| \leq \left|\frac{1}{d_{ji}}\right| \frac{1}{1 - \left|c_{ji}\right| / \left|d_{ji}\right|} \tag{3.86}$$

Substituting Equation 3.86 in Equation 3.80,

$$\left|\frac{\partial \hat{y}(n)}{\partial c_{ji}}\right| < \left|\frac{w_j}{d_{ji}}\right| \frac{1}{1 - \left|c_{ji}/d_{ji}\right|} \tag{3.87}$$

Let, $L_{3,max}(n) = \max_{i,j} \left|\frac{\partial \hat{y}(n)}{\partial c_{ji}}\right|$. The maximum norm is therefore given by Equation 3.88.

$$\max_{n} \left\| \frac{\partial \hat{y}(n)}{\partial \boldsymbol{c}} \right\| = \sqrt{N_w N_i}\, L_{3,max} \tag{3.88}$$

where

$$L_{3,max} = \max_{n} \big(L_{3,max}(n) \big)$$

Substituting Equation 3.88 in Equation 3.77, the adaptive learning rate is selected as shown in Equation 3.89.

$$0 < \mu_c(n) < \frac{2}{N_w N_i L_{3,max}^2} \tag{3.89}$$

3.5 Simulations

In this section, grey box modelling of a DC motor with nonlinear friction, represented by Equations 3.90 and 3.91, was carried out. The parameters of the DC Motor can be found in Appendix C.

$$J\frac{d\omega}{dt} = K_t i - B\omega - \tau_f(\omega) - \tau_L \tag{3.90}$$

$$L\frac{di}{dt} = V - Ri - K_b\omega \tag{3.91}$$

The nonlinear friction of the motor represented by $\tau_f(\omega)$ is assumed to be unknown and the load torque τ_L is selected as zero. As such, based on the a priori knowledge of the linear system structure, the linear system is discretized using the Euler Forward method as well as the Bilinear Transformation method in order to determine the training inputs required to train the RWN. The transfer function of the linear DC Motor without nonlinear friction is given in Equation 3.92.

$$\frac{\omega(s)}{V(s)} = \frac{K_t}{(Ls+R)(Js+B)+K_tK_b} \tag{3.92}$$

3.5.1 Nonlinear DC motor discretized using Bilinear transformation. In order to optimize the training of the RWN, the training inputs can be determined by discretizing the DC Motor using the bilinear transform, given in Equation 3.93 in order to determine the necessary training inputs. Equation 3.94 shows the discretized linear DC Motor model.

$$s = \frac{2}{T}\frac{z-1}{z+1} \tag{3.93}$$

$$\frac{\omega(z)}{V(z)} = \frac{W(1 + 2z^{-1} + z^{-2})}{X + Yz^{-1} + Zz^{-2}} \tag{3.94}$$

where

$W = T^2K_t$ $X = 4LJ + 2TBL + 2TRJ + T^2BR + T^2K_tK_b,$

$Y = -8LJ + 2T^2BR + 2T^2K_tK_b$ $Z = 4LJ - 2TBL - 2TRJ + T^2BR + T^2K_tK_b$

From Equation 3.94, it can be seen that the speed is a function of the following inputs given in Equation 3.95.

$$\omega(n) = f(V(n), V(n-1), V(n-2), \omega(n-1), \omega(n-2)) \tag{3.95}$$

Online training was carried out using a chirp signal of magnitude 12 amd frequency 0.1-4Hz over an interval of 1s and the RWN was selected with 3 neurons in the hidden layer. The nonlinear frictional torque, $\tau_f(\omega)$, was selected as a combination of viscous friction and coulomb friction and is shown in Figure 44.

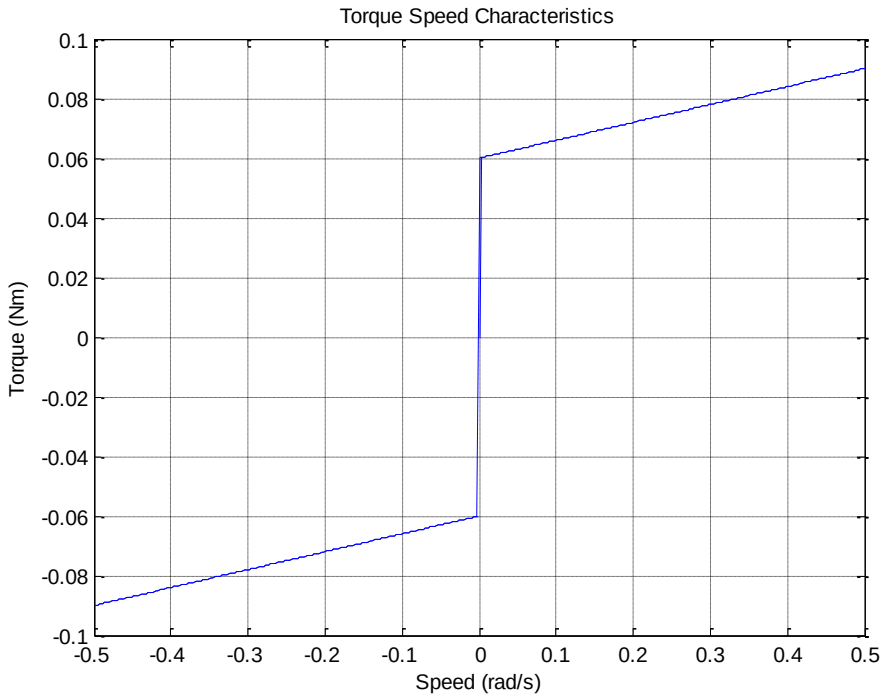


Figure 44: Frictional Torque

The network was trained as shown in Figure 45 and the MSE after training for 10000 iterations is shown in Figure 46. The MSE after training was found to be 6.4572e-6.

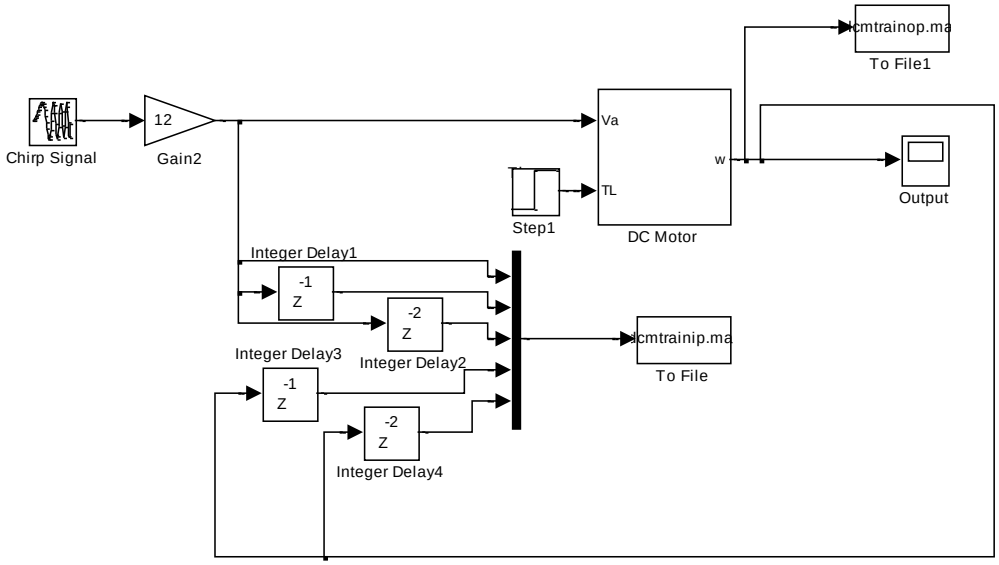


Figure 45: Training RWN for DC Motor, Bilinear Discretization

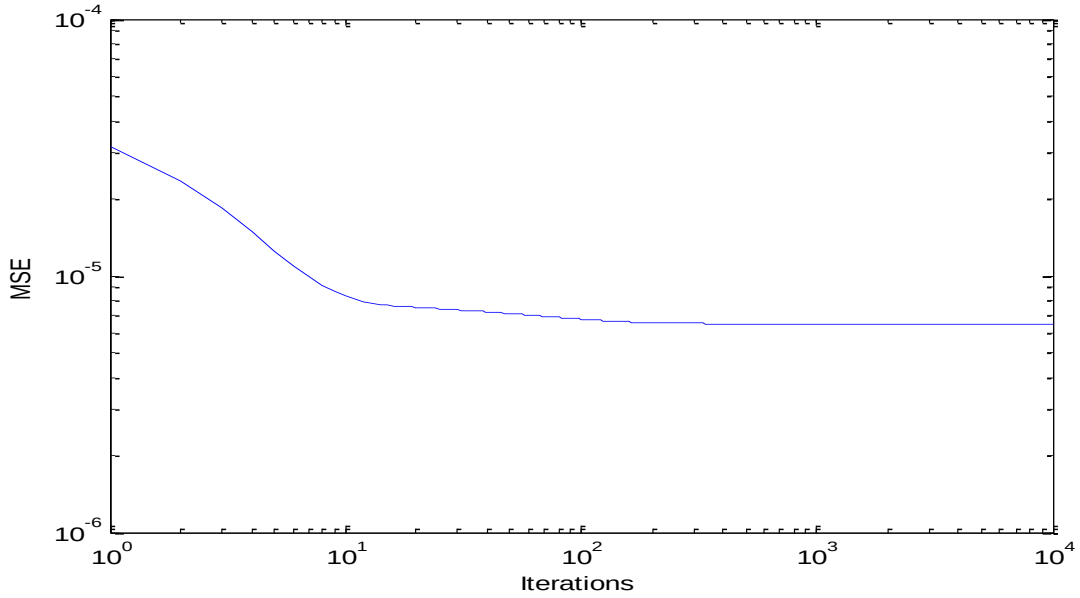


Figure 46: MSE for DC Motor RWN, Bilinear Discretization

Since μ_a and μ_w are constant values and $\mu_c = \mu_m$, the change in μ_m and μ_d over time is observed. It was observed that the parameters only changed in the first iteration and then remained constant for the rest of the training. The change in parameters over the first cycle is shown in Figure 47.

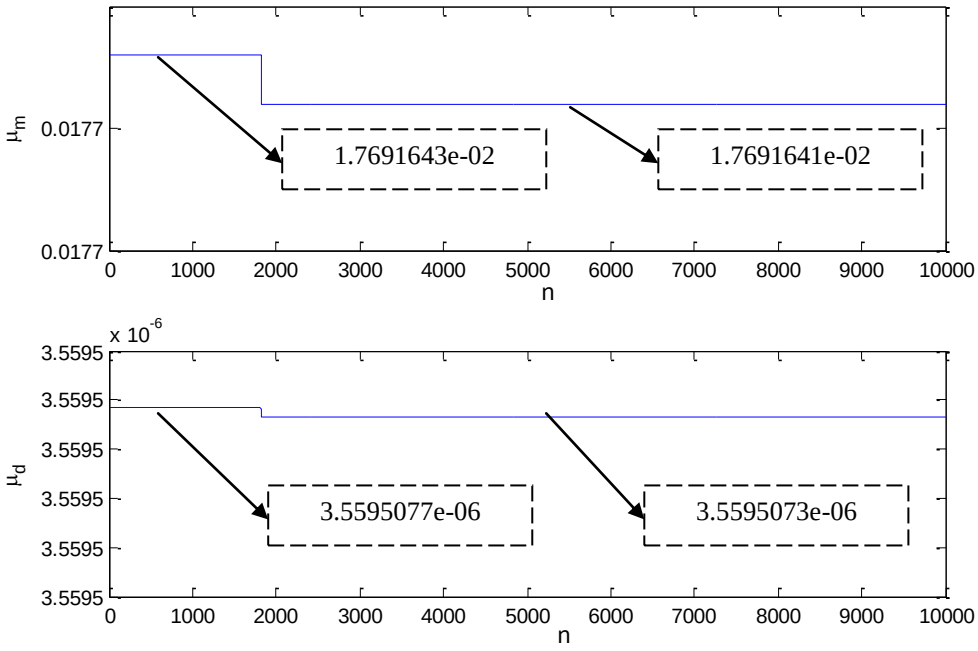


Figure 47: Learning Rates for Translation and Dilation over First Training Cycle

Testing was then carried out as shown in Figure 48, using a sinusoidal input of magnitude 3V and frequency 1Hz. From Figure 49, it can be seen that the network has been trained to represent the DC Motor with nonlinear friction. The MSE after testing was found to be 1.2912e-5.

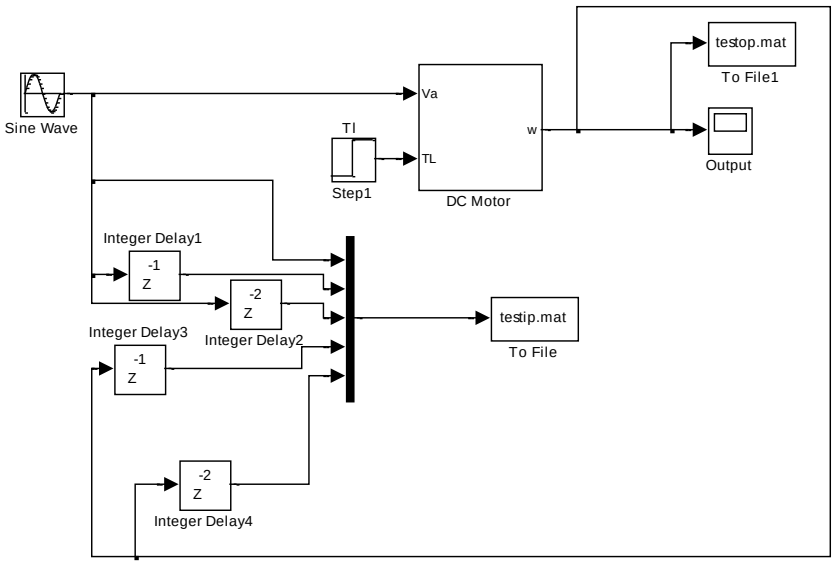


Figure 48: Testing RWN for DC Motor, Bilinear Discretization

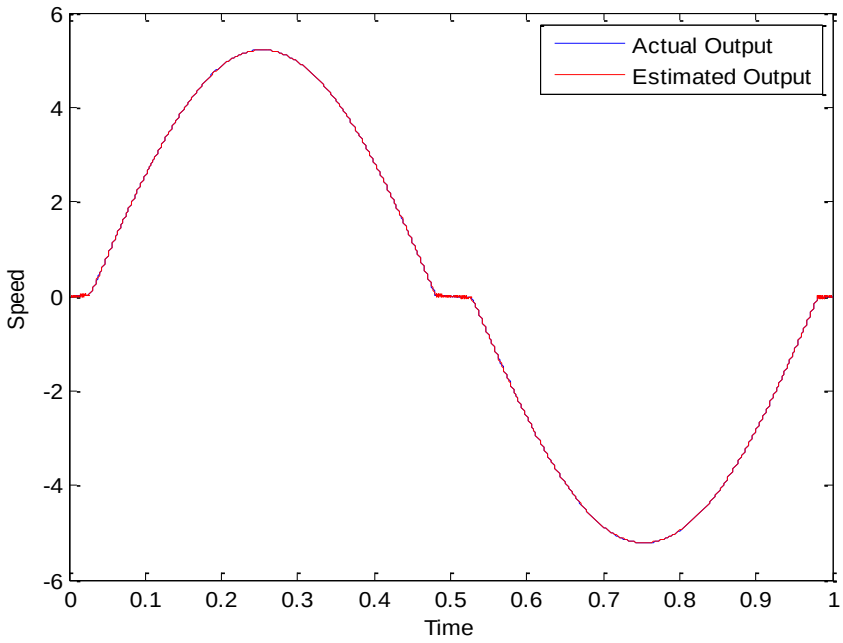


Figure 49: DC Motor RWN Test Output, Bilinear Discretization

3.5.2 Nonlinear DC motor discretized using Euler Forward transformation. In this section, the training inputs were selected based on the Euler Forward discretization of the DC Motor given in Equation 3.96. Equation 3.97 shows the discretized linear DC Motor model.

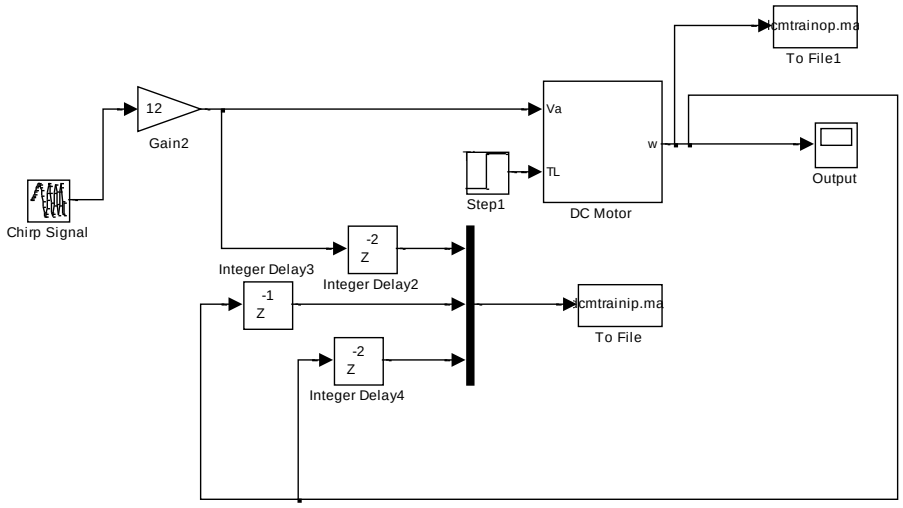
$$s = \frac{z-1}{T} \tag{3.96}$$

$$\frac{\omega(z)}{V(z)} = \frac{T^2 K_t z^{-2}}{LJ + (-2LJ + TBL + TRJ)z^{-1} + (LJ - TBL - TRJ + T^2BR + T^2K_tK_b)z^{-2}} \tag{3.97}$$

From Equation 3.97 it can be seen that the speed of the DC Motor is a function of the inputs shown in Equation 3.98.

$$\omega(n) = f(V(n-2), \omega(n-1), \omega(n-2)) \tag{3.98}$$

The RWN with 3 neurons in the hidden layer was trained as shown in Figure 50, using a chirp signal of magnitude 12 and frequency 0.1-4Hz over an interval of 1s, and the MSE after training for 10000 iterations is shown in Figure 51. The MSE after training was found to be 6.8587e-6.



.

Figure 50: Training RWN for DC Motor, Euler Forward Discretization

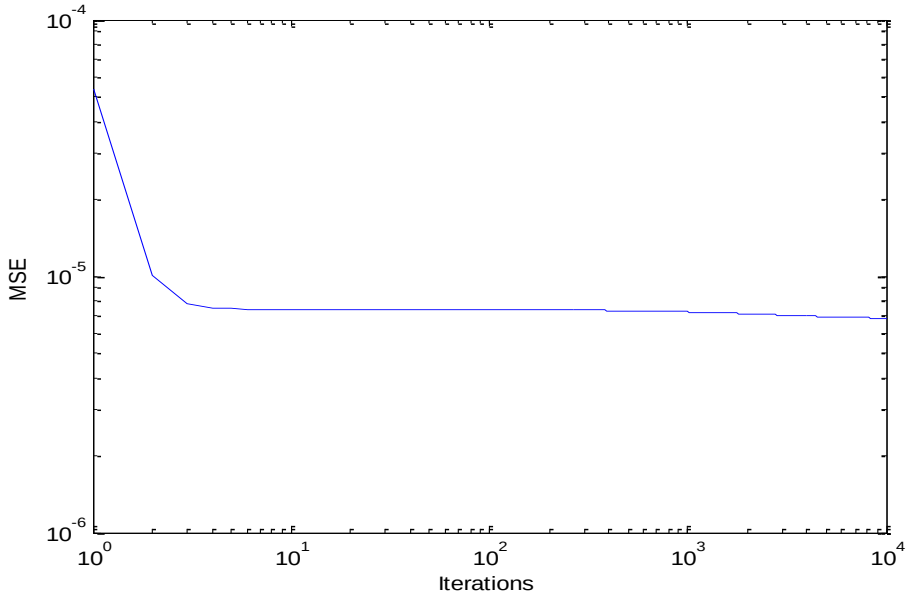


Figure 51: MSE for DC Motor RWN, Euler Forward Discretization

Since μ_a and μ_w are constant values and $\mu_c = \mu_m$, the change in μ_m and μ_d over time is observed. It was observed that based on the initial conditions selected, the parameters remained constant over the course of the training as shown in Figure 52.

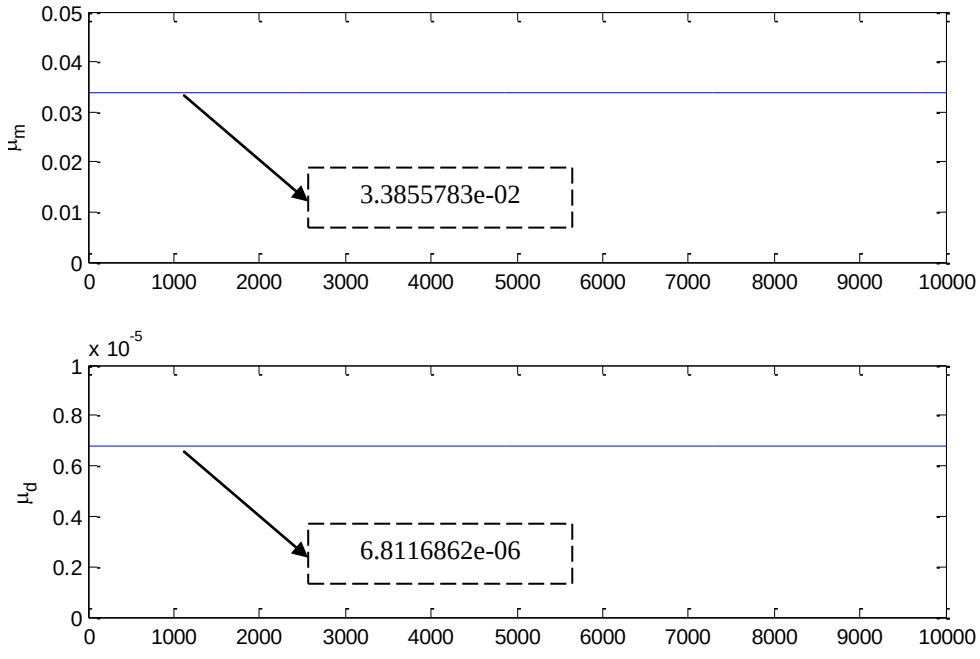


Figure 52: Learning Rates for Translation and Dilation over First Training Cycle

Testing was then carried out as shown in Figure 53, using a sinusoidal input of magnitude 3V and frequency 1Hz. From Figure 54, it can be seen that the network has been trained to represent the DC Motor with nonlinear friction. The MSE after testing was found to be 1.6747e-5.

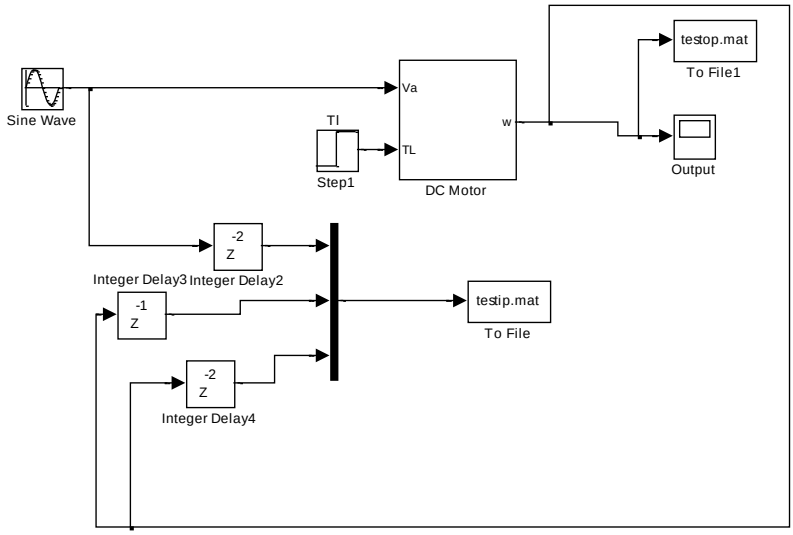


Figure 53: Testing RWN for DC Motor, Euler Forward Discretization

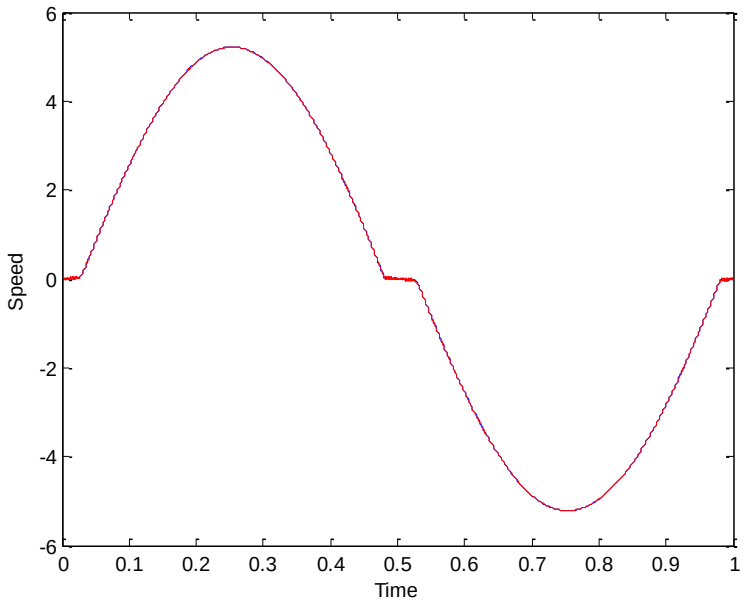


Figure 54: DC Motor RWN Test Output, Euler Forward Discretization

From the two results, it can be seen that a priori knowledge of the DC motor structure enables the selection of the most relevant inputs to ensure that the RWN provides a highly accurate black box model of the DC Motor with nonlinearities such as viscous and coulomb friction.

3.5.3 Friction. In this section, the RWN was trained to learn the nonlinear friction of a DC Motor as shown in Figure 55. Here it is assumed that the frictional torque $\tau_f(\omega)$ is a measurable value.

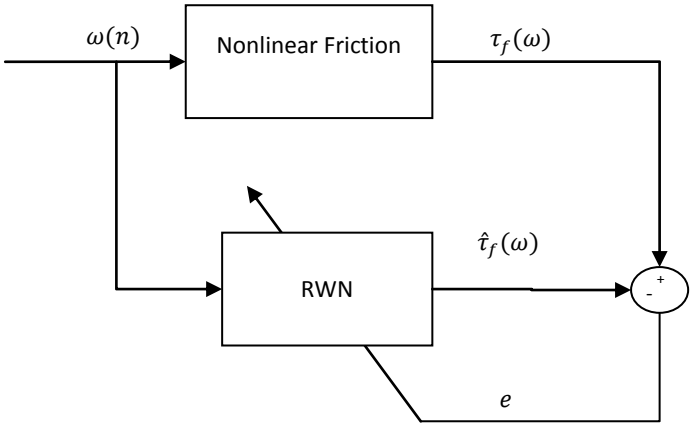


Figure 55: Nonlinear Friction Identification

The frictional function in consideration is the Tustin Armstrong Helouvry model where the signum function is approximated using a hyperbolic tan function, represented using Equation 3.99 and shown in Figure 56.

$$\tau_f(\omega) = \tanh(\beta\omega) \times \left(T_c + (T_s - T_c)e^{-\alpha|\omega|^2}\right) + B\omega \tag{3.99}$$

where β is a large positive constant, T_c represents the Coulomb friction, T_s represents the static friction coefficient, B represents the viscous friction coefficient and α represents the stiction coefficient. For the simulation, β is taken as 10.

