University of Nevada, Reno

Neural Networks: Training and Application to Nonlinear System Identification and Control

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Electrical Engineering

By

Hamid Khodabandehlou

Dr. M. Sami Fadali/Dissertation Advisor

May, 2018

THE GRADUATE SCHOOL

We recommend that the dissertation
prepared under our supervision by

HAMID KHODABANDEHLOU

Entitled

Neural Networks: Training And Application To Nonlinear System Identification And Control

be accepted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

Dr. M. Sami Fadali, Advisor

Dr. Bahram Parvin, Committee Member

Dr. Hanif Livani, Committee Member

Dr. Hao Xu, Committee Member

Dr. Gokhan Pekcan, Graduate School Representative

David W. Zeh, Ph. D., Dean, Graduate School
May, 2018

Abstract

This dissertation investigates training neural networks for system identification and classification. The research contains two main contributions as follow:

1. Reducing number of hidden layer nodes using a feedforward component

This research reduces the number of hidden layer nodes and training time of neural networks to make them more suited to online identification and control applications by adding a parallel feedforward component. Implementing the feedforward component with a wavelet neural network and an echo state network provides good models for nonlinear systems.

The wavelet neural network with feedforward component along with model predictive controller can reliably identify and control a seismically isolated structure during earthquake. The network model provides the predictions for model predictive control. Simulations of a 5-story seismically isolated structure with conventional lead-rubber bearings showed significant reductions of all response amplitudes for both near-field (pulse) and far-field ground motions, including reduced deformations along with corresponding reduction in acceleration response. The controller effectively regulated the apparent stiffness at the isolation level. The approach is also applied to the online identification and control of an unmanned vehicle. Lyapunov theory is used to prove the stability of the wavelet neural network and the model predictive controller.

2. Training neural networks using trajectory based optimization approaches

Training neural networks is a nonlinear non-convex optimization problem to determine the weights of the neural network. Traditional training algorithms can be inefficient and can get trapped in local minima. Two global optimization approaches are adapted to train neural networks and avoid the local minima problem. Lyapunov theory is used to prove the stability of the proposed methodology and its convergence in the presence of measurement errors. The first approach transforms the constraint satisfaction problem into unconstrained optimization. The constraints define a quotient gradient system (QGS) whose stable equilibrium points are local minima of the unconstrained optimization. The QGS is integrated to determine local minima and the local minimum with the best generalization performance is chosen as the optimal solution. The second approach uses the QGS together with a projected gradient system (PGS). The PGS is a nonlinear dynamical system, defined based on the optimization problem that searches the components of the feasible region for solutions. Lyapunov theory is used to prove the stability of PGS and QGS and their stability under presence of measurement noise.

Acknowledgement

I would like to express my sincerest gratitude to those who have helped me to complete my degree. I would like to thank Dr. Mohammed Sami Fadali, my Ph.D. advisor for offering research work and his invaluable guidance, encouragement, valuable suggestions and support through these years.

I would like to thank the committee members, Dr. Mohammed Sami Fadali, Dr. Bahram Parvin, Dr. Hanif Livani, Dr. Gokhan Pekcan and Dr. Hao Xu for their support in completing this work.

Last but not the least, my deepest thanks are expressed to my family and friends for their endless encouragement and support.

.

Table of Contents

List of Tables

Table of Figures

Chapter 1. Introduction

1.1 Background

Artificial neural networks can approximate any nonlinear system with the desired accuracy and have been extensively used for the identification of nonlinear systems [1], [2], [3]. Despite the advantages of artificial neural networks in nonlinear system identification, their slow convergence rate, multi-layer structure and computational complexity pose considerable difficulties for online identification and control of nonlinear systems. To cope with these shortcomings, researchers proposed different strategies to reduce the training time of the neural networks. Steck et al. [4] argued that parallel implementation of a recursive least squares and a traditional neural network is much faster than a multilayer neural network with the Marquardt-Levenberg algorithm. They proposed a method that trains neural networks faster than the error back propagation algorithm.

Zhang and Beneveniste proposed wavelet neural networks as an alternative to traditional neural networks for nonlinear system identification [5]. In these wavelet neural networks (WNNs), the wavelets may have different scale and shift parameters. They showed that WNNs can approximate any nonlinear function, as can neural networks with other basis functions. They also showed that, for a given nonlinear system and desired approximation quality, WNNs can have fewer nodes than other artificial neural networks. Hence, WNNs are often a better choice for online identification and control of nonlinear systems.

Training neural networks is a nonlinear optimization problem that can be solved using input-output measurements of the modeled system. The optimization is usually minimizing a cost function of the modeling error. Therefore, after measuring the input-output pairs, a suitable optimization approach must be chosen to optimize the cost function and find the weights of the neural network.

It is of fundamental importance to avoid local minima of the cost function, which are a major problem for Newton-based methods, and prove the stability of the training approach. Broadly speaking, there are two approaches to avoid local minima:

a) Solving optimization problems with different initial values.
b) Using global optimization approaches.

Multiple initial values do not guarantee finding the global minimum. In contrast, global optimization approaches for training neural network eliminate the local minimum issue of Newton-based methods. However, the stability of the new training method must be guaranteed. In most real-world applications, measurements include noise and uncertainties. Measurement noise can affect the stability of training method and make it unstable, therefore, training must remain stable under measurement noise. Hence, to propose the new training approach three steps are needed:

1) Adapt the optimization approach to solve neural network training problem.
2) Prove the stability of proposed method.
3) Prove the stability of method under presence of measurement noise

In many system identification applications, the system is not purely nonlinear and exhibits a combination of linear and nonlinear behavior. Identification of linear model using nonlinear approach is not efficient and leads to increased training time and model complexity. Parallel implementation of linear and nonlinear system identification approach, can overcome the increased model complexity and reduce the training time significantly in such applications.

Researchers have proposed a wide variety of schemes for training neural networks [6]. Classic methods for training neural networks use gradient based algorithms to solve this optimization problem. Rumelhart proposed using the steepest descent algorithm to minimize the error function. The method is known as error backpropagation [7].

Although error backpropagation is easy to implement, it is not an efficient and reliable algorithm. Error backpropagation can get stuck in a local minimum with poor performance [8]. Researchers have proposed several approaches, including supervised learning, to overcome the local minima issue of the error backpropagation algorithm [8],[9],[10].

The performance of the error backpropagation algorithm depends on the initial choice of network parameters. However, there is no standard theoretical approach to choose appropriate initial values. To improve the performance, reliability and convergence rate of error backpropagation, researchers have used various methods including an adaptive learning rate, adaptation of momentum term, etc. Simulation results show that an adaptive learning rate makes error backpropagation more efficient and more robust [11], [12].

To overcome the deficiencies of gradient based methods, supervised learning algorithms were introduced to train and learn the internal structure of neural networks [13],[14],[15]. Moller

proposed the scaled conjugate gradient (SCG) algorithm for fast supervised learning [15],[16]. He demonstrated that SCG outperforms conjugate gradient algorithms and standard error backpropagation and does not have critical user dependent parameters [16]. Conjugate gradient methods are easy to implement. Their convergence is faster than error backpropagation, hence they are more suitable for large scale problems [17].

Quasi-Newton methods are an alternative to conjugate gradient methods and are more sophisticated. Quasi newton methods rely on an approximation of Hessian matrix. The approximation can be close to singular in some cases making the results inaccurate and unreliable [18],[19].

To overcome the local minima and overfitting issues that reduce the efficiency and generalization capability of the neural network, global optimization techniques have been used to train the networks [20],[21]. Global optimization techniques can be divided into two major approaches, (i) deterministic methods such as cutting plane methods, branch and bound methods, interval methods etc., and (ii) stochastic approaches such as genetic algorithms, evolutionary programming, simulated annealing etc [22], [23], [24]. Researchers have used both deterministic and stochastic global optimization techniques for training neural networks [25], [26], [27]. Hybrid approaches are the combination of deterministic and stochastic optimization approaches. Hybrid approaches have advantages of deterministic and stochastic approaches and have been successfully applied to neural networks training problem [28], [29].

Although a wide variety of local, global and hybrid optimization approaches have been used for neural network training, there are promising optimization approaches in the mathematics literature that have not been exploited for this optimization problem. Quotient gradient system (QGS) is a trajectory-based method to find feasible solutions of constraint satisfaction problems. QGS searches for the feasible solutions of the CSP along the trajectories of a nonlinear dynamical system [30]. Dynamical Trajectory based methodology is another approach which uses trajectories of two nonlinear dynamical systems to find connected components of feasible region and search the components for local minima of the optimization problem [31].

After obtaining neural network models, control strategies must be used to obtain the desired time response for a nonlinear system. Model predictive control (MPC) is a well-known control approach where the controller predicts future control inputs using its knowledge of the plant dynamics and

the desired output [32]. The prediction is typically based on minimizing a performance measure such as the error or the control effort [33]. Findeisen and Allgöwer noted that any approach that deals with nonlinear systems requires the use of a suitable adaptive strategy to ensure satisfactory performance [34]. In many MPC applications, the mathematical model of the system cannot be reliably determined. Even in situations where mathematical models are available, there are conditions under which model parameters change or unmodeled dynamics are excited rendering controllers based on the nominal dynamics ineffective. This makes it difficult to apply traditional MPC approaches and makes model-free methods an attractive alternative [35].

Gu and Hu used a WNN for the identification and predictive control of nonlinear systems [36]. Their WNN avoids convergence to a local minimum, which commonly occurs with gradient descent algorithms. They observed that WNNs simplify determining the number of hidden layer nodes and initializing the weights. Sousa et al. used a WNN for robot model identification and control and compared their results with those of standard feedforward neural network with back propagation. They proved the stability of the closed-loop control system using the second method of Lyapunov [37]. Yoo et al. used a self-recurrent wavelet neural network with adaptive learning rate as a model identifier [38]. They used their model to design a generalized predictive controller for nonlinear systems and proved its closed-loop stability by the Lyapunov method.

In many control applications, communications between sensor and controller and between controller and actuator are through a communication network. These networked control systems are very common in industrial control applications. The network can be private or can be shared between several controllers and plants. Using a shared network for several plants and controllers results in lower installation costs. In some control applications, the information from one plant is necessary to control other plants. Hence, using a shared network can lead to a flexible control architecture [39].

Although the idea of using a shared network can result in a flexible control architecture, it can cause complications in the design because of network delays and data dropouts. Networked induced delays are one of the most important effects of the networked control approach. The delays are small when the network is private but increase as the traffic of the network increases. Because network induced delays can be large enough to make the control system unstable, design methodologies for NCS must compensate for these delays. A common and straightforward means

of coping with delays is to predict delayed or missing information from plant to controller or from controller to actuator [39], [40].

Li and Liu used a simplified generalized predictive controller to control a second order system over the network [40]. Xue and Liu trained a neural network to calculate the predictive control value and predictive feedback to control a system over a network with fixed network delay [41].

Jinhui and Yuanqing used an output feedback strategy to calculate the control input packets and solved the controller design problem using Markovian jump system theory. They showed that the packet based approach can cope with the network delay as well as packet loss issue [42].

Bianchi et al. used MPC to control traffic over a communication channel with network delay and packet loss. They used a buffer to compensate for the network delay and showed that the performance of their controller is close to the performance of an ideal controller [43]. Yang et al. proposed a predictive output feedback scheme for networked control systems [44]. They presented a new technique to compensate for the network delay in discrete time models and formulated the system as a Markovian jump system.

We use a wavelet neural network with feedforward component to identify the models of highly nonlinear systems [137], [244]. The feedforward component reduces the number of hidden layer nodes and reduces the training time of the neural network making it suited to online identification and control applications. Khodabandehlou et al. used wavelet neural network with feedforward component and model predictive controller to control the seismically isolated structure during earthquake [244]. Their simulation results show that wavelet neural network based predictive controller can effectively control the structure during near fault and far field motions. Khodabandehlou and Fadali used the wavelet neural network with feedforward component and model predictive controller with extended prediction horizon to networked control of an autonomous vehicle and proved the stability of the controller using Lyapunov stability theory [61]. Their simulation results shows that model predictive controller with extended prediction horizon can compensate for the effects of network delay and packet loss and achieve a satisfactory tracking performance.

1.2 Thesis Organization

Most real world applications are not purely nonlinear systems and show the combination of linear and nonlinear behavior. We use parallel implementation of linear and nonlinear modeling to identify the model of a system that demonstrates some linear behavior. The first step uses a linear model to model the input-output data of the system and the residual error is calculated. Then a wavelet neural network or echo state network is used to map the input data to residual error. The resulting network is used for nonlinear system identification application and a model predictive controller is designed based on the wavelet neural network model. To address the critical issue of the stability of the model and controller, Lyapunov theory is used.

Chapter 2 examines two popular neural networks that have been successfully utilized in a wide variety of applications: echo state networks (ESN) and wavelet neural networks (WNN). It introduces innovations in the structure of ESN that result in major improvements in their performance. By adding a feedforward component to the networks, and by tuning their weights using recursive least squares, we can drastically reduce the number of hidden layer neurons while reducing their error. We demonstrate the improvement in the performance of ESN and compare their performance to that of WNN by application to the identification of an un-manned vehicle. Different ESN and WNN are utilized to identify the model of the vehicle and examine the effects of adding a feedforward term to ESN and WNN. Our results show that, even though both networks yield acceptable performance, ESN outperform WNN in terms of accuracy but have a slightly higher computational cost.

Chapter 3 presents an online identification and control scheme for a 5-story benchmark structure during earthquake. A WNN is used to identify the structural system and the model predictive controller uses a WNN to predict the future outputs of the system. The WNN and the model predictive controller are trained by the gradient descent algorithm to minimize performance indices. The traditional recursive least squares algorithm trains the feedforward component. Due to the general structure of the controller and efficiency of the WNN with feedforward component, the controller performance is satisfactory even under the strict condition of fixed learning rate. Simulation results demonstrates the efficiency of the proposed control scheme.

Chapter 4 proposes networked identification and control of unmanned vehicle using wavelet neural network and model predictive controller. The controller uses extended prediction horizon to reduce the effect of network delay and packet loss on the tracking performance of the controller.

The model predictive controller uses the wavelet neural network model to predict the future outputs of the system over an extended prediction horizon and calculates the optimal future inputs by minimizing a controller cost function using gradient descent algorithm. We apply the proposed methodology to the online identification and control of an unmanned autonomous vehicle over a communication network with fixed and random delays and packet loss. Simulation results show that the controller with extended prediction horizon can effectively control the system in the presence of fixed or random network delay and packet loss. Lyapunov theory is used to prove the stability of the wavelet neural network with feedforward component and stability of the model predictive controller.

Chapter 5 introduces a new approach to train a fully recurrent artificial neural network by solving a constraint satisfaction problem using the quotient gradient method. The quotient gradient method is a trajectory-based methodology for global optimization that does not suffer from the problem of local minima encountered in Newton based methods. Simulation results show that the network trained with the quotient gradient method perform better than traditional error backpropagation and genetic algorithm. The method is also easier to implement in comparison to other global optimization techniques such as genetic algorithms. The stability of the proposed methodology and its stability in presence of measurement error is proved using Lyapunov theory.

Chapter 6 proposes a new method to train recurrent neural networks using dynamical trajectory based optimization. The optimization method utilizes a projected gradient system (PGS) and a quotient gradient system (QGS) to determine the feasible regions of an optimization problem and search the feasible regions for local minima. By exploring the feasible regions, local minima are identified and the local minimum with the lowest cost is chosen as the global minimum of the optimization problem. Lyapunov theory is used to prove the stability of the local minima and their stability in the presence of measurement errors. Numerical examples show that the new approach provides better results than Newton-based methods

Chapter 7 applies the quotient gradient system to train a two-stage partially recurrent neural network for anomaly classification in power networks. The classification of anomalies or sudden changes in power networks versus normal abrupt changes or switching actions is essential to take appropriate maintenance actions that guarantee the quality of power delivery. This issue has increased in importance and has become more complicated with the proliferation of volatile resources that introduce variability, uncertainty, and intermittency in circuit behavior that can be

observed as variations in voltage and current phasors. This makes diagnostics applications more challenging. This study proposes using quotient gradient system (QGS) to train two-stage partially recurrent neural network to improve anomaly classification rate in power distribution networks using high-fidelity data from micro-phasor measurement units (μPMUs). QGS is a systematic approach to finding solutions of constraint satisfaction problems. We transform the μPMUs data from the power network into a constraint satisfaction problem and use QGS to train a neural network by solving the resulting optimization problem. Simulation results show that the proposed supervised classification method reliably distinguishes between different anomalies in power distribution networks. Comparison with other neural network classifiers shows that QGS trained networks provide significantly better classification. Sensitivity analysis is performed concerning the number of μPMUs, reporting rates, noise level and early versus late data stream fusion frameworks.

Chapter 2. Echo State versus Wavelet Neural Networks: Comparison and Application to Nonlinear System Identification

2.1 Introduction

Artificial neural networks have been proven to be general function approximators. Cybenko and Hornik demonstrated that any nonlinear function could be approximated with a three layer artificial neural network with desired accuracy [1],[2].However, the multilayer structure of neural networks increases their learning time and computational burden. Steck et al. showed that parallel implementation of feedforward neural network and recursive least squares can decrease the learning time significantly while reducing the number of nodes in the hidden layer [4].

In this chapter, we examine the performance to two classes of neural networks: echo state networks that are based on reservoir computing, and wavelet networks that use wavelets as their activation function. In particular, we are interested in the use of these network for nonlinear system identification.

Reservoir computing (RC) is an increasingly popular approach for processing time series [45]. In RC models, the input signal is applied to a reservoir. The reservoir is a random dynamical system with recurrent states. The reservoir maps the input to a higher dimensional space, then the states of the reservoir are mapped to the desired output. The weights from input to reservoir are fixed and the training is only required for the mapping from the states of the reservoir to the outputs [46]. Echo state networks (ESN) are among the simplest forms of the reservoir reservoir computing models. ESN are recurrent neural networks whose to-output weights are efficiently trained using recursive least squares [45], [47].

ESN have been successfully used in modeling, guidance, control, and other applications. Jaeger and Haas used ESN to obtain a model for a nonlinear chaotic communication system. They showed that the accuracy of predicting a benchmark chaotic time series is improved by a factor of 2400 over other techniques [48].

Bush and Anderson used ESN to approximate the Q-function of a mass-spring-damper system, and showed that ESN can approximate the Q-function as well as a feedforward neural network.

They argued that ESN is superior to recurrent back propagation neural networks for temporal feature generation and mapping [49].

Jaeger proposed a nonlinear system identification approach based on echo state networks. He used recursive least squares to train the network weights and applied his approach to a tenth order NARMA system to show its effectiveness [50].

Salmen et al. used an ESN for motor control application. Their simulation and physical implementation results showed that the ESN controller is superior to PID controllers in terms of reducing oscillations around the desired velocity, squared control error, and time to reach the desired velocity [51]. Hartland et al. used ESN for robot navigation behavior acquisition. They showed that ESN based control provides better performance than naive non-recurrent perceptron's [46]. Tong et al. applied ESN in a grammatical structure learning problem. They showed that ESN can perform word prediction better than bigrams and trigrams [52]. Cernansky et al. proposed a simplified ESN that can model the system as well as standard ESN. Their simulation results showed that, at least in some tasks, a simple reservoir can achieve comparable results to standard ESN [53].

Rodan and Tino proposed a minimum complexity echo state network for the identification of nonlinear models. They showed that a simple deterministic reservoir can perform as well as a standard echo state network and that the short term memory capacity of the simplified reservoir can be close to optimal. They supported their claim by applying minimum complexity ESN to several nonlinear system identification benchmarks [54].

In recent years, other neural network models have been proposed for system identification. Rosa et al. proposed using deep learning with randomized algorithms for training neural model for nonlinear system identification [55], [56],[57]. In their approach, input data is used to decide the hidden layers and hidden weights and randomized algorithms are used to decide the output weights. Simulation results show that deep learning with randomized algorithm has superior performance to classical approaches such as gradient descent algorithm. Another approach is kernel methods, such as support vector regression and regularization networks. They construct a linear combination of kernel functions to identify the nonlinear model [58].

Kandroodi et al. applied artificial neural networks to the identification and predictive control of a stirred tank reactor. They applied a multilayer perceptron, a radial basis function network and an

Adaptive Neuro-Fuzzy Inference System (ANFIS) for system identification and neural network predictive control of a stirred tank reactor [59].

Zhang and Benveniste proposed wavelet networks as an alternative to standard back-propagation neural networks for system identification. They demonstrated that wavelet neural networks are general function approximator [5].

Ghadirian et al. proposed a fully recurrent wavelet neural network for identification of dynamic multi-input-multi-output (MIMO) nonlinear systems. They showed that their fully recurrent wavelet neural network has less error and lower convergence time in comparison to other fully recurrent neural networks [60].

Khodabandehlou and Fadali used wavelet neural network to identify the model of an autonomous vehicle and designed a generalized predictive controller based on the wavelet neural network model. Their simulation results show that wavelet networks can effectively identify the model of a nonlinear system in presence of measurement noise, network delay and packet loss [61].

In this study we propose a modification to and compare the ESN and the wavelet neural network. We add a feedforward parallel component to each networks that enhances their modeling capabilities. We explore the performance of the networks in identification of a nonlinear MIMO system. Our simulation results show that the added feedforward term can reduce the number of hidden nodes in wavelet neural network and reduce the dimension of the reservoir in ESN significantly. This increased efficiency is achieved while also reducing the identification errors in ESN.

2.2 Echo State Networks

ESN are recurrent neural networks that can provide an excellent representation for nonlinear dynamics. The architecture of the network is shown in Figure 1. The network has an input layer, an output layer and a middle or internal layer known as the reservoir. At time k, the network maps an input vector $\underline{\boldsymbol{u}}(k) = [u_1(k), u_2(k), \dots u_N(k)]^T$ to an internal state vector $\underline{\boldsymbol{x}}(k) = [x_1(k), x_2(k), \dots, x_M(k)]^T$ with the nonlinear dynamics

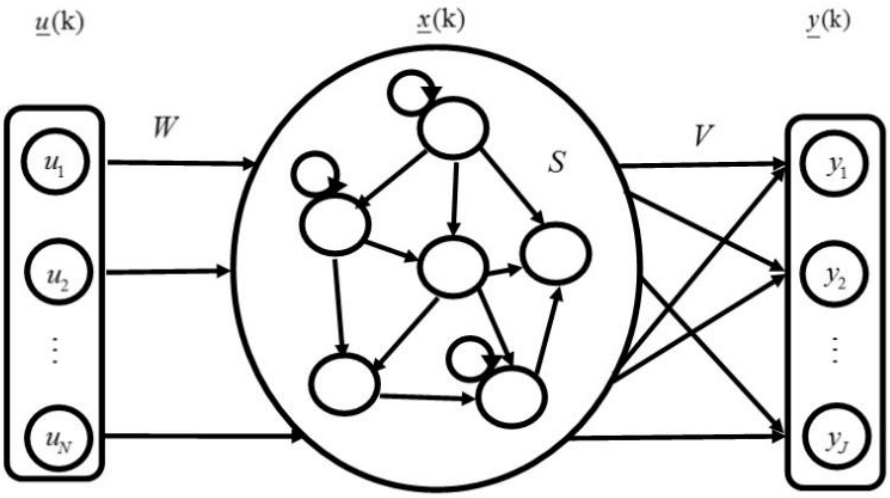

Figure 1. Echo State Network architecture

$$\underline{\boldsymbol{x}}(k) = f(W\underline{\boldsymbol{u}}(k) + S\underline{\boldsymbol{x}}(k-1)) \tag{2.1}$$

with input weight matrix $W = \left[w_{i,j}\right]_{M\times N}$ and state weight matrix $S = \left[s_{i,j}\right]_{M\times M}$. f is the reservoir activation function which is usually a sigmoidal function. W is fixed with uniformly distributed random values. W is typically scaled as $W \leftarrow \iota W/|\lambda_{max}|$ where $|\lambda_{max}|$ is spectral radius of W and $0 < \iota < 1$ is a scaling factor [45]. The output of the network is given by

$$\underline{\boldsymbol{y}}(k) = V\underline{\boldsymbol{x}}(k) \tag{2.2}$$

where $\underline{\boldsymbol{y}}(k) = \left[y_1(k), y_2(k), \ldots, y_J(k)\right]^T$ and $V = \left[v_{i,j}\right]_{J\times M}$ is the output weight matrix. The output weights of the ESN can be trained online with the recursive least squares algorithm as well as least squares for offline training. If the state weight matrix S of the internal units is a full matrix, then the reservoir is fully connected and recurrent. In this study we use a standard reservoir, delay line reservoir with feedback connection and simple cycle reservoir. Figure 2 and Figure 3 depict the structure of the delay line with feedback connection (DLRF) and simple cycle reservoirs (SCR). In DLRF, the internal units are arranged in cascade with connection weight r and each unit is has a feedback connection of weight b. Internal SCR units are also arranged in cascade with connection weight r but there is a single feedback loop from the output of the last unit to the first unit with weight r [54].