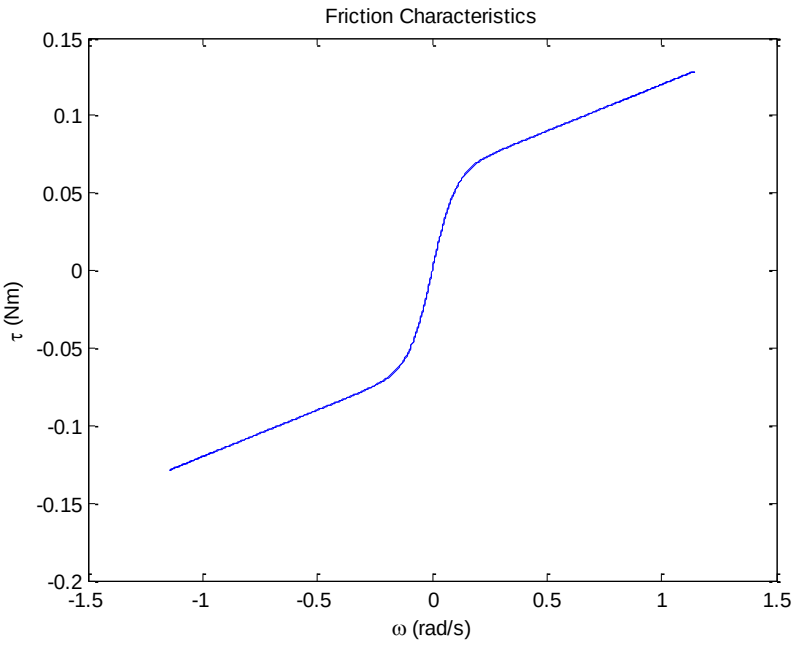


Figure 56: Friction Characteristics

The network used for training is shown in Figure 57. A 1V sine wave with frequency of 1Hz was applied to the DC motor for generating the training signal ω which is shown in Figure 58.

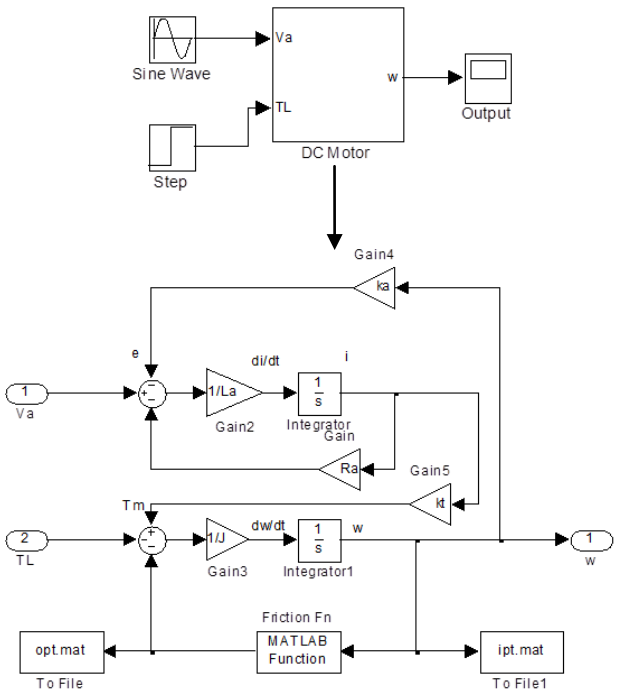


Figure 57: Generating Training Data for Friction Identification

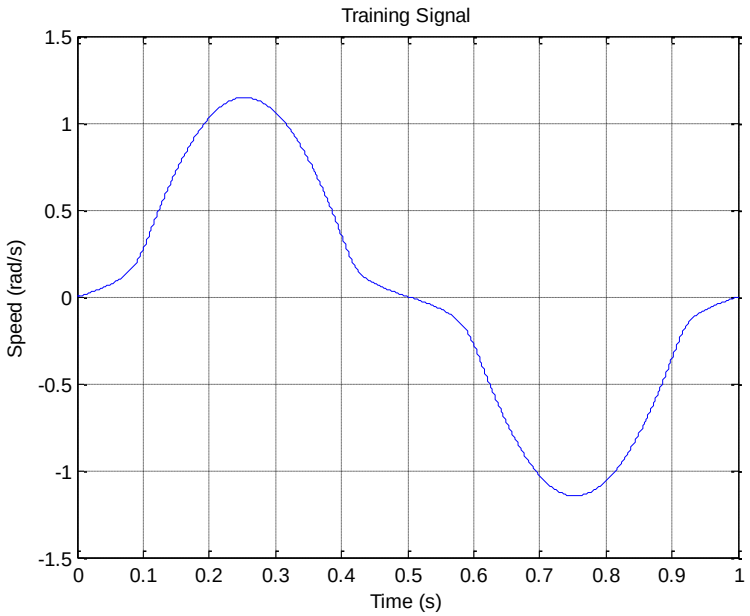


Figure 58: Training Signal for Friction Identification

Training was carried out for 100000 iterations for varying number of neurons in the hidden layer. The MSE after training is shown in Figure 59 and given in Table 11.

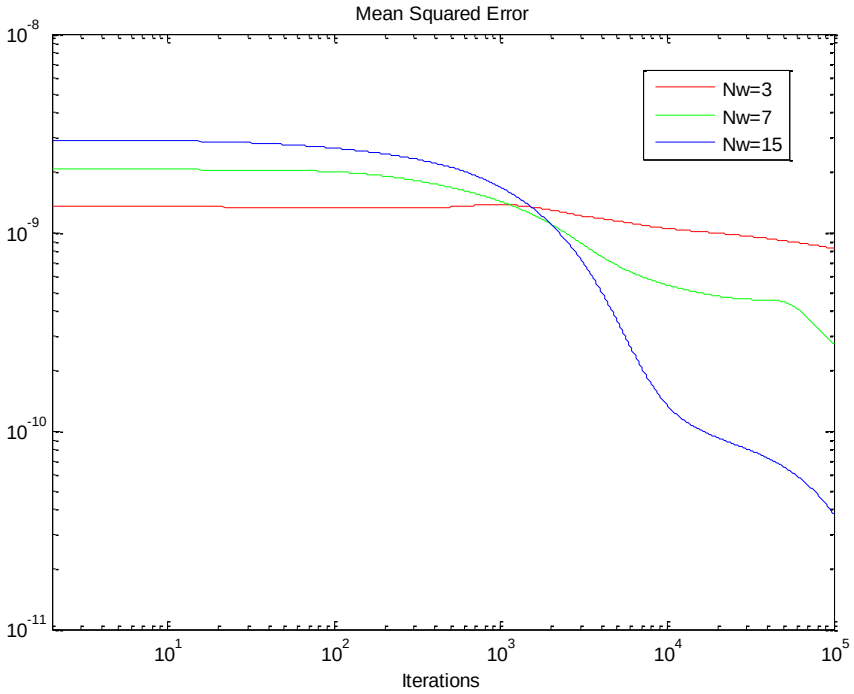


Figure 59: MSE for Varying Nw for Friction Identification

Table 11: MSE for Varying Nw for Friction Identification

Nw	MSE
3	8.2753e-10
7	2.7731e-10
15	3.7961e-11

Since μ_a and μ_w are constant values and $\mu_c = \mu_m$, the change in μ_m and μ_d over time is observed. Figures 60-62 shows the learning rate change during the first training cycle and the last training cycle for both parameters for each of the three network sizes tested.

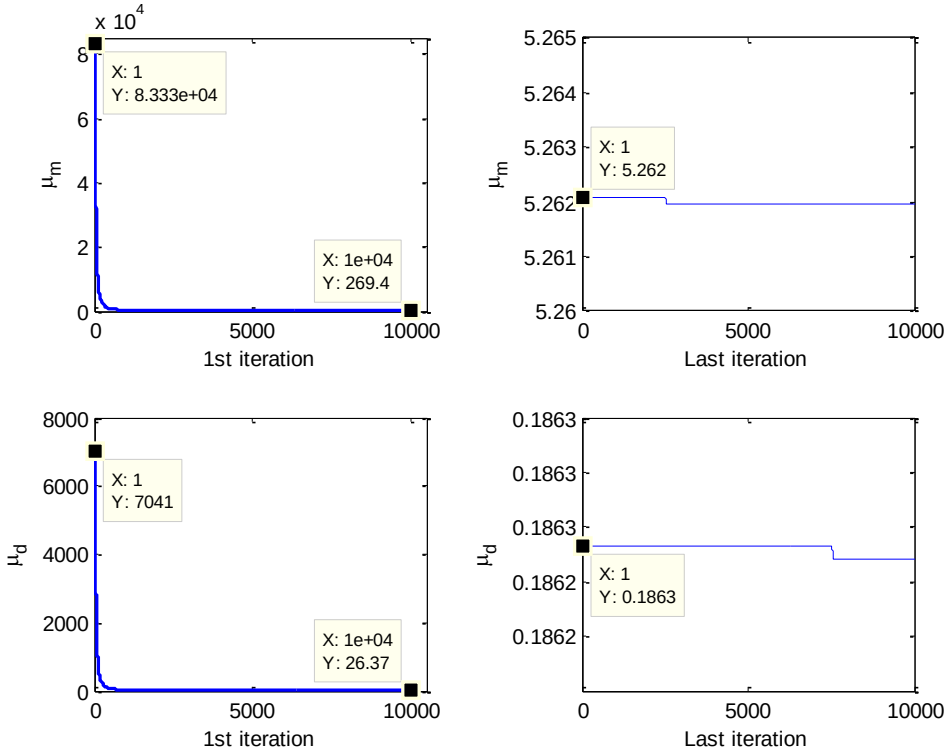


Figure 60: Learning Rates for Translation and Dilation over First and Last Training Cycle, Nw=3

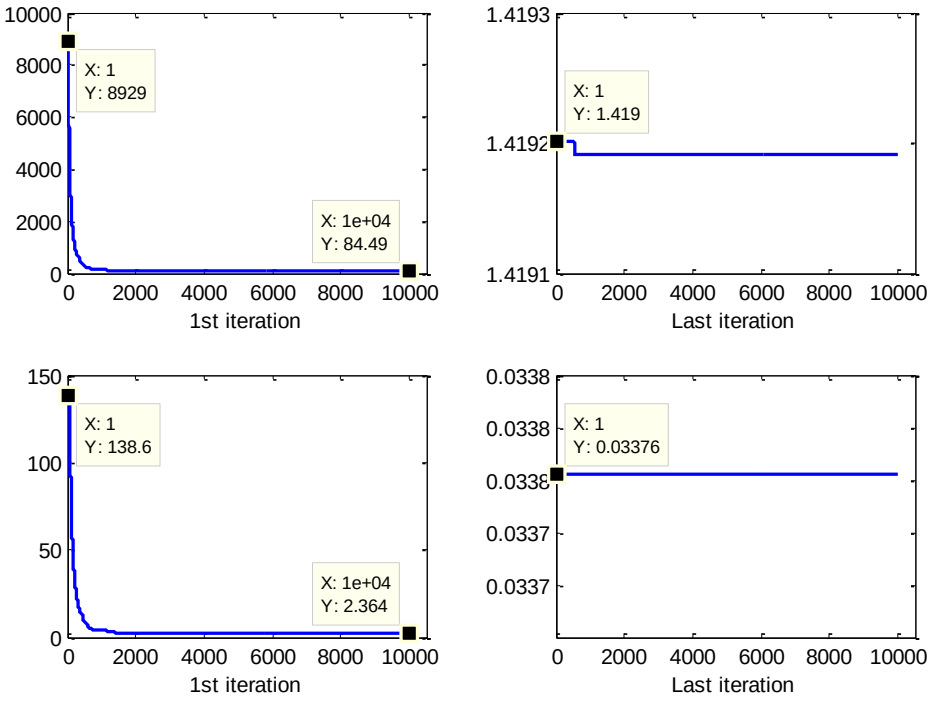


Figure 61: Learning Rates for Translation and Dilation over First and Last Training Cycle, Nw=7

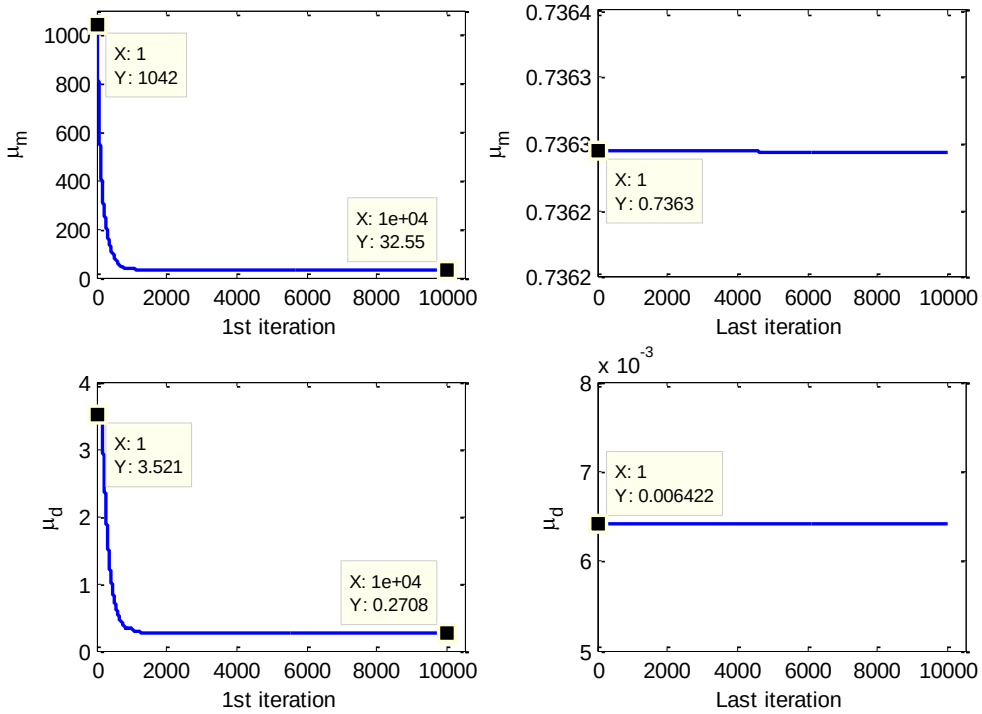


Figure 62: Learning Rates for Translation and Dilation over First and Last Training Cycle, Nw=15

Testing was then carried out by applying a chirp signal of magnitude 0.75V and frequency ranging from 0.1-4Hz to the DC Motor to generate the test signal shown in Figure 63.

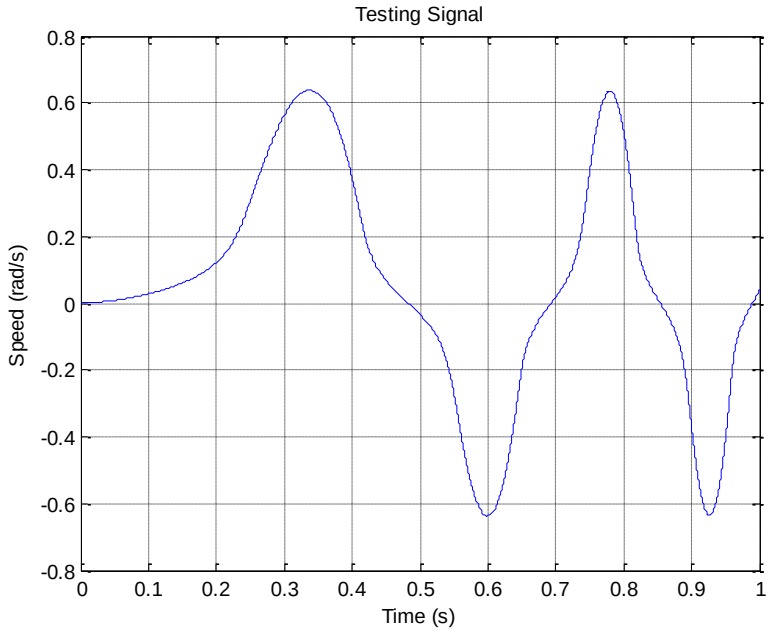


Figure 63: Testing Signal for Friction Identification

The error after testing is shown in Table 12 and the torque speed characteristics of the RWN are shown in Figure 64. From Figure 65 it can be seen that using 15 neurons in the hidden layer allowed for accurate representation of the frictional function.

Table 12: MSE for Varying Nw after Testing for Friction Identification

Nw	**MSE**
3	1.3724e-4
7	1.4662e-5
15	4.9190e-7

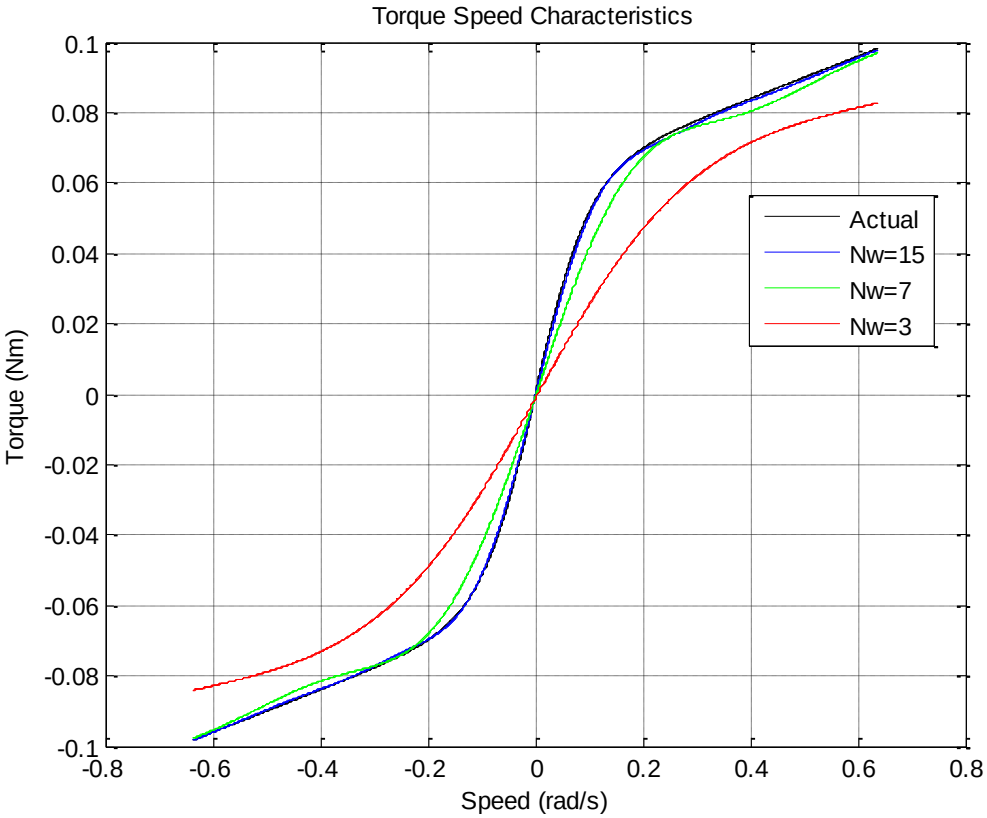


Figure 64: Torque Speed Characteristic for Varying Nw after Testing for Friction Identification

Chapter 4: DC Motor Identification Using Structured Recurrent Wavelet Network

From the results of Chapter 3, it is evident that a DC motor with nonlinearities can be modelled using a recurrent wavelet network. In this section, a structured recurrent wavelet network is designed to allow for simultaneous linear and nonlinear parameter identification of a DC motor with nonlinear frictional characteristics.

4.1 DC Motor Model Derivation

The DC Motor, as shown in Figure 65, is represented by the Equations 4.1 and 4.2.

$$J\frac{d\omega}{dt} = K_t i - B\omega - \tau_f(\omega) \tag{4.1}$$

$$L\frac{di}{dt} = V - Ri - K_b\omega \tag{4.2}$$

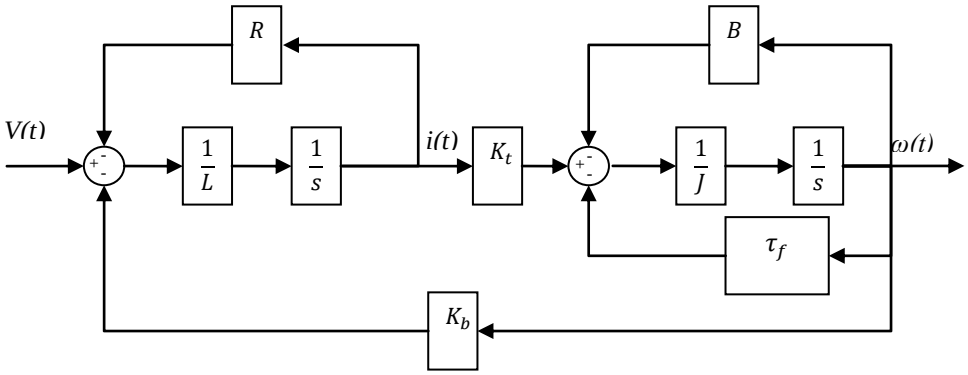


Figure 65: Continuous Time Model of DC Motor

The State Space model of the DC Motor is given in Equations 4.3 and 4.4.

$$\begin{bmatrix} \frac{d\omega}{dt} \\ \frac{di}{dt} \end{bmatrix} = \begin{bmatrix} -\frac{B}{J} & \frac{K_t}{J} \\ -\frac{K_b}{L} & -\frac{R}{L} \end{bmatrix} \begin{bmatrix} \omega \\ i \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix} V + \begin{bmatrix} -\frac{1}{J} \\ 0 \end{bmatrix} \tau_f(\omega) \tag{4.3}$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \omega \\ i \end{bmatrix} \tag{4.4}$$

The DC Motor system is discretized using the Euler-forward method which is represented in Equation 4.5 and shown in block diagram form in Figure 66. Here, T represents the sampling time which is selected to be 5 to 10 times less than the electrical time constant of the DC Motor.

$$y(n) = y(n-1) + Tx(n-1) \tag{4.5}$$

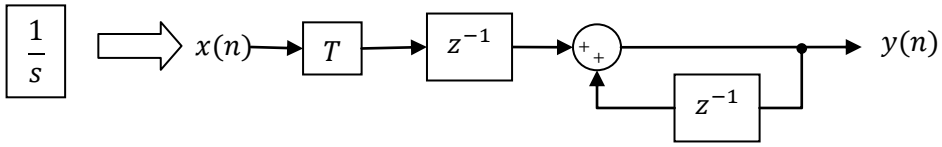


Figure 66: Discretized Integrator using Euler-forward Method

The discretized DC Motor model is shown in Figure 67.

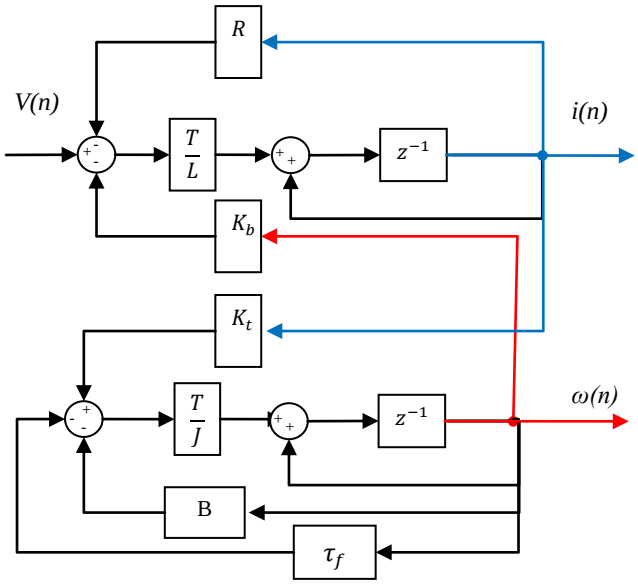


Figure 67: Step 1 of Discretization of DC Motor System

After discretization, the next step involves developing a model for a structured recurrent network which can be trained to learn both the linear and nonlinear parameters of the DC Motor system. Figure 68 and 69 show the intermediate steps in developing the structured network which is shown in Figure 70.