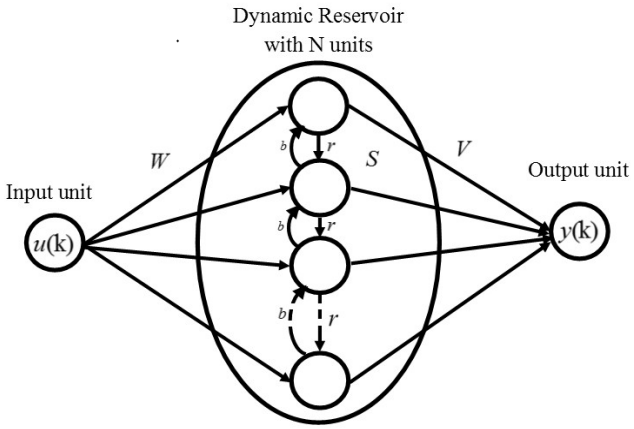

Figure 2. Delay line reservoir with feedback connections (DLRF)

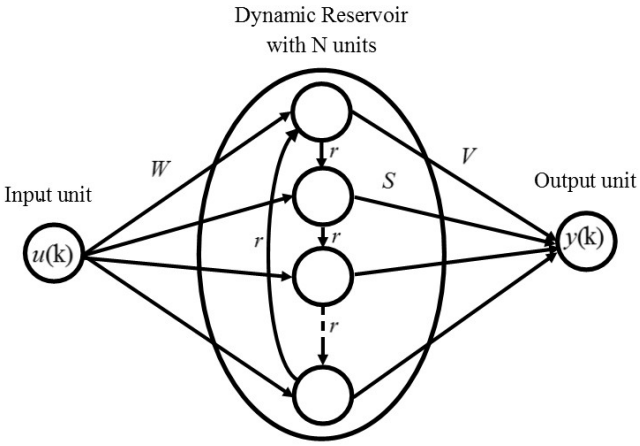

Figure 3. Simple cycle reservoir (SCR)

2.3 ESN with Feedforward Term

ESN have a simple learning algorithm with fixed input layer and middle layer weights that do not need training. However, the size of reservoir is often a significant disadvantage of ESN. This shortcoming can be mitigated by using recursive learning approaches such as recursive least squares. In dealing with complex systems, the size of the reservoir may increase drastically. To cope with this issue, we add a feedforward term to ESN. The internal unit update equation will be the same as (1) but the output equation becomes

$$\underline{\boldsymbol{y}}(k) = V\underline{\boldsymbol{x}}(k) + Q\underline{\boldsymbol{u}}(k) = [V \vdots Q]\begin{bmatrix}\underline{\boldsymbol{x}}(k)\\ \underline{\boldsymbol{u}}(k)\end{bmatrix} \tag{2.3}$$

where Q and V are state and input gain matrices respectively. For online training, first we tune the Q matrix using recursive least squares as in the standard ESN. To tune the matrix V, we consider the residual

$$\underline{e_r}(k) = \underline{y}(k) - Q\underline{u}(k) = V\underline{x}(k) \tag{2.4}$$

2.4 Wavelet Neural Network

Wavelet networks are back-propagation networks that use wavelets as the activation function in the hidden layer. Because wavelets have two extra parameters that scale and shift their inputs, wavelet networks have more flexibility than feedforward neural networks with sigmoidal activation functions. Zhang and Benveniste showed that wavelet networks are general function approximators that often have fewer nodes in comparison to neural networks with other basis functions [5]. Figure 4 shows the wavelet network architecture. The input output equation of the network is

$$\underline{\boldsymbol{y}} = S\Psi\left(A^{-1}\left(W\underline{\boldsymbol{u}} - \underline{\boldsymbol{b}}\right)\right) \tag{2.5}$$

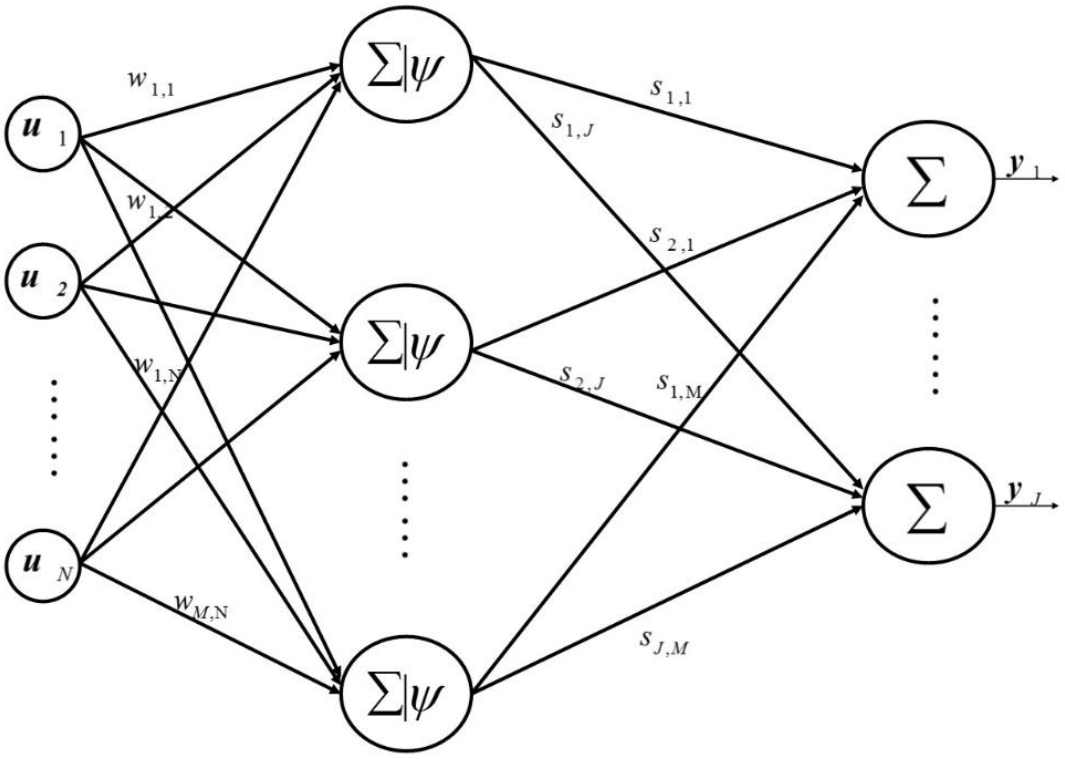

Figure 4. Wavelet Neural Network

W is the input weight matrix, S is the output layer weight matrix, A is a diagonal matrix whose diagonal elements are wavelet scale parameters, and $\underline{\boldsymbol{b}}$ is the vector of the shift parameters. Ψ is the activation function. $\boldsymbol{u}$ is the $N \times 1$ network input vector and $\boldsymbol{y}$ is the $J \times 1$ network output vector. With M hidden layer nodes, S, W, A and $\underline{\boldsymbol{b}}$ are $J \times M$, $M \times N$, $M \times M$ and $M \times 1$ matrices respectively. The activation function of each node is of the form

$$\psi_{a,b} = \psi\left(\frac{t - b_i}{a_{ii}}\right) \ ,1 \leq i \leq M \tag{2.6}$$

where ψ is chosen as the Mexican hat wavelet

$$\psi(t) = \frac{2\pi^{\frac{1}{4}}}{\sqrt{3}}(1 - t^2)e^{-\frac{t^2}{2}} \tag{2.7}$$

All the network parameters are trained with the error back propagation algorithm. The cost function for training the network parameters is the sum of square errors:

$$J = \frac{1}{2}\sum_{k=1}^{K}\left(\underline{\hat{\boldsymbol{y}}}(k) - \underline{\boldsymbol{y}}(k)\right)^T(\underline{\hat{\boldsymbol{y}}}(k) - \underline{\boldsymbol{y}}(k)) \tag{2.8}$$

where $\hat{\boldsymbol{y}}(k)$ is the network output vector and $\boldsymbol{y}(k)$ is the measured output at time step k. Every network parameter is updated by the gradient descent algorithm

$$o_{k+1} = o_k - \gamma \nabla_o(J) \tag{2.9}$$

$\gamma \in (0,1]$ is the learning rate for gradient descent that specifies the convergence speed of the algorithm. We define the partial error as

$$e_{yj}(k) = \hat{y}_j(k) - y_j(k) \tag{2.10}$$

where $\hat{y}_j(k)$ is the j^{th} measured output and $y_j(k)$ is the j^{th} output of the network at time step k, $u_{k,l}$ is the l^{th} input and $Q_{j,l}$ is (j, l) element of Q. The derivative of the cost function with respect to each network parameter is

$$\nabla_o(J) = \sum_{j=1}^{J} e_{yj}(k)\frac{\partial \hat{y}_j(k)}{\partial o} \tag{2.11}$$

2.5 Wavelet Neural Network with Feedforward Term

Figure 5 shows the structure of the WNN with feedforward term of gain Q. As in the case of ESN with RLS, we first establish a linear regression between the network inputs and outputs with the RLS algorithm and the partial error defined as

$$\underline{\boldsymbol{e_r}}(k) = \underline{\boldsymbol{y}}(k) - Q\underline{\boldsymbol{u}}(k) \tag{2.12}$$

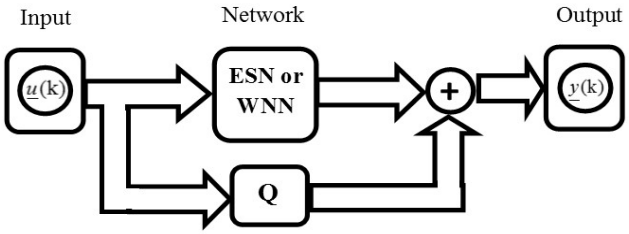

Figure 5. ESN/WNN with feedforward term

WNN are trained to learn the behavior of the partial error. The computational complexity of WNN and ESN is proportional to the number of hidden layer nodes/reservoir internal units squared, i.e. $O(M^2)$. The computational complexity of the RLS algorithm is proportional to number of measurements squared, i.e. $O(N^2)$. Adding the feedforward term changes the number of internal units from M to M_{new} Therefore, adding the feedforward component to both types of networks changes the computational complexity of the problem from $O(M^2)$ to $O(M_{new}^2) + O(N^2)$. For the ESN network we have $M_{new} \ll M$ and for the WNN number of hidden nodes decreases significantly. Assuming $M_{new} = cM, c \leq 1$, the ratio of the computational complexity of the network with feedforward term to the computational complexity of the standard network is $c^2 + N^2/M^2$. Thus, when the number of inputs is much smaller than the number of internal units, the computational complexity decreases by the ratio c^2. Figure 6 shows the change in computational complexity with c and $d = N/M$. From Figure 6, it can be concluded that if the number of inputs is close to the number of hidden layer nodes, adding a feedforward term to standard network may increase the computational burden. However, if the number of inputs is much lower than the number of internal units and the system is not highly nonlinear, adding a feedforward term to the ESN or WNN can significantly reduce the computational burden

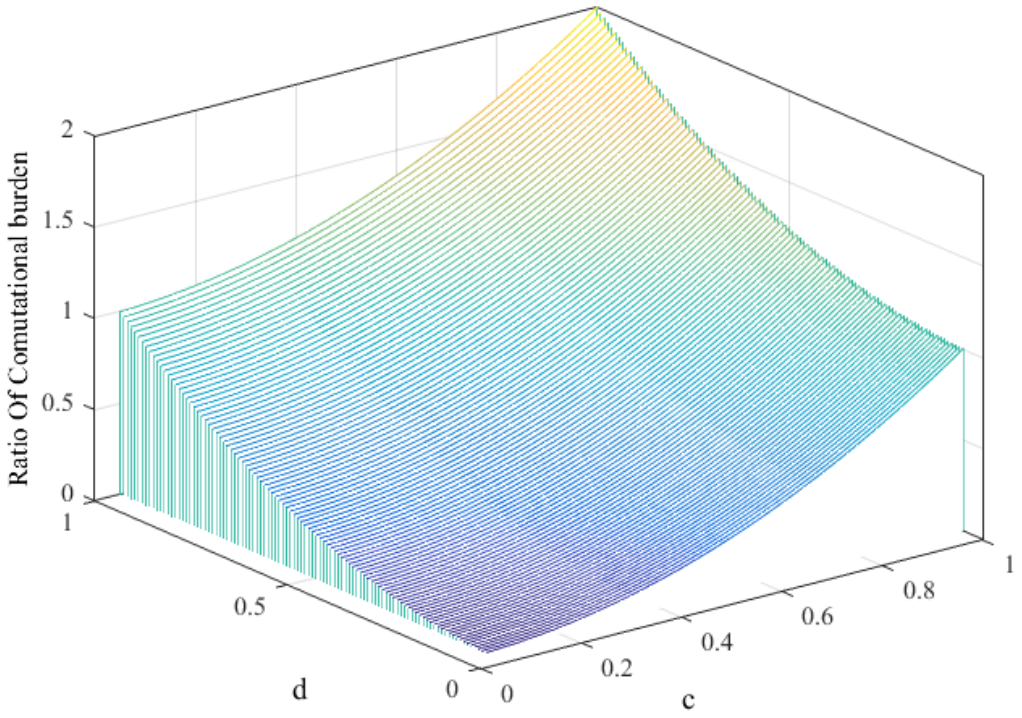

Figure 6. Changes in the computational load

2.6 Simulation Results

2.6.1 Example 1: Unmanned Vehicle

As our first example, we use the simplified nonlinear vehicle model of Figure 7 chosen from [36].The model has three state variables and two control inputs. The control inputs are the vehicle speed v, and the steering angle α, and the model is completely controllable with the two inputs. The state equation of the system is

$$\underline{\boldsymbol{x}}(k+1) = \begin{bmatrix} x(k+1) \\ y(k+1) \\ \theta(k+1) \end{bmatrix} = \begin{bmatrix} x(k) + Tv(k)\cos(\theta(k))\cos(\alpha(k)) \\ y(k) + Tv(k)\sin(\theta(k))\cos(\alpha(k)) \\ \theta(k) + Tv(k)\sin(\alpha(k))/D \end{bmatrix} \tag{2.13}$$

where T is sampling period and D is the distance between the two shafts of the vehicle. We assume D to be 3 meters, which is reasonable for a midsize car, and use a sampling period of 1 ms. To explore the effect of adding a feedforward term to the ESN and the wavelet neural network, we

use both networks to identify the model of un-manned vehicle. In both cases the input to the network is $\underline{\boldsymbol{u}}(k) = [v(k), \alpha(k), x(k), y(k), \theta(k)]^T$.

All the WNN matrices are initialized with uniformly distributed random numbers except for A which is initially the identity matrix. The learning rate for the gradient descent algorithm is $\gamma = 0.1$. For ESN, we chose $b = 0.7$ and $r = 0.1$. The scaling factor for a standard reservoir is $\iota = 0.9$. The measurement noise is zero-mean white Gaussian noise with covariance 0.01.

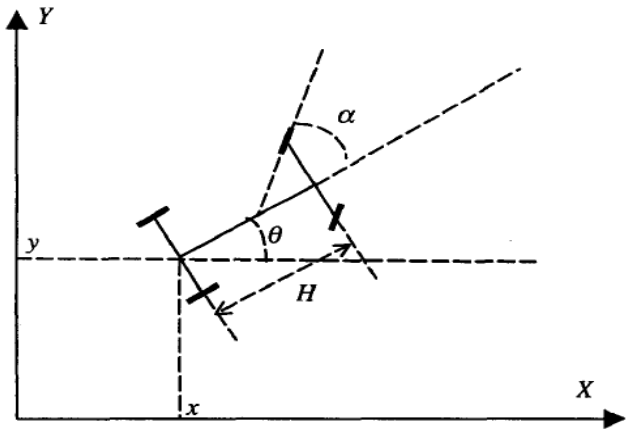

Figure 7. Autonomous Vehicle [36]

Table 1 summarizes the simulation results for the identification of the unmanned vehicle model. In the table, SESN stands for ESN with standard reservoir and FF-T stands for Feedforward Term. Due to the fixed structure of the reservoir and the untrainable input layer of the ESN, the computational complexity and execution time of the ESN with different reservoirs are almost the same. The sum of square error (SSE) of the ESN with feedforward term is lower than the SSE standard ESN, DLRB and simple cycle reservoir. The SSE of the WNN with a feedforward term is higher than the SSE of the standard WNN. The SSE of the WNN with feedforward term can be reduced by increasing the number of nodes in the hidden layer at the cost of increased computational time.

Table 1. Identification of unmanned vehicle

Network Type	Simulation results			
	Computational Cost	*M*	*N*	*SSE*
WNN	$O(M^2)$	60	5	0.5246

Network Type	Simulation results			
	Computational Cost	*M*	*N*	*SSE*
WNN with FF-T	$O(M^2) + O(N^2)$	10	5	2.1237
SESN	$O(M^2)$	100	5	2.8801
SESN with FF-T	$O(M^2) + O(N^2)$	10	5	1.8130
DLRF	$O(M^2)$	100	5	3.1955
DLRF with FF-T	$O(M^2) + O(N^2)$	10	5	1.8446
SCR	$O(M^2)$	100	5	2.1135
SCR with FF-T	$O(M^2) + O(N^2)$	10	5	1.8230

2.6.2 Example 2: Nonlinear MIMO System

For our second example, we use the two-input-two-output nonlinear MIMO system [59]

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 0.5\frac{x_1(k)}{1+x_2(k)^2} \\ 0.5\frac{x_1(k)x_2(k)}{1+x_2(k)^2} \end{bmatrix} + \begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix} \tag{2.14}$$

The network input includes the current and past states, as well as the control input and is given by $\boldsymbol{u}(k) = [u_1(k), u_2(k), x_1(k), x_2(k), x_1(k-1), x_2(k-1)]^T$. All the network parameters are initialized as in the unmanned vehicle example and the measurement noise is assumed to be zero-mean white Gaussian noise with a variance of 0.02. Table 2 summarizes the results of the identification of the nonlinear MIMO model. As in the unmanned vehicle example, adding a feedforward term has a smaller effect on the computational load for WNN than for ESN. For all networks the number of hidden units, and consequently the computational load, is reduced. The reduction is not as drastic as in Example 1. The feedforward term causes an increase in the SSE for WNN and a decrease for ESN. Although the changes are less than those observed in Example 1, they are still significant.

Table 2. Identification of MIMO system

Network Type	Simulation results			
	Computational Cost	*M*	*N*	*SSE*
WNN	$O(M^2)$	35	6	0.9327
WNN with FF-T	$O(M^2) + O(N^2)$	10	6	2. 5733
SESN	$O(M^2)$	60	6	2.8497
SESN with FF-T	$O(M^2) + O(N^2)$	20	6	2.1593
DLRF	$O(M^2)$	60	6	3.8754
DLRF with FF-T	$O(M^2) + O(N^2)$	20	6	2.8398
SCR	$O(M^2)$	60	6	2.9503
SCR with FF-T	$O(M^2) + O(N^2)$	20	6	1.8879

2.7 Conclusion

This study compares the performance of enhanced ESN and WNN. It proposes adding a feedforward term to ESN and to WNN and examines the effect of this modification on the performance of the two networks. In particular, we examine the performance of the two networks for nonlinear system identification. For WNN, the feedforward term reduces the number of hidden layer nodes but at the cost of an increase in SSE. For ESN with standard, DLRF and simple cycle reservoirs, the SSE decreases while the size of reservoir decreases drastically, which leads to a much lower computational burden. Our results show that adding a feedforward term to ESN offers significant advantages. This can lead to wider applicability of ESN in online system identification and control applications in which computational time is a major issue. The results also suggest that the reduction in the computational complexity of the ESN and WNN depends on the nonlinear dynamics with less reduction for more severe nonlinearities. Future work will explore convergence conditions for ESN with a feedforward term and the applicability of ESN to online identification and control of unmanned aerial vehicles

Chapter 3. Active Neural Predictive Control of Seismically Isolated Structures

3.1 Introduction

Seismic isolation is recognized as one of the most effective mitigation strategies for structural and nonstructural systems [62],[63]. In general, seismic isolation systems shift the fundamental period of the isolated structure away from the range of predominant excitation periods, which results in reduced acceleration demand. However, as a result of the lengthened period, the base displacement of the isolated system becomes larger, and in some cases may lead to instability of the isolation system. This issue arises in case of near-field ground motions, which are dominated by long velocity pulses. For the case of long period structures, the velocity pulse tends to develop large displacements [64]. The pulse displacement is usually associated with the fault-normal direction, where high spectral acceleration components are observed in the long period range [65]. These long period spectral acceleration components tend to resonate with conventionally isolated structures, leading to an excessive base displacement that may destabilize the structure. To maintain the isolation deformations to within acceptable limits, two potential alternatives are to increase the effective stiffness of the isolation system or to provide additional damping mechanisms. However, increasing the stiffness is counterintuitive. For this reason, many of the isolation systems provide various sources of damping mechanism to overcome the unacceptably large isolation deformations [66],[67]. This damping may be provided by the isolator directly or by means of supplemental damping devices. However, increasing the damping capacity of the isolated structures by means of supplementary dampers leads to an increase in the superstructure's accelerations and inter-story drifts [68]-[70]. In such circumstances, conventional seismic isolation systems alone, such as rubber bearings and friction pendulum, may not be the best alternative for seismic response mitigation.

Various alternative hybrid strategies have been proposed and studied analytically and experimentally to overcome the shortcomings of the conventional seismic isolation system (e.g. large isolator deformations). Some of these strategies combine conventional isolation system with actively or semi-actively controllable devices [71],…,[81]. Furthermore, control strategies using adaptive neural networks and their application in various types of structural systems, including seismically isolated structures, have been extensively studied during the last two decades [82],

….,[89] .While varying levels of response reduction have been demonstrated by means of active control strategies, there is still a need to develop efficient, consistent and robust techniques to address the issues stated earlier.

Despite the availability of various adaptive control methods, their application requires an accurate mathematical model of the system [34]. However, model uncertainty or unmodeled dynamics in real-world applications can cause a deterioration in the performance of model-based control systems. This makes model-free methods more attractive. Artificial neural networks can accurately approximate nonlinear systems and have therefore received considerable attention in the identification of nonlinear systems. Han et al. designed a generalized predictive controller for a second order nonlinear system based on a feedforward neural network identifier. Their simulation results show that neural network control is robust and can meet the design specifications [90].

The main drawbacks of neural networks are their slow convergence rate, multi-layer structure, and computational complexity. Zhang and Beneveniste proposed wavelet networks to eliminate or mitigate these disadvantages [5]. Wavelet networks use wavelets with different scale and shift parameters as activation functions and are thus a good alternative to neural networks for identification of nonlinear systems. They also argued that in higher dimensional problems, wavelet networks typically have fewer hidden layer nodes than other neural networks.

The published research on the development and application of wavelet neural network controllers and system identification is abundant. Sousa et al. identified the model of a robot using a wavelet network and used the identified model to design a dynamic controller [37]. They proved the stability of their controller using the second method of Lyapunov. They argued that in spite of the difficulty of their design and stability analysis, wavelet network controllers provide more flexibility than standard adaptive control approaches. Zayeni and Ahmadi applied a radial wavelet network to the identification of nonlinear system [91]. The structure and learning method of their network are similar to those of radial basis function networks but their activation functions are wavelets. Simulation results show that these networks can identify models of complex nonlinear systems. Khodabandehlou and Fadali used wavelet network and generalized predictive controller for online identification and networked control of an unmanned vehicle [61]. Their simulation results show satisfactory performance under fixed and random network delay. For more

publications on the application of wavelet networks to system identification and control, interested readers are referred to [88], [92],…,[96].

This study proposes a new model predictive controller (MPC) that uses a wavelet neural network for output identification of nonlinear dynamics and uses it for MPC control of seismically isolated structures. The wavelet neural network includes a back propagation neural network and a feedforward term trained using recursive least squares. This configuration reduces the number of hidden layer nodes significantly. The computational complexity of the error backpropagation algorithm is $O(n^3)$ with *n* hidden layer nodes. Consequently, reducing *n* reduces the computational complexity of the wavelet neural network drastically and allows its utilization in online identification and control. This parallel configuration is, to the best of our knowledge new, and is one of the main contributions of this study. In addition, this study presents the first application of this new configuration to seismically isolated structures.The approach is applied computationally to the control of a seismically isolated five story structure subjected to ground motions with both far-field (FF) and near-field (NF) characteristics. Simulation results demonstrate that significant reductions of all response amplitudes were achieved particularly for near-field (pulse) ground motions, including reduced deformations along with corresponding reduction in acceleration response

3.2 Wavelet Neural Network

Model predictive control (MPC) relies on using an accurate system model to predict future outputs. In cases where such a model is not available, system identification becomes an essential part of model predictive control. In this study, a 3-layer wavelet neural network in parallel with a feedforward component trained using the recursive least squares (RLS) algorithm is used to identify the model parameters. The RLS algorithm establishes a linear relationship between the inputs and outputs and the wavelet neural network minimizes the identification error. Figure 8 depicts the structure of the wavelet neural network.