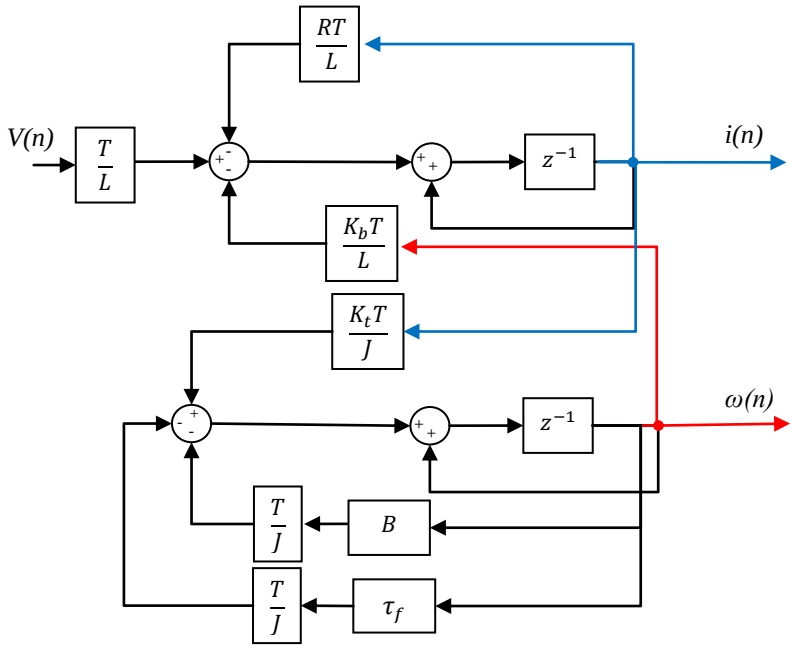


Figure 68: Step 2 of Discretization of DC Motor Model

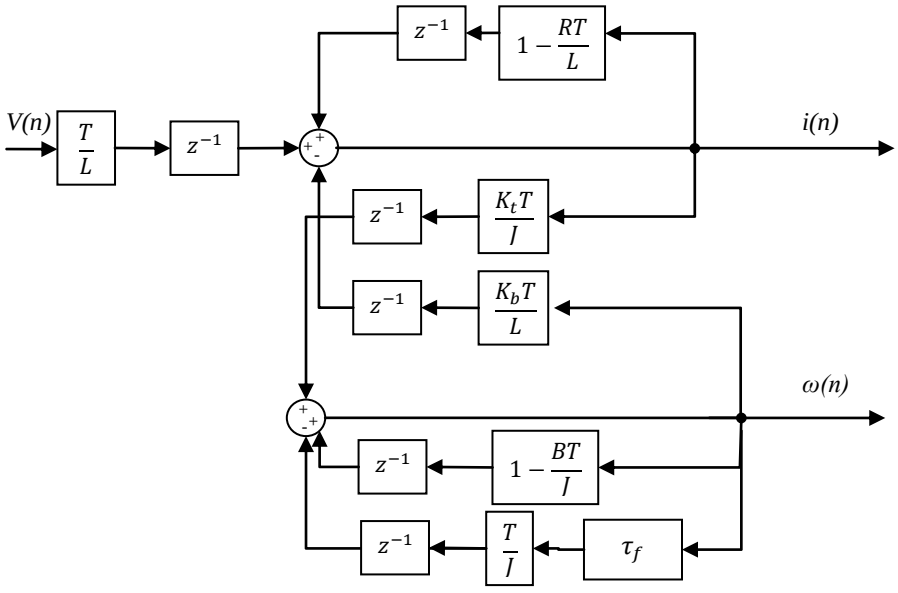


Figure 69: Step 3 of Discretization of DC Motor Model

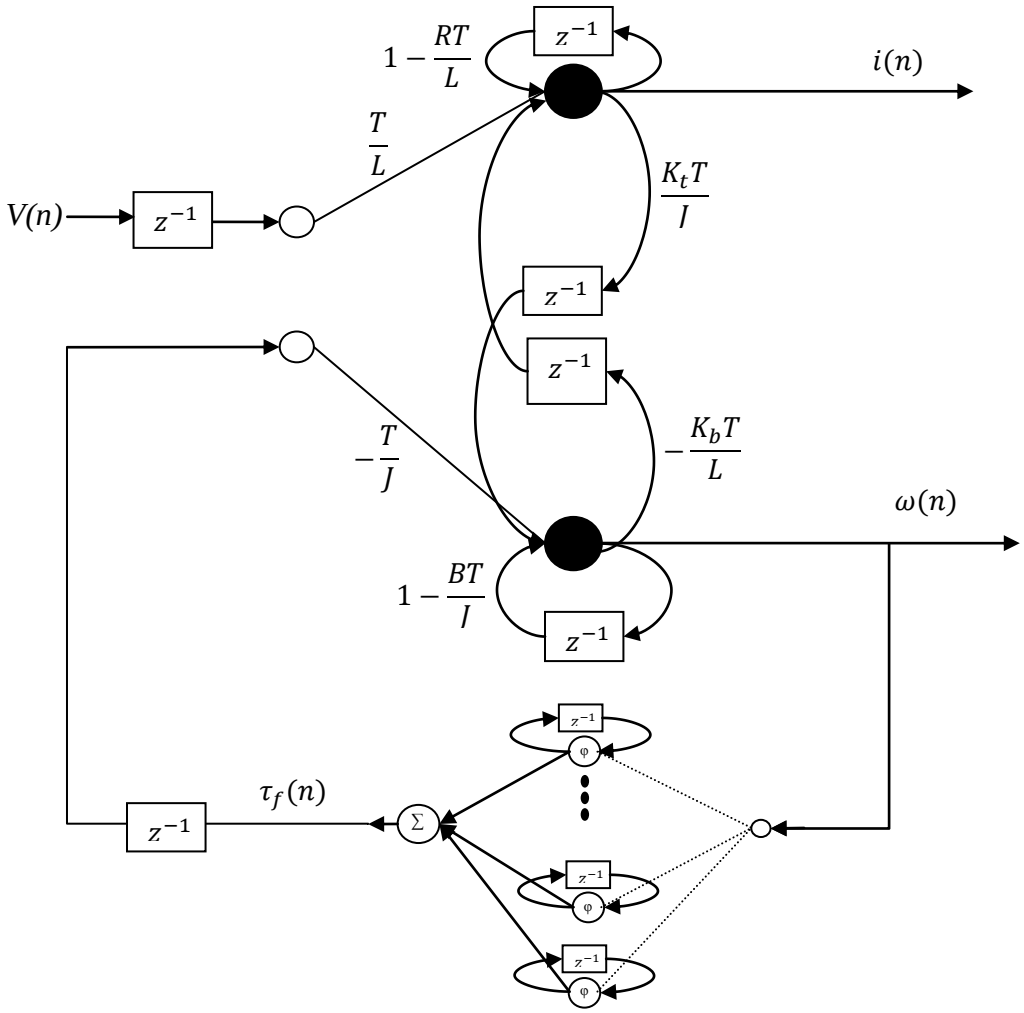


Figure 70: Structured Recurrent Wavelet Network Representing DC Motor Model

4.2 Architecture for DC Motor Parameter Identification

The following structure, shown in Figure 71, shows the online training structure for the structured recurrent wavelet network representing the DC Motor. This structure allows for simultaneous learning of both the linear and nonlinear parts of the system. For the purpose of this research it is assumed that the time constant of the motor being modelled is available. In order to train the network, since $\omega(n)$ is a measurable output of the system, $\widehat{\tau_f}$ is obtained as a function of $\omega(n)$ rather than the approximated speed $\widehat{\omega}(n)$.

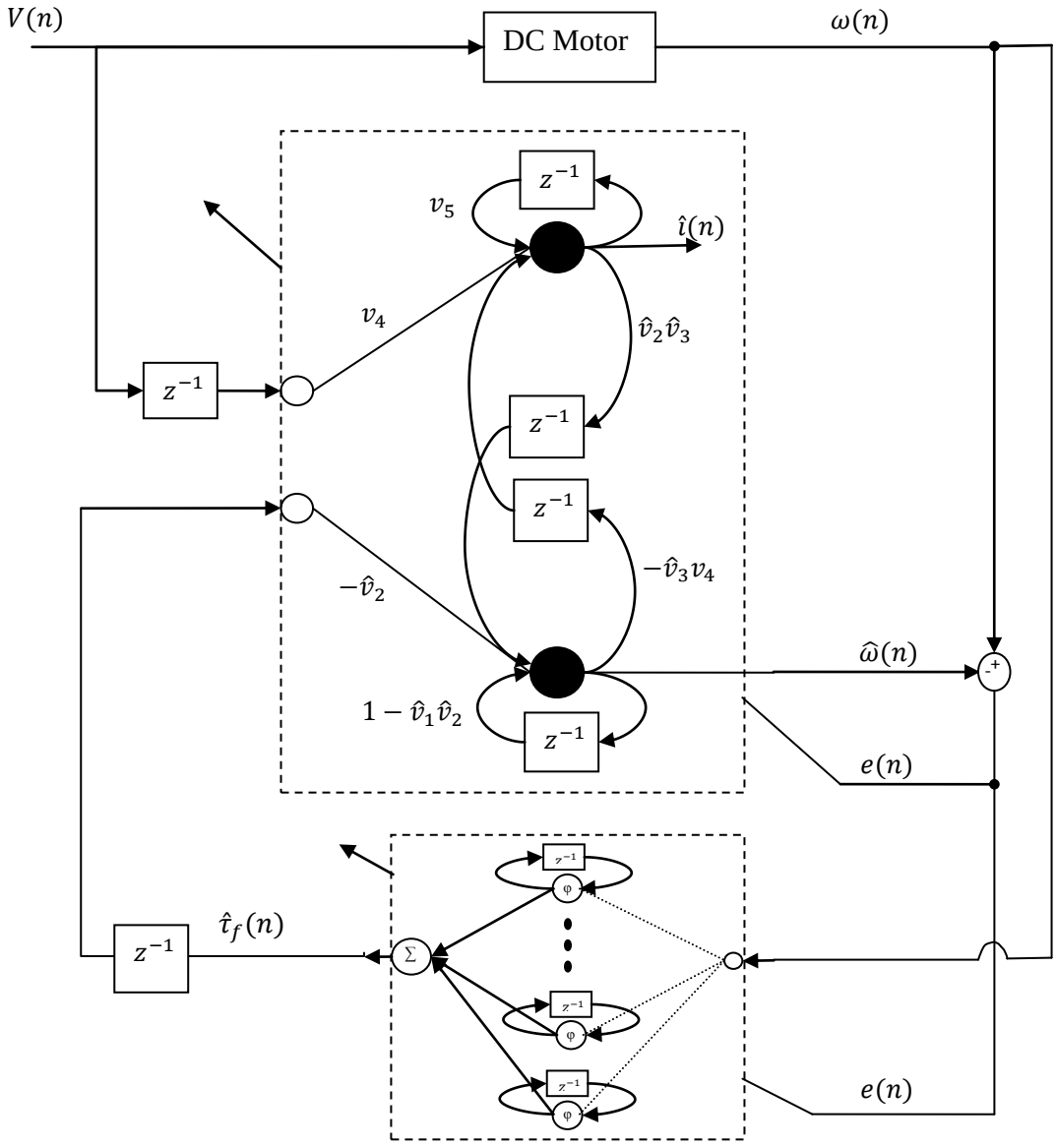


Figure 71: DC Motor Parameter Identification Training Structure

The linear network parameters are given by the vector $\hat{v}$ and the parameters of the recurrent wavelet network are given by the vector θ as given below.

$$\hat{v} = [\hat{v}_1 \quad \hat{v}_2 \quad \hat{v}_3 \quad v_4 \quad v_5] = \left[B \quad \frac{T}{J} \quad K_t \quad \frac{T}{L} \quad 1 - \frac{RT}{L}\right]$$

where v_4 and $v_5$are assumed to be known a priori.

$$\hat{\theta} = \left[\hat{w}_j \quad \hat{m}_{ji} \quad \hat{d}_{ji} \quad \hat{c}_{ji}\right]^T.$$

The network can be represented in state space form by Equations 4.6 and 4.7.

$$\begin{bmatrix}\hat{x}_1(n)\\ \hat{x}_2(n)\end{bmatrix} = \begin{bmatrix}1-\hat{v}_1\hat{v}_2 & \hat{v}_2\hat{v}_3\\ -\hat{v}_3 v_4 & v_5\end{bmatrix}\begin{bmatrix}\hat{x}_1(n-1)\\ \hat{x}_2(n-1)\end{bmatrix} + \begin{bmatrix}0\\ v_4\end{bmatrix}u(n-1) + \begin{bmatrix}-\hat{v}_2\\ 0\end{bmatrix}\hat{\tau}_f(n-1) \tag{4.6}$$

$$\hat{y}(n) = [1 \quad 0]\begin{bmatrix}\hat{x}_1(n)\\ \hat{x}_2(n)\end{bmatrix} \tag{4.7}$$

where

$$\hat{x}(n) = \begin{bmatrix}\hat{x}_1(n)\\ \hat{x}_2(n)\end{bmatrix} = \begin{bmatrix}\widehat{\omega}(n)\\ \hat{\imath}(n)\end{bmatrix} \qquad\qquad u(n) = V(n)$$

The state space equations can then be rewritten in matrix form as shown in Equations 4.8 to 4.10.

$$\hat{x}(n) = \boldsymbol{A}\hat{x}(n-1) + \boldsymbol{B}u(n-1) + \boldsymbol{H}\hat{\tau}_f(n-1) \tag{4.8}$$

$$\hat{y}(n) = \boldsymbol{C}\hat{x}(n) \tag{4.9}$$

$$\hat{\tau}_f(n) = \sum_{j=1}^{N_w} \widehat{w}_j \Phi_j(n) \tag{4.10}$$

4.3 Training Algorithm

A simplified training structure is provided in Figure 72 which shows the structure in terms of Equations 4.8 to 4.10. Training of the structured network is carried out using the gradient descent algorithm which involves adjusting the network parameters, $\hat{v}$ and $\hat{\theta}$ to ensure the minimization of a cost function given by Equation 4.11.

$$J = \frac{1}{2}(y(n) - \hat{y}(n))^2 = \frac{1}{2}e^2(n) \tag{4.11}$$

where e is the error between the desired output and the output of the network.

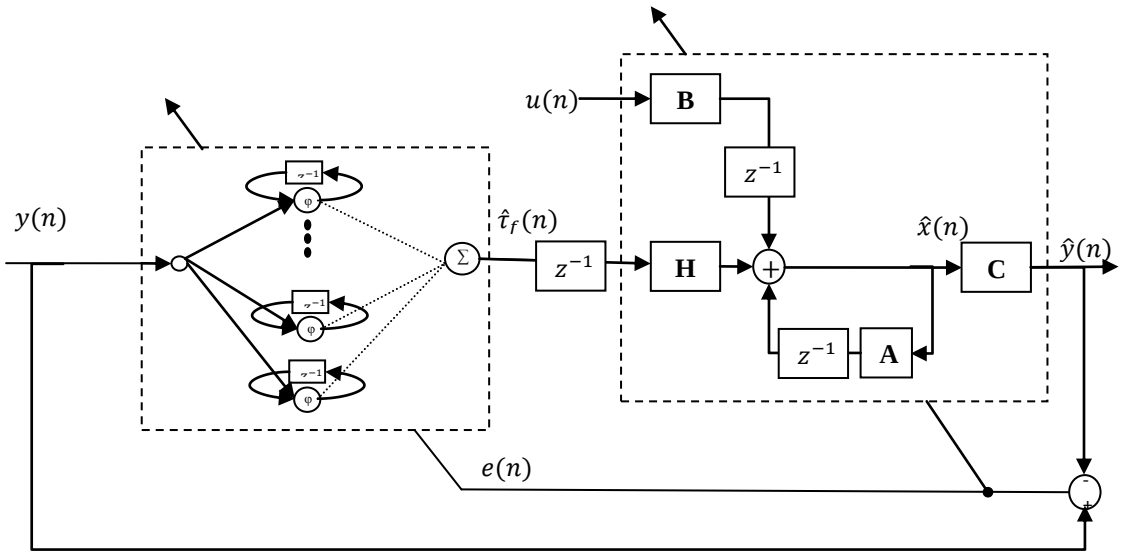


Figure 72: Simplified DC Motor Parameter Identification Training Structure

The partial derivative of the cost function with respect to the linear network parameters, v_i, is given in Equation 4.12.

$$\frac{dJ}{d\hat{v}} = -e(n)\frac{d\hat{y}(n)}{d\hat{v}} \tag{4.12}$$

The partial derivative of the network output $\hat{y}$ with respect to the linear network parameters is given in Equations 4.13 to 4.18. Since $\hat{\tau}_f(n-1)$ is a function of $\omega(n-1)$, $\frac{d\hat{\tau}_f(n-1)}{d\hat{v}_i} = 0$.

$$\frac{d\hat{y}(n)}{d\hat{v}_1} = \frac{d\hat{x}_1(n)}{d\hat{v}_1} = (1-\hat{v}_1\hat{v}_2)\frac{d\hat{x}_1(n-1)}{d\hat{v}_1} + \hat{v}_2\hat{v}_3\frac{d\hat{x}_2(n-1)}{d\hat{v}_1} - \hat{v}_2\hat{x}_1(n-1) \tag{4.13}$$

$$\frac{d\hat{x}_2(n)}{d\hat{v}_1} = -\hat{v}_3 v_4\frac{d\hat{x}_1(n-1)}{d\hat{v}_1} + v_5\frac{d\hat{x}_2(n-1)}{d\hat{v}_1} \tag{4.14}$$

$$\frac{d\hat{y}(n)}{d\hat{v}_2} = \frac{d\hat{x}_1(n)}{d\hat{v}_2} = (1-\hat{v}_1\hat{v}_2)\frac{d\hat{x}_1(n-1)}{d\hat{v}_2} + \hat{v}_2\hat{v}_3\frac{d\hat{x}_2(n-1)}{d\hat{v}_2} - \hat{v}_1\hat{x}_1(n-1) + \hat{v}_3\hat{x}_2(n-1) - \hat{\tau}_f(n-1) \tag{4.15}$$

$$\frac{d\hat{x}_2(n)}{d\hat{v}_2} = -\hat{v}_3 v_4 \frac{d\hat{x}_1(n-1)}{d\hat{v}_2} + v_5 \frac{d\hat{x}_2(n-1)}{d\hat{v}_2} \tag{4.16}$$

$$\frac{d\hat{y}(n)}{d\hat{v}_3} = \frac{d\hat{x}_1(n)}{d\hat{v}_3} = (1-\hat{v}_1\hat{v}_2)\frac{d\hat{x}_1(n-1)}{d\hat{v}_3} + \hat{v}_2\hat{v}_3\frac{d\hat{x}_2(n-1)}{d\hat{v}_3} + \hat{v}_2\hat{x}_2(n-1) \tag{4.17}$$

$$\frac{d\hat{x}_2(n)}{d\hat{v}_3} = -\hat{v}_3 v_4 \frac{d\hat{x}_1(n-1)}{d\hat{v}_3} + v_5 \frac{d\hat{x}_2(n-1)}{d\hat{v}_3} - v_4\hat{x}_1(n-1) \tag{4.18}$$

The recursive equation of the system can be rewritten such that,

$$P_i(n) = \frac{d\hat{y}(n)}{d\hat{v}_i} = \frac{d\hat{x}_1(n)}{d\hat{v}_i} \qquad\qquad Q_i(n) = \frac{d\hat{x}_2(n)}{d\hat{v}_i}$$

The linear parameters, v_i, of the structured network are updated using Equation 4.19.

$$\hat{v}_i(n+1) = \hat{v}_i(n) - \mu\frac{dJ}{d\hat{v}_i} = \hat{v}_i(n) + \mu_{\hat{v}_i}(n)e(n)\frac{d\hat{y}(n)}{d\hat{v}_i} \tag{4.19}$$

The partial derivative of the cost function with respect to the parameters of the recurrent wavelet network used to model the friction is given in Equation 4.20.

$$\frac{dJ}{d\hat{\theta}} = -e(n)\frac{d\hat{y}(n)}{d\hat{\theta}} \tag{4.20}$$

The partial derivative of the network output $\hat{y}$ with respect to the parameters of the recurrent wavelet network is given in Equations 4.21 to 4.28.

$$\frac{d\hat{y}(n)}{d\hat{w}_j} = \frac{d\hat{x}_1(n)}{d\hat{w}_j} = (1-\hat{v}_1\hat{v}_2)\frac{d\hat{x}_1(n-1)}{d\hat{w}_j} + \hat{v}_2\hat{v}_3\frac{d\hat{x}_2(n-1)}{d\hat{w}_j} - \hat{v}_2\frac{d\hat{\tau}_f(n-1)}{d\hat{w}_j} \tag{4.21}$$

$$\frac{d\hat{x}_2(n)}{d\hat{w}_j} = -\hat{v}_3 v_4 \frac{d\hat{x}_1(n-1)}{d\hat{w}_j} + v_5 \frac{d\hat{x}_2(n-1)}{d\hat{w}_j} \tag{4.22}$$

$$\frac{d\hat{y}(n)}{d\hat{m}_j} = \frac{d\hat{x}_1(n)}{d\hat{m}_j} = (1-\hat{v}_1\hat{v}_2)\frac{d\hat{x}_1(n-1)}{d\hat{m}_j} + \hat{v}_2\hat{v}_3\frac{d\hat{x}_2(n-1)}{d\hat{m}_j} - \hat{v}_2\frac{d\hat{\tau}_f(n-1)}{d\hat{m}_j} \tag{4.23}$$

$$\frac{d\hat{x}_2(n)}{d\hat{m}_j} = -\hat{v}_3 v_4 \frac{d\hat{x}_1(n-1)}{d\hat{m}_j} + v_5 \frac{d\hat{x}_2(n-1)}{d\hat{m}_j} \tag{4.24}$$

$$\frac{d\hat{y}(n)}{d\hat{d}_j} = \frac{d\hat{x}_1(n)}{d\hat{d}_j} = (1 - \hat{v}_1\hat{v}_2)\frac{d\hat{x}_1(n-1)}{d\hat{d}_j} + \hat{v}_2\hat{v}_3\frac{d\hat{x}_2(n-1)}{d\hat{d}_j} - \hat{v}_2\frac{d\hat{\tau}_f(n-1)}{d\hat{d}_j} \quad (4.25)$$

$$\frac{d\hat{x}_2(n)}{d\hat{d}_j} = -\hat{v}_3 v_4\frac{d\hat{x}_1(n-1)}{d\hat{d}_j} + v_5\frac{d\hat{x}_2(n-1)}{d\hat{d}_j} \quad (4.26)$$

$$\frac{d\hat{y}(n)}{d\hat{c}_j} = \frac{d\hat{x}_1(n)}{d\hat{c}_j} = (1 - \hat{v}_1\hat{v}_2)\frac{d\hat{x}_1(n-1)}{d\hat{c}_j} + \hat{v}_2\hat{v}_3\frac{d\hat{x}_2(n-1)}{d\hat{c}_j} - \hat{v}_2\frac{d\hat{\tau}_f(n-1)}{d\hat{c}_j} \quad (4.27)$$

$$\frac{d\hat{x}_2(n)}{d\hat{c}_j} = -\hat{v}_3 v_4\frac{d\hat{x}_1(n-1)}{d\hat{c}_j} + v_5\frac{d\hat{x}_2(n-1)}{d\hat{c}_j} \quad (4.28)$$

The recursive equation of the system can be rewritten such that,

$$P_{\hat{\theta}_j}(n) = \frac{d\hat{y}(n)}{d\hat{\theta}_j} = \frac{d\hat{x}_1(n)}{d\hat{\theta}_j} \qquad Q_{\hat{\theta}_j}(n) = \frac{d\hat{x}_2(n)}{d\hat{\theta}_j}$$

The parameters of the RWN are then updated using Equation 4.29.

$$\hat{\theta}(n+1) = \hat{\theta}(n) - \mu\frac{dJ}{d\hat{\theta}} = \hat{\theta}(n) + \mu_{\hat{\theta}_i}(n)e(n)\frac{d\hat{y}(n)}{d\hat{\theta}} \quad (4.29)$$

4.4 Convergence and Stability Analysis

In order to guarantee convergence of the proposed simultaneous identification structure, adaptive learning rates are derived from the discrete Lyapunov stability theorem as described in Chapter 3. The adaptive learning rates for the parameters, $\hat{v}_i$ and $\hat{\theta}_i$ are computed as shown in Equation 4.30 and 4.31. The ALRs are updated as per Figure 43.

$$0 < \mu_{\hat{v}_i}(n) < \frac{2}{\max_n \left\| \frac{\partial\hat{y}(n)}{\partial\hat{v}_i} \right\|^2} \quad (4.30)$$

$$0 < \mu_{\hat{\theta}_i}(n) < \frac{2}{\max_n \left\| \frac{\partial\hat{y}(n)}{\partial\hat{\theta}_i} \right\|^2} \quad (4.31)$$

4.5 Adaptive Learning Rates

The following two theorems are used to derive the adaptive learning rates. Theorem 1 shown in Equation 4.32 states that:

$$\lim_{k\to\infty} A^k = 0 \quad iff\ \rho(A) = \max_{1\le i\le n}(|\lambda_i|) < 1 \tag{4.32}$$

where $\rho(A)$ is the spectral radius of the matrix A [27]. From this it can be concluded that when $\rho(A) < 1$, Equation 4.33 is valid.

$$\lim_{k\to\infty} \|A^k\| = 0 \tag{4.33}$$

The eigen decomposition theorem states that any non-singular matrix A^k can be decomposed as shown by Equation 4.34.

$$A^k = MD^kM^{-1} \tag{4.34}$$

where *D* is a diagonal matrix where the diagonal elements are equal to the eigenvalues of *A* and *M* is the matrix of eigenvectors of *A*. If a matrix does not have a set of linearly independent eigenvectors, the matrix is not diagonalizable. Matrices that are not diagonalizable do not have an eigen decomposition.

- $\mu_{\hat{v}_1}$

The learning rate, $\mu_{\hat{v}_1}$, is selected so as to satisfy Equation 4.35.

$$0 < \mu_{\hat{v}_1} < \frac{2}{\max_n \left\| \frac{\partial \hat{y}(n)}{\partial \hat{v}_1} \right\|^2} \tag{4.35}$$

In order to compute the norm of $\frac{\partial \hat{y}(n)}{\partial \hat{v}_1}$, the solution of Equation 4.36 must be computed.

$$\begin{bmatrix} P_1(n) \\ Q_1(n) \end{bmatrix} = \begin{bmatrix} 1-\hat{v}_1\hat{v}_2 & \hat{v}_2\hat{v}_3 \\ -\hat{v}_3 v_4 & v_5 \end{bmatrix} \begin{bmatrix} P_1(n-1) \\ Q_1(n-1) \end{bmatrix} + \begin{bmatrix} -\hat{v}_2 \\ 0 \end{bmatrix} \hat{x}_1(n-1) \tag{4.36}$$

where

$$P_1(n) = \frac{d\hat{x}_1(n)}{d\hat{v}_1} \; and \; Q_1(n) = \frac{d\hat{x}_2(n)}{d\hat{v}_1}$$

Equation 4.36 can be rewritten as shown in Equation 4.37 and 4.38

$$R_1(n) = AR_1(n-1) + E_1\hat{x}_1(n-1) \tag{4.37}$$

$$P_1(n) = CR_1(n) \tag{4.38}$$

where $R_1(n) = \begin{bmatrix} P_1(n) \\ Q_1(n) \end{bmatrix}$

The solution to the Equation 4.38 is shown in Equation 4.39.

$$P_1(n) = C\left(A^n R_1(0) + \sum_{m=0}^{n-1} A^{n-m-1} E_1 \hat{x}_1(m)\right) \tag{4.39}$$

Since $R_1(0) = 0$, Equation 4.39 can be simplified to Equation 4.40.

$$P_1(n) = C\sum_{m=0}^{n-1} A^{n-m-1} E_1 \hat{x}_1(m) \tag{4.40}$$

Changing the index such that *r=n-m-1*, Equation 4.40 can be rewritten as shown in Equation 4.41.

$$P_1(n) = C\sum_{r=0}^{n-1} A^r E_1 \hat{x}_1(n-r-1) \tag{4.41}$$

The norm of $P_1(n)$ is given in Equation 4.42.

$$\|P_1(n)\| = \left\| C\left(\sum_{r=0}^{n-1} A^r \, E_1 \hat{x}_1(n-r-1)\right)\right\| \tag{4.42}$$

Using Rule 1 and 2 of matrix norms given in Appendix D, Equation 4.42 can be further decomposed as shown in Equations 4.43.

$$\|P_1(n)\| \leq \|C\|.\|E_1\|.\|\hat{x}_1(n-1)\| + \cdots + \|C\|.\|A^{n-1}\|.\|E_1\|.\|\hat{x}_1(0)\| \tag{4.43}$$

Since $\|C\| = 1,\ \|E_1\| = |\hat{v}_2|$, Equation 4.43 can be simplified to Equation 4.44.

$$\|P_1(n)\| \leq |\hat{v}_2|(\|\hat{x}_1(n-1)\| + \|A\|.\|\hat{x}_1(n-2)\| + \cdots + \|A^{n-1}\|.\|\hat{x}_1(0)\|) \quad (4.44)$$

This can be rewritten as shown in Equation 4.45.

$$\|P_1(n)\| \leq |\hat{v}_2|(1 + \|A\| + \cdots + \|A^{n-1}\|).\|\hat{x}_1\|_{max} \quad (4.45)$$

where

$$\|\hat{x}_1\|_{max} = \max_n \|\hat{x}_1(n)\|$$

Equation 4.45 can then be written in the form of a series summation as shown in Equation 4.46.

$$\|P_1(n)\| \leq |\hat{v}_2|\left(\sum_{r=0}^{n-1}\|A^r\|\right).\|\hat{x}_1\|_{max} \quad (4.46)$$

Using Theorem 1, the sum of the finite series will be less than or equal to the infinite sum of the series as shown in Equation 4.47.

$$\|P_1(n)\| \leq |\hat{v}_2|\left(\sum_{r=0}^{\infty}\|A^r\|\right).\|\hat{x}_1\|_{max} \quad (4.47)$$

From eigenvalue decomposition, Equation 4.47 can then be written as shown in Equation 4.48,

$$\|P_1(n)\| \leq |\hat{v}_2|\left(\sum_{r=0}^{\infty}\|MD^rM^{-1}\|\right).\|\hat{x}_1\|_{max} \quad (4.48)$$

Using Rules 1 to 3 from Appendix D, Equation 4.48 can be written as Equation 4.49.

$$\|P_1(n)\| \leq |\hat{v}_2|\|M\|.\left(\sum_{r=0}^{\infty}\|D\|^r\right).\|M^{-1}\|.\|\hat{x}_1\|_{max} \quad (4.49)$$

This can be further simplified as shown in Equation 4.50, using Rule 4 from Appendix D.

$$\|P_1(n)\| \leq |\hat{v}_2| \left(\frac{\|M\|.\|M^{-1}\|}{1-\rho(A)} \right) \|\hat{x}_1\|_{max} \tag{4.50}$$

Using Equation 4.50, the maximum norm can be computed as shown in Equation 4.51 where $P_{1,max} = \max_n \|P_1(n)\|$.

$$P_{1,max} = \max_n \left\| |\hat{v}_2| \left(\frac{\|M\|.\|M^{-1}\|}{1-\rho(A)} \right) \|\hat{x}_1\|_{max} \right\| \tag{4.51}$$

Substituting Equation 4.51 in Equation 4.35, the adaptive learning rate is selected as shown in Equation 4.52.

$$0 < \mu_{\hat{v}_1} < \frac{2}{P_{1,max}^2} \tag{4.52}$$

- $\mu_{\hat{v}_2}$

  The learning rate, $\mu_{\hat{v}_2}$, is selected so as to satisfy Equation 4.53.

$$0 < \mu_{\hat{v}_2} < \frac{2}{\max_n \left\| \frac{\partial \hat{y}(n)}{\partial \hat{v}_2} \right\|^2} \tag{4.53}$$

In order to compute the norm of $\frac{\partial \hat{y}(n)}{\partial \hat{v}_2}$, the solution of the Equation 4.54 must be computed.

$$\begin{bmatrix} P_2(n) \\ Q_2(n) \end{bmatrix} = \begin{bmatrix} 1-\hat{v}_1\hat{v}_2 & \hat{v}_2\hat{v}_3 \\ -\hat{v}_3 v_4 & v_5 \end{bmatrix} \begin{bmatrix} P_2(n-1) \\ Q_2(n-1) \end{bmatrix} + \begin{bmatrix} \hat{v}_3 \\ 0 \end{bmatrix} \hat{x}_2(n-1) + \begin{bmatrix} -\hat{v}_1 \\ 0 \end{bmatrix} \hat{x}_1(n-1) + \begin{bmatrix} -1 \\ 0 \end{bmatrix} \hat{\tau}_f(n-1) \tag{4.54}$$

where

$$P_2(n) = \frac{d\hat{y}(n)}{d\hat{v}_2} \; and \; Q_2(n) = \frac{d\hat{x}_2(n)}{d\hat{v}_2}$$

Equation 4.54 can be rewritten as shown in Equation 4.55 and 4.56 where where $R_2(n) = \begin{bmatrix} P_2(n) \\ Q_2(n) \end{bmatrix}$.

$$R_2(n) = AR_2(n-1) + E_2\hat{x}_2(n-1) + E_3\hat{x}_1(n-1) + E_4\hat{\tau}_f(n-1) \tag{4.55}$$

$$P_2(n) = CR_2(n) \tag{4.56}$$

The solution to the Equation 4.56 is shown in Equation 4.57.

$$P_2(n) = C\left(A^n R_2(0) + \sum_{m=0}^{n-1} A^{n-m-1}(E_2\hat{x}_2(m) + E_3\hat{x}_1(m) + E_4\hat{\tau}_f(m))\right) \tag{4.57}$$

Changing the index such that *r=n-m-1* and since $R_2(0) = 0$, the norm of $P_2(n)$ given in Equation 4.57 can be written as shown in Equation 4.58.

$$\|P_2(n)\| \leq \left\| C\sum_{r=0}^{n-1} A^r\,(E_2\hat{x}_2(n-r-1) + E_3\hat{x}_1(n-r-1) + E_4\hat{\tau}_f(n-r-1))\right\| \tag{4.58}$$

Using Rule 1 and 2 of matrix norms given in Appendix D and the fact that $\|C\| = \|E_4\| = 1, \|E_2\| = |\hat{v}_3|, \|E_3\| = |\hat{v}_1|$ Equation 4.58 can be further decomposed as shown in Equation 4.59.

$$\|P_2(n)\| \leq (1 + \|A\| + \cdots + \|A^{n-1}\|).\,(|\hat{v}_3|\|\hat{x}_2\|_{max} + |\hat{v}_1|\|\hat{x}_1\|_{max} + \left\|\hat{\tau}_f\right\|_{max}) \tag{4.59}$$

where

$$\|\hat{x}_1\|_{max} = \max_n\|\hat{x}_1(n)\| \qquad \|\hat{x}_2\|_{max} = \max_n\|\hat{x}_2(n)\| \qquad \left\|\hat{\tau}_f\right\|_{max} = \max_n\left\|\hat{\tau}_f(n)\right\|$$

Equation 4.59 can then be written in the form of a series summation as shown in Equation 4.60.

$$\|P_2(n)\| \leq \left(\sum_{r=0}^{n-1}\|A^r\|\right).\,(|\hat{v}_3|\|\hat{x}_2\|_{max} + |\hat{v}_1|\|\hat{x}_1\|_{max} + \left\|\hat{\tau}_f\right\|_{max}) \tag{4.60}$$

Using Theorem 1 and eigenvalue decomposition, Equation 4.60 can be simplified as shown in Equation 4.61.

$$\|P_2(n)\| \leq \left(\sum_{r=0}^{\infty} \|MD^r M^{-1}\|\right).(|\hat{v}_3|\|\hat{x}_2\|_{max} + |\hat{v}_1|\|\hat{x}_1\|_{max} + \left\|\hat{\tau}_f\right\|_{max}) \tag{4.61}$$

Using Rules 1-4 from Appendix D, Equation 4.61 can be written as Equation 4.62.

$$\|P_2(n)\| \leq \left(\frac{\|M\|.\|M^{-1}\|}{1-\rho(A)}\right).(|\hat{v}_3|\|\hat{x}_2\|_{max} + |\hat{v}_1|\|\hat{x}_1\|_{max} + \left\|\hat{\tau}_f\right\|_{max}) \tag{4.62}$$

Using Equation 4.62, the maximum norm can be computed as shown in Equation 4.63 where $P_{2,max} = \max_n \|P_2(n)\|$,.

$$P_{2,max} = \max_n \left\| \left(\frac{\|M\|.\|M^{-1}\|}{1-\rho(A)}\right)(|\hat{v}_3|\|\hat{x}_2\|_{max} + |\hat{v}_1|\|\hat{x}_1\|_{max} + \left\|\hat{\tau}_f\right\|_{max}) \right\| \tag{4.63}$$

Substituting Equation 4.63 in Equation 4.53, the adaptive learning rate is selected as shown in Equation 4.64.

$$0 < \mu_{\hat{v}_2} < \frac{2}{P_{2,max}^2} \tag{4.64}$$

- $\mu_{\hat{v}_3}$

  The learning rate, $\mu_{\hat{v}_3}$, is selected so as to satisfy Equation 4.65.

$$0 < \mu_{\hat{v}_3} < \frac{2}{\max_n \left\| \frac{\partial \hat{y}(n)}{\partial \hat{v}_3} \right\|^2} \tag{4.65}$$

In order to compute the norm of $\frac{\partial \hat{y}(n)}{\partial \hat{v}_3}$, the solution of the Equation 4.66 must be computed.

$$\begin{bmatrix} P_3(n) \\ Q_3(n) \end{bmatrix} = \begin{bmatrix} 1-\hat{v}_1\hat{v}_2 & \hat{v}_2\hat{v}_3 \\ -\hat{v}_3 v_4 & v_5 \end{bmatrix} \begin{bmatrix} P_3(n-1) \\ Q_3(n-1) \end{bmatrix} + \begin{bmatrix} \hat{v}_3 \\ 0 \end{bmatrix} \hat{x}_2(n-1) + \begin{bmatrix} 0 \\ -v_4 \end{bmatrix} \hat{x}_1(n-1) \tag{4.66}$$

where

$$P_3(n) = \frac{d\hat{y}(n)}{d\hat{v}_3} \; and \; Q_3(n) = \frac{d\hat{x}_2(n)}{d\hat{v}_3}$$

Equation 4.66 can be rewritten as shown in Equation 4.67 and 4.68.

$$R_3(n) = AR_3(n-1) + E_2\hat{x}_2(n-1) + E_5\hat{x}_1(n-1) \tag{4.67}$$

$$P_3(n) = CR_3(n) \tag{4.68}$$

where $R_3(n) = \begin{bmatrix} P_3(n) \\ Q_3(n) \end{bmatrix}$.

The solution to the Equation 4.68 is shown in Equation 4.69.

$$P_3(n) = C\left(A^n R_3(0) + \sum_{m=0}^{n-1} A^{n-m-1}(E_2\hat{x}_2(m) + E_5\hat{x}_1(m))\right) \tag{4.69}$$

Changing the index such that *r=n-m-1* and since $R_3(0) = 0$, the norm of $P_3(n)$ is given in Equation 4.70.

$$\|P_3(n)\| \leq \left\| C\sum_{r=0}^{n-1} A^r\,(E_2\hat{x}_2(n-r-1) + E_5\hat{x}_1(n-r-1)) \right\| \tag{4.70}$$

Using Rule 1 and 2 of matrix norms given in Appendix D and the fact that $\|C\| = 1, \|E_2\| = |\hat{v}_3|, \|E_5\| = |v_4|$ Equation 4.70 can be further decomposed as shown in Equation 4.71.

$$\|P_3(n)\| \leq (1 + \|A\| + \cdots + \|A^{n-1}\|).\,(|\hat{v}_3|\|\hat{x}_2\|_{max} + |v_4|\|\hat{x}_1\|_{max}) \tag{4.71}$$

where

$$\|\hat{x}_1\|_{max} = \max_n \|\hat{x}_1(n)\| \qquad \|\hat{x}_2\|_{max} = \max_n \|\hat{x}_2(n)\|$$

Equation 4.71 can then be written in the form of a series summation as shown in Equation 4.72.

$$\|P_3(n)\| \leq \left(\sum_{r=0}^{n-1} \|A^r\|\right).\,(|\hat{v}_3|\|\hat{x}_2\|_{max} + |v_4|\|\hat{x}_1\|_{max}) \tag{4.72}$$

Using Theorem 1 and eigenvalue decomposition, Equation 4.72 can be rewritten as shown in Equation 4.73.

$$\|P_3(n)\| \leq \left(\sum_{r=0}^{\infty} \|MD^rM^{-1}\|\right).(|\hat{v}_3|\|\hat{x}_2\|_{max} + |v_4|\|\hat{x}_1\|_{max}) \tag{4.73}$$

Using Rules 1-4 from Appendix D, Equation 4.73 can be written as Equation 4.74.

$$\|P_3(n)\| \leq \left(\frac{\|M\|.\|M^{-1}\|}{1-\rho(A)}\right).(|\hat{v}_3|\|\hat{x}_2\|_{max} + |v_4|\|\hat{x}_1\|_{max}) \tag{4.74}$$

Using Equation 4.74, the maximum norm can be computed as shown in Equation 4.75 where $P_{3,max} = \max_n \|P_3(n)\|$,.

$$P_{3,max} = \max_n \left\| \left(\frac{\|M\|.\|M^{-1}\|}{1-\rho(A)}\right)(|\hat{v}_3|\|\hat{x}_2\|_{max} + |v_4|\|\hat{x}_1\|_{max}) \right\| \tag{4.75}$$

Substituting Equation 4.75 in Equation 4.65, the adaptive learning rate is selected as shown in Equation 4.76.

$$0 < \mu_{\hat{v}_3} < \frac{2}{P_{3,max}^2} \tag{4.76}$$

- $\mu_{\hat{w}}$

  The learning rate, $\mu_{\hat{w}}$, is selected so as to satisfy Equation 4.77.

$$0 < \mu_{\hat{w}} < \frac{2}{\max_n \left\|\frac{\partial \hat{y}(n)}{\partial \hat{w}}\right\|^2} \tag{4.77}$$

In order to compute the norm of $\frac{\partial \hat{y}(n)}{\partial \hat{w}}$, first $\left\|\frac{\partial \hat{y}(n)}{\partial \hat{w}_j}\right\|$ needs to be computed by solving Equation 4.78.

$$\begin{bmatrix} P_{\hat{w}_j}(n) \\ Q_{\hat{w}_j}(n) \end{bmatrix} = \begin{bmatrix} 1-\hat{v}_1\hat{v}_2 & \hat{v}_2\hat{v}_3 \\ -\hat{v}_3 v_4 & v_5 \end{bmatrix} \begin{bmatrix} P_{\hat{w}_j}(n-1) \\ Q_{\hat{w}_j}(n-1) \end{bmatrix} + \begin{bmatrix} -\hat{v}_2 \\ 0 \end{bmatrix} \frac{d\hat{\tau}_f(n-1)}{d\hat{w}_j} \tag{4.78}$$

where

$$P_{\hat{w}_j}(n) = \frac{\partial \hat{x}_1(n)}{\partial \hat{w}_j} = \frac{\partial \hat{y}(n)}{\partial \hat{w}_j} \; and \; Q_{\hat{w}_j}(n) = \frac{\partial \hat{x}_2(n)}{\partial \hat{w}_j}$$

Equation 4.78 can be rewritten as shown in Equation 4.79 and 4.80

$$R_{\hat{w}_j}(n) = AR_{\hat{w}_j}(n-1) + HT_{\hat{w}_j}(n-1) \tag{4.79}$$

$$P_{\hat{w}_j}(n) = CR_{\hat{w}_j}(n) \tag{4.80}$$

where $R_{\hat{w}_j}(n) = \begin{bmatrix} P_{\hat{w}_j}(n) \\ Q_{\hat{w}_j}(n) \end{bmatrix}$.

The solution to the Equation 4.80 is shown in Equation 4.81.

$$P_{\hat{w}_j}(n) = C\left(A^n R_{\hat{w}_j}(0) + \sum_{m=0}^{n-1} A^{n-m-1} HT_{\hat{w}_j}(m)\right) \tag{4.81}$$

Changing the index such that *r=n-m-1* and since $R_{\hat{w}_j}(0) = 0$, the norm of $P_{\hat{w}_j}(n)$ is written as shown in Equation 4.82.

$$\left\| P_{\hat{w}_j}(n) \right\| = \left\| \sum_{r=0}^{n-1} CA^r HT_{\hat{w}_j}(n-r-1) \right\| \tag{4.82}$$

Using Rule 1 and 2 of matrix norms given in Appendix D and the fact that $\|C\| = 1$ and $\|H\| = |\hat{v}_2|$, Equation 4.82 can be further decomposed as shown in Equation 4.83.

$$\left\| P_{\hat{w}_j}(n) \right\| \leq |\hat{v}_2|.\left(\left\| T_{\hat{w}_j}(n-1) \right\| + \|A\|.\left\| T_{\hat{w}_j}(n-2) \right\| + \cdots + \|A^{n-1}\|.\left\| T_{\hat{w}_j}(0) \right\| \right) \tag{4.83}$$

This can be rewritten as shown in Equation 4.84.

$$\left\| P_{\hat{w}_j}(n) \right\| \leq |\hat{v}_2|(1 + \|A\| + \cdots + \|A^{n-1}\|).\left\| T_{\hat{w}_j} \right\|_{max} \tag{4.84}$$

where

$$\left\| T_{\hat{w}_j} \right\|_{max} = \left\| \Phi_j \right\|_{max} = \max_n \left\| \Phi_j \right\| = 1$$

The derivation procedure is continued as before to compute the norm as shown in Equation 4.85.

$$\left\| P_{\hat{w}_j}(n) \right\| \leq |\hat{v}_2| . \left(\frac{\|M\| . \|M^{-1}\|}{1 - \rho(A)} \right) \tag{4.85}$$

The maximum norm, at any time n, of the elements of the vector is given in Equation 4.86 where $P_{\hat{w}_{max}}(n) = \max_j \left\| P_{\hat{w}_j}(n) \right\|$.

$$P_{\hat{w}_{max}}(n) = \max_j \left\| |\hat{v}_2| . \left(\frac{\|M\| . \|M^{-1}\|}{1 - \rho(A)} \right) \right\| \tag{4.86}$$

The maximum norm of the vector is therefore given by Equation 4.87.

.

$$\max_n \left\| \frac{\partial \hat{y}(n)}{\partial \hat{w}} \right\| = \sqrt{N_w}\, P_{\hat{w}_{max}} \tag{4.87}$$

where

$$P_{\hat{w}_{max}} = \max_n \left(P_{\hat{w}_{max}}(n) \right)$$

Substituting Equation 4.87 in Equation 4.77 the adaptive learning rate is selected as shown in Equation 4.88.

$$0 < \mu_{\hat{w}} < \frac{2}{N_w P_{\hat{w},max}^2} \tag{4.88}$$

- $\mu_{\hat{m}}$

  The learning rate, $\mu_{\hat{m}}$, is selected so as to satisfy Equation 4.89.

$$0 < \mu_{\hat{m}} < \frac{2}{\max_n \left\| \frac{\partial \hat{y}(n)}{\partial \hat{m}} \right\|^2} \tag{4.89}$$

In order to compute the norm of $\frac{\partial \hat{y}(n)}{\partial \hat{m}}$, first $\left\|\frac{\partial \hat{y}(n)}{\partial \hat{m}_j}\right\|$needs to be computed by solving Equation 4.90.

$$\begin{bmatrix} P_{\hat{m}_j}(n) \\ Q_{\hat{m}_j}(n) \end{bmatrix} = \begin{bmatrix} 1-\hat{v}_1\hat{v}_2 & \hat{v}_2\hat{v}_3 \\ -\hat{v}_3 v_4 & v_5 \end{bmatrix} \begin{bmatrix} P_{\hat{m}_j}(n-1) \\ Q_{\hat{m}_j}(n-1) \end{bmatrix} + \begin{bmatrix} -\hat{v}_2 \\ 0 \end{bmatrix} \frac{d\hat{\tau}_f(n-1)}{d\hat{m}_j} \tag{4.90}$$

where

$$P_{\hat{m}_j}(n) = \frac{\partial \hat{x}_1(n)}{\partial \hat{m}_j} \; and \; Q_{\hat{m}_j}(n) = \frac{\partial \hat{x}_2(n)}{\partial \hat{m}_j}$$

Equation 4.90 can be rewritten as shown in Equation 4.91 and 4.92 where

where $R_{\hat{m}_j}(n) = \begin{bmatrix} P_{\hat{m}_j}(n) \\ Q_{\hat{m}_j}(n) \end{bmatrix}$.

$$R_{\hat{m}_j}(n) = AR_{\hat{m}_j}(n-1) + HT_{\hat{m}_j}(n-1) \tag{4.91}$$

$$P_{\hat{m}_j}(n) = CR_{\hat{m}_j}(n) \tag{4.92}$$

where

$$T_{\hat{m}_j}(n-1) = \frac{d\hat{\tau}_f(n-1)}{d\hat{m}_j} = \hat{w}_j . \frac{\partial \Phi_j(n-1)}{d\hat{m}_j} = \hat{w}_j . V_{1,j}(n-1)$$

The solution to the Equation 4.92 is shown in Equation 4.93.

$$P_{\hat{m}_j}(n) = C\left(A^n R_{\hat{m}_j}(0) + \sum_{m=0}^{n-1} A^{n-m-1} H T_{\hat{m}_j}(m)\right) \tag{4.93}$$

Changing the index such that *r=n-m-1* and since $R_{\hat{m}_j}(0) = 0$, the norm of $P_{\hat{m}_j}(n)$ is as shown in Equation 4.94.

$$\left\|P_{\hat{m}_j}(n)\right\| = \left\|\sum_{r=0}^{n-1} CA^r H T_{\hat{m}_j}(n-r-1)\right\| \tag{4.94}$$

Equation 4.94 can be further decomposed as shown in Equation 4.95.

$$\left\|P_{\hat{m}_j}(n)\right\| \leq |\hat{v}_2| . \left(\left\|T_{\hat{m}_j}(n-1)\right\| + \|A\| . \left\|T_{\hat{m}_j}(n-2)\right\| + \cdots + \|A^{n-1}\| . \left\|T_{\hat{m}_j}(0)\right\|\right) \tag{4.95}$$

This can be rewritten as shown in Equation 4.96.

$$\left\| P_{\hat{m}_j}(n) \right\| \leq |\hat{v}_2|.(1 + \|A\| + \cdots + \|A^{n-1}\|).\left\| T_{\hat{m}_j} \right\|_{max} \tag{4.96}$$

where

$$\left\| T_{\hat{m}_j} \right\|_{max} = \max_n \left\| \hat{w}_j V_{1,j} \right\|$$

The derivation procedure is continued as before to compute the norm as shown in Equation 4.97.

$$\left\| P_{\hat{m}_j}(n) \right\| \leq |\hat{v}_2|.\left(\frac{\|M\|.\|M^{-1}\|}{1 - \rho(A)} \right).\left\| T_{\hat{m}_j} \right\|_{max} \tag{4.97}$$

The maximum norm, at any time n, of the elements of the vector is given in Equation 4.98 where $P_{\hat{m}_{max}}(n) = \max_j \left\| P_{\hat{m}_j}(n) \right\|$.

$$P_{\hat{m}_{max}}(n) = \max_j \left\| |\hat{v}_2|.\left(\frac{\|M\|.\|M^{-1}\|}{1 - \rho(A)} \right).\left\| T_{\hat{m}_j} \right\|_{max} \right\| \tag{4.98}$$

The maximum norm of the vector is therefore given by Equation 4.99.

$$\max_n \left\| \frac{\partial \hat{y}(n)}{\partial \hat{m}} \right\| = \sqrt{N_w}\, P_{\hat{m}_{max}} \tag{4.99}$$

where

$$P_{\hat{m}_{max}} = \max_n \left(P_{\hat{m}_{max}}(n) \right)$$

Substituting Equation 4.99 in Equation 4.89, the adaptive learning rate is selected as shown in Equation 4.100.

$$0 < \mu_{\hat{m}} < \frac{2}{N_w P_{\hat{m},max}^2} \tag{4.100}$$

- $\mu_{\hat{d}}$

The learning rate, $\mu_{\hat{d}}$, is selected so as to satisfy Equation 4.101.

$$0 < \mu_{\hat{d}} < \frac{2}{\max_n \left\| \frac{\partial \hat{y}(n)}{\partial \hat{d}} \right\|^2} \tag{4.101}$$

In order to compute the norm of $\frac{\partial \hat{y}(n)}{\partial \hat{d}}$, first $\left\| \frac{\partial \hat{y}(n)}{\partial \hat{d}_j} \right\|$ needs to be computed by solving Equation 4.102.

$$\begin{bmatrix} P_{\hat{d}_j}(n) \\ Q_{\hat{d}_j}(n) \end{bmatrix} = \begin{bmatrix} 1 - \hat{v}_1\hat{v}_2 & \hat{v}_2\hat{v}_3 \\ -\hat{v}_3 v_4 & v_5 \end{bmatrix} \begin{bmatrix} P_{\hat{d}_j}(n-1) \\ Q_{\hat{d}_j}(n-1) \end{bmatrix} + \begin{bmatrix} -\hat{v}_2 \\ 0 \end{bmatrix} \frac{d\hat{\tau}_f(n-1)}{d\hat{d}_j} \tag{4.102}$$

where

$$P_{\hat{d}_j}(n) = \frac{\partial \hat{x}_1(n)}{\partial \hat{d}_j} \; and \; Q_{\hat{d}_j}(n) = \frac{\partial \hat{x}_2(n)}{\partial \hat{d}_j}$$

Equation 4.102 can be rewritten as shown in Equation 4.103 and 4.104

$$R_{\hat{d}_j}(n) = AR_{\hat{d}_j}(n-1) + HT_{\hat{d}_j}(n-1) \tag{4.103}$$

$$P_{\hat{d}_j}(n) = CR_{\hat{d}_j}(n) \tag{4.104}$$

where

$$R_{\hat{d}_j}(n) = \begin{bmatrix} P_{\hat{d}_j}(n) \\ Q_{\hat{d}_j}(n) \end{bmatrix} \quad \text{and} \quad T_{\hat{d}_j}(n-1) = \frac{d\hat{\tau}_f(n-1)}{d\hat{d}_j} = \hat{w}_j . \frac{\partial \Phi_j(n-1)}{d\hat{d}_j} = \hat{w}_j . V_{2,j}(n-1)$$

The solution to the Equation 4.104 is shown in Equation 4.105.

$$P_{\hat{d}_j}(n) = C\left(A^n R_{\hat{d}_j}(0) + \sum_{m=0}^{n-1} A^{n-m-1} H T_{\hat{d}_j}(m) \right) \tag{4.105}$$

Changing the index such that *r=n-m-1* and since $R_{\hat{d}_j}(0) = 0$, the norm of $P_{\hat{d}_j}(n)$ is written as shown in Equation 4.106.

$$\left\| P_{\hat{d}_j}(n) \right\| = \left\| \sum_{r=0}^{n-1} CA^r H T_{\hat{d}_j}(n-r-1) \right\| \tag{4.106}$$

Equation 4.106 can be further decomposed as shown in Equation 4.107.

$$\left\|P_{\hat{a}_j}(n)\right\| \leq |\hat{v}_2|.\left(\left\|T_{\hat{a}_j}(n-1)\right\| + \|A\|.\left\|T_{\hat{a}_j}(n-2)\right\| + \cdots + \|A^{n-1}\|.\left\|T_{\hat{a}_j}(0)\right\|\right) \tag{4.107}$$

This can be rewritten as shown in Equation 4.108.

$$\left\|P_{\hat{a}_j}(n)\right\| \leq |\hat{v}_2|.(1 + \|A\| + \cdots + \|A^{n-1}\|).\left\|T_{\hat{a}_j}\right\|_{max} \tag{4.108}$$

where

$$\left\|T_{\hat{a}_j}\right\|_{max} = \max_n \|\hat{w}_j V_{2,j}\|$$

The derivation procedure is continued as before to compute the norm as shown in Equation 4.109.

$$\left\|P_{\hat{a}_j}(n)\right\| \leq |\hat{v}_2|.\left(\frac{\|M\|.\|M^{-1}\|}{1-\rho(A)}\right).\left\|T_{\hat{a}_j}\right\|_{max} \tag{4.109}$$

The maximum norm, at any time n, of the elements of the vector is given in Equation 4.110 where $P_{\hat{a}_{max}}(n) = \max_j \left\|P_{\hat{a}_j}(n)\right\|$.

$$P_{\hat{a}_{max}}(n) = \max_j \left\| |\hat{v}_2|.\left(\frac{\|M\|.\|M^{-1}\|}{1-\rho(A)}\right).\left\|T_{\hat{a}_j}\right\|_{max} \right\| \tag{4.110}$$

The maximum norm of the vector is therefore given by Equation 4.111.

$$\max_n \left\|\frac{\partial \hat{y}(n)}{\partial \hat{d}}\right\| = \sqrt{N_w}\, P_{\hat{a}_{max}} \tag{4.111}$$

where

$$P_{\hat{a}_{max}} = \max_n\left(P_{\hat{a}_{max}}(n)\right)$$

Substituting Equation 4.111 in Equation 4.101, the adaptive learning rate is selected as shown in Equation 4.112.

$$0 < \mu_{\hat{a}} < \frac{2}{N_w P_{d,max}^2} \tag{4.112}$$

- $\mu_{\hat{c}}$

The learning rate, $\mu_{\hat{c}}$, is selected so as to satisfy Equation 4.113.

$$0 < \mu_{\hat{c}} < \frac{2}{\max_n \left\| \frac{\partial \hat{y}(n)}{\partial \hat{c}} \right\|^2} \tag{4.113}$$