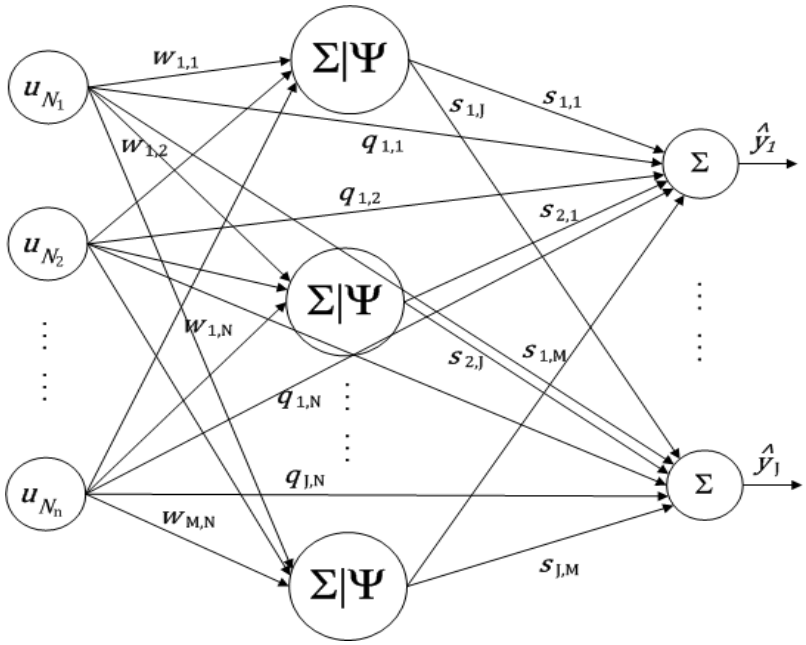

Figure 8. Wavelet Neural Network structure

The input-output equation of the network can be expressed as

$$\hat{y}_j(k) = \sum_{i=1}^{J} s_{j,i} \sum_{t=1}^{n} \psi(w_{i,t} \times \mathrm{u}_{\mathrm{N_t}} - p_i)/r_{i,i}) + \sum_{t=1}^{n} q_{j,t} \times \mathrm{u}_{\mathrm{N_t}} \ , 1 \le j \le J \tag{3.1}$$

$$\hat{\mathbf{y}}(k) = \left[\hat{y}_1(k), \hat{y}_2(k), \dots, \hat{y}_J(k)\right]^T \tag{3.2}$$

where $\hat{\mathbf{y}}$ is the network output and $\mathbf{u}_N = \left[\mathrm{u}_{N_t}\right]$ is the $n \times 1$ network input vector. Assuming that the network has J outputs and m hidden layer nodes, $\mathbf{S} = [s_{i,j}]$ is a $J \times m$ matrix, $\mathbf{W} = [w_{i,j}]$ is $m \times n$ matrix, $\mathbf{p} = [p_i]$ is $m \times 1$ vector, and $\mathbf{R} = [r_{i,j}]$ is a $m \times m$ diagonal matrix whose diagonal elements are positive, and off-diagonals are zero and $\mathbf{Q} = [q_{i,j}]$ is $J \times n$ matrix. The activation function of the hidden layer nodes,ψ, is

$$\psi_{p,r} = \psi\left(\frac{t-p}{r}\right) \tag{3.3}$$

The activation function is chosen to be the Mexican hat wavelet

$$\psi(t) = \frac{2\pi^{\frac{1}{4}}}{\sqrt{3}}(1-t^2)e^{-\frac{t^2}{2}} \tag{3.4}$$

The well-known error back propagation algorithm is used to train the wavelet neural network weights while **Q** is trained using the RLS algorithm. RLS learns the behavior of the system much

faster than the wavelet network and traditional neural networks. Consequently, the wavelet neural network learns the behavior of the system faster, which makes the model more appropriate for online identification and control. In addition, particularly when the system behaves linearly under certain operating conditions, the parallel configuration reduces the number of hidden layer nodes drastically, and consequently the training time of the WNN drops significantly. This makes the algorithm more efficient for real time applications. The reduction of number of hidden layer nodes reduces the model complexity for achieving certain desired approximation accuracy and does not sacrifice accuracy to reduce the model complexity.

The cost function for training the network parameter is assumed to be sum of squared errors

$$J(k) = \frac{1}{2}\sum_{k=1}^{N}\sum_{j=1}^{m}(\hat{y}_j(k) - y_j(k))^2 \tag{3.5}$$

where $y_j(k)$ is the j^{th} output of the system and $\hat{y}_j(k)$ is the j^{th} output of the wavelet neural network at time step k. The gradient descent method is used to update the network parameters

$$\sigma_{k+1} = \sigma_k - \gamma\frac{\partial J(k)}{\partial \sigma_k} \tag{3.6}$$

where γ is the gradient descent algorithm learning rate. By defining the partial error as

$$e_{yj}(k) = \hat{y}_j(k) - y_j(k) - \sum_{t=1}^{n} q_{j,t}\mathrm{u}_{N_t} \tag{3.7}$$

The change in each parameter can be calculated using the chain rule as [5][61].

$$\frac{\partial J}{\partial \sigma_k} = \sum_{j=1}^{J} e_{yj}(k)\frac{\partial \hat{y}_j(k)}{\partial \sigma_k} \tag{3.8}$$

3.3 Wavelet Neural Network based MPC

Predictive control uses an explicit model of the system to predict future outputs of the system and uses them to calculate future inputs. It assumes that the inputs and outputs of the system can be measured and used to update the network parameters. In each iteration, the controller updates the network parameters and uses the wavelet neural network to predict future outputs over a prediction horizon N_p, then calculates the future inputs over a control horizon $N_u \leq N_p$ by minimizing the cost function

$$J_c = \frac{1}{2}\sum_{i=1}^{N_p}\sum_{j=1}^{m} l_j e_{yj,c}(k+i)^2 + \frac{1}{2}\sum_{i=1}^{N_u}\rho\Delta\mathbf{u}(k+i-1)^2 \tag{3.9}$$

where $e_{yj,c}(k+i)$ is the error between desired value and predicted value of the j^{th} output of the system at time $k+i$ and l_j is penalty on this error. In the structure model, the desired value for acceleration and displacement is zero, therefore minimizing prediction error is equivalent to minimize the response of the system. However minimizing acceleration to zero may lead to unacceptable control force, therefore cost functions implies a penalty on control force increments to limit the control force. $\Delta\mathbf{u}$ is the change in the control input and ρ is the penalty on the change in the control input. A long control horizon leads to a smooth and small control input but decreases the tracking performance. A long prediction horizon leads to smooth control input also but decreases the tracking speed. The controller cost function J_c can be written in terms of the control as

$$J_c = \frac{1}{2}\left[\sum_{j=1}^{J}\left(\mathbf{e}_{yj,c}^T(k+1)\mathbf{L}\mathbf{e}_{yj,c}(k+1)\right) + \rho\left(\mathbf{H}\mathbf{u}(k)\right)^T(\mathbf{H}\mathbf{u}(k))\right] \tag{3.10}$$

in which **L** is a diagonal matrix with l_j's as its diagonal elements and

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ -1 & 1 & 0 & \dots & 0 \\ 0 & -1 & 1 & \dots & 0 \\ 0 & \dots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & -1 & 1 \end{bmatrix} \tag{3.11}$$

$$\mathbf{e}_{yj,c}(k+1) = \mathbf{y}_{j_d}(k+1) - \hat{\mathbf{y}}_j(k+1)\ , 1 \le j \le J \tag{3.12}$$

where y_{jd} is the desired value of the j^{th} output and $\hat{y}_j$ is its predicted value as follows:

$$\mathbf{y}_{j_d}(k+1) = \left[y_{jd}(k+1), , \dots, y_{jd}(k+N_p)\right]^T\ , 1 \le j \le J \tag{3.13}$$

$$\hat{\mathbf{y}}_j(k+1) = \left[\hat{y}_j(k+1), \dots, \hat{y}_j\left(k+N_p\right)\right]^T, 1 \le j \le J \tag{3.14}$$

$$\mathbf{u}(k) = [u(k), u(k+1), \dots, u(k+N_u - \ 1)]^T, 1 \le i \le m \tag{3.15}$$

The change in control input can be calculated by optimizing J_c via gradient descent based on the following generalized form

$$\frac{\partial J_c}{\partial \mathbf{u}(k)} = -\sum_{j=1}^{J} \mathbf{G}_{yj,\boldsymbol{u}}^{T} \mathbf{Le}_{yj,c}(k+1) \tag{3.16}$$

$$\Delta \mathbf{u}(\mathrm{k}) = (I + \lambda \rho \mathbf{H}^{T})^{-1} \times \frac{\partial J_c}{\partial \mathbf{u}(k)} \;\; and \;\; \mathbf{u}(k+1) = \mathbf{u}(k) + \Delta \mathbf{u}(\mathrm{k}) \tag{3.17}$$

$$\mathbf{G}_{\boldsymbol{y_j},u} = \left[g_{y_j,u}(s,l) \right] \; such \; that \; g_{\boldsymbol{y_j},u}(t,l) = \begin{cases} \dfrac{\partial \hat{y}_j(k+t)}{\partial u(k+l-1)} & , s \geq l \\ 0 & , s < l \end{cases} \tag{3.18}$$

$$1 \leq j \leq J, 1 \leq s \leq N_p, 1 \leq l \leq N_u$$

where λ is the learning rate of the gradient descent algorithm [61]. All the derivatives are calculated using the chain rule

$$\frac{\partial \hat{y}_j(k+q)}{\partial u(k+r)} = \frac{\partial \hat{y}_j(k+q)}{\partial \hat{y}_j(k+q-1)} \times \frac{\partial \hat{y}_j(k+q-1)}{\partial \hat{y}_j(k+q-2)} \times \ldots \ldots \times \frac{\partial \hat{y}_j(k+r+2)}{\partial \hat{y}_j(k+r+1)} \times \frac{\partial \hat{y}_j(k+r+1)}{\partial u(k+r)} \tag{3.19}$$

3.3.1 Alternative Controllers

A benchmark comparison of the proposed WNN-based controller is demonstrated with respect to linear quadratic Gaussian (LQG) and multivariable proportional-integral (PI) controllers. LQG control assumes that enough state variables of the structure, that is enough velocities and displacements, can be measured to make the system observable. The measurements are fed to a Kalman filter which estimates the states and control inputs of the system. In practice, accelerations are the only measured variables and velocities and displacements are obtained by integrating the accelerations. Integration introduces errors in the inputs to the Kalman filter and reduce the quality of LQG control.

Designing a PI controller for single-input-multi-output systems is a challenging problem. The system must be completely controllable whereas the seismically isolated structural model has five uncontrollable modes. To overcome this, Kalman decomposition is used to find the controllable subsystem of the structure. The controllable subsystem of the structure is a single-input-six-output system. A multivariable PI controller is then designed for the controllable subsystem using linear quadratic methods. Details of the two controllers are available in [97] and are omitted for brevity.

3.4 Computational Simulation Model

The computational simulation model represents the five-story seismically isolated structure that was developed for an experimental program by Kelly and Tsai [98] as shown in Figure 9. The model was analyzed as a conventionally isolated structure with lead-rubber bearings (LRB) only, and with LRBs coupled with actuators that provide the active control forces. In general, various types of devices may be employed in active control. While a specific type has not been assumed nor modelled, the most suitable devices in these types of applications may be hydraulic (or electrical) actuators. The lumped system properties are summarized in Table 3. The post-yield stiffness, k_b, of the isolation system was originally selected so that the fundamental period of the structure is 2.5 sec once the lead plug yields. The characteristic strength, Q_y, is selected to be 10% of the building's weight and the post-yield to pre-yield stiffness ratio, α, is taken as 8.5%. These values were recommended by Ramallo et al. to achieve acceptable control of the base displacement without excessive structural accelerations for both moderate and severe seismic events [99]. The inherent damping of the structure is assumed to be 2%.

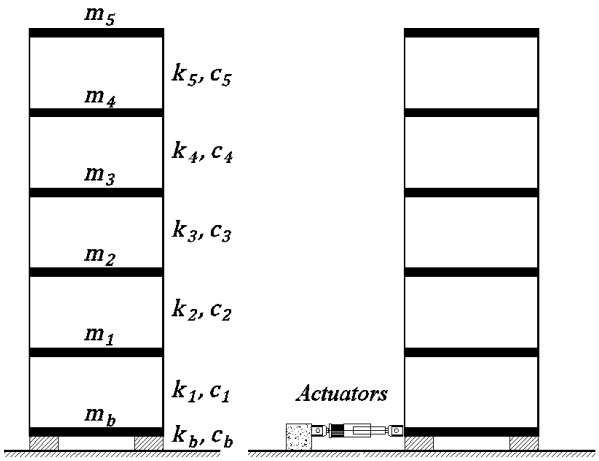

Figure 9. Simulation model, (a) LRB isolated, (b) LRB isolated and controlled

Table 3. Properties of the simulation model

Floor Mass (kg)	Stiffness Coefficients (kN/m)	Damping Coefficients (kN.s/m)
$m_b = 6800$	$k_b = 232$	$c_b = 3.74$
$m_1 = 5897$	$k_1 = 33732$	$c_1 = 67$
$m_2 = 5897$	$k_2 = 29093$	$c_2 = 58$
$m_3 = 5897$	$k_3 = 28621$	$c_3 = 57$
$m_4 = 5897$	$k_4 = 24954$	$c_4 = 50$
$m_5 = 5897$	$k_5 = 19059$	$c_5 = 38$

3.4.1 Equation of the Motion

The all-inclusive equations of motion of the seismically isolated structure (Figure 9) can be written as:

$$\mathbf{M}_s\ddot{\mathbf{u}}_s^t + \mathbf{C}_s\dot{\mathbf{u}}_s + \mathbf{K}_s\mathbf{u}_s = \mathbf{0} \tag{3.20}$$

$$m_b\ddot{u}_b^t + F_d(c_b(t), \dot{u}_b) + F_s(\alpha(t), u_b, \dot{u}_b, z) - \mathbf{I}_s^T\mathbf{V}_s = 0 \tag{3.21}$$

in which $\mathbf{M}_s$, $\mathbf{C}_s$ and $\mathbf{K}_s$ are the mass, damping, and stiffness matrices of the superstructure respectively. Equation (20) governs the superstructure motion while (21) defines the base slab motion. Furthermore, m_b is the base slab mass, $F_d(c_b(t), \dot{u}_b)$ and $F_s(\alpha(t), u_b, \dot{u}_b, z)$ are the damping force and the restoring force of the substructure (isolation level), respectively. It is noted that the damping force term F_d is provided also for cases where additional source of damping at the isolation level might be considered. $\ddot{\mathbf{u}}_s$, $\dot{\mathbf{u}}_s$ and $\mathbf{u}_s$ are the acceleration, velocity and displacement vectors of the superstructure with respect to the base slab, respectively, and $\ddot{u}_b$, $\dot{u}_b$ and u_b are those of substructure with respect to the ground. The superscript, t denotes the total displacement with respect to a fixed reference. $\mathbf{I}_s$ is the influence array of the superstructure motion on the substructure motion and $\mathbf{V}_s$ is the vector of shear forces induced on the superstructure

$$\mathbf{V_s} = -\mathbf{M_s}\ddot{\mathbf{u}}_s^t \tag{3.22}$$

The total displacement vectors can be expressed as

$$\mathbf{u}_s^t = \mathbf{u}_s + \mathbf{I}_b u_b + \mathbf{I}_1 u_g \tag{3.23}$$

$$u_b^t = u_b + \mathbf{I}_2 u_g \tag{3.24}$$

where $\mathbf{I}_b$ is the influence array of the base slab motion on the DOFs of the superstructure, $\mathbf{I}_1$ and $\mathbf{I}_2$ are the influence arrays of the ground motion on the superstructure and base slab DOFs respectively, and u_g is the total ground displacement. Substituting for $\mathbf{u}_s^t$ and u_b^t and rearranging, the governing equations of motion can be written as

$$\mathbf{M}_s\ddot{\mathbf{u}}_s + \mathbf{C}_s\dot{\mathbf{u}}_s + \mathbf{K}_s\mathbf{u}_s = -\mathbf{M}_s(\mathbf{I}_b\ddot{u}_b + \mathbf{I}_1\ddot{u}_g) \tag{3.25}$$

$$\begin{aligned} \boldsymbol{m_b\ddot{u}_b} + \boldsymbol{F_d(c_b,\dot{u}_b)} + \boldsymbol{F_s(\alpha,u_b,\dot{u}_b,z)} \\ = -\boldsymbol{m_b\mathbf{I}_2\ddot{u}_g} - \mathbf{I}_s^T\mathbf{M}_s(\ddot{\mathbf{u}}_s + \mathbf{I}_b\ddot{u}_b + \mathbf{I}_1\ddot{u}_g) \end{aligned} \tag{3.26}$$

The restoring force $F_s(\alpha, u_b, \dot{u}_b, z)$ represents the true hysteresis behavior of conventional LRB isolation systems. The well-known Bouc-Wen model [100] has been used in this study

$$F_s = k_b u_b + (1-\alpha)k_e x_y z \tag{3.27}$$

$$\dot{z} = A\dot{u}_b - \beta|\dot{u}_b|z|z|^{n-1} - \gamma\dot{u}_b|z|^n \tag{3.28}$$

where α is the ratio of the post-yield to the pre-yield stiffness of the isolation system, x_y defines the yield displacement of the isolators and z is dimensionless parameter that defines the hysteresis of the isolation system, and n, A, β and γ are constant parameters that control the shape of the hysteresis loops. The parameter z is found by solving the nonlinear differential equation (2.28). For the elastic stiffness to be modeled properly ($A = \beta + \gamma$), and for the unloading to follow the elastic stiffness ($\beta = \gamma$). Equations (2.25) and (2.26) can be written in matrix form

$$\begin{aligned} &\overbrace{\begin{bmatrix} \mathbf{M}_s & \mathbf{M}_s\mathbf{I}_b \\ \mathbf{I}_s^T\mathbf{M}_s & m_b + \mathbf{I}_s^T\mathbf{M}_s\mathbf{I}_b \end{bmatrix}}^{\bar{\mathbf{M}}}\begin{bmatrix}\ddot{\mathbf{u}}_s \\ \ddot{u}_b\end{bmatrix} + \overbrace{\begin{bmatrix}\mathbf{C}_s & \mathbf{0} \\ \mathbf{0} & c_b\end{bmatrix}}^{\bar{\mathbf{C}}}\begin{bmatrix}\dot{\mathbf{u}}_s \\ \dot{u}_b\end{bmatrix} + \overbrace{\begin{bmatrix}\mathbf{K}_s & \mathbf{0} \\ \mathbf{0} & k_b\end{bmatrix}}^{\bar{\mathbf{K}}}\begin{bmatrix}\mathbf{u}_s \\ u_b\end{bmatrix} \\ &= \overbrace{\begin{bmatrix} -\mathbf{M}_s\mathbf{I}_1 \\ -m_b\mathbf{I}_2 - \mathbf{I}_s^T\mathbf{M}_s\mathbf{I}_1 \end{bmatrix}}^{\mathbf{E}_g}\ddot{u}_g + \overbrace{\begin{bmatrix}\mathbf{0} \\ -\mathbf{I}_c\end{bmatrix}}^{\mathbf{E}_c} f_c \end{aligned} \tag{3.29}$$

where $\mathbf{I}_c$ is the location matrix of the restoring force of the isolation system and the control force. Equation (2.29) can be represented in state-space as

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}f_c + \mathbf{E}\ddot{u}_g \tag{3.30}$$

$$\mathbf{y} = \mathbf{C}_y\mathbf{x} + \mathbf{D}_y f_c + \mathbf{E}_y\ddot{u}_g + \mathbf{v} \tag{3.31}$$

where $\mathbf{x} = [\mathbf{u}_s^T, u_b, \dot{\mathbf{u}}_s^T, \dot{u}_b]^T$ is the state vector, $\mathbf{y}$ represents the vector of measurements, and $\mathbf{v}$ is the measurement noise. The state matrices are defined as

$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & \boldsymbol{I} \\ -\widetilde{\mathbf{M}}^{-1}\widetilde{\mathbf{K}} & -\widetilde{\mathbf{M}}^{-1}\widetilde{\mathbf{C}} \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \mathbf{0} \\ \widetilde{\mathbf{M}}^{-1}\mathbf{E}_c \end{bmatrix}, \quad \mathbf{E} = \begin{bmatrix} \mathbf{0} \\ -\widetilde{\mathbf{M}}^{-1}\mathbf{E}_g \end{bmatrix} \tag{3.32}$$

and C_y; D_y and E_y are obtained from the measured state. Assuming that the floor and base slab accelerations are measured:

$$\mathbf{C}_y = [-\bar{\mathbf{M}}^{-1}\widetilde{\mathbf{K}} \quad -\bar{\mathbf{M}}^{-1}\widetilde{\mathbf{C}}], \mathbf{D}_y = [-\bar{\mathbf{M}}^{-1}\mathbf{E}_c], \mathbf{E}_y = \mathbf{0}, \bar{\mathbf{M}} = \begin{bmatrix} \mathbf{M}_s & \mathbf{0} \\ \mathbf{I}_s^T\mathbf{M}_s & m_b \end{bmatrix} \tag{3.33}$$

In case of passive isolators, $(f_c = (1-\alpha)k_e x_y z)$ represents the hysteretic behavior, otherwise, the control force is superimposed on f_c. The state of the structure is used to generate its input-output data for identification and to update the parameters of the wavelet neural network as described earlier.

The computational simulation of both the uncontrolled (seismically isolated with conventional LRBs) and the controlled (with or without LRBs) structure are performed using Matlab/Simulink [101]. A block diagram of the online identification and control implementation is shown schematically in Figure 10.

The neural network input vector includes the control force and the history of acceleration. The input to the network is defined as

$$\mathbf{u}_N(k) = [f_c(k-1), \mathbf{y}(k-1), \dots, \mathbf{y}(k-10)] \tag{3.34}$$

where $\mathbf{y}$ is a 6x1 vector of accelerations:

$$\mathbf{y}(\mathrm{k}) = [y_1(k), y_2(k), \dots, y_6(k)]^T \tag{3.35}$$

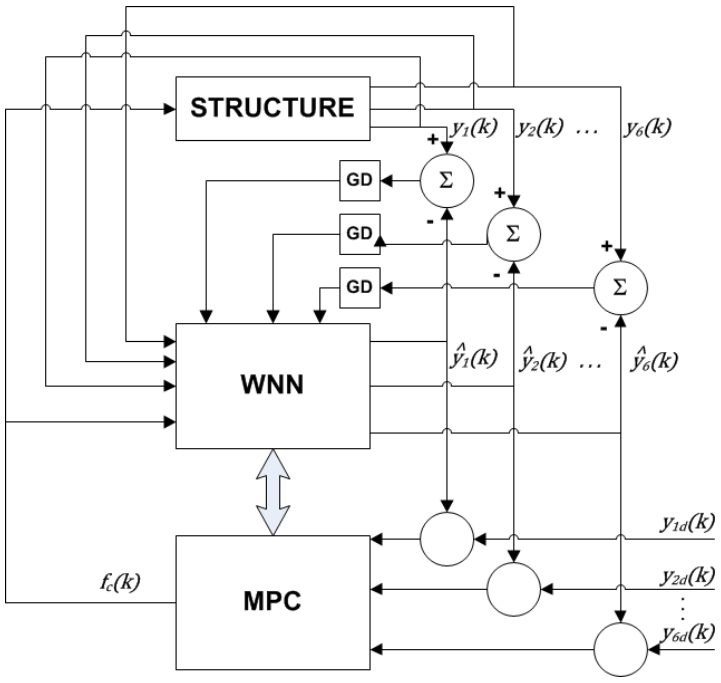

Figure 10. Block diagram of the online identification and control scheme

For the simulations in this study, 10 hidden layer neurons and a fixed learning rate of $\gamma = 0.2$ was specified. Both the prediction horizon N_p and the control horizon N_u were set equal to 10. The control input weighting factors for different cases investigated were chosen as $\rho = 10^{-1}$ for a medium-level penalty on the acceleration response, and $\rho = 10^{-4}$ for high penalty on the acceleration response. Since the accelerations of the base isolated structure are almost the same, penalty on the output prediction error L is assumed to be the identity matrix to impose the same penalty on all of the accelerations. The neural network parameters are randomly initialized except for setting the matrix **R** equal to the identity and the vector **q** equal to zero.

Because the RLS algorithm can quickly establish a linear relationship between inputs and outputs, no pre-tuning is needed on the WNN. However, for real world applications, the number of hidden layer nodes can be effectively predetermined using an approximate computational simulation model of the structure.

Last but not the least, for accurate system identification, the excitation should be sufficiently large to excite significant modes of vibration of the system, which is always the case for high intensity ground motions of interests. However, a key factor that may affect the performance of controller implementation is the noise in output measurements. Therefore, the simulations were repeated for two cases, with and without measurement noise to demonstrate the robustness of the

proposed control strategy in that regard. Robustness with respect to variations in system parameters (structural stiffness, isolation properties, etc.) is not considered in this study since the nonlinearities are assumed to be only associated with the isolation bearings. However, a more comprehensive assessment is part of an ongoing study that investigates full-scale implementation of the proposed control strategy, including the effects of variations in system parameters.