In order to compute the norm of $\frac{\partial \hat{y}(n)}{\partial \hat{c}}$, first $\left\| \frac{\partial \hat{y}(n)}{\partial c_j} \right\|$ needs to be computed by solving Equation 4.114.

$$\begin{bmatrix} P_{\hat{c}_j}(n) \\ Q_{\hat{c}_j}(n) \end{bmatrix} = \begin{bmatrix} 1 - \hat{v}_1\hat{v}_2 & \hat{v}_2\hat{v}_3 \\ -\hat{v}_3 v_4 & v_5 \end{bmatrix} \begin{bmatrix} P_{\hat{c}_j}(n-1) \\ Q_{\hat{c}_j}(n-1) \end{bmatrix} + \begin{bmatrix} -\hat{v}_2 \\ 0 \end{bmatrix} \frac{d\widehat{\tau_f}(n-1)}{d\hat{c}_j} \tag{4.114}$$

where

$$P_{\hat{c}_j}(n) = \frac{\partial \hat{x}_1(n)}{\partial \hat{c}_j} \; and \; Q_{\hat{c}_j}(n) = \frac{\partial \hat{x}_2(n)}{\partial \hat{c}_j}$$

Equation 4.114 can be rewritten as shown in Equation 4.115 and 4.116.

$$R_{\hat{c}_j}(n) = AR_{\hat{c}_j}(n-1) + HT_{\hat{c}_j}(n-1) \tag{4.115}$$

$$P_{\hat{c}_j}(n) = CR_{\hat{c}_j}(n) \tag{4.116}$$

where

$$R_{\hat{c}_j}(n) = \begin{bmatrix} P_{\hat{c}_j}(n) \\ Q_{\hat{c}_j}(n) \end{bmatrix} \quad \text{and} \quad T_{\hat{c}_j}(n-1) = \frac{d\widehat{\tau_f}(n-1)}{d\hat{c}_j} = \widehat{w}_j . \frac{\partial \Phi_j(n-1)}{d\hat{c}_j} = \widehat{w}_j . V_{3,j}(n-1)$$

The solution to the Equation 4.116 is shown in Equation 4.117.

$$P_{\hat{c}_j}(n) = C\left(A^n R_{\hat{c}_j}(0) + \sum_{m=0}^{n-1} A^{n-m-1} H T_{\hat{c}_j}(m) \right) \tag{4.117}$$

Changing the index such that *r=n-m-1* and since $R_{\hat{c}_j}(0) = 0$, the norm of $P_{\hat{c}_j}(n)$ is written as shown in Equation 4.118.

$$\left\| P_{\hat{c}_j}(n) \right\| = \left\| \sum_{r=0}^{n-1} C A^r H T_{\hat{c}_j}(n-r-1) \right\| \tag{4.118}$$

Equation 4.118 can be further decomposed as shown in Equation 4.119.

$$\left\|P_{\hat{c}_j}(n)\right\| \leq |\hat{v}_2|.\left(\left\|T_{\hat{c}_j}(n-1)\right\| + \|A\|.\left\|T_{\hat{c}_j}(n-2)\right\| + \cdots + \|A^{n-1}\|.\left\|T_{\hat{c}_j}(0)\right\|\right) \tag{4.119}$$

This can be rewritten as shown in Equation 4.120.

$$\left\|P_{\hat{c}_j}(n)\right\| \leq |\hat{v}_2|.(1 + \|A\| + \cdots + \|A^{n-1}\|).\left\|T_{\hat{c}_j}\right\|_{max} \tag{4.120}$$

where

$$\left\|T_{\hat{c}_j}\right\|_{max} = \max_n \left\|\hat{w}_j V_{3,j}\right\|$$

The derivation procedure is continued as before to compute the norm as shown in Equation 4.121.

$$\left\|P_{\hat{c}_j}(n)\right\| \leq |\hat{v}_2|.\left(\frac{\|M\|.\|M^{-1}\|}{1-\rho(A)}\right).\left\|T_{\hat{c}_j}\right\|_{max} \tag{4.121}$$

The maximum norm, at any time n, of the elements of the vector is given in Equation 4.122 where $P_{\hat{c}_{max}}(n) = \max_j \left\|P_{\hat{c}_j}(n)\right\|$.

$$P_{\hat{c}_{max}}(n) = \max_j \left\|\,|\hat{v}_2|.\left(\frac{\|M\|.\|M^{-1}\|}{1-\rho(A)}\right).\left\|T_{\hat{c}_j}\right\|_{max}\right\| \tag{4.122}$$

The maximum norm of the vector is therefore given by Equation 4.123.

$$\max_n \left\|\frac{\partial \hat{y}(n)}{\partial \hat{c}}\right\| = \sqrt{N_w}\, P_{\hat{c}_{max}} \tag{4.123}$$

where

$$P_{\hat{c}_{max}} = \max_n\left(P_{\hat{c}_{max}}(n)\right)$$

Substituting Equation 4.123 in Equation 4.113, the adaptive learning rate is selected as shown in Equation 4.124.

$$0 < \mu_{\hat{c}} < \frac{2}{N_w P_{\hat{c}_{max}}^2} \tag{4.124}$$

Chapter 5: Case Studies and Simulation Analysis

In this chapter, simulations for different case studies are carried out to assess the learning method and network structure developed in Chapter 4. In the first case study, the linear parameters are assumed known and the system is used to learn the nonlinear friction only in order to determine the optimum number of wavelets and mother wavelet for training. In the second case study, the ability of the network to learn the linear parameters of a frictionless DC motor is studied. The third case study determines the effectiveness of the network in learning the linear network parameters for a motor with viscous friction. The final case study demonstrates the ability of the network to simultaneously learn both linear and nonlinear system parameters for a DC motor with friction. The motor parameters are provided in Appendix C and the sampling rate T is selected as 0.1ms.

5.1 Nonlinear Friction Identification of DC Motor

In this section, a recurrent wavelet network (RWN) is used to learn the DC motor friction assuming all the linear parameters of the network are known. The network learning structure is shown in Figure 73 and the friction function is given in Equation 5.1 where β is 10.

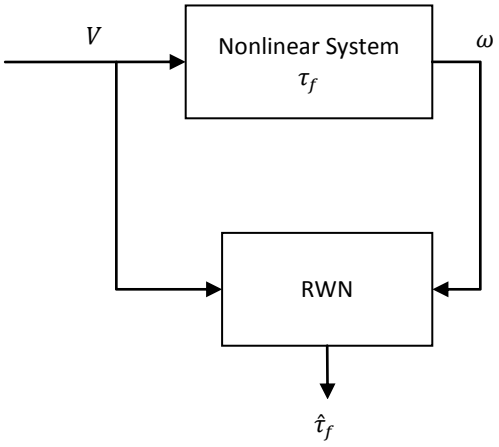


Figure 73: Nonlinear Friction Identification

$$\tau_f(\omega) = \tanh(\beta\omega) \times \left(T_c + (T_s - T_c)e^{-\alpha|\omega|^2}\right) + B\omega \tag{5.1}$$

Training was carried out by applying a 1V sine wave of frequency 1Hz to the DC Motor to generate the training signal shown in Figure 74.

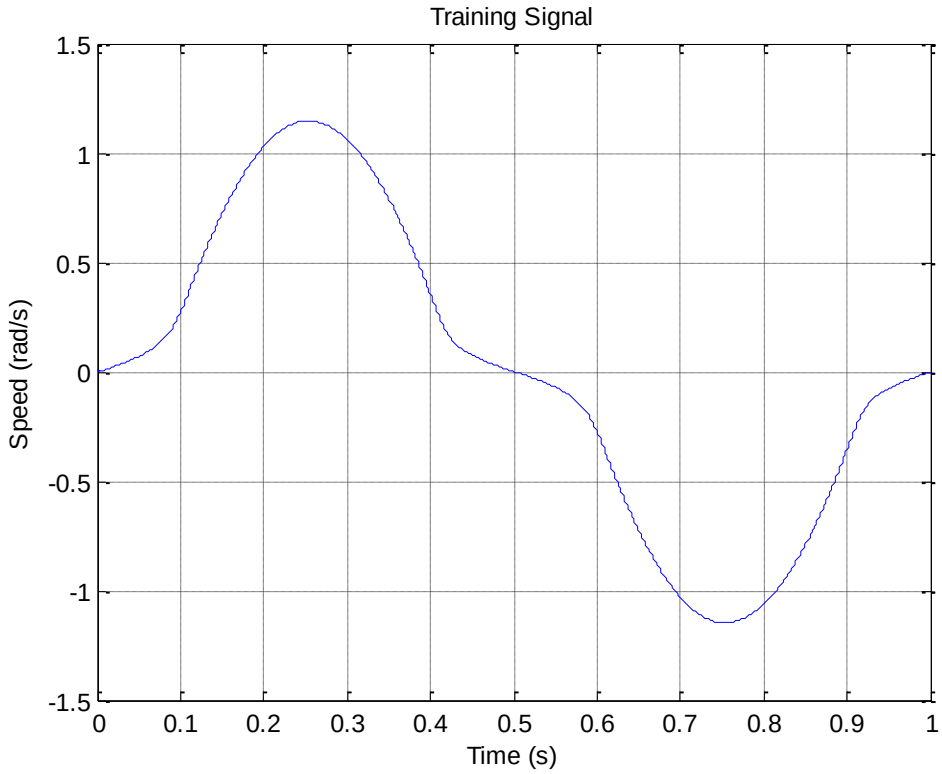


Figure 74: Training Signal for Nonlinear Frictional Function Identification

Training was carried out for 100000 iterations using the first derivative of the Gaussian as the mother wavelet, for varying number of neurons in the hidden layer in order to determine the optimum size for the RWN. The MSE after training is given in Table 13 and shown in Figure 75.

Table 13: MSE after Training for Varying Nw for Nonlinear Frictional Function Approximation

Nw	**MSE**
7	8.3151e-5
15	3.3040e-5
31	1.4045e-5

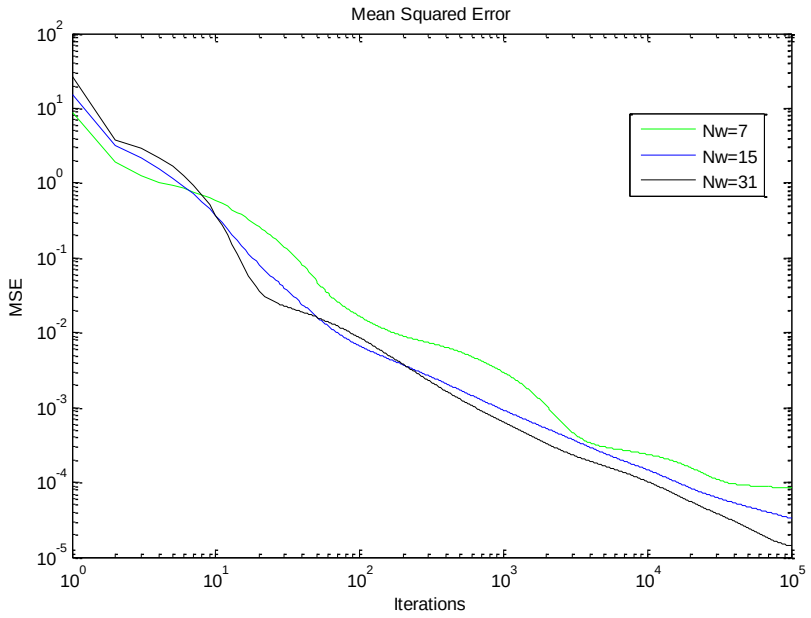


Figure 75: MSE for Varying Nw for Nonlinear Frictional Function Approximation

The adaptive learning rates for the network with 15 wavelets in the hidden layer are shown in Figure 76. It should be noted that $\mu_c = \mu_m$.

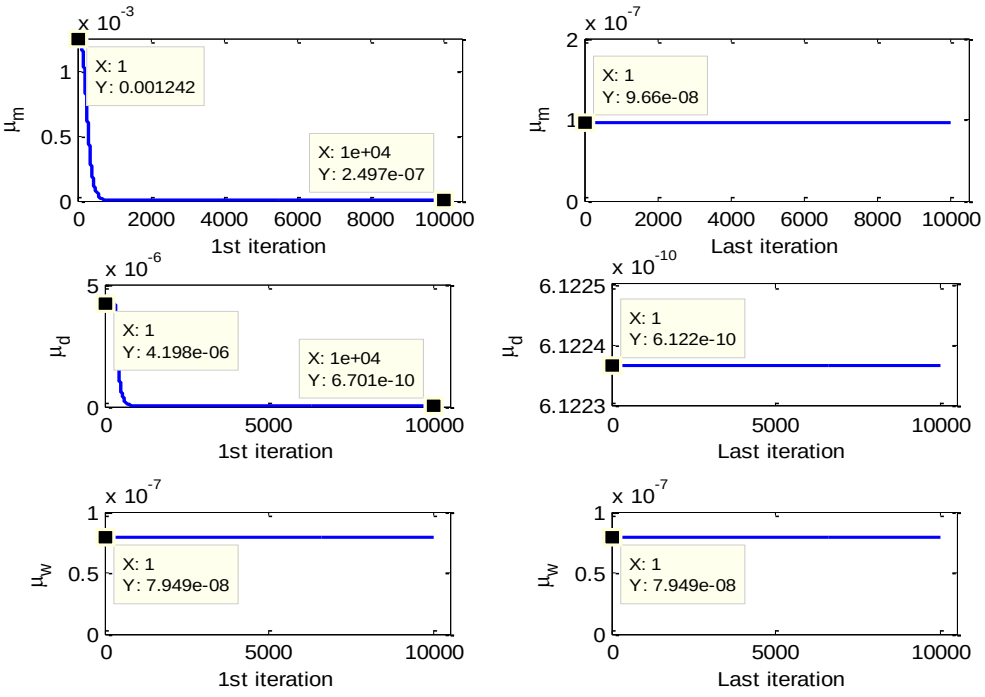


Figure 76: ALRs for Nonlinear Function Approximation over First and Last Training Cycle, Nw=15

Testing was then carried out where the testing signal shown in Figure 77 was generated by applying a 0.75V chirp signal of frequency ranging from 0.1-4Hz to the DC Motor.

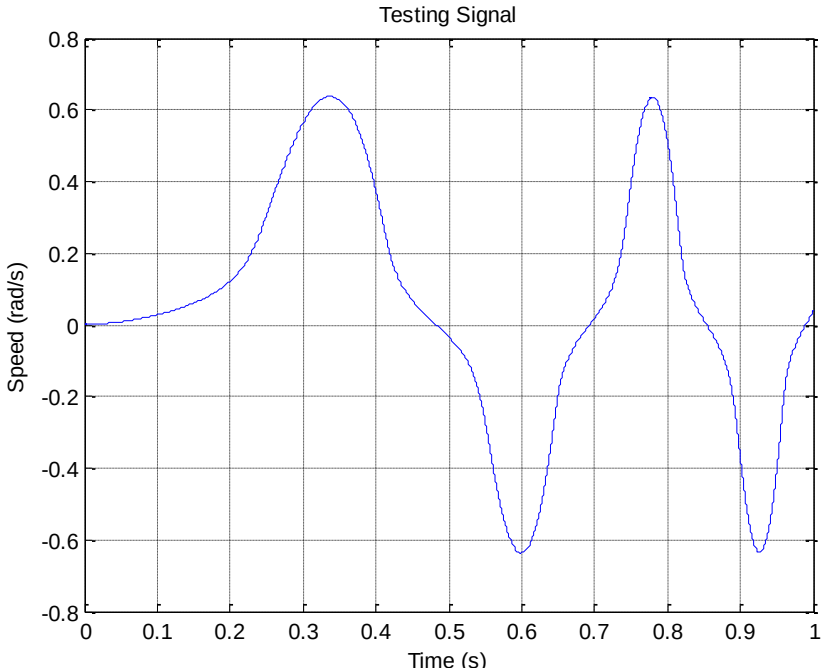


Figure 77: Testing Signal for Nonlinear Frictional Function Approximation

The MSE after testing is shown in Table 14 and the torque speed characteristics of the trained network after testing are shown in Figure 78.

Table 14: MSE after Testing for Varying Nw for Nonlinear Frictional Function Approximation

Nw	**MSE**
7	1.5425e-4
15	5.4618e-5
31	4.2923e-5

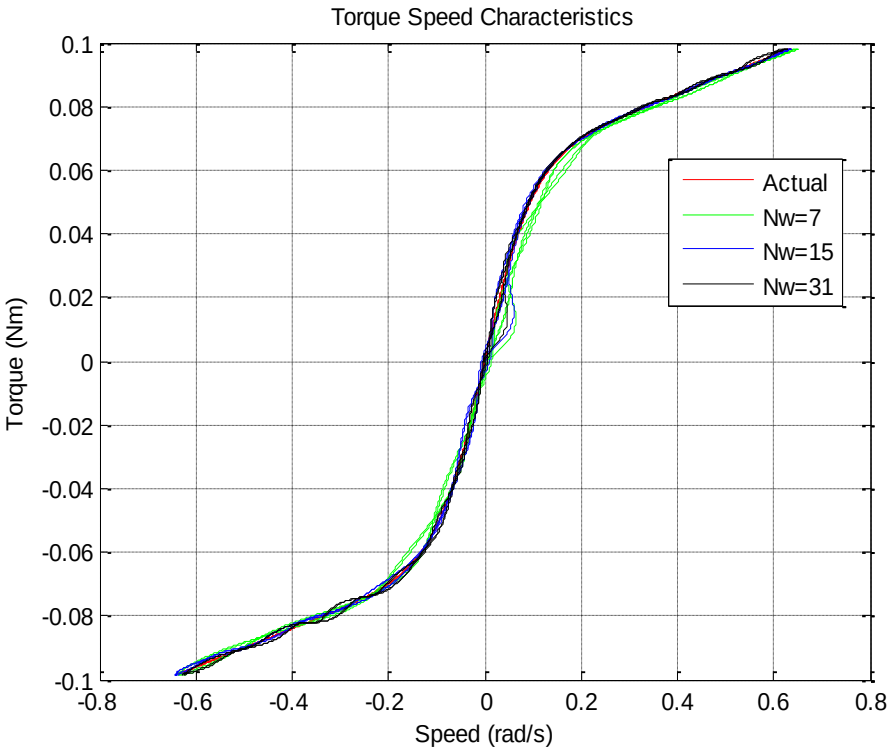


Figure 78: Torque Speed Characteristics of DC Motor

From the results, it can be seen that increasing the number of neurons in the hidden layer improves the accuracy of the RWN, however this is done at the expense of training time with larger networks taking significantly longer to train.

The network was then trained using different mother wavelets in order to optimize the network structure. The number of neurons in the hidden layer was set to 15 and training was carried out for 100000 iterations with the training signal shown in Figure 79.

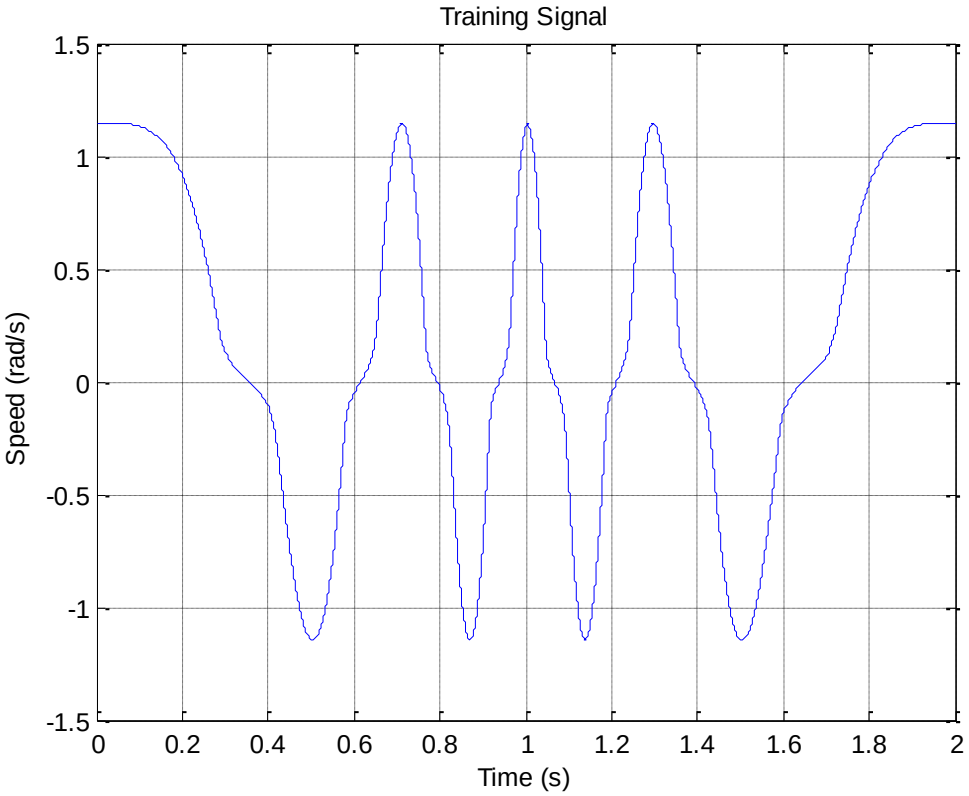


Figure 79: Training Signal for Nonlinear Frictional Function Identification

Training was carried out for three different activation functions and the MSE after training is shown in Table 15 and in Figure 80.

Table 15:MSE after Training for Different Activation Functions for Nonlinear Function Approximation

Activation Function	**MSE**
1st Derivative of Gaussian	3.1385e-5
Mexican Hat	4.8973e-4
Morlet	21.0638

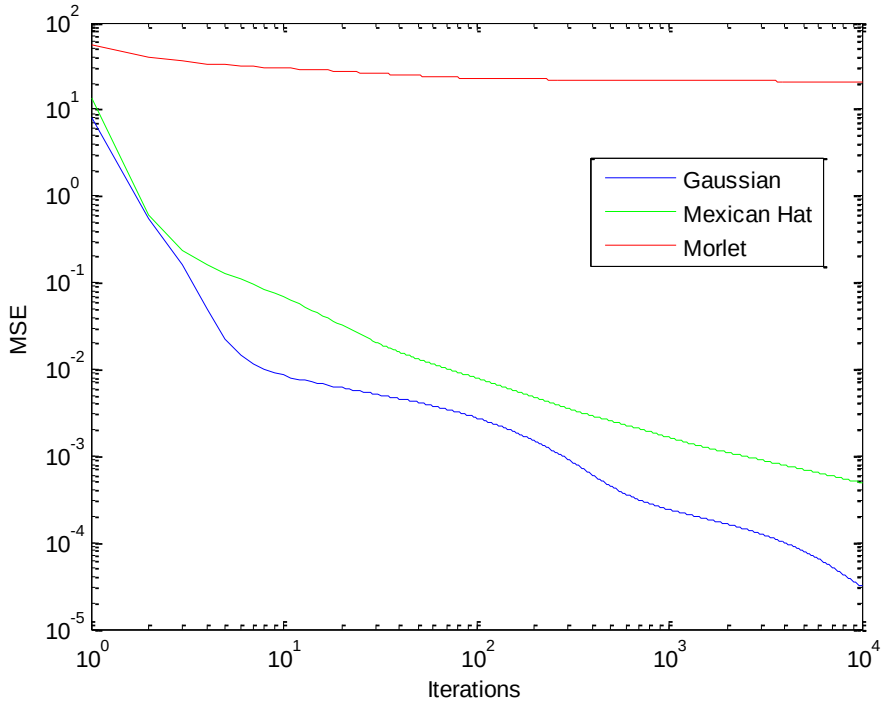


Figure 80: MSE for Different Activation Functions for Nonlinear Frictional Function Approximation

Testing was carried out using a sine wave as shown in Figure 81.

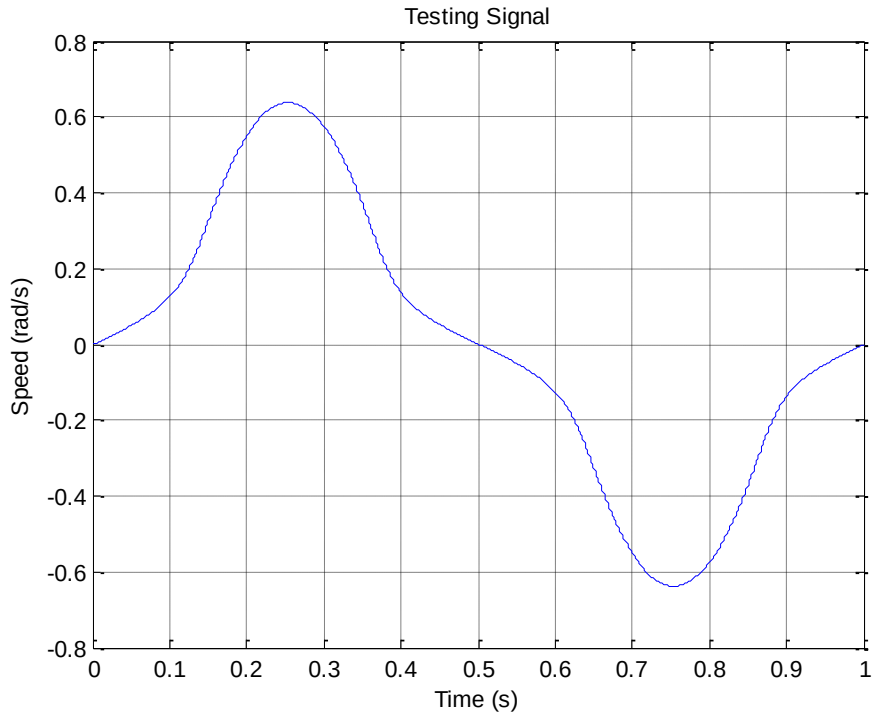


Figure 81: Testing Signal for Nonlinear Frictional Function Identification

The MSE after testing is given in Table 16 and the torque speed characteristics are shown in Figure 82 from where is can be seen that the first derivative of the Gaussian provides a better approximation for the nonlinear friction.

Table 16: MSE after Testing for Different Activation Functions for Nonlinear Frictional Function Approximation

Activation Function	**MSE**
1st Derivative of Gaussian	2.5062e-4
Mexican Hat	0.0027

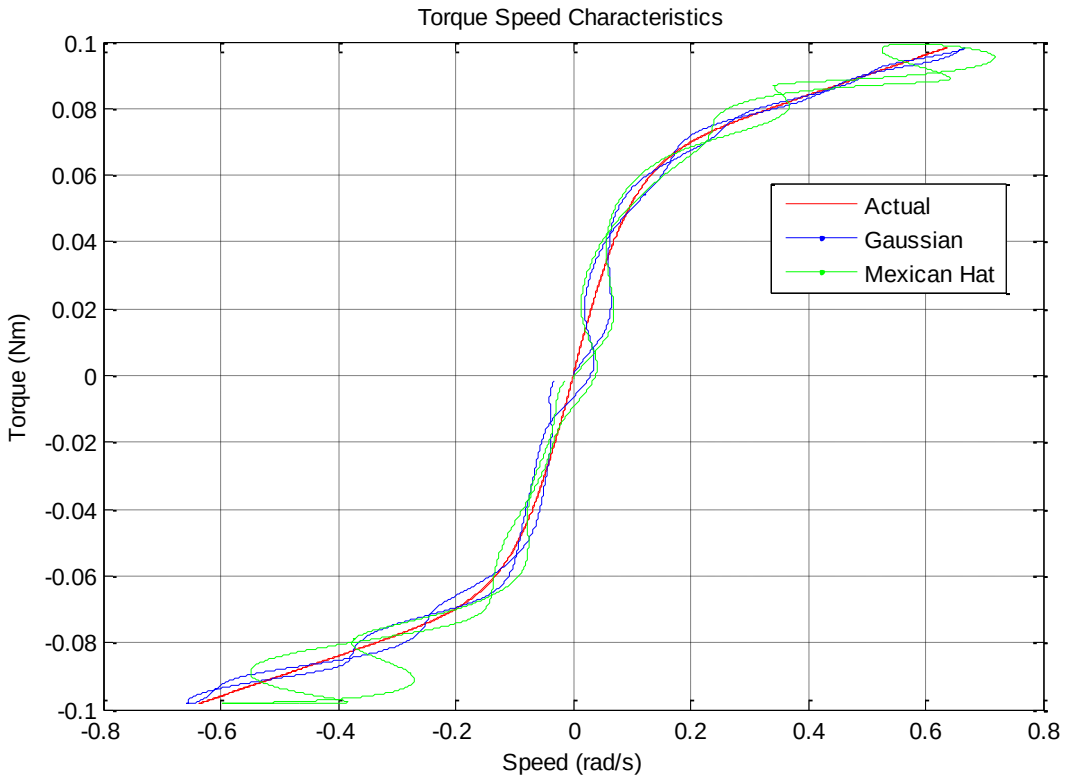


Figure 82: Torque Speed Characteristics

5.2 Linear Parameter Identification of Frictionless DC Motor

In this section, the linear mechanical parameters of a frictionless DC motor are learned, assuming the motor resistance and inductance are known. The network learning structure for parameter identification is shown in Figure 83. The training signal used, shown in Figure 84, is a chirp signal of magnitude 1V and frequency range from 0 to 4 Hz.

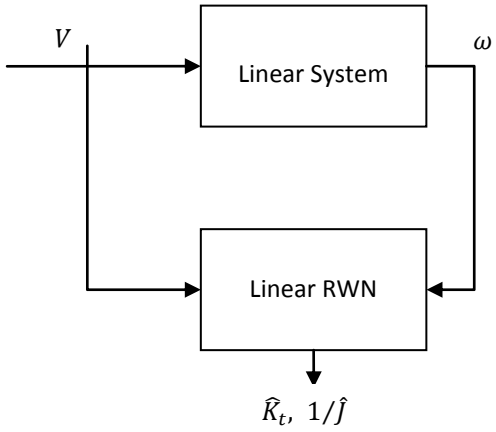


Figure 83: Linear Parameter Identification for Frictionless DC Motor

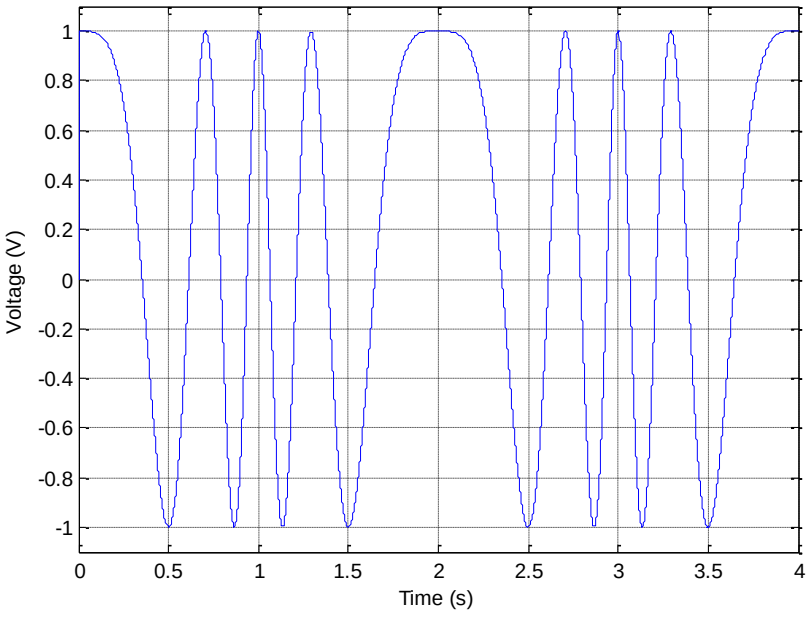


Figure 84: Training Signal for DC Motor Parameter Training

Online training is carried out for 50 iterations assuming initial conditions within 20% and within 50% of the actual values. Figure 85 shows the mean squared error during training. Figure 86 shows the adaptive learning rates for the parameters during the 1st training cycle, after which the ALRs remained constant. Figure 87 shows the convergence of the network parameters during the first ten and last ten cycles of training.

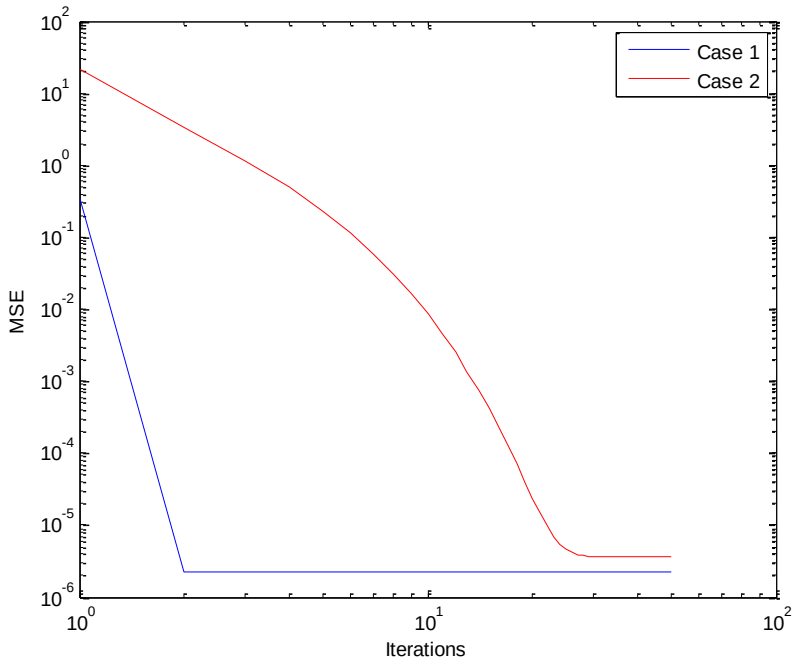


Figure 85: MSE for Training Linear Parameters of Frictionless DC Motor

Table 17 shows the results after training and it can be seen that the network is able to accurately learn the motor inertia as well as the motor torque constant.

Table 17: Results of Training Linear Parameters of Frictionless DC Motor

Case 1	**Actual Value**	**Initial Value**	**Final Value**	**Error (%)**
J	1.8e-4	1.44e-4	1.8e-4	0
K_t	0.0550	0.0440	0.0550	0
Case 2	**Actual Value**	**Initial Value**	**Final Value**	**Error (%)**
J	1.8e-4	9e-5	1.8e-4	0
K_t	0.0550	0.0275	0.0550	0

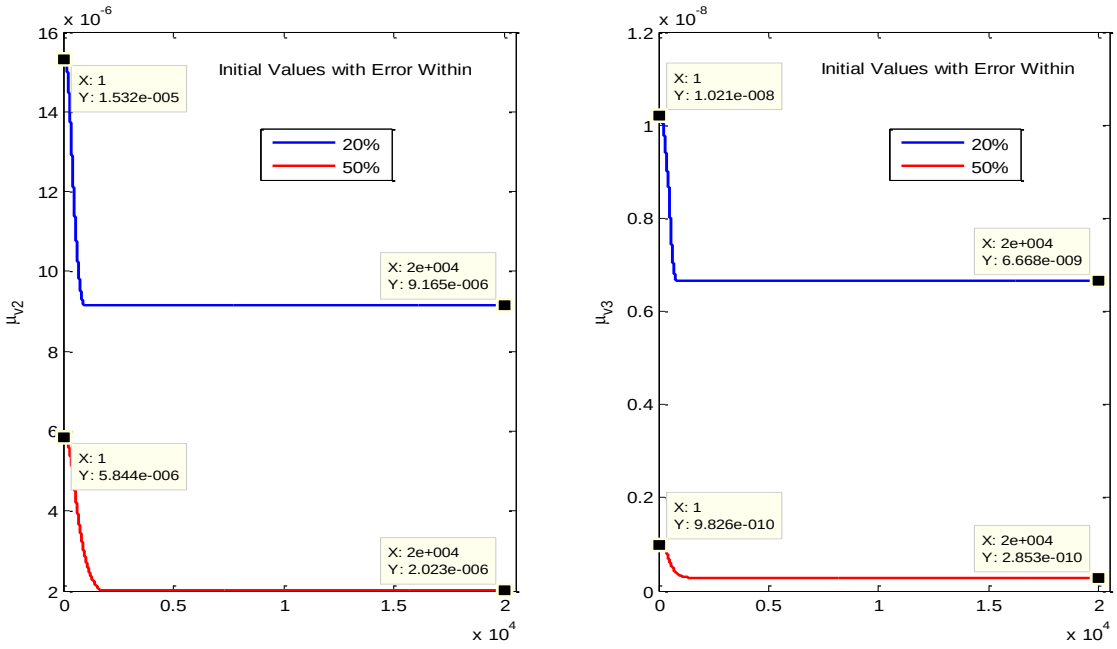


Figure 86: ALRs for Linear Parameter Identification of Frictionless DC Motor

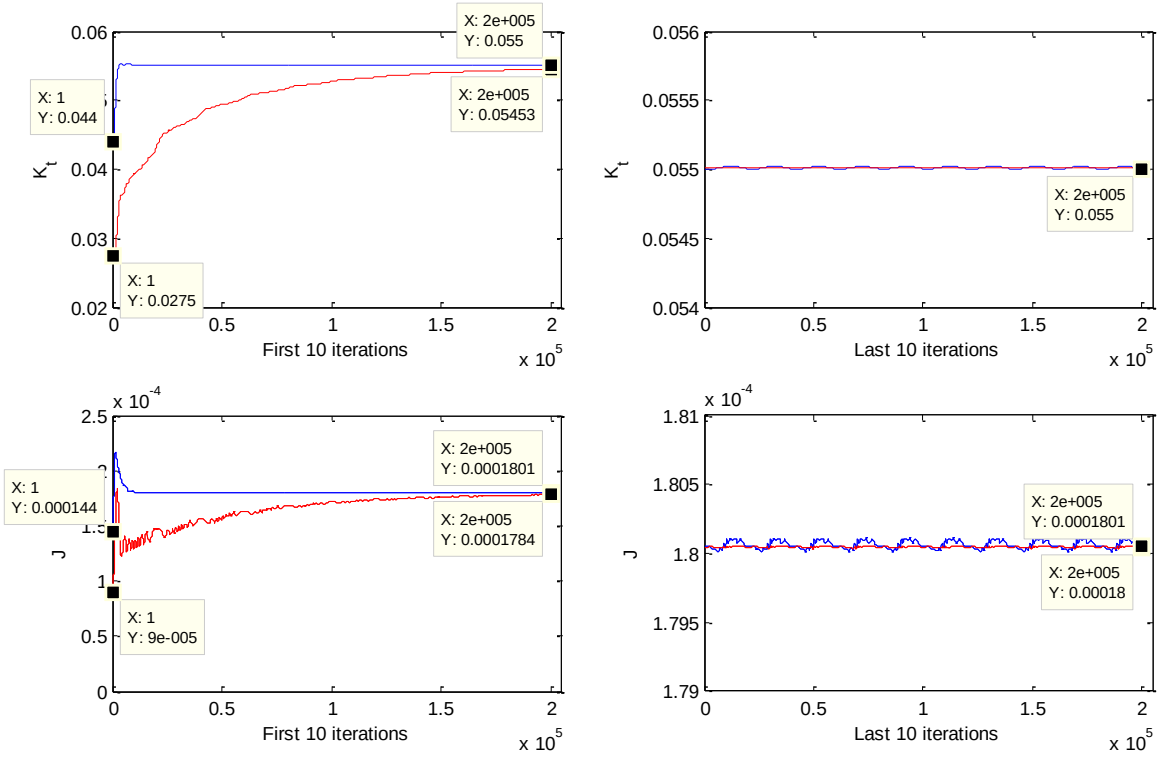


Figure 87: Convergence of Kt and J for Parameter Identification of Frictionless DC Motor

5.3 Linear Parameter Identification of DC Motor with Viscous Friction

In this case, the network is used to learn the motor inertia and viscous friction of a DC Motor assuming the torque constant, resistance and inductance are known a priori and that there is no nonlinear friction. The network learning structure is shown in Figure 88.

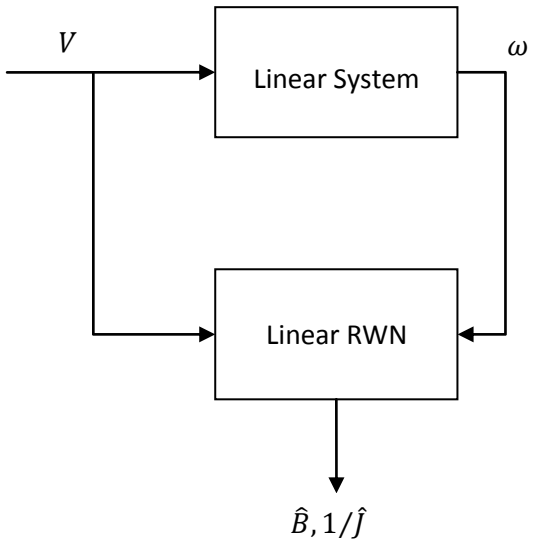


Figure 88: Network Learning Structure for DC Motor with Viscous Friction

Training was carried out for 50 iterations and Figure 89 shows the MSE during training. Table 18 shows the results of the simulations. The adaptive learning rates over the first and last iteration for three different cases in which the initial conditions were assumed within 10%, 20% and upto 40% of the actual values are shown in Figure 90. The convergence of the network parameters over the first and last ten cycles of training is shown in Figure 91.

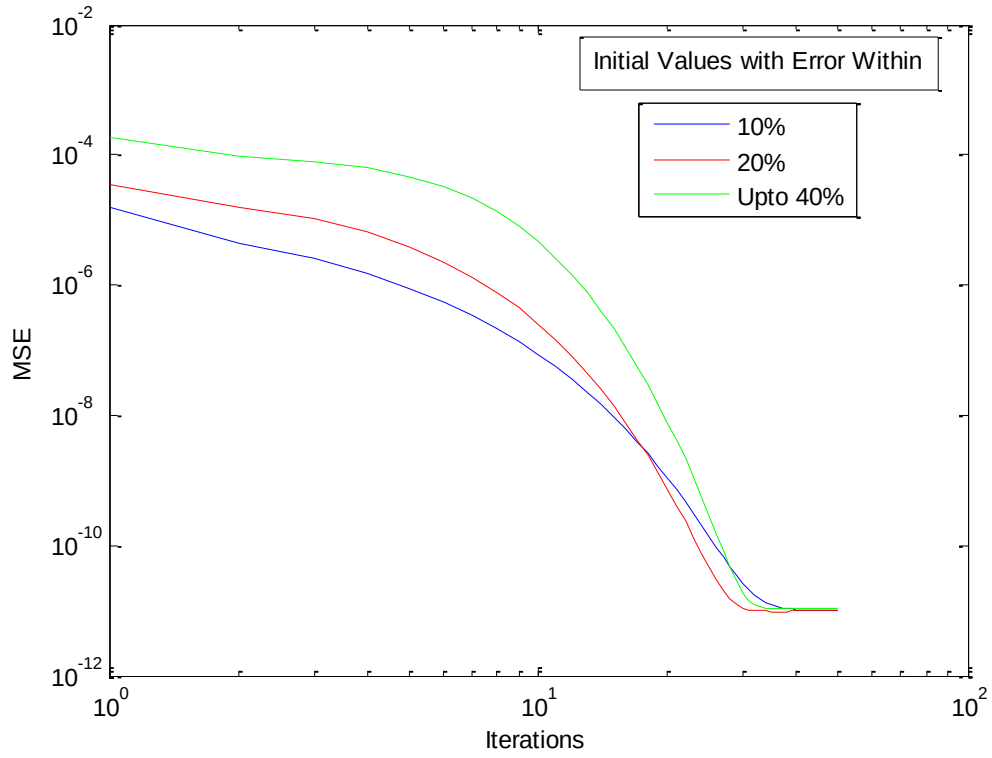


Figure 89: MSE for Training DC Motor with Viscous Friction

Table 18: Results of Training Linear Parameters of DC Motor with Known Viscous Friction

Case 1	**Actual Value**	**Initial Value**	**Final Value**	**Error (%)**
J	1.8e-4	1.98e-4	1.8e-4	0
B	0.060	0.0540	0.060	0
Case 2	**Actual Value**	**Initial Value**	**Final Value**	**Error (%)**
J	1.8e-4	1.44e-4	1.8e-4	0
B	0.060	0.0720	0.060	0
Case 3	**Actual Value**	**Initial Value**	**Final Value**	**Error (%)**
J	1.8e-4	1.08e-4	1.8e-4	0
B	0.060	0.0780	0.06	0

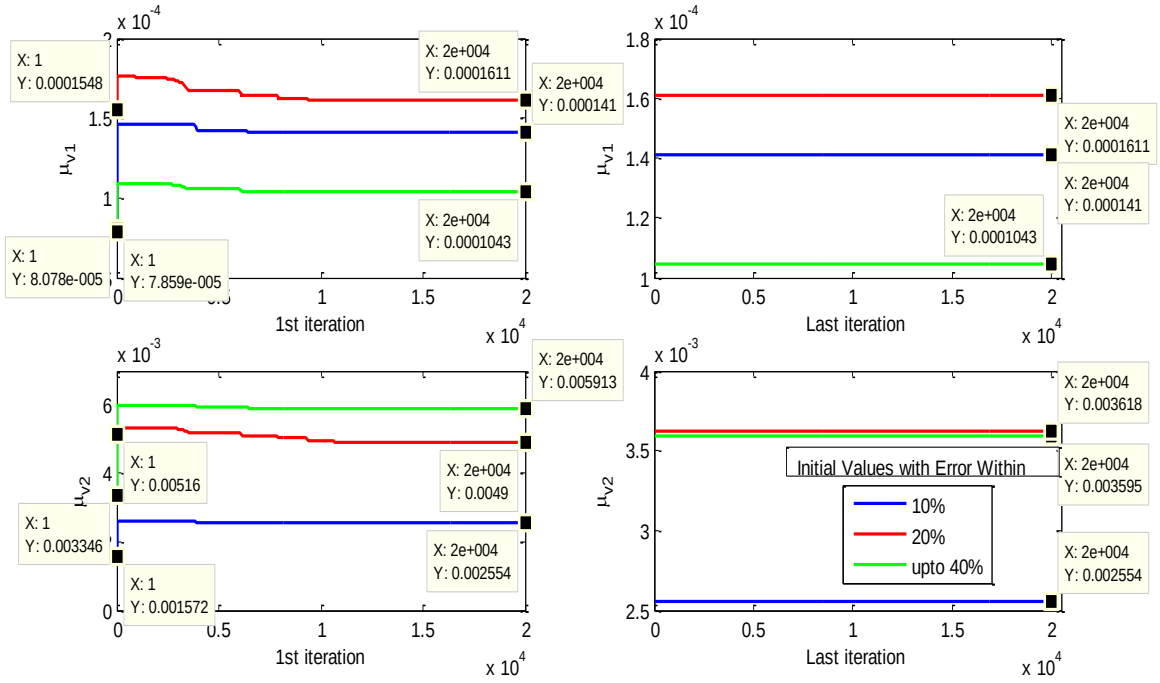


Figure 90: ALRs for Linear 2 Parameter Identification of DC Motor with Viscous Friction

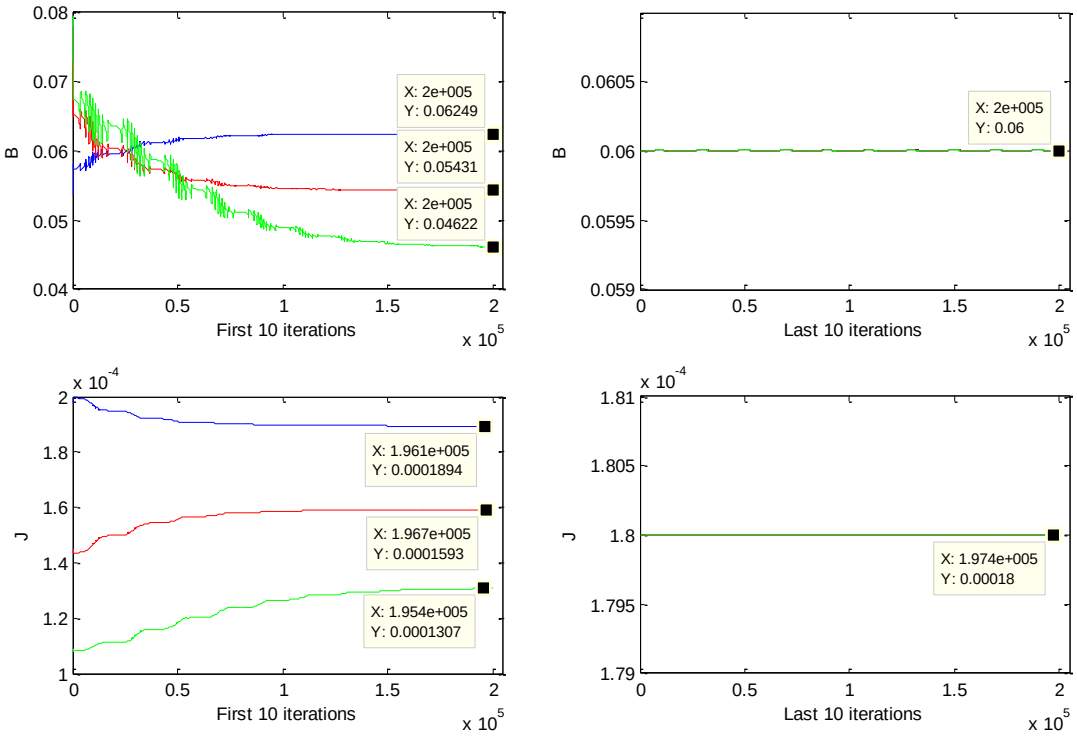


Figure 91: Convergence of B and J for Parameter Identification of DC Motor with Viscous Friction

The next case study involves identification of the motor torque, viscous friction and motor inertia of the DC motor assuming the resistance and induction are known a priori. The network learning structure is given in Figure 92. Figure 93 and Table 19 show the results of the simulations for initial conditions within 10% and 20% of the actual values. The adaptive learning rates and convergence of network parameters are shown in Figures 94 and 95 respectively. Based on the results, it can be seen that the method can be extended to also learn the motor torque constant, however the time taken for the network to converge to the exact values will be much higher as shown in Figure 93.

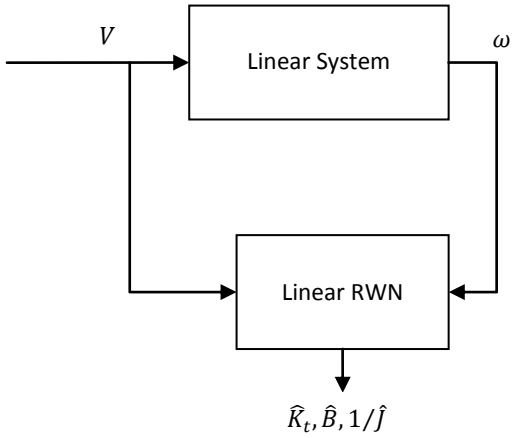


Figure 92: Network Learning Structure for Linear Parameter Identification

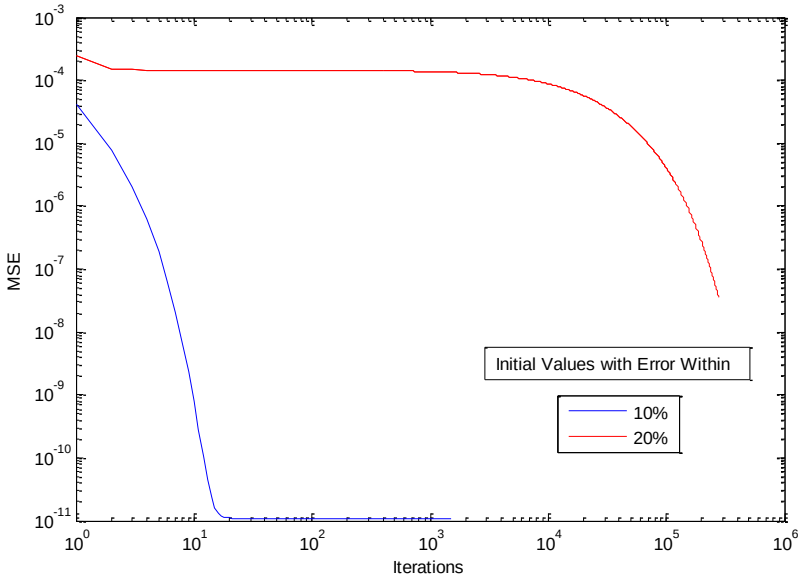


Figure 93: MSE for Training Linear Parameters of DC Motor with Unknown Viscous Friction

Table 19: Results of Training Linear Parameters of DC Motor with Unknown Viscous Friction

Case 1	**Actual Value**	**Initial Value**	**Final Value**	**Error (%)**
J	1.8e-4	1.98e-4	1.84e-4	2.2
K_t	0.0550	0.0495	0.0562	2.18
B	0.060	0.0540	0.0611	1.83
Case 2	**Actual Value**	**Initial Value**	**Final Value**	**Error (%)**
J	1.8e-4	2.16e-4	1.86e-4	3.76
K_t	0.0550	0.044	0.0566	2.86
B	0.060	0.048	0.0615	2.49

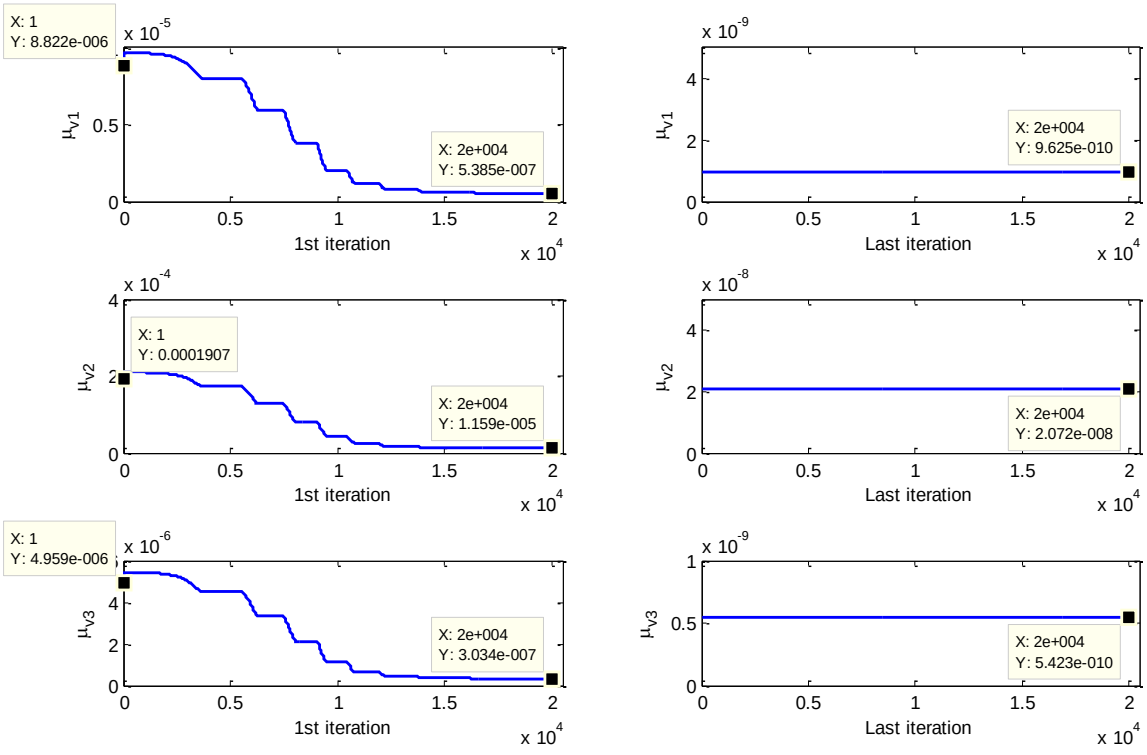


Figure 94: ALRs for Linear 3 Parameter Identification of DC Motor with Viscous Friction

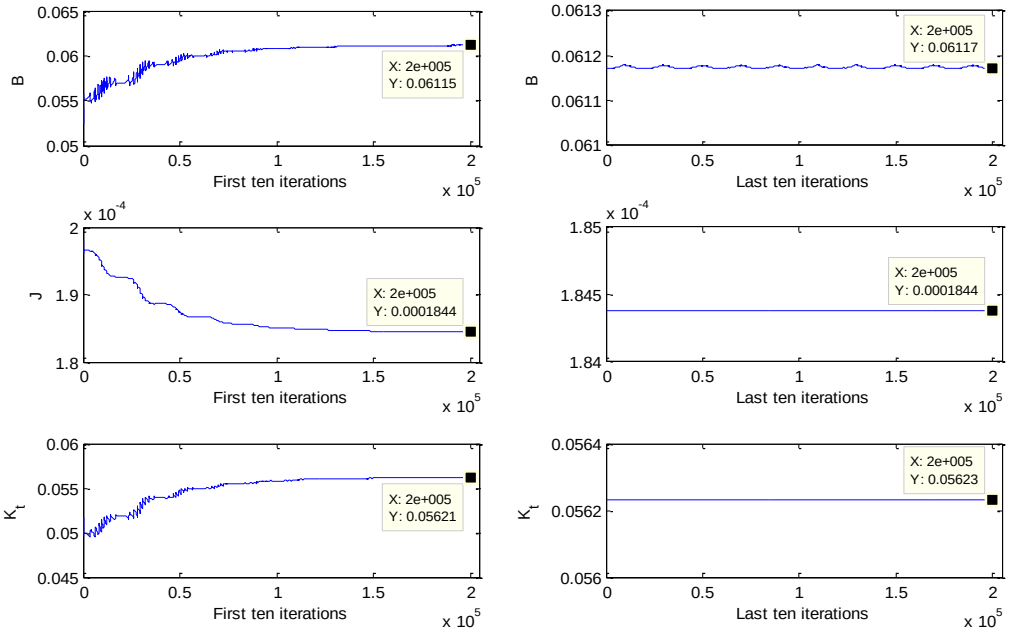


Figure 95: Convergence of B, J and Kt for DC Motor with Viscous Friction

5.4 Simultaneous Linear and Nonlinear Parameter Identification of DC Motor

In this section, simultaneous linear and nonlinear parameters identification of the DC Motor is carried out as shown in Figure 96. The results of sections 5.1 to 5.3 are used to optimize the network structure. The number of neurons in the hidden layer was set to 15, the first derivative of the Gaussian was used as the mother wavelet and the network was used to learn the motor inertia, viscous friction and coulomb friction.