Table 4. Details of the ground motion records

EQ ID	Record Name	M_w	Recording Station	Dur.(s)	R (km)	PGA (g)	PGV (cm/s)	PGD (cm)
FAR-FIELD (FF) GROUND MOTIONS								
F01	01-Northridge, 1994	6.7	Beverly Hills-Mul.	23.95	13.3	0.62	40.7	8.56
F02	02-Northridge, 1994	6.7	Canyon Count.-WLC	19.95	26.5	0.48	44.9	12.5
F03	03-Duzce, Turkey, 1999	7.1	Bolu	55.85	41.3	0.82	62.1	13.6
F04	06-Imperial Valley, 1979	6.5	El Centro Array 11	39.03	29.4	0.38	42.1	18.6
F05	12-Landers, 1992	7.3	Coolwater	27.96	82.1	0.42	42.3	13.8
F06	18-Cape Mendocino, 1992	7.0	Rio Dell Overpass	35.90	22.7	0.55	41.9	19.5
F07	20-Chi-Chi, Taiwan, 1999	7.6	TCU045	89.98	77.5	0.51	40.0	14.3
NEAR-FIELD (NF) WITH PULSE GROUND MOTIONS								
N01	02-Imperial Valley, 1979	6.5	El Centro Array 7	36.80	27.6	0.46	109.3	44.7
N02	04-Superstition Hills, 1987	6.5	Parachute Test Site	22.30	16.0	0.45	111.9	52.8
N03	06-Erzincan, Turkey, 1992	6.7	Erzincan	21.30	9.00	0.52	95.5	27.7
N04	09-Northridge, 1994	6.7	Rinaldi Receiving	14.93	10.9	0.83	166.0	28.1
N05	10-Northridge, 1994	6.7	Sylmar - Olive View	39.90	16.8	0.84	129.4	39.9
N06	12-Chi-Chi, Taiwan, 1999	7.6	TCU065	89.98	26.7	0.81	126.2	92.6
N07	13-Chi-Chi, Taiwan, 1999	7.6	TCU102	89.98	45.6	0.30	112.5	89.2

3.4.2 Ground Motions

To illustrate the effectiveness of the proposed WNN-based control implementation, the seismically isolated structure is subjected to a series of recorded far-field and near-field ground

motions. A total of seven far-field and seven near-field [pulse] type recorded ground motions are selected as summarized in Table 4 and 5%-damped acceleration response spectra are plotted in Figure 11. The ground motions were selected as a subset of FEMA P695 records [102]. The ground motions were neither time-scaled nor amplitude-scaled since the main purpose of the study is to provide a relative comparison of achieved controlled response versus conventional seismic isolation. Selected ground motions in the near-field [pulse] type group were targeted to have peak ground velocities (PGV) greater than 100 cm/s (with one exception; 1992 Erzincan) and peak ground acceleration (PGA) to PGV ratios less than 0.7 g/cm/s. For the far-field type ground motions, the corresponding targets were PGV≥40 cm/s and PGA/PGV≤1.2 g/cm/s. Whereas both [high] PGV and [low] PGA/PGV parameters are considered as "damage" indicators, the average PGAs of the two sets of ground motions were selected to be comparable; 0.55g (FF) vs 0.60g (NF).

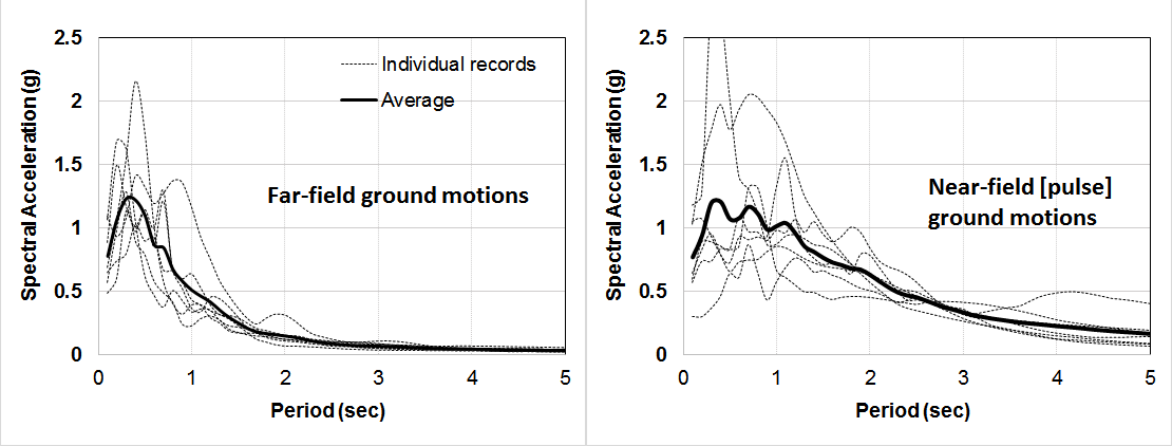

Figure 11. Response spectra of the selected ground motions

3.5 Performance of Conventional and Controlled Structure

In the simulation, the only observable responses were assumed to be floor and isolation level accelerations as indicated in equations (2.31) through (2.34). Therefore, the penalty factor ρ that is used in the minimization of the control cost function J_c applies to the measured acceleration response in equation (2.10). Smaller ρ values imply smaller desired acceleration response. Furthermore, WNN-based control does not require real-time simulation of the structure. However, a feasible weight ρ, learning rate λ and number of hidden layer nodes m can be determined offline (tuning) using an approximate simulation model before implementation. This parameters are fixed during real time application and the network weights are the only parameters that need to be

updated every time step. In this study, first a lower-bound $\rho = 10^{-4}$ was selected to examine the efficacy of the control method to reduce accelerations irrespective of the associated displacements. This is referred to as the *Cont-C2* case. In an attempt to reduce displacements while maintaining accelerations, base shear and foundation shear forces under acceptable limits, a larger $\rho = 10^{-1}$ was used. This case is referred to as *Cont-C1*.

Thus, the three different cases investigated are: 1) conventional seismically-isolated structure with LRBs (*Passive*), 2) structure with LRBs coupled with control with high penalty on acceleration response (*Cont-C2*; $\rho = 10^{-4}$), and 3) structure with LRBs coupled with control with medium-level penalty on acceleration response (*Cont-C1*; $\rho = 10^{-1}$). Each case was subjected to both NF and FF ground motions. A summary of the simulation results is presented in Figure 12 and Figure 16 for the NF and FF ground motions, respectively. These figures present the average of maximum (a) floor and base [isolator] displacements, (b) interstory drifts, (c) displacements relative to isolator, (d) floor and base accelerations, (e) [superstructure] base shear, and (f) total shear force at the foundation level which includes the isolator force and control force for the controlled cases.

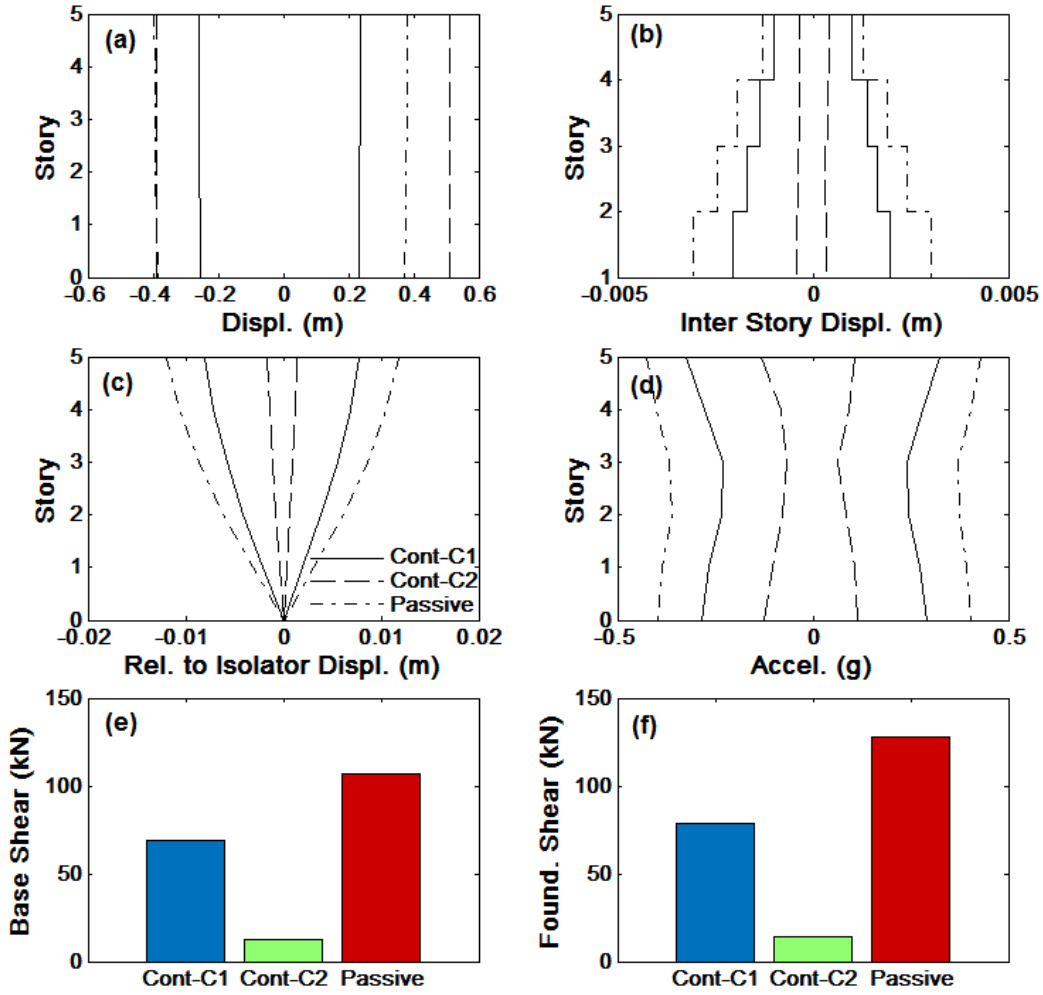

Figure 12. Comparison of average maximum response – Near-field ground motions. (a) story and base [isolator] displacements, (b) interstory drifts, (c) displacements relative to isolator, (d) story and base [isolator level] accelerations, (e) [superstructure] base

As can be seen in Figure 12(a-d), the overall displacement response is reduced by an average of 40% in the *Cont-C1* case with similar reductions in acceleration response along the height of the structure in comparison to the *Passive* case. In particular, the simultaneous reduction in the base displacement and the floor accelerations under pulse-type NF ground motions is notable. Sample displacement response history comparisons for two of the NF ground motions are shown in Figure 13 where the pulse-type nature of the earthquakes is evident. The simulation results demonstrate that *Cont-C2* implementation reduces the resonance behavior of the seismic isolation system induced by long-period ground motions. While the initial displacement pulse was not reduced significantly, subsequent larger isolation deformations were mitigated. These response reductions are achieved by the control force which effectively regulates the apparent isolation

stiffness in real-time (Figure 14). Furthermore, reduced floor accelerations result in lower superstructure base shear as well as foundation forces (Figure 12(e) and (d)).

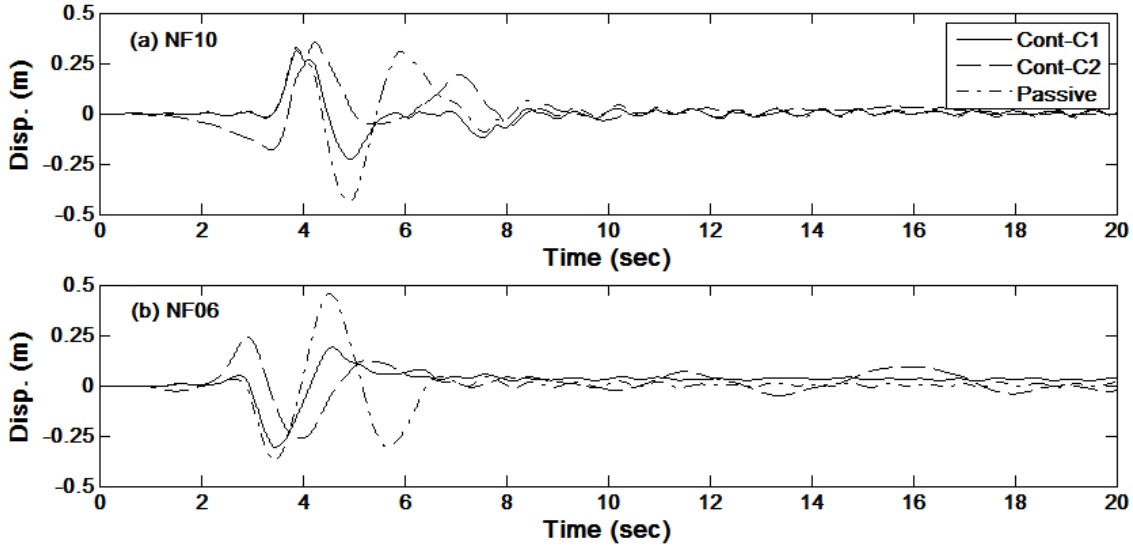

Figure 13. Sample isolator displacement response history comparisons. (a) NF05, (b) NF03

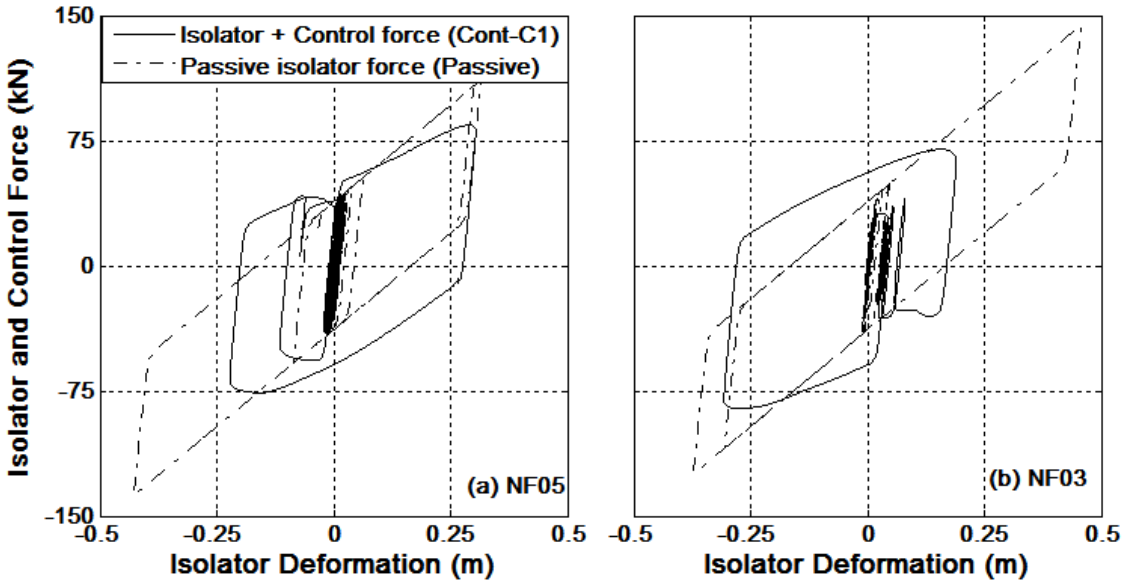

Figure 14. Sample isolator deformation versus isolator/control force deformation response. (a) NF05, (b) NF03

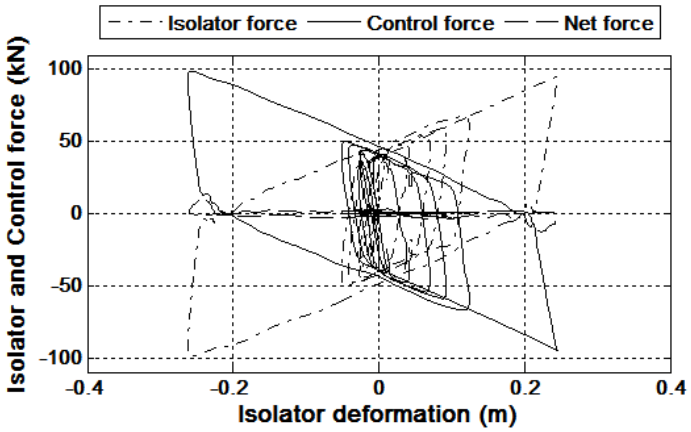

Figure 15. Sample isolator deformation versus isolator/control force deformation response; Cont-C2 case under NF03 ground motion

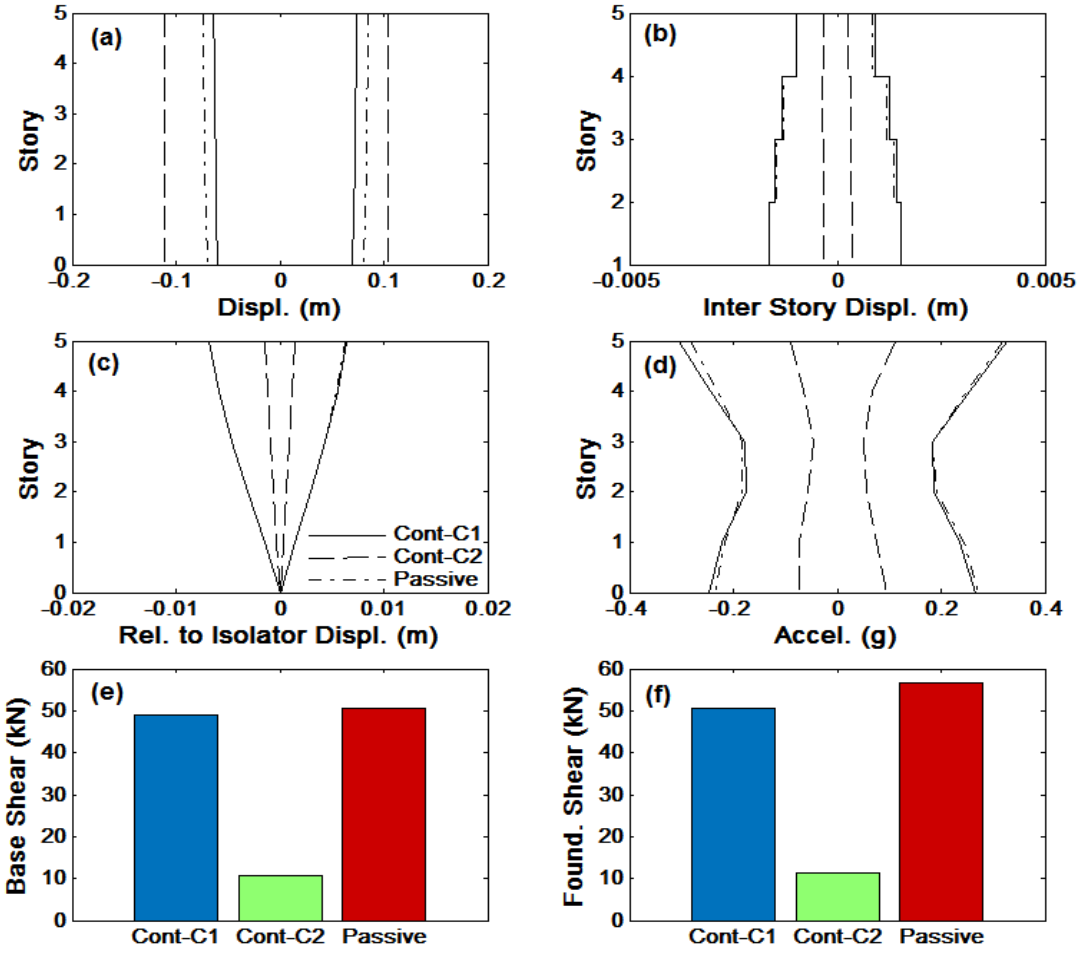

Figure 16 . Comparison of average maximum response – Far-field ground motions. (a) story and base [isolator] displacements, (b) interstory drifts, (c) displacements relative to isolator, (d) story and base [isolator level] accelerations, (e) [superstructure] base

The *Cont-C2* case employs a low weight ρ for control input with an objective to completely attenuate the acceleration response. As shown in Figure 12, this objective was achieved to the extent possible which further resulted in significantly reduced interstory drifts and base shear forces, albeit at the expense of larger isolator deformations compared to *Cont-C1* and *Passive* cases. In theory, the floor accelerations would tend to zero as the effective stiffness at the base of the structure reduces to zero. Our simulations show that WNN control dictates out-of-phase control forces with respect to isolator forces, which results in essentially zero effective stiffness in the isolation system (Figure 15).

Similar observations can be made in case of far-field (FF) ground motions. However, WNN-based control does not lead to significant changes in response quantities when compared to the conventional isolated system. This is primarily due to the fact that the passive isolator properties were deemed near optimal to achieve small displacements, accelerations, hence small forces, particularly for moderate level earthquakes as is the case with the FF ground motions [99]. Evidently, the seismic demand on the isolated structure due to FF ground motions is relatively insignificant in comparison to NF ground motions (Figure 16). This is confirmed by the present study as the average maximum control force in *Cont-C2* remained less than 13 kN versus 40 kN of isolator force with approximately 8% reduction in base displacements only. Finally, *Cont-C2* implementation resulted in significant reduction of all response quantities except for increased base displacements. Clearly, this may be considered a desired outcome as the benefits of reduced floor accelerations and interstory drifts outweigh the slight increase in the base displacements for certain types of buildings with deformation and acceleration sensitive equipment.

Table 5. Normalized response comparisons

	Near-field					**Far-field**				
	Cont-C1	Cont-C1 + Noise	Cont-C2	LQG	PI	Cont-C1	Cont-C1 + Noise	Cont-C2	LQG	PI
+ve isolator displ	0.625	0.612	1.383	1.018	0.947	0.868	0.836	1.298	1.304	1.094
-ve isolator displ	0.660	0.624	1.012	1.307	1.123	0.860	0.830	1.586	1.278	1.239
+ve story displ*	0.662	0.620	0.122	0.516	0.650	1.027	1.058	0.234	0.339	0.345
-ve story displ*	0.678	0.643	0.138	0.613	0.723	1.004	1.015	0.212	0.311	0.356

+ve story drift	0.760	0.739	0.311	0.516	0.650	1.078	1.099	0.320	0.339	0.345
-ve story drift	0.755	0.777	0.251	0.613	0.723	1.023	1.033	0.347	0.311	0.356
+ve story accel	0.755	0.778	0.267	0.609	0.719	1.023	1.032	0.348	0.298	0.343
-ve story accel	0.760	0.740	0.312	0.517	0.653	1.078	1.099	0.334	0.323	0.328
Peak base shear	0.643	0.596	0.119	0.613	0.723	0.973	0.987	0.212	0.336	0.356
Peak found. shear	0.618	0.560	0.108	0.628	0.738	0.894	0.895	0.199	0.362	0.381

* Relative to isolators

To demonstrate the relative efficiency as well as the effect of potential noise in the output measurements (acceleration response), additional cases were considered and simulated as listed in Table 5. The table summarizes normalized response quantities with respect to the conventional passive case. It can be seen that classical LQG and multivariable PI controllers can achieve the desired response reduction in general, but fail to reduce deformation response at the isolation bearing level. In contrast, the achieved response reductions of all quantities by the Cont-C1 case with and without noise are similar, demonstrating the insensitivity of the proposed WNN-based controller to the noise in the measurements.

3.6 Conclusion

The primary objective of this study was to assess the applicability of the proposed Wavelet Neural Network (WNN)-based control to reduce the isolator deformations (base displacements) in seismically isolated structures subjected to near-field ground motions. This can be achieved using conventional techniques by providing higher levels of damping at the isolation level, but only at the expense of increasing floor accelerations and interstory drifts. Clearly, given the uncertainty and variability of ground motion characteristics, the control and reduction of both the displacements and accelerations require active control. For this purpose, a wavelet neural network comprising a wavelet back propagation network in parallel with a feedforward component trained using recursive least squares is introduced. The feedforward component significantly reduces the number of hidden layer nodes, and provides fast efficient learning. The computational complexity of the error backpropagation algorithm is $O(n^3)$ with n hidden layer nodes. Consequently, the computational complexity of the wavelet neural network is reduced drastically with the number of

hidden layer neurons. Hence, the parallel wavelet network is suited to online identification and control. The WNN-based control with MPC was used for online identification and control of a nonlinear structural system. The only input to the network was assumed to be monitored floor accelerations.

The efficacy of the method was demonstrated through a series of computational simulations. Two control cases that reduce the acceleration response and mitigate deformations were evaluated. Both cases demonstrate the effectiveness of WNN-based control in comparison to conventional isolation. They also highlight the efficiency and flexibility of the proposed approach to achieve multiple performance objectives. All response quantities were significantly reduced for both near-field (NF) and far-field (FF) ground motions, in particular an average of 40% reduction in isolator deformations, with corresponding reductions in floor accelerations, was observed. A comparison between the average responses to NF and FF ground motions suggests that one of the control cases is capable of reducing large base displacements due to NF ground motions without compromising performance under FF ground motions.

The controller performance was dictated by the established performance objective through a penalty factor and WNN learning rate. The controller effectively regulated the apparent stiffness at the isolation level. This observed feature of the WNN-based control makes the method a desirable hybrid seismic isolation alternative in general, and particularly a good candidate for lightweight structural system and equipment isolation. Finally, the proposed control method is fast, accurate, and robust, which allows implementation for large-scale dynamic systems. Furthermore, controllers that optimize other performance indices (objectives) can be readily implemented to provide more targeted response control.

Chapter 4. Networked Model Predictive Control Using a Wavelet Neural Network

4.1 Introduction

Networked control systems are control systems in which the controller, actuator and sensor are connected through a communication network. The shared network connection between different components of the control loop yields a flexible architecture and reduced installation and maintenance costs [39]. The theory of networked control systems combines control system theory and communication theory [39]. Time delay is a salient feature of any digital control system. This delay can be due to either plant delay or computational delay [103]. The computational delay can adversely affect controller performance or cause closed loop instability [104].

The control and stability of time-delayed systems has been widely studied [105], and various control and optimization algorithms have been proposed to provide satisfactory stable performance [106]. Astrom and Wittenmark studied effects of computational delay on digital controller design [107].

Although recent advances in digital processors have mitigated the effects of computational delay, network transport delay must still be considered in the design of networked control systems [103][108], [109], [110]. Delay switching based methods and parameter uncertainty based methods are two other alternatives to deal with the network induced delays [111],[112][113]. The approach of [114] uses parameter uncertainty based to deal with network-induced delay and uses linear matrix inequalities (LMI) to prove the existence of a stable state feedback controller. Wang and Yang [115] model the delay as a Markov chain and model the control loop as a Markov jump system then stabilize the closed loop system using an output feedback.

Although switching based methods for time delay systems are less conservative, they are more computationally costly and are difficult to implement [115],[116]. A combination of switching and parameter uncertainty approaches is used in [117] to avoid the computational complexity of switching based approach and the conservativeness of parameter uncertainty based approach.

Traditional digital control uses uniform sampling of the measurements over time. Although this makes analysis easy, it is not optimal in terms of network traffic [118],[119]. Astrom and Bernhardsson proposed an event triggered sampling scheme to decrease the network traffic by reducing the number of packets sent over the network [118]. The main idea of event triggered base

systems is to obtain a new measurement when the closed loop system does not satisfy desired performance criteria [120]. Researchers have proposed different sampling approaches such as deadband sampling [121], self-triggered sampling [122]-[124] and error energy sampling [125] to optimize the network resource usage.

Heemels and Donkers used periodic event triggered control for a piecewise linear system and for an impulsive system, and analyzed the stability of the controller for both systems [126]. Wang et al. proved that a network control system with L1 adaptive controller and event trigger sampling scheme can be arbitrarily close to a desired stable reference system under certain conditions [127]. Peng and Hong designed H_∞ controller with non-uniform sampling period. They sampled the states of the system nonuniformly, modeled the networked control system as a time-delay system, and proved the ultimate boundedness of their controller [128]. Another method of designing H_∞ controllers based on Markovian modeling of sensor and actuator was presented in [129]. An observer based H_∞ controller for continuous time networked control system was presented in [130]. A new model for continuous time networked control system was introduced and the observer based controller was designed based on a new Lyapunov functional. A method of designing an L_2 controller for decentralized event-triggered control system was presented in [24].

Wang et al. designed an event triggered model predictive controller for wireless networked control [132]. They derived trigger conditions and proved the stability of their controller by choosing the objective function of MPC as a Lyapunov function. A networked predictive controller to dampen power system inter-area oscillations was presented in [133]. The network predictive controller uses a generalized predictive control scheme to calculate the optimal control input for constant and random network delay. The stability analysis of a networked control system with a predictive-observer based controller was presented in [134]. They proved stability using two different Lyapunov functions, a function derived from network conditions, and a common quadratic Lyapunov function.

Cao et al. used a Gaussian process model of the unknown dynamics of a quadrotor with model predictive control [135]. Their methodology handles the model uncertainty and is computationally efficient. A locally weighted learning model predictive control (LWL-MPC) is presented in [136]. The model can effectively learn nonlinear and time varying dynamics online.

In this study, we use a wavelet neural network with feedforward component for online nonlinear system identification. Zhang and Benveniste argued that wavelet neural networks may have fewer

nodes that other artificial neural networks [5]. The feedforward component drastically reduces the number of hidden layer nodes and consequently reduces the training time of the wavelet neural network. The improved computational efficiency of the wavelet networks with feedforward component makes it ideal for online identification and control applications [137]. The model predictive controller uses wavelet neural network to predict the future outputs of the system over an extended prediction horizon and minimizes a cost function to find the optimal control action. Lyapunov stability theory is used to prove the stability of the model predictive controller.

To demonstrate the efficacy of our networked control approach, we apply it to the control of an unmanned autonomous vehicle. Two scenarios of fixed and random network delay are simulated. Simulation results show that the model predictive controller with extended prediction horizon can successfully mitigate the effect of fixed and random network delay. A preliminary version of this work was presented in [61], [250]. This paper extends the preliminary work of [61]and adds the following:

- A new formulation of the controller equations which is computationally efficient
- A proof of the stability of the controller based on the new formulation
- Improved simulation results with a new network designed that reduces the tracking error.

4.2 Networked Control System

A block diagram of networked control of an autonomous vehicle is shown in Figure 17. The measurements are sent over the communication channel to the WNN and MPC. Measurements are received after a network delay and they may get lost due to packet loss in the communication channel. The WNN weights are updated after receiving the measurement, then the MPC predicts the future outputs of the system. Using the predicted outputs and the desired future outputs of the system, the controller calculates the future optimal control inputs by minimizing a controller cost function.

Due to network delay, the system does not promptly receive the control input. Hence the controller needs to make more predictions to compensate for the effect of delay from sensor to controller and controller to actuators.

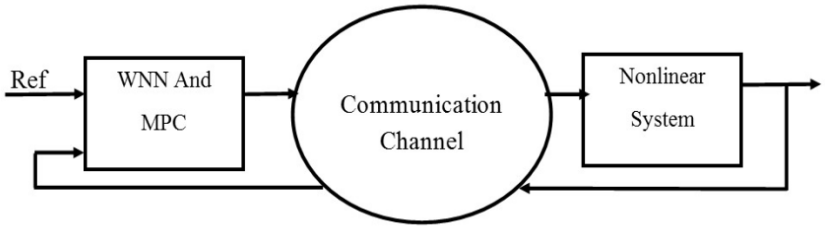

Figure 17. General scheme of plant and controller

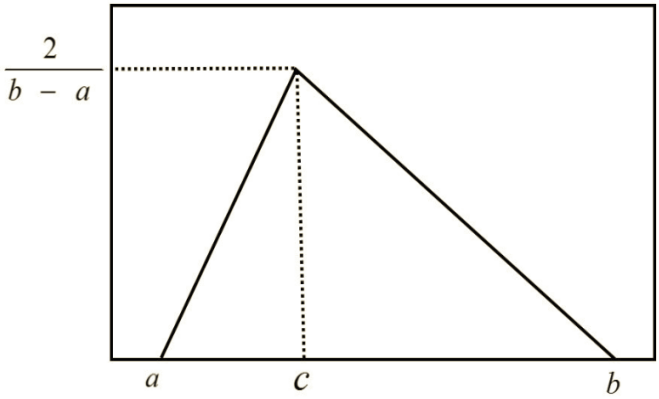

Figure 18 . Probability density function of triangular distribution

We use a feedforward wavelet neural network with one hidden layer and feedforward component to identify the model of nonlinear system. The wavelet neural network structure is shown in Figure 19. The input-output equation of the WNN is described as

$$\hat{\boldsymbol{y}}(k) = \boldsymbol{S}\boldsymbol{\psi}(\boldsymbol{u}_N) + \boldsymbol{Q}\boldsymbol{u}_N(k) \tag{4.1}$$

where $\boldsymbol{u}_N(k) = \left[u_{N_1}, \dots, u_{N_n}\right]^T$ is the input vector to the network and $\hat{\boldsymbol{y}}(k) = \left[\hat{y}_1(k), \hat{y}_2(k), \dots, \hat{y}_J(k)\right]^T$ is networks output. n is number of inputs to the network and J is number of network outputs. Activation function of hidden layer nodes are assumed to be Mexican hat wavelet

$$\psi_i(t_i) = \frac{2\pi^{\frac{1}{4}}}{\sqrt{3}}(1 - t_i^2)e^{-\frac{t_i^2}{2}} \tag{4.2}$$

$$t_i = \frac{\boldsymbol{w}_i^T \boldsymbol{u}_N - b_i}{a_{i,i}}, i = 1, \dots, m \tag{4.3}$$

where m is number of hidden layer nodes. With m nodes in the hidden layer, $\boldsymbol{S}$, $\boldsymbol{W}$ and $\boldsymbol{Q}$ will be $J \times m, m \times n$ and $J \times n$ matrices respectively. $\boldsymbol{b} = [b_i]_{m\times 1}$ is vector of shift parameter of wavelets and $\boldsymbol{A} = diag([a_{1,1}, \dots, a_{m,m}])$ is diagonal matrix whose diagonal elements are scale parameter of hidden layer activation functions.

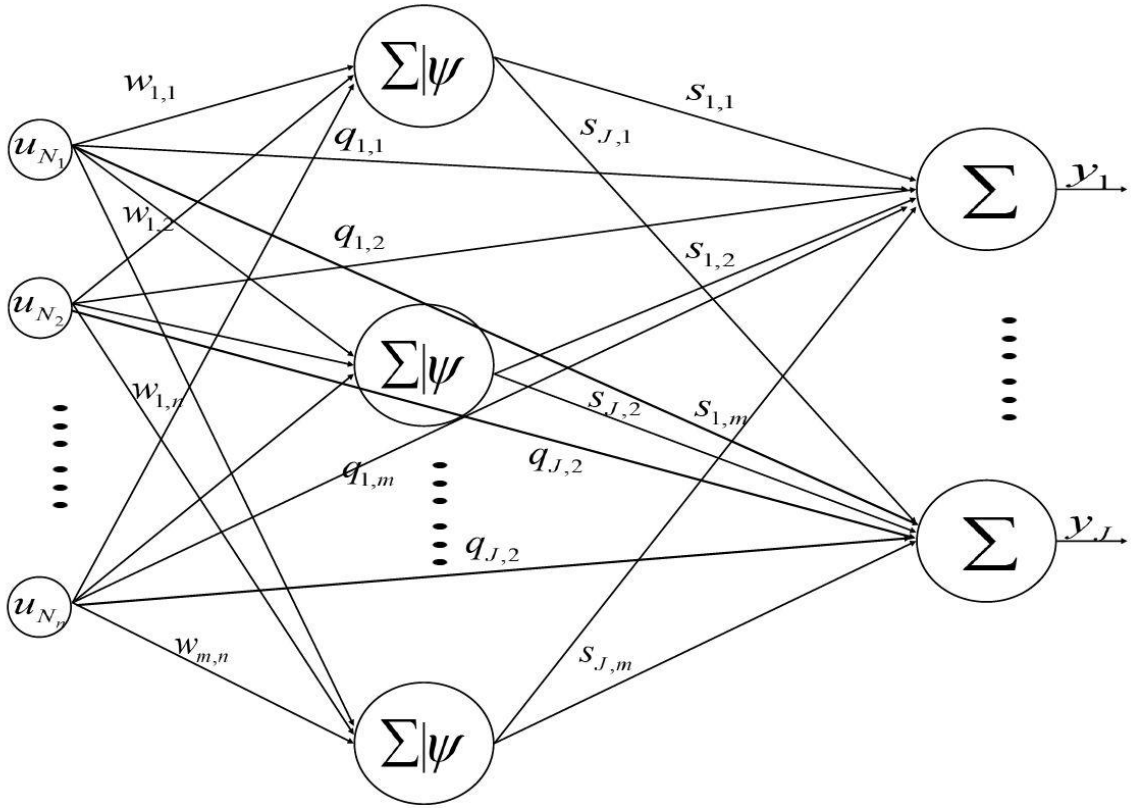

Figure 19 . Structure of wavelet neural network [61]

The feedforward component weights is tuned using well known recursive least squares algorithm and the rest of the parameters are tuned using error backpropagation algorithm. Due to fast convergence of recursive least squares algorithm no pre tuning is needed for wavelet neural network. This makes the network more interesting for online identification and control applications.

The cost function for training wavelet neural network is the sum of squared errors (SSE)

$$J_N = \frac{1}{2}\sum_{k=1}^{N} \|\hat{\boldsymbol{y}}(k) - \boldsymbol{y}(k)\|^2 \tag{4.4}$$

where $\boldsymbol{y}(k)$ is the vector of measured outputs and $\hat{\boldsymbol{y}}(k)$ is the output of the WNN. Network parameters are updated by the gradient descent algorithm

$$\sigma_{t+1} = \sigma_t - \gamma \nabla_\sigma (J_N) \tag{4.5}$$

γ is the learning rate for gradient descent algorithm with $\gamma \in (0,1]$. After training the feedforward component using the recursive least squares algorithm, the network modeling error can be calculated as

$$\boldsymbol{e}_y(k) = \hat{\boldsymbol{y}}(k) - \boldsymbol{y}(k) - Q\boldsymbol{u}_N(k) \tag{4.6}$$

$$\boldsymbol{e}_{\boldsymbol{y}}(k) = \left[\boldsymbol{e}_{y_1}, \ldots, \boldsymbol{e}_{y_J}\right]^T \tag{4.7}$$

The cost function for training the neural network can be rewritten in terms of the error as

$$J_N = \frac{1}{2}\sum_{k=1}^{N}\left\|\boldsymbol{e}_y(k)\right\|^2 \tag{4.8}$$

The chain rule is used to calculate the gradient of the cost function with respect to the WNN parameters

$$\nabla_\sigma(J_N) = \boldsymbol{e}_y^T(k)\partial\hat{\boldsymbol{y}}(k)/\partial\sigma \tag{4.9}$$

4.2.1 Stability Analysis

The following proposition provides stability conditions for the learning rate of the WNN.

Proposition: Suppose that $\boldsymbol{o}_t$ is a vector of network parameters that affect the t^{th} output, $\boldsymbol{o}_t = [a_i, b_i, w_{i,j}, q_{i,j}, s_{i,j}]$, and $\mu = \max_t \left\|\frac{\partial\hat{y}_t(k)}{\partial\boldsymbol{o}_t(k)}\right\|_2^2 , t = 1,2,\dots,J$, where $\hat{y}_t(n)$ is the t^{th} network output at time step n. Then the WNN is stable if the learning rate satisfies $\gamma < 2/\mu$.

Proof:

Lyapunov's method is used to analyze the stability of the WNN. The positive definite Lyapunov function for the WNN is defined as

$$V(k) = \frac{1}{2}\sum_{t=1}^{J} e_t(k)^2 \tag{4.10}$$

where $e_t(n)$ is the output error of wavelet network. The change in the Lyapunov function can be expressed as

$$\begin{aligned}\Delta V(k) = V(k+1) - V(k) &= \frac{1}{2}\sum_{t=1}^{J} e_t(k+1)^2 - e_t(k)^2 \\ &= \sum_{t=1}^{J}\Delta e_t\left(e_t(k) + \frac{1}{2}\Delta e_t(k)\right)\end{aligned} \tag{4.11}$$

$\boldsymbol{o}_t$ is a vector that contains all network parameters that affect the t^{th} output, $\boldsymbol{o}_t = [a_i, b_i, w_{i,j}, q_{i,j}, s_{i,j}]$, the output error of the network can be approximated as

$$\Delta e_t(k) = e_t(k+1) - e_t(k) \cong \frac{\partial e_t(k)}{\partial \boldsymbol{o}_t(k)}\Delta\boldsymbol{o}_t(k) \tag{4.12}$$

Using (6), (7) and (24), the change in the Lyapunov function can be rewritten as

$$\Delta V(k) = \sum_{t=1}^{J} \left[\frac{\partial e_t(k)}{\partial \boldsymbol{o}_t(k)}\right]^T \gamma\, e_t(k) \frac{\partial \hat{y}_t(k)}{\partial \boldsymbol{o}_t(k)} \left[e_t(k) + \frac{1}{2}\frac{\partial e_t(k)}{\partial \boldsymbol{o}_t(k)} \gamma e_t(n) \frac{\partial \hat{y}_t(k)}{\partial \boldsymbol{o}_t(k)}\right] \tag{4.13}$$

$e_t(k) = y_t(k) - \hat{y}_t(k)$ results in $\frac{\partial e_t(k)}{\partial \boldsymbol{o}_t(k)} = -\frac{\partial \hat{y}_t(k)}{\partial \boldsymbol{o}_t(k)}$, therefore $\Delta V(k)$ can be expressed as

$$\begin{aligned}\Delta V(k) = \sum_{t=1}^{J} e_t(k)^2 \gamma \Bigg[&\left[\frac{\partial e_t(k)}{\partial \boldsymbol{o}_t(k)}\right]^T \frac{\partial \hat{y}_t(k)}{\partial \boldsymbol{o}_t(k)} \\ &+ \frac{1}{2}\left[\frac{\partial e_t(k)}{\partial \boldsymbol{o}_t(k)}\right]^T \frac{\partial \hat{y}_t(k)}{\partial \boldsymbol{o}_t(k)} \gamma \left[\frac{\partial e_t(k)}{\partial \boldsymbol{o}_t(k)}\right]^T \frac{\partial \hat{y}_t(k)}{\partial \boldsymbol{o}_t(k)}\Bigg]\end{aligned} \tag{4.14}$$

By combining similar terms and the fact that $\frac{\partial e_t(k)}{\partial \boldsymbol{o}_{t_i}(k)} = -\frac{\partial \hat{y}_t(k)}{\partial \boldsymbol{o}_{t_i}(k)}$,$\Delta V(n)$ can be simplified to

$$\Delta V(k) = -\sum_{t=1}^{J} e_t(k)^2 \gamma \left\|\frac{\partial \hat{y}_t(k)}{\partial \boldsymbol{o}_t(k)}\right\|_2^2 \left[1 - \frac{1}{2}\gamma \left\|\frac{\partial \hat{y}_t(k)}{\partial \boldsymbol{o}_t(k)}\right\|_2^2\right] \tag{4.15}$$

For $\Delta V(n)$ to be negative definite

$$\left[1 - \frac{1}{2}\gamma \left\|\frac{\partial \hat{y}_t(k)}{\partial \boldsymbol{o}_t(k)}\right\|_2^2\right] < 0 \tag{4.16}$$

Suppose that $\mu = \max_t \left\|\frac{\partial \hat{y}_t(k)}{\partial \boldsymbol{o}_t(k)}\right\|_2^2$, $t = 1,2,\dots,J,$ by choosing $\gamma < 2/\mu$, the wavelet network will be stable.

4.3 Model Predictive Controller

To design a model predictive controller, an accurate model of the system is needed. We assume that the outputs of the system can be measured in each iteration to update the feedforward component and WNN weights. The updated WNN is used to predict the future outputs of the system and the MPC uses the predicted outputs to calculate future inputs by minimizing the controller cost function. The controller cost function is given by

$$J_c(k) = \frac{1}{2}\sum_{i=1}^{N_p}\sum_{j=1}^{J} \xi e_{yj,c}(k+i)^2 + \frac{1}{2}\sum_{j=1}^{n_u}\sum_{i=1}^{N_u} \rho \Delta u_j(k+i-1)^2 \tag{4.17}$$

where $\boldsymbol{e}_{yj,c}$ is the prediction error defined as the error between predicted outputs, $\hat{\boldsymbol{y}}_j(n+1)$, and desired outputs $\boldsymbol{y}_{j_d}(n+1)$.

$\Delta u_j(n), j = 1, \dots, n_u$ is the increment in the j^{th} input. $\xi$and ρ are penalty factors on the j^{th} tracking error and the i^{th} input change, respectively. Both factors are assumed to be in the range $(0,1]$. J is the number of outputs and n_u is the number of inputs. Small values of ρ lead to smaller and smoother control action while large values lead to faster tracking but may cause controller instability. Small values of ξ lead to smaller tracking error and larger control action while larger values of ξ increase the tracking error while reducing the magnitude of the control input. N_p is the prediction horizon and N_u is the control horizon. Large values of the prediction horizon lead to smoother control action but increase the tracking error while smaller values lead to better tracking and larger control input.

The neural network input, $\boldsymbol{u}_N(k)$, consist previous inputs and output measurements. Hence, it can be partitioned as

$$\boldsymbol{u}_N(k) = \left[\boldsymbol{u}_p^T(k), \boldsymbol{y}^T(k-1)\right]^T \tag{4.18}$$

By defining the controller error as $\boldsymbol{e}_c(k) = \hat{\boldsymbol{y}}(k) - \boldsymbol{y}(k)$ and

$$\boldsymbol{u}(k) = \left[\boldsymbol{u}_p^T(k), \boldsymbol{u}_p^T(k-1)\right]^T \tag{4.19}$$

$$\boldsymbol{H} = [\boldsymbol{I}_{n_u \times n_u} \quad -\boldsymbol{I}_{n_u \times n_u}] \tag{4.20}$$

where $I_{n_u \times n_u}$ is $n_u \times n_u$ identity matrix, the controller cost function can be rewritten as

$$J_c(k) = \frac{1}{2}\sum_{j=1}^{N_p} \xi \boldsymbol{e}_c^T(k+j)\boldsymbol{e}_c(k+j) + \frac{1}{2}\sum_{i=1}^{N_u} \rho\big(\boldsymbol{H}\boldsymbol{u}(k+i-1)\big)^T\big(\boldsymbol{H}\boldsymbol{u}(k+i-1)\big) \tag{4.21}$$

To find the optimal value of increment in control input, we need to optimize $J_c(k)$ with respect to $\boldsymbol{u}_p(k)$

$$\begin{aligned}\frac{\partial J_c(k)}{\partial \boldsymbol{u}_p(k)} = \frac{1}{2}\xi \sum_{j=1}^{N_p}\left[\frac{\partial \boldsymbol{e}_c(k+j)}{\partial \boldsymbol{u}_{\boldsymbol{p}}(k)}\right]^T \boldsymbol{e}_c(k+j) \\ + \frac{1}{2}\sum_{i=1}^{N_u} \rho\left[\frac{\partial \boldsymbol{u}(k+i-1)}{\partial \boldsymbol{u}_p(k)}\right]^T \times \boldsymbol{H}^T\boldsymbol{H}\boldsymbol{u}(k+i-1)\end{aligned} \tag{4.22}$$

Therefore

$$\partial J_c(k)/\partial \boldsymbol{u}_p(k) = \xi \sum_{j=1}^{N_p} \left[\frac{\partial \boldsymbol{e}_c(k+j)}{\partial \boldsymbol{u_p}(k)}\right]^T \boldsymbol{e}_c(k+j) - N_u \rho \Delta \boldsymbol{u}_p(k) \tag{4.23}$$

using the update equation, $\Delta \boldsymbol{u}_p(k) = -\eta \partial J_c(k)/\partial \boldsymbol{u}_p(k)$, and $\boldsymbol{e}_c(k) = \hat{\boldsymbol{y}}(k) - \boldsymbol{y}(k)$ we will have

$$\Delta \boldsymbol{u}_p(k) = -\eta\xi(1 - N_u \eta\rho)^{-1} \sum_{j=1}^{N_p} \left[\frac{\partial \hat{\boldsymbol{y}}(k+j)}{\partial \boldsymbol{u_p}(k)}\right]^T \boldsymbol{e}_c(k+j) \tag{4.24}$$

and the total control input is calculated as

$$\boldsymbol{u}_p(k+1) = \boldsymbol{u}_p(k) + \Delta \boldsymbol{u}_p(k) \tag{4.25}$$

The standard solution of this optimization problem calculates $\Delta \boldsymbol{u}_p(k+i), i = 0, \dots, N_u - 1$ then applies $\Delta \boldsymbol{u}_p(k)$ to system and discards the other calculated values. Our formulation only calculates $\Delta \boldsymbol{u}_p(k)$, which makes the algorithm computationally efficient. $\frac{\partial \hat{\boldsymbol{y}}(k+j)}{\partial \boldsymbol{u_p}(k)}$ can be calculated using the chain rule

$$\frac{\partial \hat{\boldsymbol{y}}(k+j)}{\partial \boldsymbol{u}_p(k)} = \frac{\partial \hat{\boldsymbol{y}}(k+1)}{\partial \boldsymbol{u}_N(k)} \frac{\partial \hat{\boldsymbol{y}}(k+2)}{\partial \hat{\boldsymbol{y}}(k+1)} \times \dots \times \frac{\partial \hat{\boldsymbol{y}}(k+j)}{\partial \hat{\boldsymbol{y}}(k+j-1)} \tag{4.26}$$

The derivatives in the chain rule can be calculated as

$$\begin{aligned} \frac{\partial \hat{\boldsymbol{y}}(k+1)}{\partial \boldsymbol{u}_p(k)} = \left[\boldsymbol{I}_{n_p \times n_p} \quad \boldsymbol{0}_{n_p \times n - n_p}\right] \times [\boldsymbol{Q}^T \\ + \boldsymbol{W}^T \boldsymbol{A}^{-1} diag\left(\psi'\left(\boldsymbol{A}^{-1}(\boldsymbol{W}\boldsymbol{u}_N(k) - b)\right)\right) \times \boldsymbol{S}^T] \end{aligned} \tag{4.27}$$

$$\begin{aligned} \frac{\partial \hat{\boldsymbol{y}}(k+j+1)}{\partial \hat{\boldsymbol{y}}(k+j)} = \left[\boldsymbol{0}_{J \times n_p} \quad \boldsymbol{I}_{J \times J} \quad \boldsymbol{0}_{J \times n - J - n_p}\right] \\ \times [\boldsymbol{Q}^T + \boldsymbol{W}^T \boldsymbol{A}^{-1} diag\left(\psi'\left(\boldsymbol{A}^{-1}(\boldsymbol{W}\boldsymbol{u}_N(k+j) - \boldsymbol{b})\right)\right) \times \boldsymbol{S}^T] \end{aligned} \tag{4.28}$$

4.3.1 Stability Analysis

According to proposition 1, the wavelet neural network in convergent, therefore, the identification error, which is difference between network output and desired output is bounded. We assume that $\|\boldsymbol{e}_c(k)\| < k_e$. Due to convergence of the neural network and bounded derivative

of the neural network activation function, $\frac{\partial \hat{\boldsymbol{y}}(k+j)}{\partial \boldsymbol{u_p}(k)}$ is bounded. We assume that $\left\|\frac{\partial \hat{\boldsymbol{y}}(k+j)}{\partial \boldsymbol{u_p}(k)}\right\| < k_y$. Using this assumptions and (4.24)

$$\begin{aligned}\left\|\Delta \boldsymbol{u}_p(k)\right\| &= \left\|\eta\xi(1-N_u\eta\rho)^{-1}\sum_{j=1}^{N_p}\left[\frac{\partial \hat{\boldsymbol{y}}(k+j)}{\partial \boldsymbol{u}_p(k)}\right]^T \boldsymbol{e}_c(k+j)\right\| \\ &\leq \eta\xi(1-N_u\eta\rho)^{-1}N_p k_y k_e\end{aligned} \tag{4.29}$$

By defining $\boldsymbol{P}_j = \frac{\partial \hat{\boldsymbol{y}}(k+j)}{\partial \boldsymbol{u_p}(k)}\left[\frac{\partial \hat{\boldsymbol{y}}(k+1)}{\partial \boldsymbol{u}_p(k)}\right]^T, j = 1,..,N_p$ and

$$\lambda_{min} = \left(\min_{j\in\{1,..,N_p\}} \lambda(\boldsymbol{P}_j)\right)^2 \tag{4.30}$$

and

$$\lambda_{max} = \max_{j\in\{1,..,N_p\}} \lambda\left(\frac{\partial \hat{\boldsymbol{y}}(k+j)}{\partial \boldsymbol{u_p}(k)}\right) \times max\, \lambda(\boldsymbol{P}_1) \tag{4.31}$$

The stability of the model predictive controller is stated as Theorem 1.