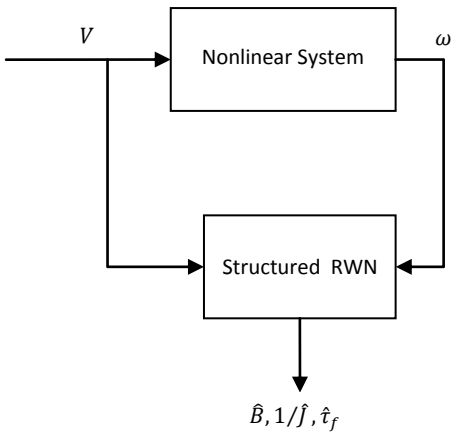


Figure 96: Network Learning Structure for Linear and Nonlinear DC Motor Parameter Identification

In order to initialize the nonlinear friction learning RWN which is used to learn only the coulomb friction and stiction, if any, training was carried out keeping the initial estimated values of the linear parameters constant as shown in Figure 97. Training was carried out to initialize the parameters $\hat{\theta} = \{\hat{w}_j, \hat{m}_j, \hat{d}_j, \hat{c}_j\}$.

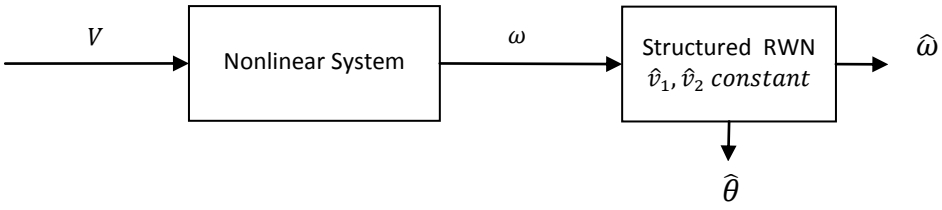


Figure 97: Initialization of RWN for Simultaneous Identification

The training signal was generated by applying a 1V chirp signal to the DC motor of frequency 0-4Hz as shown in Figure 98 and the torque speed characteristic of the DC Motor under consideration is shown in Figure 99. Training was carried out for 10000 iterations and the MSE for the initialization of the RWN is shown in Figure 100.

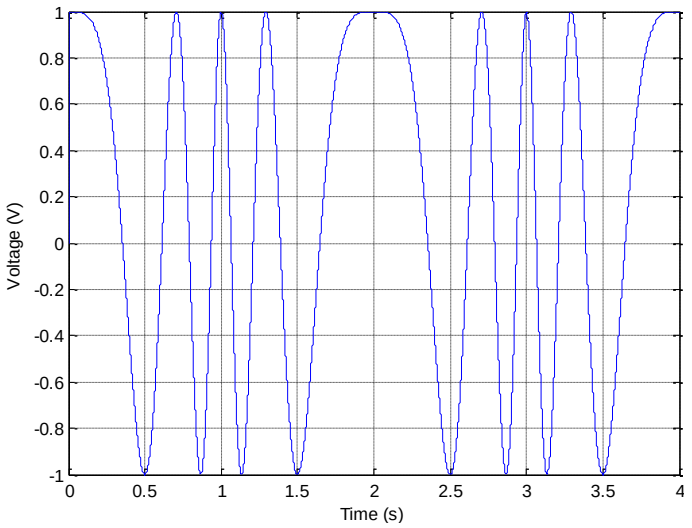


Figure 98: Training Signal for Simultaneous DC Motor Parameter Identification

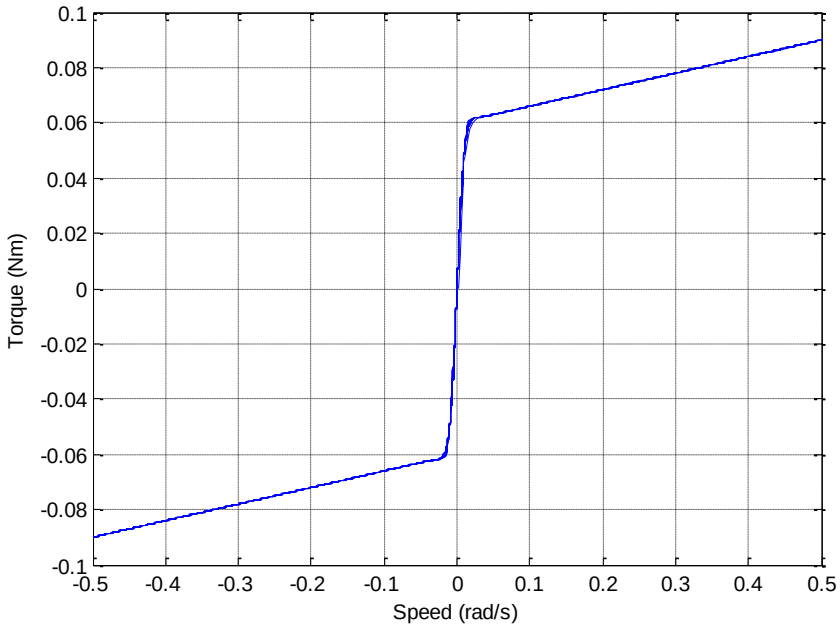


Figure 99: Torque Speed Characteristics of DC Motor

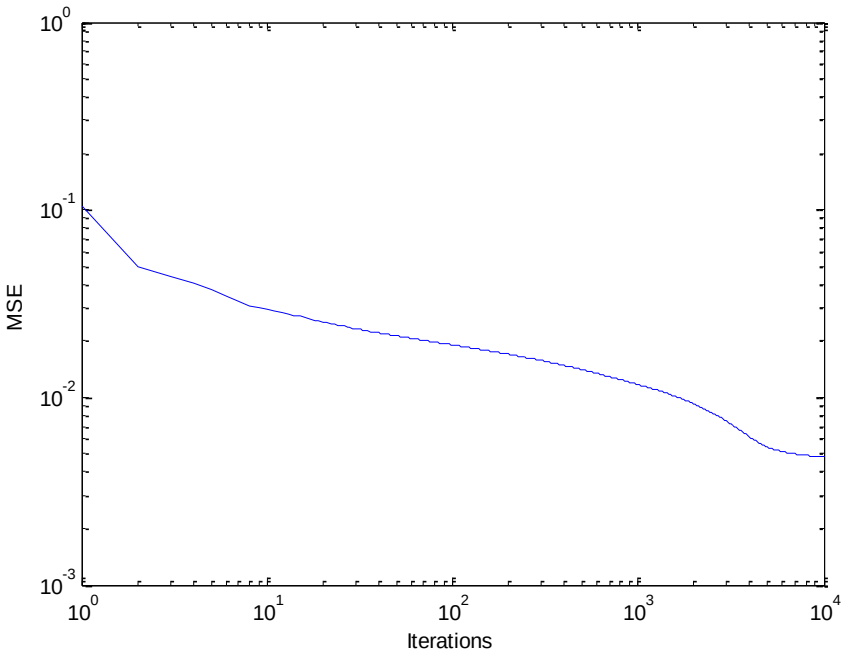


Figure 100: MSE for Initialization of Friction RWN

The initialized parameters were then used in the simultaneous training for both linear and nonlinear parameters. Training was carried out for 200000 iterations and the MSE after training is shown in Figure 101.

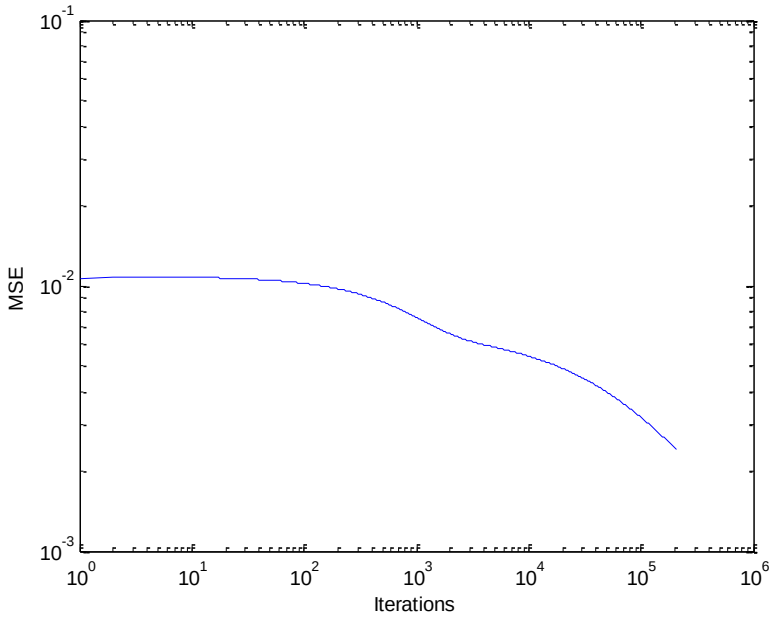


Figure 101: MSE for Simultaneous Parameter Identification

Figure 102 shows the adaptive learning rates for all the network parameters for the first training iteration. It was observed that the learning rates did not change after the first iteration. The convergence of network parameters is shown in Figure 103.

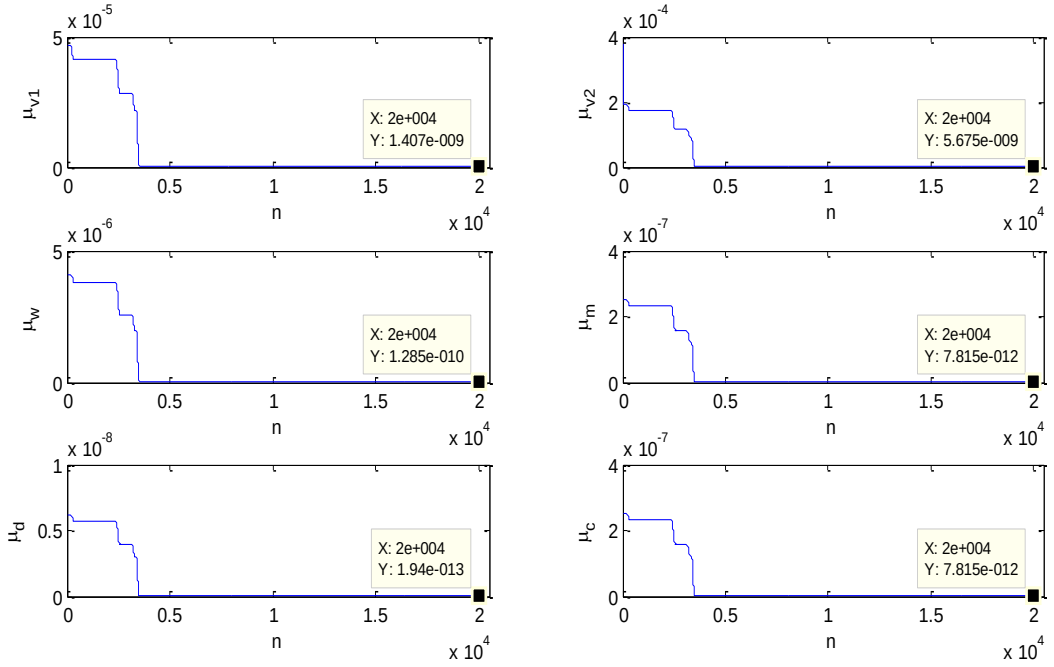


Figure 102: ALRs for Simultaneous Linear and Nonlinear Parameter Identification of DC Motor

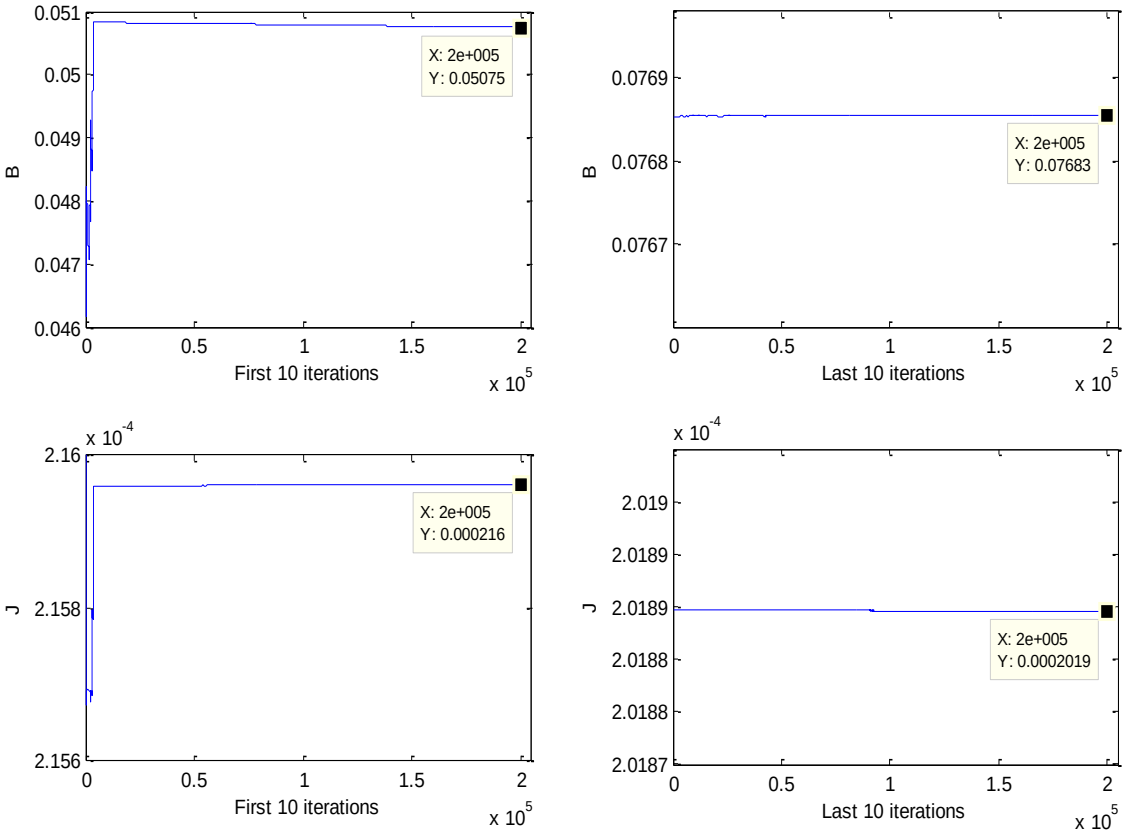


Figure 103: Convergence of B, J for DC Motor with Nonlinear Friction

Table 20 shows the values of the linear motor parameters after training. From Table 20, it can be seen that while the value of J is converging to the expected value, the value for the viscous friction is not.

Table 20: Results of Simultaneous Training of Linear and Nonlinear DC Motor Parameters

Variable	Actual Value	Initial Value	Final Value	Error (%)
J	1.8e-4	2.16e-4	2.01e-4	11
B	0.060	0.048	0.0735	22.5

Testing of the network was carried by applying a 1V sine wave of frequency 1Hz to the system and Figure 104 shows the comparison between the desired torque speed characteristic and the torque speed characteristic of the nonlinear network responsible for learning the coulomb and stiction friction. The MSE after testing was found to be 0.0017.

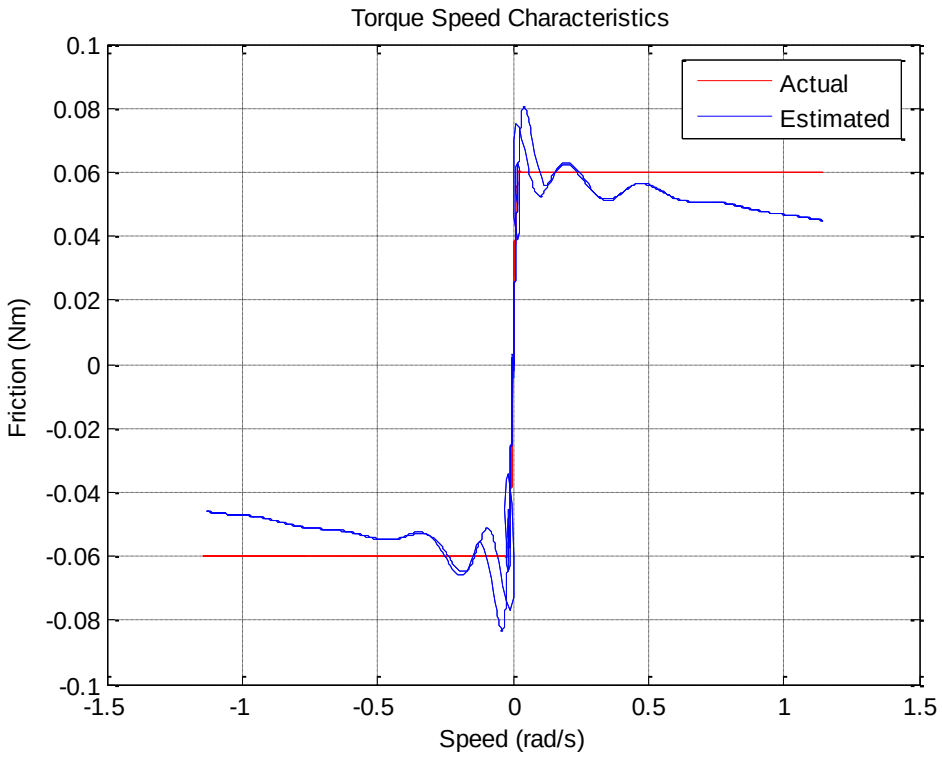


Figure 104: Estimated Frictional Torque Speed Characteristic

The results for the nonlinear friction characteristic can be explained as follows. From Equation 5.2, it is seen that $\hat{x}_1(n)$ has a linear relationship with $\hat{v}_1$ but a nonlinear relationship with $\hat{\tau}_f(n-1)$.

$$\hat{x}_1(n) = \hat{x}_1(n-1) - \hat{v}_2(\hat{v}_1\hat{x}_1(n-1) - \hat{v}_3\hat{x}_2(n-1) + \hat{\tau}_f(n-1)) \tag{5.2}$$

Equation 5.2 can be rewritten as given by Equation 5.3.

$$\hat{x}_1(n) = \hat{x}_1(n-1) - \hat{v}_2\big(\hat{v}_{11}\hat{x}_1(n-1) - \hat{v}_3\hat{x}_2(n-1) + \hat{\tau}_{fe}(n-1)\big) \tag{5.3}$$

The estimated frictional torque, $\hat{\tau}_{fe}$, and the viscous friction can therefore be written as shown in Equation 5.4 and 5.5.

$$\hat{\tau}_{fe}(n-1) = \hat{\tau}_f(n-1) + \hat{v}_{12}\hat{x}_1(n-1) \tag{5.4}$$

$$\hat{v}_1 = \hat{v}_{11} + \hat{v}_{12} \tag{5.5}$$

Based on these equations, it is seen that the system is not unique in that the nonlinear RWN may learn some or all of the parameter $\hat{v}_1$. Figure 105 shows the combined frictional characteristic which is identical to the desired friction, showing that the network has successfully learnt the viscous and coulomb friction. Figure 106 shows the comparison of the results obtained with the desired frictional torque speed characteristic.

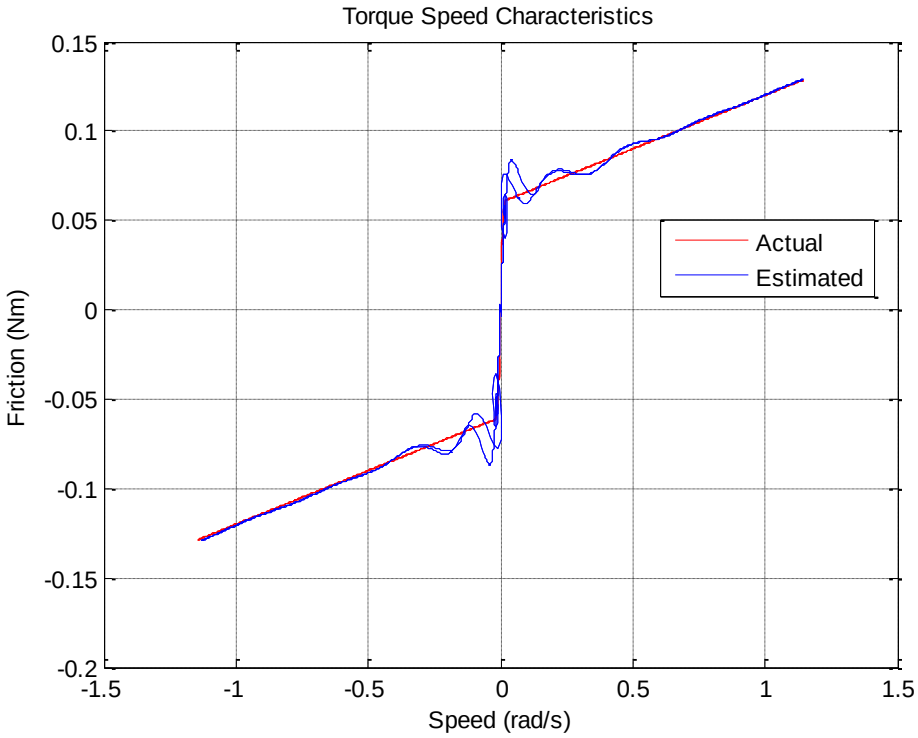


Figure 105: Estimated Combined Frictional Characteristic for Simultaneous Identification

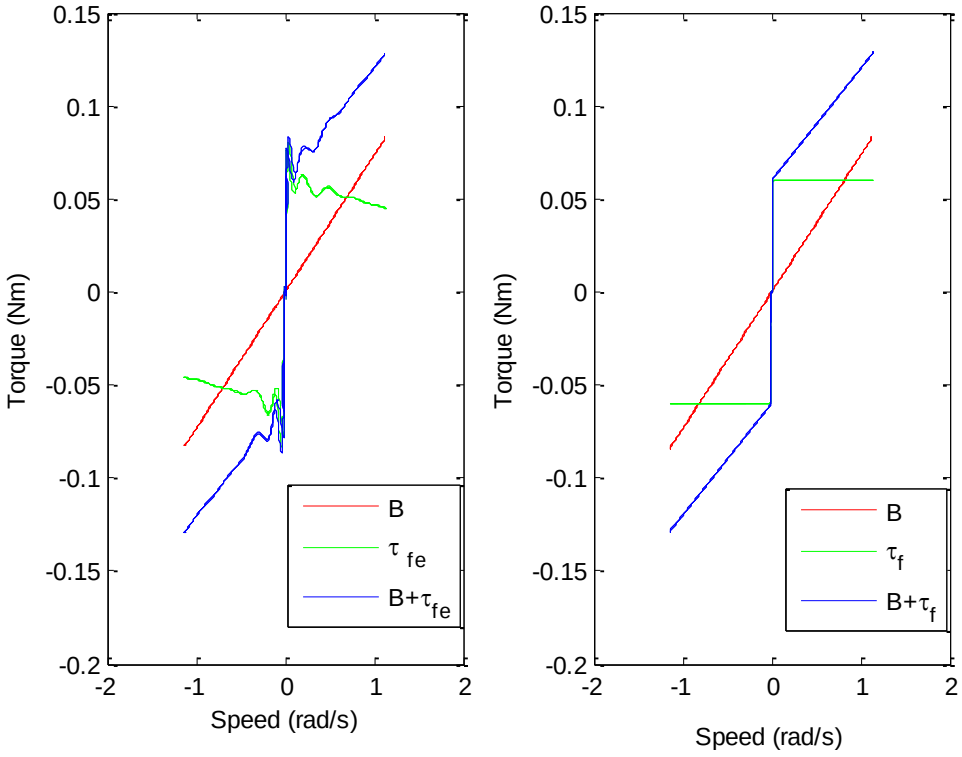


Figure 106: Comparison of Combined Frictional Characteristic

Chapter 6: Conclusion and Future Work

6.1 Conclusion

The aim of this thesis is to develop a recurrent wavelet network based algorithm for the modelling and identification of nonlinear time varying electromechanical systems. Black box modelling was first carried out with conventional wavelet networks in order to study and compare the different initialization algorithms available as well as to determine the effect of changing the mother wavelet and network size on the rate of convergence. It was observed that compared to the heuristic initialization method, the dyadic grid method provided faster convergence and a lower mean squared error. Comparison between three of the most commonly used mother wavelets, namely the first derivative of the Gaussian, the Morlet and Mexican Hat wavelets, showed that the effectiveness of mother wavelet depends largely on the type of function being modelled. In most cases, it was observed that the first derivative of the Gaussian performs well in terms of convergence speed and mean squared error. Similarly, the effect of the network size on the network performance was seen to depend on the type of function being modelled. In some cases, larger networks performed better, giving a smaller training mean square error. However, in other instances, a larger network size was seen to provide a negligible improvement in the mean square error and this was done at the expense of the speed of convergence. The effect of changing the learning rates on the network performance was also studied and simulation results showed that selection of learning rates significantly impacted the network performance and rate of convergence.