Theorem 1: The wavelet based model predictive controller is stable if $\eta < \frac{\lambda_{\min}}{\xi\lambda_{max}+\rho N_u\lambda_{min}}$

Proof:

consider the discrete Lyapunov function $V(k) = \frac{1}{2}\boldsymbol{e}_c(k)^T e_c(k)$. $V(k)$ is a positive define function of the states. The change in the Lyapunov function is

$$\begin{aligned}\Delta V(k) = V(k+1) - V(k) &= \frac{1}{2}\boldsymbol{e}_c(k+1)^T\boldsymbol{e}_c(k+1) - \frac{1}{2}\boldsymbol{e}_c(k)^T\boldsymbol{e}_c(k) \\ &= \frac{1}{2}\Delta\boldsymbol{e}_c(k+1)(2\boldsymbol{e}_c(k+1) + \Delta\boldsymbol{e}_c(k+1))\end{aligned} \tag{4.32}$$

where $\Delta\boldsymbol{e}_c(k+1) = \boldsymbol{e}_c(k+1) - \boldsymbol{e}_c(k)$. Using the approximation

$$\Delta\boldsymbol{e}_c(k+1) = \frac{\partial \boldsymbol{e}_c(k+1)}{\partial \boldsymbol{u}_p(k)}\Delta\boldsymbol{u}_p(k) \tag{4.33}$$

and the fact that $\frac{\partial \boldsymbol{e}_c(k+1)}{\partial \boldsymbol{u}_p(k)} = \frac{\partial \hat{\boldsymbol{y}}(k+1)}{\partial \boldsymbol{u}_p(k)}$, $\Delta V(k)$ can be rewritten as

$$\Delta V(k) = \frac{1}{2}\left[\frac{\partial \hat{\boldsymbol{y}}(k+1)}{\partial \boldsymbol{u}_p(k)}\Delta\boldsymbol{u}_p(k)\right]^T(2\boldsymbol{e}_c(k+1) + \frac{\partial \hat{\boldsymbol{y}}(k+1)}{\partial \boldsymbol{u}_p(k)}\Delta\boldsymbol{u}_p(k)) \tag{4.34}$$

Substituting from (4.24) yields

$$\Delta V(k) = \frac{1}{2}\,\|\Delta \boldsymbol{e}_c(k)\|^2 - \frac{\eta\xi}{(1 - N_u\eta\rho)}\left[\frac{\partial \hat{\boldsymbol{y}}(k+1)}{\partial \boldsymbol{u}_p(k)}\sum_{j=1}^{N_p}\left[\frac{\partial \hat{\boldsymbol{y}}(k+j)}{\partial \boldsymbol{u_p}(k)}\right]^T \boldsymbol{e}_c(k+j)\right]^T \boldsymbol{e}_c(k+1) \tag{4.35}$$

To have a negative definite $\Delta V(k)$ we require

$$\frac{1}{2}\,\|\Delta \boldsymbol{e}_c(k)\|^2 < \frac{\eta\xi}{(1 - N_u\eta\rho)} \times \sum_{j=1}^{N_p} \boldsymbol{e}_c(k+j)^T \frac{\partial \hat{\boldsymbol{y}}(k+j)}{\partial \boldsymbol{u_p}(k)}\left[\frac{\partial \hat{\boldsymbol{y}}(k+1)}{\partial \boldsymbol{u}_p(k)}\right]^T \boldsymbol{e}_c(k+1) \tag{4.36}$$

Using (4.30)

$$\sum_{j=1}^{N_p} \boldsymbol{e}_c(k+j)^T \frac{\partial \hat{\boldsymbol{y}}(k+j)}{\partial \boldsymbol{u_p}(k)}\left[\frac{\partial \hat{\boldsymbol{y}}(k+1)}{\partial \boldsymbol{u}_p(k)}\right]^T \boldsymbol{e}_c(k+1) \geq N_p\lambda_{min}k_e \tag{4.37}$$

And using (4.24) and (4.29)

$$\begin{aligned}\|\Delta \boldsymbol{e}_c(k)\|^2 &= \left(\frac{\eta\xi}{1 - N_u\eta\rho}\right)^2 \\ &\times \sum_{j=1}^{N_p} \boldsymbol{e}_c(k+j)^T \frac{\partial \hat{\boldsymbol{y}}(k+j)^T}{\partial \boldsymbol{u_p}(k)}\, P_1^T \sum_{j=1}^{N_p}\left[\frac{\partial \hat{\boldsymbol{y}}(k+j)}{\partial \boldsymbol{u_p}(k)}\right]^T \times \boldsymbol{e}_c(k+j) \\ &= \left(\frac{\eta\xi}{1 - N_u\eta\rho}\right)^2 \times k_\Delta\end{aligned} \tag{4.38}$$

By combining (4.36), (4.37) and (4.38), to have negative definite $\Delta V(k)$ we require

$$\eta < \frac{N_p k_e \lambda_{min}}{k_\Delta \xi + \rho N_u N_p k_e \lambda_{min}} \tag{4.39}$$

Using (4.38), the upper bound of k_Δ can be calculated as

$$k_\Delta \leq N_p k_e \lambda_{max} \tag{4.40}$$

Therefore, to have negative definite $\Delta V(k)$, the upper bound of learning rate is

$$\eta < \frac{\lambda_{min}}{\xi\lambda_{max} + \rho N_u \lambda_{min}} \tag{4.41}$$

4.4 Simulation Results

A simple model of an unmanned autonomous vehicle is presented in Figure 20. The system has two control inputs, the speed of autonomous vehicle $v(n)$ and the steering angle $\alpha(k)$. The dynamics of the autonomous vehicle is completely controllable through the two control inputs.

The input-output equation of the vehicle is described by

$$\boldsymbol{x}(n+1) = \begin{bmatrix} x(n+1) \\ y(n+1) \\ \theta(n+1) \end{bmatrix} = \begin{bmatrix} x(n) + Tv(n)cos(\theta(n)cos(\alpha(n))) \\ y(n) + Tv(n)sin(\theta(n)cos(\alpha(n))) \\ \theta(n) + Tv(n)sin(\alpha(n)))/D \end{bmatrix} + \boldsymbol{\omega}(n) \tag{4.42}$$

where T is the sampling period, D is the vehicle length and $\boldsymbol{\omega}(n)$ is white measurement noise. The sampling period is chosen as $T = 5ms$ and the vehicle length is assumed to be $D = 300cm$ [61].

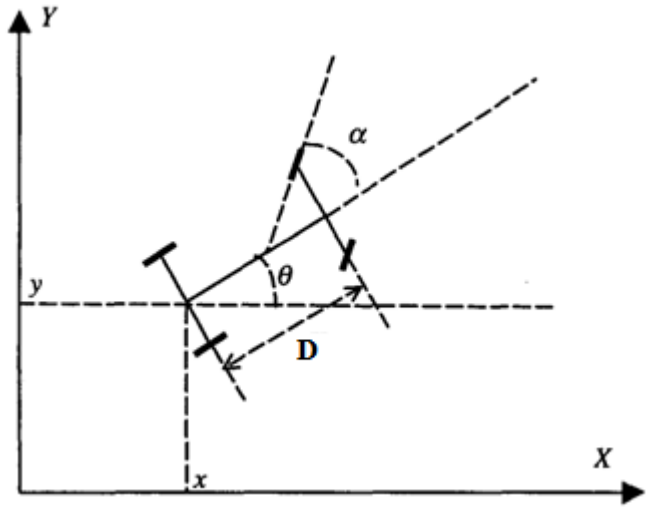

Figure 20. Autonomous vehicle [36]

To demonstrate the performance of model predictive control using wavelet neural network with feedforward component, we apply our networked control methodology to the autonomous vehicle. We present two simulations scenarios. In the first scenario, the delay from sensor to model and delay form controller to actuator are fixed. This corresponds to the case of a private network for the control system. In the second scenario, network delay is random with a triangular probability distribution function. In both scenarios, measurement noise is assumed to be Gaussian white noise with a variance of 0.1. In both scenarios delayed measurements are discarded.

In both scenarios, all the network parameters are initialized with random values from a normal distribution with variance 0.5 and the WNN is not pretuned. The input to the wavelet neural network is $\boldsymbol{u}_N(k) = [v(k), \alpha(k), x(k), y(k), \theta(k)]^T$ and the target output of the wavelet neural network is $[x(k+1), y(k+1), \theta(k+1)]^T$. The optimum number of hidden layer nodes was found to be $m = 5$. The learning rate of the network parameters is assumed to be $\gamma = 0.1$. In the controller cost function, the weight of the control inputs is $\rho_1 = \rho_2 = 0.1$ and the weight of the prediction errors is $\xi_j = 0.01, j \in \{1, \dots, J\}$. A tradeoff between control input magnitude, tracking

performance and computational burden is needed to choose appropriate values for N_p and N_u. In this simulations, $N_p = N_u = 15$ was found to yield satisfactory performance. The autonomous vehicle is assumed to have a length $D = 300cm$ and the sampling period is chosen as $T_s = 5ms$.

4.4.1 Fixed Network Delay

In the first scenario, both sensor to controller and controller to actuator network delays are assumed to be $0.1s$. This is a large delay for a sampling period of $T_s = 5ms$ where the MPC receives the measurement and the actuator receives the control action after a delay of 20 sampling periods. To mitigate the effect of this large network delay, the controller predicts 55 samples and uses the last $N_p = 15$ samples to calculate the control action. This provides good compensation for the network delays, assuming that the prediction accuracy is satisfactory and allows the autonomous vehicle to follow the desired trajectory.

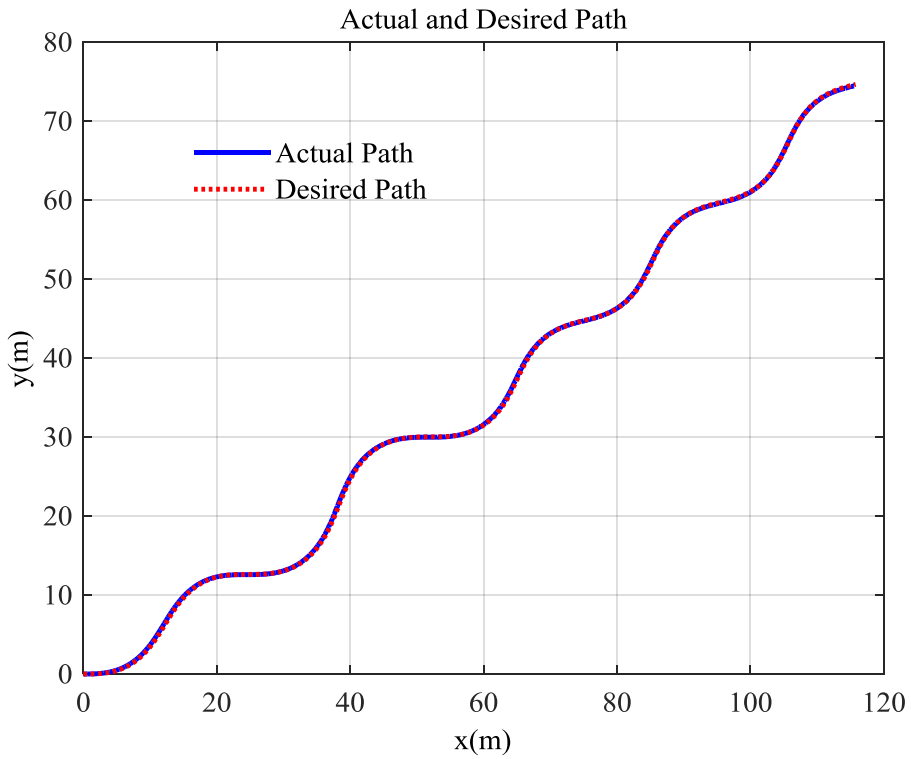

Figure 21 . Tracking of a curved line

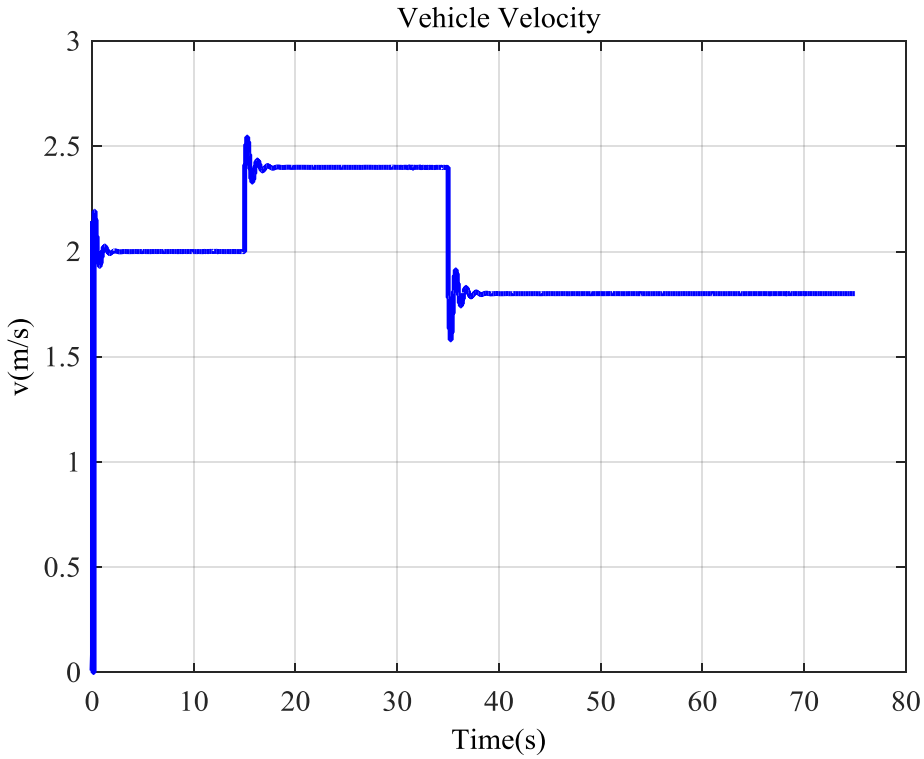

Figure 22. Vehicle Velocity Calculated by Model Predictive controller

Figure 21. shows a desired curved path together with the vehicle path as it tracks it. Figure 22 and Figure 23 show the control inputs for the vehicle. The vehicle must increase and decrease its speed at the appropriate time to be able to track the desired path. Figure 22 shows that using extended prediction and control horizons, controller is able to increase and decrease the velocity at the right time to minimize the tracking error. Figure 24 shows the tracking error of the curved path. At time $t = 0$, because there is no pretuning on the network parameters, the tracking error is large, but as time progresses the controller identifies the model of the system and reduces the tracking error. The tracking error after 70 seconds is about $20cm$ in the x and y directions, which is small in comparison to vehicle length of $D = 300cm$. The tracking error decreases slowly due to large network delay and the online identification process but it asymptotically approaches zero after the shown simulation period.

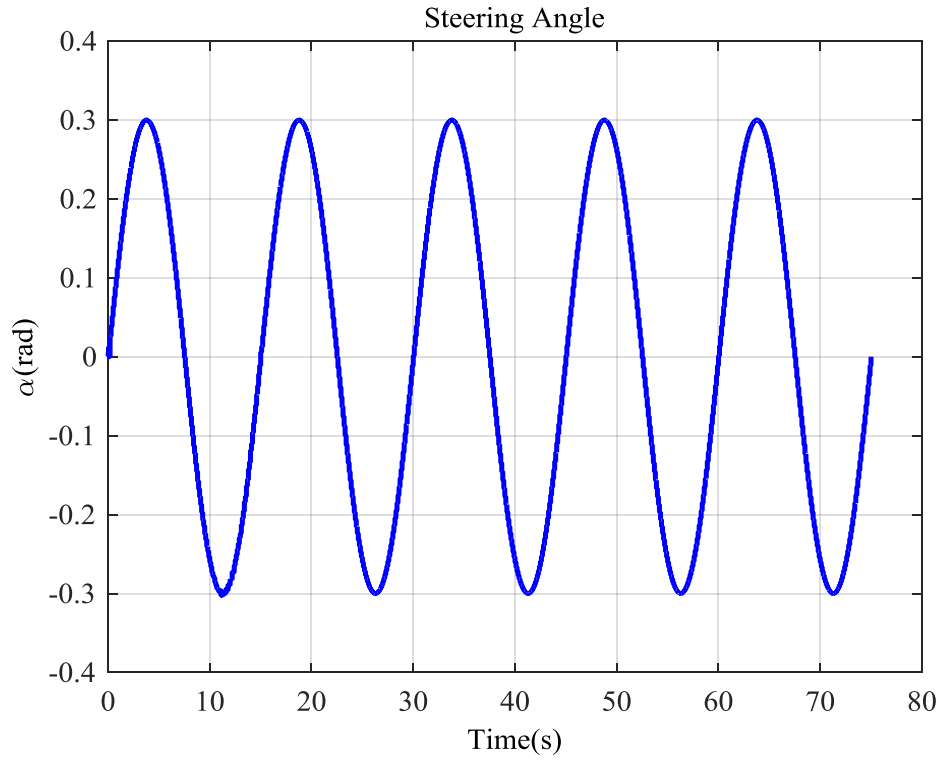

Figure 23 . Steering angle Calculated by Model Predictive controller

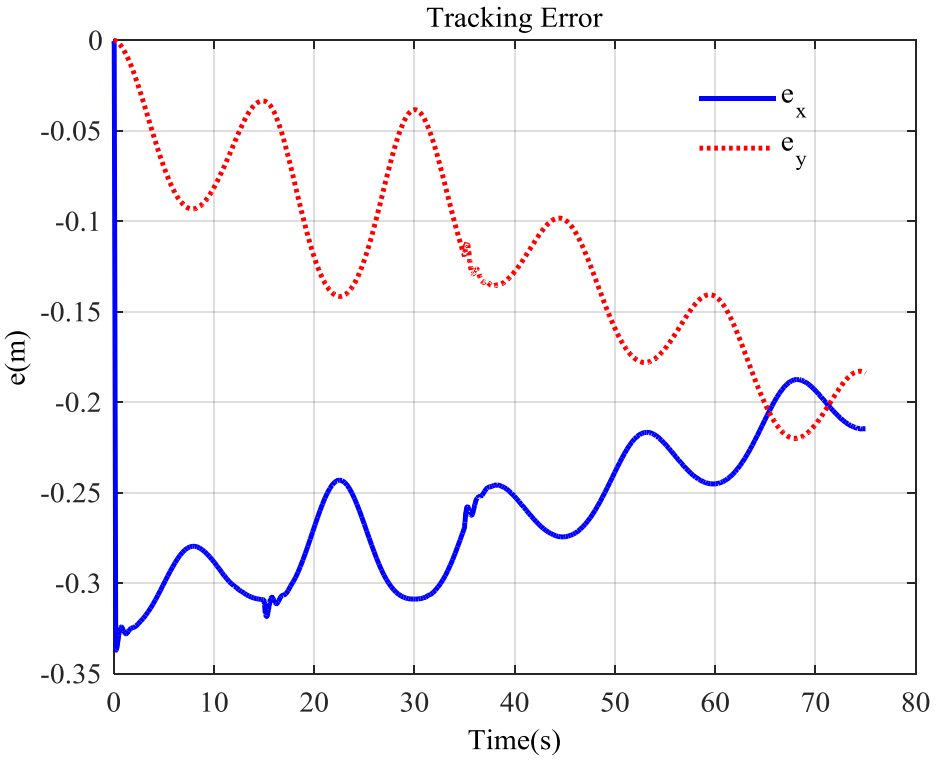

Figure 24 . Tracking error of curved path

4.4.2 Random Network Delay

In the second scenario, network delay is random with triangular probability density function. Thus, only the upper and lower bound of the network delay are known. The packet loss occurs in the network. Packet loss and network delay can be combined and considered as a single delay both for sensor to MPC or for controller to actuator. Therefore, network parameters are updated every τ seconds with τ having a triangular probability density function. The sum of network delay and packet loss follows a triangular probability density function with lower limit of $a = 0.005$, upper limit $b = 0.1$, and mode $c = 0.05$. For a sampling period of $T_s = 5ms$, the minimum delay both from sensor to MPC and from MPC to actuator is at least one sampling period. To compensate for the network delay and packet loss in the network, the controller predicts 35 samples beyond the current time, which is equal to twice the mean network delay. This improves the tracking performance of the controller.

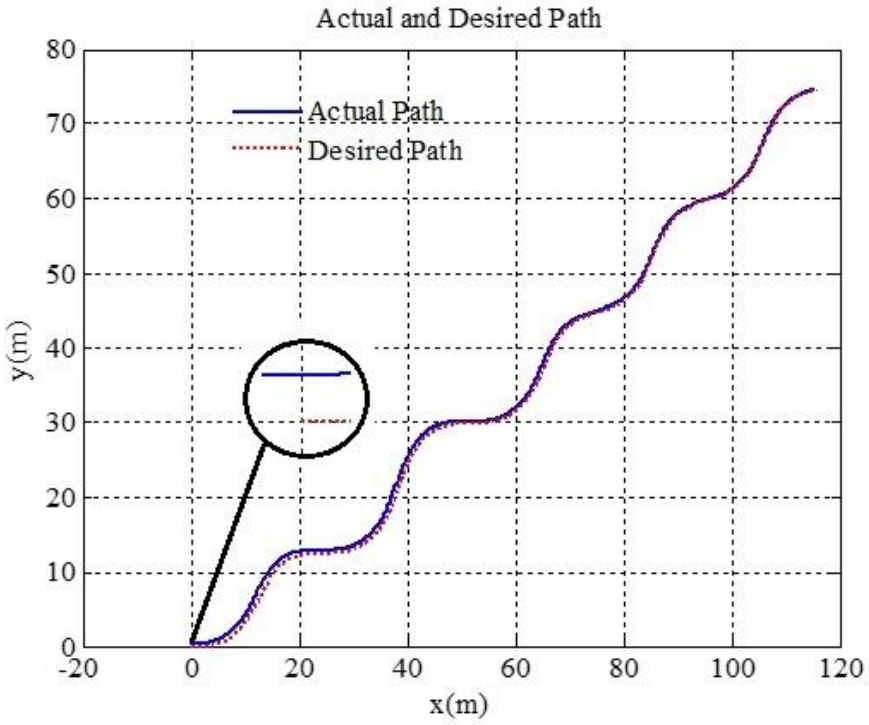

Figure 25. Tracking of a curved line

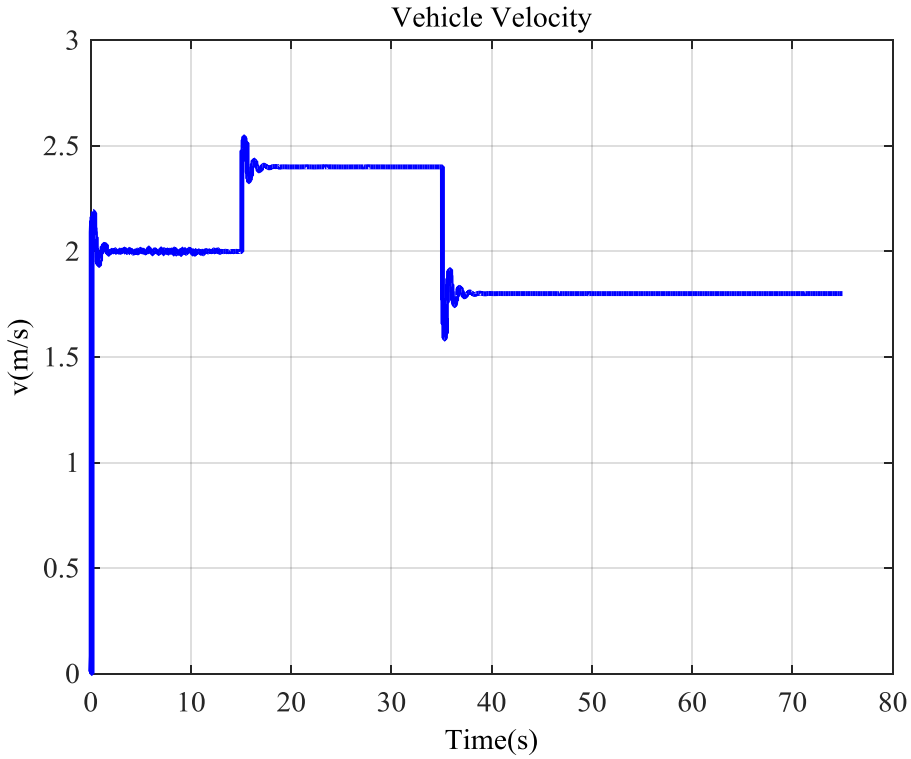

Figure 26. Vehicle velocity

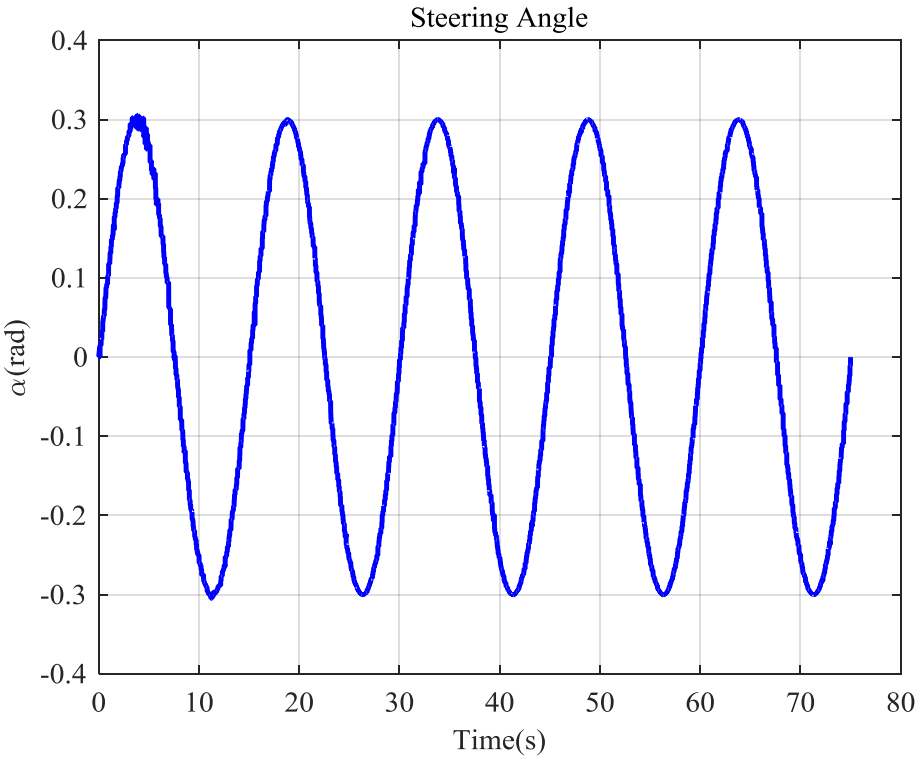

Figure 27. Steering angle

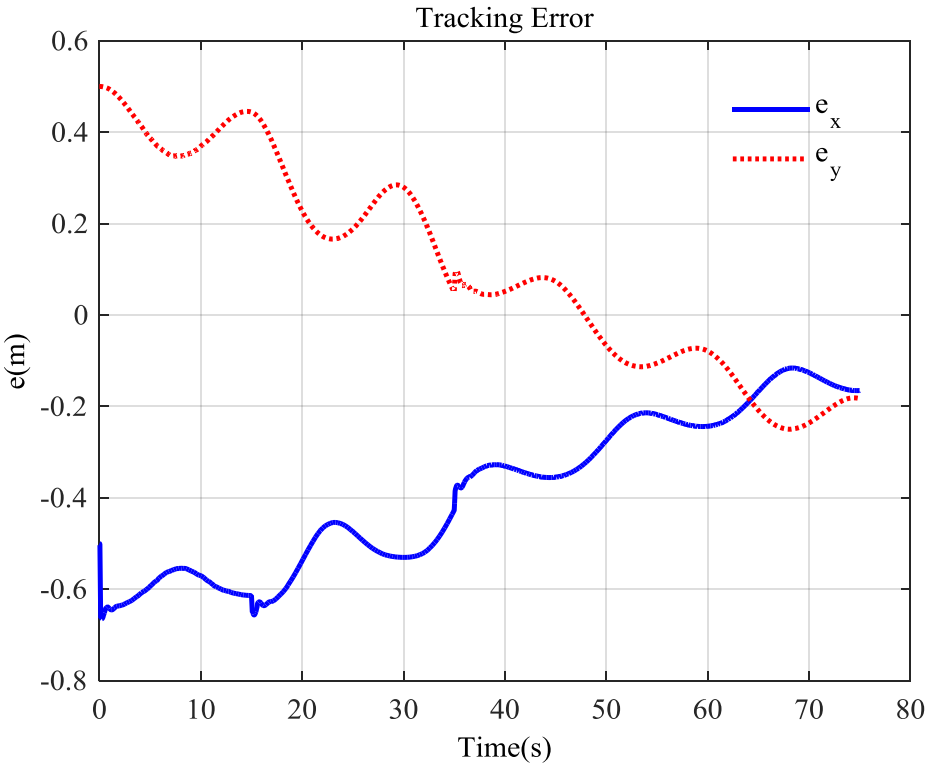

Figure 28. Tracking error for random delay scenario

Figure 25 shows the tracking result for random network delay. The vehicle is not on the desired path at the beginning of the simulation but the controller can successfully guide the vehicle toward desired path and track it. Since the vehicle is not on the desired track and the WNN was not pretuned, there is a big tracking error at $t = 0$. However, the network quickly learns the behavior of vehicle and the controller reduces the error. Figure 26 and Figure 27 show the control input calculated by the MPC. As in the first scenario, the controller produces smooth control action to track the desired path. The vehicle velocity is dependent on the desired path which may require it to speed up in some parts and slow down in other parts as shown in Figure 26 Figure 27 shows the smooth steering angle calculated by the controller. Figure 28 shows the tracking error for random network delay scenario. The final tracking error in the x direction is about $e_x = 16cm$ and final tracking error in y direction is $e_y = 18cm$. Considering the vehicle length of $D = 300cm$ and large network delay and packet loss, the error is acceptable.

Table 6. shows the mean square identification error of x,y and θ. The very small mean square identification error shows that the wavelet neural network with feedforward component can efficiently identify the model of autonomous vehicle. In the random network delay scenario, packet

loss and the initial position error of the vehicle increase the mean square identification error. Nevertheless, the error is still very small and the network effectively identifies the model of the system.

Table 6. Mean square identification error

	x	y	θ
Fixed delay	0.0363	0.0281	0.0152
Random delay	0.0411	0.0332	0.0211

4.5 Conclusion

In this study we used model predictive controller along with wavelet neural network with feedforward component to online identification and control of nonlinear system. The feedforward component reduces the number of hidden layer nodes and accelerates the learning and therefore, makes the model more suitable for online identification and control applications. Model predictive controller uses the wavelet neural network with feedforward component to predict the future outputs of the plant over extended prediction horizon. By optimization of controller cost function over extended prediction horizon, controller finds the future control inputs. Simulation results show that this methodology can compensate the effect of fixed and random network delay and packet loss in the network and provide a satisfactory tracking performance. The Lyapunov theory is used to prove the stability of the model predictive controller. Future work will be application of the methodology to unmanned aerial vehicle.

Chapter 5. Training Recurrent Neural Networks as a Constraint Satisfaction Problem

5.1 Introduction

After Minsky and Papert showed that two-layer perceptron cannot approximate functions generally [138], it took nearly a decade for researchers to show that multilayer feedforward neural networks are universal approximators [2]. Since then, neural networks have been successfully used in various science and engineering applications ,[138]. However, training and learning the internal structure of neural networks has remained a challenging problem for researchers.

Training neural networks requires solving a nonlinear non-convex optimization problem and researchers have proposed different approaches to solving it [6]. Classical optimization methods were the first methods used for training neural networks. The most widely used training algorithm is error backpropagation which minimizes an error function using the steepest decent algorithm [139]. Although error backpropagation is easy to implement, it has all the disadvantages of Newton-based optimization algorithms including slow convergence rate and trapping in local minima. Local minima can decrease the generalization ability of the neural network [6],[139].

To cope with these deficiencies, researches proposed other training methods such as supervised learning and global optimization approaches [8],[9],[10]. Supervised learning approaches learn the internal structure of the neural network while learning internal weights of the neural network. Learning the internal structure of the neural network makes these approaches more efficient and less reliant on parameters selected by the user [13][14].

Researchers proposed different supervised learning methods such as the tiling algorithm, cascade-correlation algorithm, stepnet, and the scaled conjugate algorithm, among others [14]. While in incremental supervised learning approaches network size grows in the training phase which may result in over-fitting, some supervised learning approaches prune the over-fitted network during training [142],[143],[144],[145]. However, few of these methods have been successfully applied to large scale practical problems [14]. This is in contrast to conjugate gradient methods which are attractive for large scale problems due to their fast convergence rate [16]. Quasi-Newton methods are a sophisticated alternative to conjugate gradient methods for supervised learning, although their reliance on exact approximation of the Hessian matrix makes them inefficient in some applications [17].

Global optimization methods are another alternative to cope with deficiencies of Newton-based methods and learn the internal structure of neural networks. Genetic algorithms and simulated annealing have been widely used to train neural networks and optimize network structure [146],[147]. These approaches assume that the quality of the network is related to network topology and parameters. Alopex is another global optimization approach which trains the network using the correlation between changes in weights and changes in the error function. Due to local computations of the Alopex, it is more suitable for parallel computation [148].

Taboo search is another stochastic approach which has been frequently used to train neural networks. It can find the optimal or near optimal solution of the optimization problem [149]. Implementation of taboo search is easier than most global optimization methods and the method is generally applicable to a wide variety of optimization problems [150].

Researchers have used a combination of global optimization methods for training neural networks. GA-SA is a combination of a genetic algorithm and simulated annealing. GA-SA uses a genetic algorithm to make simulated annealing faster to reduce the training time [151],[152],[153]. NOVEL is another hybrid approach which uses a trajectory-based method to find feasible regions of the solution space and then locates local the minima in the feasible regions by local search [154],[155],[156],[157].

Backpropagation through time is an adaptation of traditional error backpropagation for recurrent neural networks [158],[163],[164][165]. Real time recurrent learning is another approach to train recurrent neural networks that is effective for training small networks but suffers from huge computational complexity in large applications [166]. Martens and Sutskever used Hessian free optimization with a damping scheme to effectively train recurrent neural networks [167]. Monner proposed generalized long short-Term Memory (LSTM)- as a training algorithm for second order recurrent neural networks [168]. The method is applicable to arbitrary second order recurrent networks. Lu et al. used low rank factorizations and parameter sharing schemes to train recurrent neural networks and showed that their approach effectively reduces the parameters of LSTM [169].

Although global optimization methods have been applied for training neural networks, there are other promising global optimization approaches that have not been used for neural network training. Quotient gradient method is a trajectory based method to find feasible solutions of

constraint satisfaction problems. QGS searches for the feasible solutions of the CSP along the trajectories of a nonlinear dynamical system [30].

This study exploits QGS to train artificial neural networks by transforming the training data set into a CSP, then transforms the resulting CSP to an unconstrained minimization problem. After constructing the unconstrained minimization problem, the nonlinear QGS dynamical system is defined. Using the fact that the equilibrium points of the QGS are local minima of the unconstrained minimization problem, a neural network can be trained by integrating QGS over time until it reaches an equilibrium point. The method is easy to implement because constructing the nonlinear dynamical system is similar to deriving the equations of the steepest descent algorithm. The algorithm finds multiple local minima of the optimization by forward and backward integration of the QGS. This provides an easy and straightforward approach to find multiple local minima of the optimization problem. However, like other global optimization methods, finding local minima takes more time than Newton-based methods. Numerical examples show that QGS outperforms error backpropagation and a genetic algorithm and the resulting network has better generalization capability. A preliminary version of the study which compares the method with error backpropagation was presented in [159].

Solving optimization problems with different initial points is one of the approaches to cope with local minima in Newton-based methods. However, the selected initial points may be in the stability region of the same stable equilibrium point, which makes this approach inefficient. QGS uses backward integration to escape from the stability region of a stable equilibrium point, then enters the stability region of another equilibrium point with forward integration. This allows QGS to explore a bigger region in its search for local minima. The simple implementation, along with the global optimization property of QGS justify its use as a new training method for artificial neural networks.

5.2 Quotient Gradient System

CSP is an active field of research in artificial intelligence and operations research. Lee and Chiang [30], used the trajectories of a nonlinear dynamical system to find the solutions of the CSP. This section reviews their work that forms the basis for our new approach to neural network training

Consider a system of nonlinear equality and inequality constraints

$$C_{\mathrm{I}}(\boldsymbol{y}) < 0$$
$$C_{\mathrm{E}}(\boldsymbol{y}) = 0, \boldsymbol{y} \in R^{\mathrm{n-l}} \tag{5.1}$$

To guarantee the existence of the solution of this CSP, C_{I} and C_{E} are assumed to be smooth. The CSP can be transformed into the unconstrained optimization problem

$$\min_{x} f(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{h}(\boldsymbol{x})\|^2, \qquad \boldsymbol{x} = (\boldsymbol{y}, \boldsymbol{s}) \in R^{\mathrm{n}} \tag{5.2}$$

$$\boldsymbol{h}(\boldsymbol{x}) = \begin{bmatrix} C_{\mathrm{I}}(\boldsymbol{y}) + \hat{\boldsymbol{s}}^2 \\ C_{\mathrm{E}}(\boldsymbol{y}) \end{bmatrix} \in R^{\mathrm{m}}, \hat{\boldsymbol{s}}^2 = (s_1^2, \dots, s_l^2)^{\mathrm{T}} \tag{5.3}$$

where the slack variable $\hat{\boldsymbol{s}}$ has been introduced to transform the inequality constraints to equality constraints. The global minimum of (5.2) is the optimal solution of the original CSP. The QGS is a nonlinear dynamical system of equations defined based on the constraint set as

$$\dot{\boldsymbol{x}} = F(\boldsymbol{x}) = -\nabla f(\boldsymbol{x}) \coloneqq -D_x \boldsymbol{h}(\boldsymbol{x})^{\mathrm{T}} \boldsymbol{h}(\boldsymbol{x}) \tag{5.4}$$

Lee and Chiang showed that stable equilibrium points of the QGS are local minimums of unconstrained minimization problem (5.2) which are possible feasible solutions of the original CSP. A solution of the QGS starting from initial point $\boldsymbol{x}(0)$ at initial time $t = 0$ is called a trajectory or orbit. An equilibrium manifold is a path connected component of $F^{-1}(0)$. Assuming that $\boldsymbol{\phi}(., \boldsymbol{x}): R \to R^{\mathrm{n}}$ is an orbit of the QGS, an equilibrium manifold $\sum$ of the QGS is *stable* if $\forall \epsilon > 0$ there exist $\delta(\epsilon) > 0$ such that

$$\boldsymbol{x} \in B_{\delta}(\sum) \Longrightarrow \boldsymbol{\phi}(t, \boldsymbol{x}) \in B_{\epsilon}(\sum), \forall t \in R \tag{5.5}$$

where $B_{\delta}(\sum) = \{\boldsymbol{x} \in R^{\mathrm{n}}: \|\boldsymbol{x} - \boldsymbol{y}\| < \delta, \forall \delta \in R^{\mathrm{n}}\}$. If δ can be chosen such that

$$\boldsymbol{x} \in B_{\delta}(\sum) \Longrightarrow \lim_{\mathrm{t}\to\infty} \boldsymbol{\phi}(t, \boldsymbol{x}) \in B_{\epsilon}(\sum) \tag{5.6}$$

the equilibrium manifold is *asymptotically stable*. An equilibrium manifold $\sum$ which is not stable is *unstable*. An equilibrium manifold is *pseudo-hyperbolic* if $\forall \boldsymbol{x} \in \sum$, the Jacobian of $F(.)$ at $\boldsymbol{x}$ has no eigenvalues with a zero real part on the normal space of $\sum$ at $\boldsymbol{x} \in R^{\mathrm{n}}$ and there exist $\epsilon > 0$ such that $\Phi_{-\infty}: B_{\epsilon}(\sum) \to \sum$ is locally homeomorphic to projection from R^{n} to R^{l} with l the dimension of the equilibrium manifold. The stability region of the stable equilibrium manifold is an open, connected and invariant set and is defined as

$$A(\sum\nolimits_{\mathrm{s}}) = \{\boldsymbol{x} \in R^{n} : \lim_{\mathrm{t}\to\infty} \boldsymbol{\phi}(t, \boldsymbol{x}) \in \sum\nolimits_{\mathrm{s}}\} \tag{5.7}$$

The boundary of a stable equilibrium manifold $\sum_{\mathrm{s}}$ is the *stability boundary* and is denoted by $\partial A(\sum_{\mathrm{s}})$. QGS is assumed to satisfy the following assumptions.

Assumptions: let $\sum_s$ be stable equilibrium manifold of QGS

(A1) If an equilibrium manifold $\sum$ has nonempty intersection with $\partial A(\sum_s)$ then $\sum \subset \partial A(\sum_s)$

(A2) All the equilibrium manifolds on $\partial A(\sum_s)$ are pseudo-hyperbolic and have the same dimension

(A3)The stable and unstable manifolds of equilibrium manifolds on $\partial A(\sum_s)$ satisfy the transversality condition.

(A4) The function H satisfies one of the following

(1) $\|\boldsymbol{h}(\boldsymbol{x})\|$ is a proper map

(2) For any $\gamma > 0$ and any closed subset K of

$$\{\boldsymbol{x} \in R^{\mathrm{n}} : \|\boldsymbol{h}(\boldsymbol{x})\| \leq \gamma,\ D\boldsymbol{h}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{h}(\boldsymbol{x}) \neq 0\}, \inf\{\|D\boldsymbol{h}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{h}(\boldsymbol{x})\| : \boldsymbol{x} \in K\} > 0$$

where $D\boldsymbol{h}(\boldsymbol{x})$ denotes the gradient of $\boldsymbol{h}(\boldsymbol{x})$.

The transversality condition of assumption A3 is defined as follows. Let M_1 and M_2 be manifolds in R^{n} of codimensions m_1 and m_2. We say that M_1 and M_2 intersect transversally if (i) for every $\bar{x} \in M_1 \cap M_2$ there exist an open neighborhood $U_{\bar{x}}$ of $\bar{x}$, and (ii) a system of functions $(h_1, \dots, h_{\mathrm{m}_1})$ for $M_1 \cap U_{\bar{\mathrm{x}}}$ and $(\rho_1, \dots, \rho_{\mathrm{m}_2})$ for $M_2 \cap U_{\bar{\mathrm{x}}}$ such that the set $\{Dh_i(\boldsymbol{x}), D\rho_j(\boldsymbol{x}), i = 1, \dots, m_1, j = 1, \dots, m_2\}$ is linearly independent for all $\boldsymbol{x} \in M_1 \cap M_2 \cap U_{\bar{\mathrm{x}}}$ (Jongen et al. 2001). The following theorem assures the stability of QGS and redefines the stability boundary under assumptions A1-A4.

Theorem 1 [30]: Let $\sum_s$ be a stable equilibrium manifold of QGS and suppose that assumptions A1-A4 hold. Then we have the following:

1) The QGS is completely stable, i.e., every trajectory of QGS converges to an equilibrium manifold
2) Let $\{\sum_{\mathrm{i}} : i = 1,2,..\}$ be the set of all equilibrium manifolds on $\partial A(\sum_s)$, then $\partial A(\sum_s) = \bigcup_i W^{\mathrm{s}}(\sum_{\mathrm{i}})$

   where $W^{\mathrm{s}}(\sum)$ is a stable manifold of pseudo-hyperbolic equilibrium manifold and is defined as

$$W^{\mathrm{s}}(\sum) = \{\boldsymbol{x} \in R^{\mathrm{n}} : \lim_{\mathrm{t}\to\infty} \phi(t, \boldsymbol{x}) \in \sum\} \tag{5.8}$$

Theorem 2 (Lee & Chiang 2001): Consider the CSP and its associated quotient gradient system. If assumptions (A1-A4) hold, then we have the following

I. Each path component of the solution set of the CSP is a stable equilibrium manifold of the QGS

II. If $\sum$ is a stable equilibrium manifold of the QGS, then $\sum$ consists of non-isolated local minima of the following minimization problem

$$\min_{x \in R^{\mathrm{n}}} V(\boldsymbol{x}) \tag{5.9}$$

where $V: R^{\mathrm{n}} \rightarrow R$ is defined as $V(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{h}(\boldsymbol{x})\|^2$

III. If $n > 2m - 1$ then $\sum$ is a component of the solution set of the CSP if and only if $\sum$ is an $n - m$ dimensional stable equilibrium manifold of the QGS

A stable equilibrium manifold of the QGS may not be in the feasible region of the CSP. In such cases, the QGS must escape from this equilibrium manifold and enter the stability region of another stable equilibrium manifold. If the new equilibrium manifold is not in the feasible region, this process is repeated until the QGS enters the stability region of a feasible equilibrium manifold or until it satisfies a stopping criterion. Once a feasible manifold is reached, QGS is integrated over time until an equilibrium point is reached. To escape from the basin of attraction of a stable equilibrium point, QGS is integrated backward in time until an unstable point is reached. Thus, solving the optimization problem becomes a series of forward and backward integrations of the QGS until the stopping criteria is satisfied [248].

5.3 Neural Networks

Function approximation is required in many fields of science and engineering. Neural networks are general function approximators and have been successfully applied to different function approximation applications [6]. Based on the nature of the application, researchers have developed different versions of neural networks such as feedforward networks, recurrent neural networks, liquid state networks and wavelet networks among the others [161].

In this study, we consider a three-layer fully recurrent neural network with smooth activation functions. Figure 29 illustrates the internal structure of the neural network.

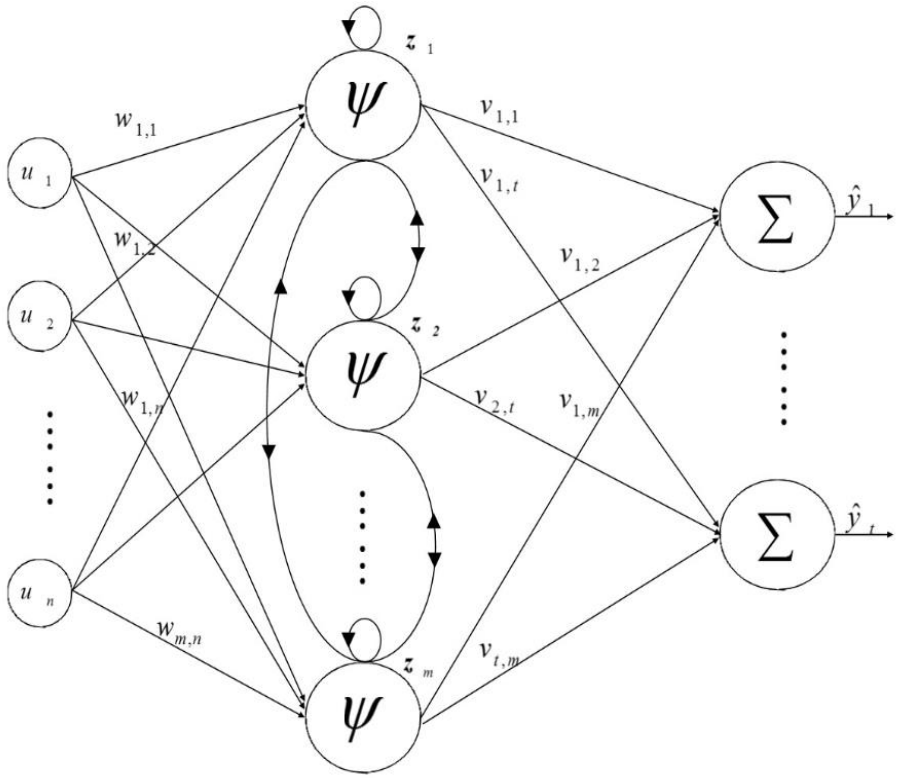

Figure 29. Artificial neural network structure

The network has input $\boldsymbol{u}(k) = [u_1(k) \quad ... \quad u_n(k)]^{\mathrm{T}}$, internal state $\boldsymbol{z}(k) = [z_1(k) \quad ... \quad z_m(k)]^{\mathrm{T}}$ and output $\hat{\boldsymbol{y}}(k) = [\hat{y}_1(k) \quad ... \quad \hat{y}_t(k)]^{\mathrm{T}}$. The input-output equation of the network is described as

$$\begin{aligned} \boldsymbol{z}(k) &= \boldsymbol{\psi}\big(W\boldsymbol{u}(k) + S\boldsymbol{z}(k-1)\big) \\ \hat{\boldsymbol{y}}(k) &= V\,\boldsymbol{z}(k) \end{aligned} \tag{5.10}$$

W, S and V are network weights matrices whose size is dependent on the number of network inputs, outputs and hidden layer nodes. For a network with n inputs, t outputs and m hidden layer nodes, $W \in R^{m\times n}$, $S \in R^{m\times m}$ and $V \in R^{t\times m}$. The cost function for training neural network is the traditional sum of squared errors (SSE)

$$SSE = \sum_{k=1}^{N} \boldsymbol{e}(k)^{\mathrm{T}}\boldsymbol{e}(k) = \sum_{k=1}^{N} \big(\hat{\boldsymbol{y}}(k) - \boldsymbol{y}(k)\big)^{\mathrm{T}} (\hat{\boldsymbol{y}}(k) - \boldsymbol{y}(k)) \tag{5.11}$$

where $\hat{\boldsymbol{y}}(k)$is the network output, $\boldsymbol{y}(k)$ is the measured output, and N is the total number of training samples.