The selection of learning rates by trial and error proved to be tedious and for the case of recurrent wavelet networks, was unable to guarantee network stability. As a result, a need for adaptive learning rates which guarantee both network convergence and stability was observed. Adaptive learning rates were derived based on the Lyapunov Stability Theorem and grey box modelling was carried out to determine the performance of recurrent wavelet networks with adaptive learning rates when learning nonlinear time varying systems. The network was used to model a DC motor with viscous and coulomb friction and was able to accurately model the system. Similarly,

the recurrent wavelet network with adaptive learning rates was used to model coulomb and viscous friction. The results proved the ability of the network to accurately model highly nonlinear systems, while ensuring stability.

The next step after carrying out black and grey box modelling, involved the development of a structured recurrent wavelet network for white box modelling of the DC motor to identify the linear and nonlinear mechanical parameters of the system. Identification of linear network parameters, assuming the friction was known a priori, was carried out first, and it was seen that the network was able to accurately converge to the desired result even when the initial values were set to differ by over 20% of the actual values. An important observation was that in order for the network to converge to the desired values, the system being modelled must be unique. Nonlinear friction identification was also carried out, assuming the linear parameters of the motor were known a priori, and the recurrent wavelet network was able to accurately approximate the friction. Further validation of the proposed method was carried out through simultaneous identification of both linear and nonlinear system parameters. It was observed that while the parameter representing the motor inertia converged to the desired value, the linear parameter representing the viscous friction and the network representing the coulomb friction and stiction did not quite converge to the expected results. This was explained mathematically by observing that the nonlinear network was also able to learn a portion of the viscous friction. The linear combination of both was seen to reproduce the desired frictional torque speed characteristics of the motor being identified.

To summarize, the contributions of this thesis are as follows:

- Developing the algorithms for the modelling and simulation of conventional and recurrent wavelet networks.
- Developing the structured recurrent wavelet network for the DC motor and simultaneous parameter identification of linear and nonlinear parameters of the DC motor.
- Derivation and application of adaptive learning rates for recurrent wavelet networks.

- The work done in this thesis can also be extended to identification of the electrical network parameters and the relevant derivation for the ALRs is provided in Appendix E.

6.2 Recommendations for Future Work

Future work may involve:

- Carrying out simultaneous parameter identification for the electrical as well as mechanical linear and nonlinear system parameters
- Investigating the effect of noise on the performance of the learning algorithm
- Developing optimization algorithms to improve the simulation and convergence speed of the network.

References

[1] K. Narendra and K. Parthasarathy, "Identification and Control of Dynamical Systems Using Neural Networks," *IEEE Trans. on Neural Networks,* vol. 1, pp. 4-27, Mar. 1990.

[2] S. Omatu, M. Khalid and R. Yusof, *Neuro-Control and its Applications: Advances in Industrial Control.* New York: Springer, 1996.

[3] O. De Jesús, "Training General Dynamic Neural Networks," Ph.D. dissertation, Dept. Elect. and Comp. Eng., Oklahoma State Univ., Stillwater, OK, 2002.

[4] O. De Jesús and M. Hagan, "Backpropagation through Time for a Broad Class of Dynamic Networks," *IEEE Trans. on Neural Networks,* vol. 18, no. 1, pp. 14-27, Jan. 2007.

[5] Q. Zhang and A. Benveniste, "Wavelet Networks," *IEEE Trans. on Neural Networks,* vol. 3, no. 6, pp. 889-898, Nov. 1992.

[6] E. Garcia-Trevino, V. Alarcon-Aquino, J. Ramirez-Cruz, "Improving Wavelet-Networks Performance with a New Correlation-based Initialization Method and Training Algorithm," in *Proc. of the 15th International Conference on Computing,* 2006.

[7] Y.C.Pati and P.S. Krishnaprasad, "Analysis and Synthesis of Feedforward Neural Networks Using Discrete Affine Wavelet Transformations," *IEEE Trans. on Neural Networks,* vol. 4, pp. 73-85, Jan. 1993.

[8] Q. Zhang, "Using Wavelet Network in Nonparametric Estimation," *IEEE Trans. on Neural Networks,* vol. 8, no. 2, pp. 227-236, 1997.

[9] Y. Oussar and G. Dreyfus, "Initialization by Selection for Wavelet Network Training," *Journal of Neurocomputing,* vol. 34, pp. 131-143, Sept. 2000.

[10] Y. Oussar, I. Rivals, L. Personnaz, G. Dreyfus, "Training Wavelet Networks for Nonlinear Dynamic Input-Output Modelling," *Journal of Neurocomputing,* vol. 20, pp. 173-188, Feb. 1998.

[11] J. Sjöberg et al., "Nonlinear Black-box Modelling in System Identification: A Unified Overview," *Automatica,* vol. 31, no. 12, pp. 1691-1724, Dec. 1995.

[12] S. Postalcioglu and Y. Becerikli, "Nonlinear System Modelling Using Wavelet Networks," in *Proc. of the International Conference on Signal Processes,* 2003.

[13] S.J. Yoo, J.B. Park and Y.H. Choi, "Direct Adaptive Control Using Self Recurrent Wavelet Neural Network Via Adaptive Learning Rates for Stable Path Tracking of Mobile Robots," *Proc. American Control Conference,* 2005, pp.288-293.

[14] S.J.Yoo, J.B. Park and Y.H. Choi, "Indirect Adaptive Control of Nonlinear Dynamic Systems Using Self Recurrent Wavelet Neural Networks Via Adaptive Learning Rates," *Information Sciences,* vol. 177, pp. 3074-3098, Feb. 2007.

[15] S.J. Yoo, J.B. Park and Y.H. Choi, "Generalized Predictive Control Based on Self Recurrent Wavelet Neural Network for Stable Path Tracking of Mobile Robots: Adaptive Learning Rates Approach," *IEEE Trans. On Circuits and Systems,* vol. 3, no. 6, pp. 1381-1394, June 2006.

[16] S.J.Yoo, J.B. Park and Y.H. Choi, "Stable Predictive Control of Chaotic Systems Using Self-Recurrent Wavelet Neural Network," *International Journal of Control, Automation and Systems,* vol. 3, no. 1, pp. 43-55, March 2005.

[17] C.Lin, C. Lee and C. Chin, "Dynamic Recurrent Wavelet Network Controllers for Nonlinear System Control," *Journal of the Chinese Institute of Engineers,* vol. 29, no. 4, pp. 747-751, 2006.

[18] C. Lu, "Design and Application of Stable Predictive Controller Using Recurrent Wavelet Neural Networks," in *IEEE Trans. on Industrial Electronics,* vol. 56, no. 9, pp. 3733-3742, Sept. 2009.

[19] R.J. Williams and D. Zipser, "A Learning Algorithm for Continually Running Fully Recurrent Neural Networks," *Neural Computation,* vol. 1, no. 2, pp. 270-280, 1989.

[20] C.C. Ku and K.Y. Lee, "Diagonal Recurrent Neural Networks for Dynamic Systems Control," in *IEEE Trans. Neural Networks,* vol. 6 (1), pp. 144–156, Jan. 1995.

[21] J. Peng and R. Dubay, "Identification and Adaptive Neural Network Control of a DC Motor System with Dead-zone Characteristics," *ISA Transactions,* vol. 50, no. 4, pp. 588-598, Oct. 2011.

[22] A.A. Al-Qassar and M. Z. Othman, "Experimental Determination of Electrical and Mechanical Parameters of DC Motor using Genetic Elman Neural Network," *Journal of Engineering Science and technology,* vol. 3, no. 2, pp. 190-196, 2008.

[23] D. Schröder, C. Hintz and M. Rau, "Intelligent Modeling, Observation, and Control for Nonlinear Systems," in *IEEE Trans. on Mechatronics,* vol. 6, no. 2, pp. 122-130, 2001.

[24] C. Hintz, M. Rau and D. Schröder, "Identification of a Nonlinear Multi Stand Rolling System by a Structured Recurrent Neural Network," *Proc. of IEEE Industry Applications Conference,* vol. 2, pp. 1121-1128, Oct. 2000.

[25] C. Endisch et. al, "Identification of Mechatronic Systems with Dynamic Neural Networks using Prior Knowledge," *Proc. of the WCEAS,* vol. 2, pp. 859-865, Oct. 2009.

[26] M.T. Hagan and M.T. Mehnaj, "Training Feedforward Networks with the Marquardt Algorithm," in *IEEE Trans. on Neural Networks,* vol. 5(6), pp. 989-993, Nov. 1994.

[27] R.A. Horn and C. R. Johnson, *Matrix Analysis,* Cambridge: Cambridge University Press, 1999.

Appendix A

MATRIX NORMS

Let A be a $N_r \times N_c$ matrix,

$$A = \begin{bmatrix} a_{11} & . & a_{1N_c} \\ . & . & . \\ a_{N_r1} & . & a_{N_rN_c} \end{bmatrix}$$

The L_1 norm, also known as the maximum absolute column sum, is given as

$$\|A\|_1 = \max_j \sum_{i=1}^{N_r} |a_{ij}| \leq N_r |a_{ij}|_{max}$$

The L_∞ norm, also known as the maximum absolute row sum, is given as

$$\|A\|_\infty = \max_i \sum_{j=1}^{N_c} |a_{ij}| \leq N_c |a_{ij}|_{max}$$

From,

$$\|A\|_2^2 \leq \|A\|_1 \|A\|_\infty$$

$$\|A\|_2^2 \leq N_r N_c |a_{ij}|_{max}^2$$

Therefore the L_2 norm is given by

$$\|A\|_2 \leq \sqrt{N_r N_c} |a_{ij}|_{max}$$

Appendix B

GENERAL SOLUTION OF RECURSIVE EQUATIONS FOR RWN

$$V_{1,ji}(n) = \frac{\varphi'_{ji}(n)}{d_{ji}}\left(c_{ji}.V_{1,ji}(n-1) - 1\right)$$

Using MAPLE, the recursive equation is solved as shown

```
> rsolve(V1(n) = (c/d)*dphi(n)*V1(n-1)-(1/d)*dphi(n), V1(n));
```

$$\frac{\left(\prod_{n0=0}^{n-1} \mathrm{dphi}(n0+1)\right)\left(\frac{c}{d}\right)^{n}\left(-\left(\sum_{n1=0}^{n-1} \frac{\mathrm{dphi}(n1+1)\left(\frac{c}{d}\right)^{(-n1)}}{\prod_{n0=0}^{n1} \mathrm{dphi}(n0+1)}\right) + \mathrm{V1}(0)\,c\right)}{c}$$

The solution of the recursive equation is as follows,

$$V_{1,ji}(n) = \frac{1}{c_{ji}}\left(\prod_{p=0}^{n-1}\varphi'_{ji}(p+1)\right)\left(\frac{c_{ji}}{d_{ji}}\right)^{n}\left[c_{ji}V_{1,ji}(0) - \sum_{r=0}^{n-1}\left(\frac{\left(\frac{c_{ji}}{d_{ji}}\right)^{-r}\varphi'_{ji}(r+1)}{\prod_{p=0}^{r}\varphi'_{ji}(p+1)}\right)\right]$$

This can be rewritten as,

$$V_{1,ji}(n) = \left(\prod_{p=0}^{n-1}\varphi'_{ji}(p+1)\right)\left(V_{1,ji}(0)\left(\frac{c_{ji}}{d_{ji}}\right)^{n}\right) - \frac{1}{d_{ji}}\sum_{p=0}^{n-1}\left(\left(\frac{c_{ji}}{d_{ji}}\right)^{n-p-1}\prod_{r=0}^{n-p-1}\varphi'_{ji}(n-r)\right)$$

$$V_{2,ji}(n) = \frac{\partial\varphi\left(z_{ji}(n)\right)}{\partial z_{ji}(n)} . \frac{1}{d_{ji}}\left(-z_{ji}(n) + c_{ji} . V_{2,ji}(n-1)\right)$$

Using MAPLE, the recursive equation is solved as shown

```
> rsolve(V2(n) = (-1/d)*dphi(n)*z(n)+(c/d)*dphi(n)*V2(n-1), V2(n));
```

$$\frac{\left(\prod_{n0=0}^{n-1} \mathrm{dphi}(n0+1)\right)\left(\frac{c}{d}\right)^{n}\left(-\left(\sum_{n1=0}^{n-1} \frac{\mathrm{dphi}(n1+1)\, z(n1+1)\left(\frac{c}{d}\right)^{(-n1)}}{\prod_{n0=0}^{n1} \mathrm{dphi}(n0+1)}\right) + \mathrm{V2}(0)\, c\right)}{c}$$

The solution of the recursive equation is as follows,

$$V_{2,ji}(n) = \frac{1}{c_{ji}}\left(\prod_{p=0}^{n-1} \varphi'_{ji}(p+1)\right)\left(\frac{c_{ji}}{d_{ji}}\right)^{n}\left[c_{ji}V_{2,ji}(0) - \sum_{r=0}^{n-1}\left(\frac{\left(\frac{d_{ji}}{c_{ji}}\right)^{r}\varphi'_{ji}(r+1)z_{ji}(r+1)}{\prod_{p=0}^{r}\varphi'_{ji}(p+1)}\right)\right]$$

This can be rewritten as

$$V_{2,ji}(n) = \left(\prod_{p=0}^{n-1} \varphi'_{ji}(p+1)\right)\left(V_{2,ji}(0)\left(\frac{c_{ji}}{d_{ji}}\right)^{n}\right) - \frac{1}{d_{ji}}\sum_{p=0}^{n-1}\left(z_{ji}(p+1)\left(\frac{c_{ji}}{d_{ji}}\right)^{n-p-1}\prod_{r=0}^{n-p-1}\varphi'_{ji}(n-r)\right)$$

$$V_{3,ji}(n) = \frac{\partial\varphi(z_{ji}(n))}{\partial z_{ji}(n)} . \frac{1}{d_{ji}} \left(\varphi\left(z_{ji}(n-1)\right) + c_{ji} V_{3,ji}(n-1)\right)$$

Using MAPLE, the recursive equation is solved as shown

```
> rsolve(V3(n) = (1/d)*dphi(n)*phi(n-1)+(c/d)*dphi(n)*V3(n-1), V3(n));
```

$$\frac{\left(\prod_{n0=0}^{n-1} \mathrm{dphi}(n0+1)\right)\left(\frac{c}{d}\right)^{n}\left(\left(\sum_{n1=0}^{n-1} \frac{\left(\frac{c}{d}\right)^{(-n1)} \phi(n1)\,\mathrm{dphi}(n1+1)}{\prod_{n0=0}^{n1} \mathrm{dphi}(n0+1)}\right) + \mathrm{V3}(0)\,c\right)}{c}$$

The solution of the recursive equation is as follows,

$$V_{3,ji}(n) = \frac{1}{c_{ji}}\left(\prod_{p=0}^{n-1} \varphi'_{ji}(p+1)\right)\left(\frac{c_{ji}}{d_{ji}}\right)^{n}\left[c_{ji} V_{1,ji}(0) + \sum_{r=0}^{n-1}\left(\frac{\left(\frac{c_{ji}}{d_{ji}}\right)^{-r} \varphi'_{ji}(r+1)\varphi_{ji}(r)}{\prod_{p=0}^{r} \varphi'_{ji}(p+1)}\right)\right]$$

This can be rewritten as,

$$V_{3,ji}(n) = \left(\prod_{p=0}^{n-1} \varphi'_{ji}(p+1)\right)\left(V_{3,ji}(0)\left(\frac{c_{ji}}{d_{ji}}\right)^{n}\right) + \frac{1}{d_{ji}} \sum_{p=0}^{n-1}\left(\varphi_{ji}(p)\left(\frac{c_{ji}}{d_{ji}}\right)^{n-1-p} \prod_{r=0}^{n-p-1} \varphi'_{ji}(n-r)\right)$$

Appendix C

DC MOTOR DATA SHEET

Type 1.13.075.XXX			016	214
Characteristics*				
Rated voltage	V	V	24	24
Rated power	P_N	W	130	200
Rated torque	T_N	mNm	400	600
Rated speed	n_N	rpm	3200	3200
Rated current	I_N	A	8.0	12.0
No load characteristics*				
No load speed	n_0	rpm	3900	3900
No load current	I_0	A	0.7	0.8
Starting characteristics*				
Starting torque	T_s	mNm	2250	3450
Starting current	I_s	A	42	64
Performance Characteristics*				
max. Output power	P_{max}	W	230	370
max. Constant torque	T_{max}	mNm	280	400
Motor parameters*				
Weight	G	g	1500	1800
Rotor inertia	J	gcm^2	1300	1800
Terminal resistance	R	Ohm	0.6	0.4
Mech. time constant	τ_m	ms	27	29
Electr. time constant	τ_e	ms	2.4	2.0
Speed regulation constant	R_m	rpm/mNm	1.7	1.1
Torque constant	k_t	mNm/A	55	55
Thermal resistance	R_{th1}	K/W	2.2	2.0
Thermal resistance	R_{th2}	K/W	2.5	2.2
Axial play		mm	< 0.01	< 0.01
Direction of rotation			bidirectional	

Appendix D

RULES FOR VECTOR AND MATRIX NORMS

Rule 1:

$$\|A + B\| \leq \|A\| + \|B\|$$

Rule 2:

$$\|AB\| \leq \|A\|.\|B\|$$

Rule 3:

$$\|A^r\| \leq \|A\|^r$$

Rule 4:

For a diagonal matrix, D, where ρ is the absolute maximum element of the matrix,

$$\sum_{r=0}^{\infty} \|D\|^r = \frac{1}{1-\rho} \quad \Leftrightarrow \ |\rho| < 1$$

Appendix E

DERIVATION OF ALR FOR ELECTRICAL PARAMETERS OF DC MOTOR

- μ_4

The learning rate, $\mu_{\hat{v}_4}$, is selected so as to satisfy Equation E.1.

$$0 < \mu_{\hat{v}_4} < \frac{2}{\max_n \left\| \frac{\partial \hat{y}(n)}{\partial \hat{v}_4} \right\|^2} \tag{E.1}$$

In order to compute the norm of $\frac{\partial \hat{y}(n)}{\partial \hat{v}_2}$, the solution of the Equation E.2 must be computed.

$$\begin{bmatrix} P_4(n) \\ Q_4(n) \end{bmatrix} = \begin{bmatrix} 1 - \hat{v}_1\hat{v}_2 & \hat{v}_2\hat{v}_3 \\ -\hat{v}_3\hat{v}_4 & \hat{v}_5 \end{bmatrix} \begin{bmatrix} P_4(n-1) \\ Q_4(n-1) \end{bmatrix} + \begin{bmatrix} 0 \\ -\hat{v}_3 \end{bmatrix} \hat{x}_1(n-1) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(n-1) \tag{E.2}$$

where

$$P_4(n) = \frac{d\hat{y}(n)}{d\hat{v}_4} \; and \; Q_4(n) = \frac{d\hat{x}_2(n)}{d\hat{v}_4}$$

Equation E.2 can be rewritten as shown in Equation E.3 and E.4 where

where $R_4(n) = \begin{bmatrix} P_4(n) \\ Q_4(n) \end{bmatrix}$.

$$R_4(n) = AR_4(n-1) + E_6\hat{x}_1(n-1) + E_7u(n-1) \tag{E.3}$$

$$P_4(n) = CR_4(n) \tag{E.4}$$

The solution to the Equation E.4 is shown in Equation E.5.

$$P_4(n) = C\left(A^nR_4(0) + \sum_{m=0}^{n-1} A^{n-m-1}(E_6\hat{x}_1(m) + E_7u(m))\right) \tag{E.5}$$

Changing the index such that *r=n-m-1* and since $R_4(0) = 0$, the norm of $P_4(n)$ given in Equation E.5 can be written as shown in Equation E.6.

$$\|P_4(n)\| \leq \left\| C \sum_{r=0}^{n-1} A^r \left(E_6 \hat{x}_1(n-r-1) + E_7 u(n-r-1)\right) \right\| \tag{E.6}$$

Using Rule 1 and 2 of matrix norms given in Appendix D and the fact that $\|C\| = \|E_7\| = 1, \|E_6\| = |\hat{v}_3|$ Equation E.6 can be further decomposed as shown in Equation E.7.

$$\|P_2(n)\| \leq (1 + \|A\| + \cdots + \|A^{n-1}\|).(|\hat{v}_3| \|\hat{x}_1\|_{max} + \|u\|_{max}) \tag{E.7}$$

where

$$\|u\|_{max} = \max_n \|u(n)\| \qquad \|\hat{x}_1\|_{max} = \max_n \|\hat{x}_1(n)\|$$

Equation E.7 can then be written in the form of a series summation as shown in Equation E.8.

$$\|P_4(n)\| \leq \left(\sum_{r=0}^{n-1} \|A^r\| \right).(|\hat{v}_3| \|\hat{x}_1\|_{max} + \|u\|_{max}) \tag{E.8}$$

Using Theorem 1 and eigenvalue decomposition, Equation E.8 can be simplified as shown in Equation E.9.

$$\|P_4(n)\| \leq \left(\sum_{r=0}^{\infty} \|M D^r M^{-1}\| \right).(|\hat{v}_3| \|\hat{x}_1\|_{max} + \|u\|_{max}) \tag{E.9}$$

Using Rules 1-4 from Appendix D, Equation E.9 can be written as Equation E.10.

$$\|P_4(n)\| \leq \left(\frac{\|M\|.\|M^{-1}\|}{1 - \rho(A)} \right).(|\hat{v}_3| \|\hat{x}_1\|_{max} + \|u\|_{max}) \tag{E.10}$$

Using Equation E.10, the maximum norm can be computed as shown in Equation E.11 where $P_{4,max} = \max_n \|P_4(n)\|$,.

$$P_{4,max} = \max_n \left\| \left(\frac{\|M\| . \|M^{-1}\|}{1 - \rho(A)} \right) (|\hat{v}_3| \|\hat{x}_1\|_{max} + \|u\|_{max}) \right\| \tag{E.11}$$

Substituting Equation E.11 in Equation E.1, the adaptive learning rate is selected as shown in Equation E.12.

$$0 < \mu_{\hat{v}_4} < \frac{2}{P_{4,max}^2} \tag{E.12}$$

- μ_5

The learning rate, $\mu_{\hat{v}_5}$, is selected so as to satisfy Equation E.13.

$$0 < \mu_{\hat{v}_5} < \frac{2}{\max_n \left\| \frac{\partial \hat{y}(n)}{\partial \hat{v}_5} \right\|^2} \tag{E.13}$$

In order to compute the norm of $\frac{\partial \hat{y}(n)}{\partial \hat{v}_5}$, the solution of the Equation E.14 must be computed.

$$\begin{bmatrix} P_5(n) \\ Q_5(n) \end{bmatrix} = \begin{bmatrix} 1 - \hat{v}_1 \hat{v}_2 & \hat{v}_2 \hat{v}_3 \\ -\hat{v}_3 \hat{v}_4 & \hat{v}_5 \end{bmatrix} \begin{bmatrix} P_5(n-1) \\ Q_5(n-1) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \hat{x}_2(n-1) \tag{E.14}$$

where

$$P_5(n) = \frac{d\hat{y}(n)}{d\hat{v}_5} \; and \; Q_5(n) = \frac{d\hat{x}_2(n)}{d\hat{v}_5}$$

Equation E.14 can be rewritten as shown in Equation E.15 and E.16 where

where $R_5(n) = \begin{bmatrix} P_5(n) \\ Q_5(n) \end{bmatrix}$.

$$R_5(n) = AR_5(n-1) + E_7 \hat{x}_2(n-1) \tag{E.15}$$

$$P_5(n) = CR_5(n) \tag{E.16}$$

The solution to the Equation E.16 is shown in Equation E.17.

$$P_5(n) = C\left(A^n R_5(0) + \sum_{m=0}^{n-1} A^{n-m-1}(E_7\hat{x}_2(m))\right) \tag{E.17}$$

Changing the index such that *r=n-m-1* and since $R_5(0) = 0$, the norm of $P_5(n)$ given in Equation E.17 can be written as shown in Equation E.18.

$$\|P_5(n)\| \leq \left\| C \sum_{r=0}^{n-1} A^r \left(E_7\hat{x}_2(n-r-1)\right) \right\| \tag{E.18}$$

Using Rule 1 and 2 of matrix norms given in Appendix D and the fact that $\|C\| = \|E_7\| = 1$, Equation E.18 can be further decomposed as shown in Equation E.19.

$$\|P_5(n)\| \leq (1 + \|A\| + \cdots + \|A^{n-1}\|).(\|\hat{x}_2\|_{max}) \tag{E.19}$$

where

$$\|\hat{x}_2\|_{max} = \max_n \|\hat{x}_2(n)\|$$

Equation E.19 can then be written in the form of a series summation as shown in Equation E.20.

$$\|P_5(n)\| \leq \left(\sum_{r=0}^{n-1} \|A^r\|\right).(\|\hat{x}_2\|_{max}) \tag{E.20}$$

Using Theorem 1 and eigenvalue decomposition, Equation E.20 can be simplified as shown in Equation E.21.

$$\|P_5(n)\| \leq \left(\sum_{r=0}^{\infty} \|MD^rM^{-1}\|\right).(\|\hat{x}_2\|_{max}) \tag{E.21}$$

Using Rules 1-4 from Appendix D, Equation E.21 can be written as Equation E.22.

$$\|P_5(n)\| \leq \left(\frac{\|M\|.\|M^{-1}\|}{1-\rho(A)}\right).(\|\hat{x}_2\|_{max}) \tag{E.22}$$

Using Equation E.22, the maximum norm can be computed as shown in Equation E.23 where $P_{5,max} = \max_n \|P_5(n)\|$,.

$$P_{5,max} = \max_n \left\| \left(\frac{\|M\|.\|M^{-1}\|}{1-\rho(A)} \right) (\|\hat{x}_2\|_{max}) \right\| \tag{E.23}$$

Substituting Equation E.23 in Equation E.13, the adaptive learning rate is selected as shown in Equation E.24.

$$0 < \mu_{\hat{v}_5} < \frac{2}{P_{5,max}^2} \tag{E.24}$$

Vita

Sarah Hussain Zahidi graduated from English Medium School in Dubai, where she secured 10 As in O Levels and 2 As in the A Level exams. She continued her education at the American University of Sharjah where she graduated Summa Cum Laude as Bachelor of Electrical Engineering in 2009.

Following her graduation, she began the Master's degree in Electrical Engineering at the American University of Sharjah where she also worked as a Graduate Teaching Assistant. During the course of her degree, Ms. Zahidi joined MARS, GCC as a Project Control Engineer and then changed roles to become a Maintenance Planning Engineer also at MARS, GCC. She was awarded the Master of Science degree in Electrical Engineering in Fall 2012.