5.4 Applying QGS to Neural Network Training

Solving the CSP is equivalent to an unconstrained minimization problem (5.2). QGS is a trajectory-based method to find the local minima of (5.2) which are the possible feasible solutions of the CSP. To train neural networks using QGS, we consider the training set as equality constraints of the CSP and then transform the CSP into unconstrained minimization problem as (5.2) and then we use the second part of Li and Chiang's work which is equilibrium points of QGS are local minimums of unconstrained minimization problem. If N measurements are available, the CSP can be written as

$$\begin{aligned} &\boldsymbol{h}(\boldsymbol{x}) = [h_{\mathrm{i}}(\boldsymbol{x})], i = 1,2, \dots, N \\ &h_{\mathrm{i}}(\boldsymbol{x}) = V\boldsymbol{\psi}\big(W\boldsymbol{u}(i) + S\boldsymbol{z}(i-1)\big) - y(i) \end{aligned} \tag{5.12}$$

The network state vector $\boldsymbol{x}$ includes all the network parameters, i. e, all entries of V, W and S**.** More specifically, if we partition V, W and S as

$$V = \begin{bmatrix} \boldsymbol{v}_1^{\mathbf{T}} \\ \vdots \\ \boldsymbol{v}_{\mathrm{m}}^{\mathbf{T}} \end{bmatrix}_{t \times m} \quad W = \begin{bmatrix} \boldsymbol{w}_1^{\mathbf{T}} \\ \vdots \\ \boldsymbol{w}_{\mathrm{m}}^{\mathbf{T}} \end{bmatrix}_{\mathrm{m} \times \mathrm{n}} \quad S = \begin{bmatrix} \boldsymbol{s}_1^{\mathbf{T}} \\ \vdots \\ \boldsymbol{s}_{\mathrm{m}}^{\mathbf{T}} \end{bmatrix}_{\mathrm{m} \times \mathrm{m}} \tag{5.13}$$

then $\boldsymbol{x}$ is defined as

$$\begin{aligned} &\boldsymbol{x} = [x_i]_{\mathrm{n_p} \times 1} = [\boldsymbol{v}_1, .., \boldsymbol{v}_{\mathrm{m}}, \boldsymbol{w}_1, \dots, \boldsymbol{w}_{\mathrm{m}}, \boldsymbol{s}_1, \dots, \boldsymbol{s}_{\mathrm{m}}]^{\mathbf{T}} \\ &n_p = m^2 + m \times (n + t) \end{aligned} \tag{5.14}$$

Since the training set does not contain any inequality constraints, slack variables are not needed. Using the training set, the QGS for training the neural network can be defined as

$$\dot{\boldsymbol{x}} = -\boldsymbol{f}(x) = -D_{\mathrm{x}}\boldsymbol{h}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{h}(\boldsymbol{x}) \tag{5.15}$$

Where

$$D_{\mathrm{x}}\boldsymbol{h}(x) = \begin{bmatrix} \dfrac{\partial h_1(\boldsymbol{x})}{\partial \boldsymbol{x}} \\ \vdots \\ \dfrac{\partial h_{\mathrm{N}}(\boldsymbol{x})}{\partial \boldsymbol{x}} \end{bmatrix}_{\mathrm{N} \times \mathrm{n_p}} \tag{5.16}$$

To train neural network using QGS, we use the fact that the equilibrium points of QGS are local minima of the unconstrained minimization problem. Therefore the algorithm needs to find an equilibrium point of QGS and then escape from that equilibrium point and move toward another equilibrium point of QGS. The first step is to integrate the QGS from a starting point, which need not be feasible, to find an equilibrium point. Next, we escape from the stability region of the stable

equilibrium point to an unstable point with backward integration of QGS in time. The eigenvalues of the Jacobian matrix can be used as a measure of stability and instability. The algorithm continues until it cannot find any new equilibrium point or until it satisfies the stopping criterion.

Because neural network training has equilibrium points which can be considered as zero-dimensional equilibrium manifolds, assumptions A1, A2 and A3 hold. When the activation function of the neural network is a one-one invertible function, $\|\boldsymbol{h}(\boldsymbol{x})\|$ is a proper map. Assumption 4 also holds because the QGS is asymptotically stable and $\|\boldsymbol{h}(\boldsymbol{x})\|$ is proper.

5.5 Stability Analysis

Any training algorithm must be stable, even in presence of measurement error and uncertainties. We use Lyapunov stability theory to prove the asymptotic stability of equilibrium points and their asymptotic stability in the presence of measurement errors.

Theorem 3: The equilibrium points of the quotient gradient system are asymptotically stable

Proof : Consider the Lyapunov function

$$V(\boldsymbol{x}) = \boldsymbol{h}^{\mathrm{T}}(\boldsymbol{x})\boldsymbol{h}(\boldsymbol{x}) \tag{5.17}$$

$V(\boldsymbol{x})$ is a locally positive definite function of the state that is equal to zero at global optima of the optimization problem. Thus, $V(\boldsymbol{x})$ is a locally positive definite function in the vicinity of each equilibrium point. The derivative of the Lyapunov function along the system trajectories is

$$\dot{V} = \left(\frac{\partial V}{\partial \boldsymbol{x}}\right)^{\mathrm{T}} \dot{x} = -\boldsymbol{h}^{\mathrm{T}} D\boldsymbol{h} D\boldsymbol{h}^{\mathrm{T}}\boldsymbol{h} = -\|D\boldsymbol{h}^{\mathrm{T}}\boldsymbol{h}\|^2 \tag{5.18}$$

The derivative of the Lyapunov function is negative definite in the vicinity of each equilibrium point of the QGS, i.e. in the vicinity of each local minimum of the optimization problem. The Jacobian $D\boldsymbol{h}$ is positive definite in the vicinity of the equilibrium points because they are minima of the cost function. Therefore, all the equilibrium points of the QGS are locally asymptotically stable. ■

Under certain conditions, the equilibrium points are exponentially stable as shown in the next theorem.

Theorem 4: The equilibrium points of the QGS are exponentially stable.

Proof: Consider the Lyapunov function of (5.17). When there is no repeated measurement, $D\boldsymbol{h}$ is full rank and therefore $D\boldsymbol{h}D\boldsymbol{h}^{\mathrm{T}}$is a positive definite matrix. Assume that $\sigma_{\min}$ is the smallest

singular value of the positive definite matrix $D\boldsymbol{h}D\boldsymbol{h}^{\mathrm{T}}$. The derivative of the Lyapunov function can be written as

$$\dot{V} = \left(\frac{\partial V}{\partial \boldsymbol{x}}\right)^{\mathrm{T}} \dot{\boldsymbol{x}} = -\boldsymbol{h}^{\mathrm{T}}(\boldsymbol{x}) D\boldsymbol{h} D\boldsymbol{h}^{\mathrm{T}} \boldsymbol{h}(\boldsymbol{x}) \leq -\sigma_{\min} \boldsymbol{h}^{\mathrm{T}}(\boldsymbol{x}) \boldsymbol{h}(\boldsymbol{x}) = -\sigma_{\min} \|\boldsymbol{h}(\boldsymbol{x})\|^2 \tag{5.19}$$

where $\sigma_{\min}$ is the smallest eigenvalue of Therefore $\dot{V} \leq -\sigma_{\min} V$ and the equilibrium points of the QGS are exponentially stable. With the bounded input and output assumption, (29) yields that $\|D\boldsymbol{h}\|_2$ is bounded. Therefore the spectral radius and consequently smallest singular value of $D\boldsymbol{h}$ are finite.

Measurement errors and noise can make the measurements inaccurate and destabilize the system. Fortunately, QGS can tolerate relatively large measurement error. In neural networks, measurement errors lead to errors in neural network inputs. Consider the QGS as a function of $\boldsymbol{x}$ and $\boldsymbol{u}$, i.e, $\dot{\boldsymbol{x}} = -f(\boldsymbol{x}, \boldsymbol{u})$ and let the measurement errors change $\boldsymbol{u}$ to $\boldsymbol{u} + \Delta\boldsymbol{u}$. Assuming that $\Delta\boldsymbol{u}$ is small

$$\dot{\boldsymbol{x}} = -\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u} + \Delta\boldsymbol{u}) = -\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) - \frac{\partial \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})}{\partial \boldsymbol{u}} \Delta\boldsymbol{u} + HOT \tag{5.20}$$

where HOT denotes higher order terms. For sufficiently small $\Delta\boldsymbol{u}$, we can neglect higher order terms and write

$$\begin{aligned} \dot{\boldsymbol{x}} &= -\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u} + \Delta\boldsymbol{u}) \cong -\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) - \frac{\partial \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})}{\partial \boldsymbol{u}} \Delta\boldsymbol{u} \\ &= -\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) + \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}, \Delta\boldsymbol{u}) \end{aligned} \tag{5.21}$$

Assuming that the activation functions of the neural network are continuously differentiable, $\boldsymbol{g}$ will be continuously differentiable and hence $\boldsymbol{g}$ is Lipschitz for all $t > 0$ and $\in T \subset R^n$, with T the domain that contains the equilibrium point. Assume that the perturbation term satisfies the linear growth bound

$$\|\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}, \Delta\boldsymbol{u})\| \leq \gamma \|\boldsymbol{x}\|, \forall t \geq 0, \forall \boldsymbol{x} \in T \tag{5.22}$$

To find a bound on the perturbation that guarantees stability, we need the following property of matrix norms

Fact: For every $A: C^n \rightarrow C^m$

$$\|A\|_2 \leq \|A\|_{\mathrm{F}} \leq \sqrt{r}\|A\|_2 \tag{5.23}$$

where $\|A\|_{\mathrm{F}}$ is the Frobenius norm of A and r is its rank.

Theorem 5: Assume that the input and output of the network are bounded, $\|\boldsymbol{y}\| \leq K_{\mathrm{y}}$ and $\|\boldsymbol{u}\| \leq K_{\mathrm{u}}$, and the corresponding neural network has n inputs, m hidden layer nodes and we have N measurements. The equilibrium of the perturbed QGS is asymptotically stable if

$$\gamma < N\sqrt{Nm}\left[\left(\sqrt{m} + \sqrt{K_y\left(\sqrt{n}K_u + m\right)}\right)^2\right] \tag{5.24}$$

Proof: Consider the Lyapunov function $V(\boldsymbol{x}) = \boldsymbol{h}^{\mathrm{T}}(\boldsymbol{x})\boldsymbol{h}(\boldsymbol{x})$. The derivative of $V(\boldsymbol{x})$ including the perturbation is

$$\dot{V} = (-\boldsymbol{h}^{\mathrm{T}}D\boldsymbol{h} + \boldsymbol{g}^{\mathrm{T}})D\boldsymbol{h}^{\mathrm{T}}\boldsymbol{h} = -\|D\boldsymbol{h}^{\mathrm{T}}\boldsymbol{h}\|^2 + \boldsymbol{g}^{\mathrm{T}}D\boldsymbol{h}^{\mathrm{T}}\boldsymbol{h} \tag{5.25}$$

for a negative definite $\dot{V}$, we need

$$\boldsymbol{g}^{\mathrm{T}}D\boldsymbol{h}^{\mathrm{T}}\boldsymbol{h} < \|D\boldsymbol{h}^{\mathrm{T}}\boldsymbol{h}\|^2 \tag{5.26}$$

This condition is satisfied if

$$\|\boldsymbol{g}\| < \|D\boldsymbol{h}^{\mathrm{T}}\boldsymbol{h}\| \leq \|\boldsymbol{h}\| \times \|D\boldsymbol{h}\| \tag{5.27}$$

Using the nonlinearity of (5.12) with a bounded output, $\boldsymbol{h}$ satisfies

$$\|\boldsymbol{h}\| \leq N(m\|\boldsymbol{x}\| + K_{\mathrm{y}}) \tag{5.28}$$

The Jacobian of the hyperbolic function gives

$$\|D\boldsymbol{h}\|_{\mathrm{F}} \leq \sqrt{Nm}(1 + \sqrt{n}\|\boldsymbol{u}\|\|\boldsymbol{x}\| + m\|\boldsymbol{x}\|) \tag{5.29}$$

Using proposition 1 gives the 2-norm bound

$$\|D\boldsymbol{h}\|_2 \leq \sqrt{Nm}(1 + \sqrt{n}\|\boldsymbol{u}\|\|\boldsymbol{x}\| + m\|\boldsymbol{x}\|) \tag{5.30}$$

By combining (5.29), (5.27), (5.26) and (5.22)

$$\gamma\|\boldsymbol{x}\| < N\sqrt{Nm}(1 + \sqrt{n}\|\boldsymbol{u}\|\|\boldsymbol{x}\| + m\|\boldsymbol{x}\|)(m\|\boldsymbol{x}\| + K_{\mathrm{y}}) \tag{5.31}$$

Using the input bound $\|\boldsymbol{u}\| \leq K_u$ gives the condition for negative definite $\dot{V}$

$$\gamma < N\sqrt{Nm}\left[\left(\sqrt{m}+\sqrt{K_{\mathrm{y}}\left(K_{\mathrm{u}}\sqrt{n}+m\right)}\right)^{2}\right] \tag{5.32}$$

5.6 Simulation Results

To illustrate the effectiveness of QGS for training neural networks, we use a QGS trained network for nonlinear system identification and compare the results with a genetic algorithm we use the results of [159]. The genetic algorithm optimization uses the MATLAB optimization toolbox with a population size of 10000, Roulette selection, adaptive feasible mutation, scattered crossover, and top fitness scaling to get the best results.

5.6.1 Example 1: Nonlinear System

Our first benchmark system is the second order nonlinear system chosen from [162]. The input-output equation of the system is described as

$$y(k+1)=\frac{y(k)y(k-1)(y(k)+0.25)}{1+y(k)^{2}+y(k-1)\text{^}2}+q(k) \tag{5.33}$$

$q(k)$ is the system input and $y(k)$ is the system output. $q(k)$

is zero-mean normally distributed with standard deviation $\sigma = 0.5$. The input to the neural network is $\boldsymbol{u}(k)=[q,q(k-1),y(k),\ldots,y(k-1)]^{T}$ and $y(k+1)$ is the target output for training. All the initial network parameter values are zero-mean normally distributed with standard deviation $\sigma = 0.5$. The optimal number of hidden layer nodes is found to be $m = 8$ and the total number of training sets is $N = 200$. The activation function of the neural network is the tangent hyperbolic function

$$\psi(x)=\tanh(x)=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{5.34}$$

After initializing with random initial values, QGS finds 47 local minima of the optimization problem. The local minimum with the best generalization capability is the global minimum or close to the optimal solution of the optimization problem.

Table 7 summarizes the mean squared error (MSE) for test data for QGS network, genetic algorithm network, and error backpropagation network. The MSE of the QGS network is less than the MSE for the genetic algorithm network and both networks outperform the backpropagation trained network. Other simulation results that are not included here for brevity, including

generalization errors, demonstrate that back propagation gives much worse results than the two other networks. Hence, we do not include back propagation in the remainder of this example.

Table 7. Mean Squared Error

Training method	QGS	GA	EBP
MSE	0.00797	0.0082	0.0187

Figure 30 shows the outputs of the system, the QGS trained network, and the genetic algorithm trained network and Figure 31 shows the same outputs for test data. While the training results is the same for both networks, Figure 31 shows that QGS trained network has better generalization performance on random input as test data and has smaller generalization error.

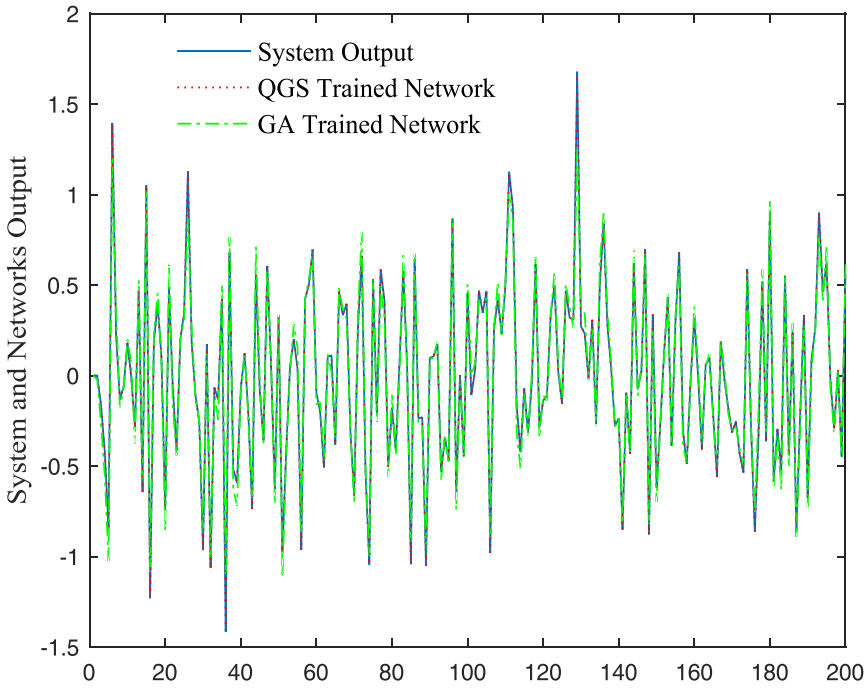

Figure 30. Outputs of the system and the trained neural

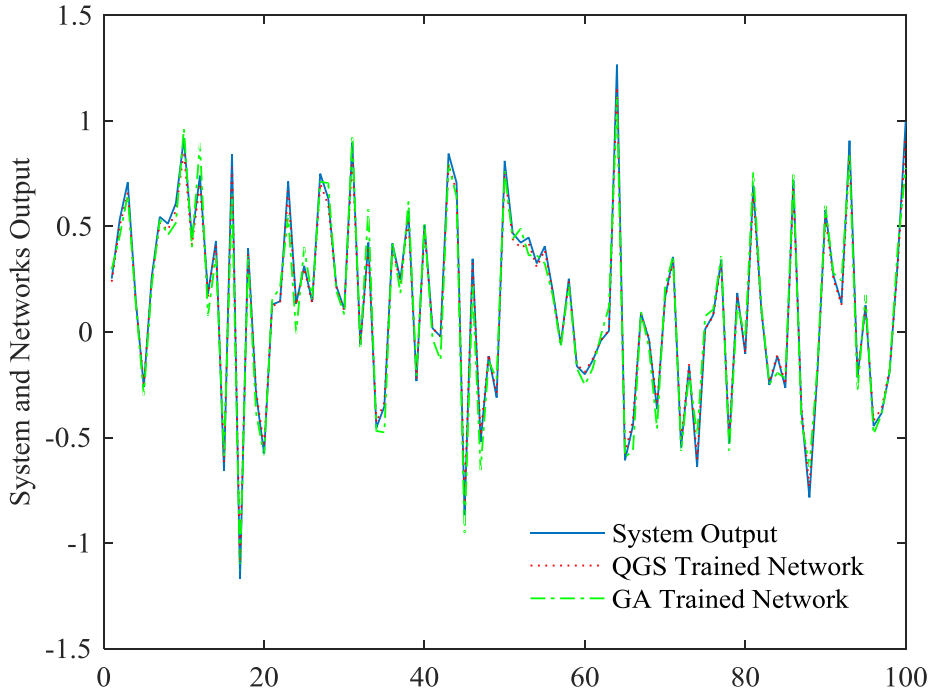

Figure 31. Outputs of the system and the trained neural networks

Figure 32 shows the generalization error for the QGS trained network and genetic algorithm trained network. While both networks have similar performance with the training data as input, Figure 32 illustrates that the QGS trained network has better generalization capability in terms of maximum generalization error percentage and mean squared error for test data. The average absolute generalization error of QGS trained network is 1.05% while average absolute generalization error of genetic algorithm trained network is 1.38%.

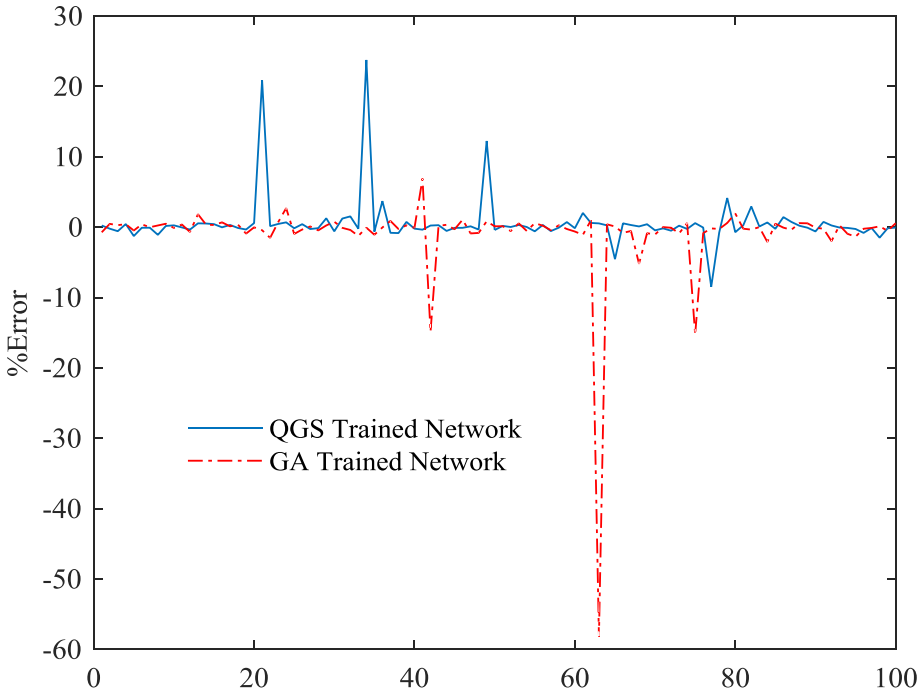

Figure 32. Generalization Error

5.6.2 Example 2: NARMA System

Our second benchmark system is a tenth order nonlinear autoregressive moving average (NARMA) process [50]. The input-output equation of the system is described as

$$y(k+1) = 0.3y(k) + .05y(k)\sum_{i=1}^{9} y(k-i) + 1.5 \times q(k-9)q(k) + 0.1 \tag{5.35}$$

$q(k)$ is the system input and $y(k)$ is the system output. $q(k)$ is zero-mean normally distributed with standard deviation $\sigma = 0.5$. The input to the system is $\boldsymbol{u}(k) = [q(k), \dots, q(k-9), y(k), \dots, y(k-4)]^T$ and $y(k+1)$ is the target output for training. All the initial network parameter values are zero-mean normally distributed with standard deviation $\sigma = 0.5$. The optimal number of hidden layer nodes is found to be $m = 6$ and the total number of training sets is $N = 100$. After initializing with random initial values, QGS finds 36 local minima for the optimization problem. The local minimum with the best generalization capability is the global minimum or close to optimal solution of the optimization problem.

Table 8 summarizes the MSE for test data for the QGS trained network, the genetic algorithm trained network and the error backpropagation trained network. The MSE of QGS is smaller than

the MSE for the genetic algorithm trained network. Both networks outperform the error backpropagation trained network. As in Example 1, other simulation results are much worse for back propagation than for the other two networks and we omit back propagation results for the remainder of this example.

Table 8. Mean square error

Training method	QGS	GA	EBP
MSE	0.0026	0.0038	0.0087

Figure 33 shows the outputs of the system, the QGS trained network, and the genetic algorithm trained network. Figure 34 shows the same outputs for test data. Figure 33 shows that QGS trained network has better performance on train data and Figure 34 shows that QGS trained network outperforms genetic algorithm trained network on random input as a test data.

Figure 35 shows the generalization error for the QGS trained network and for the genetic algorithm trained network. While both networks have similar performance with the training data as input, Figure 34 and Figure 35 show that the QGS trained network has better generalization capability. The average absolute generalization error of QGS trained network is 1.45% while average absolute generalization error of genetic algorithm trained network is 2.86%.

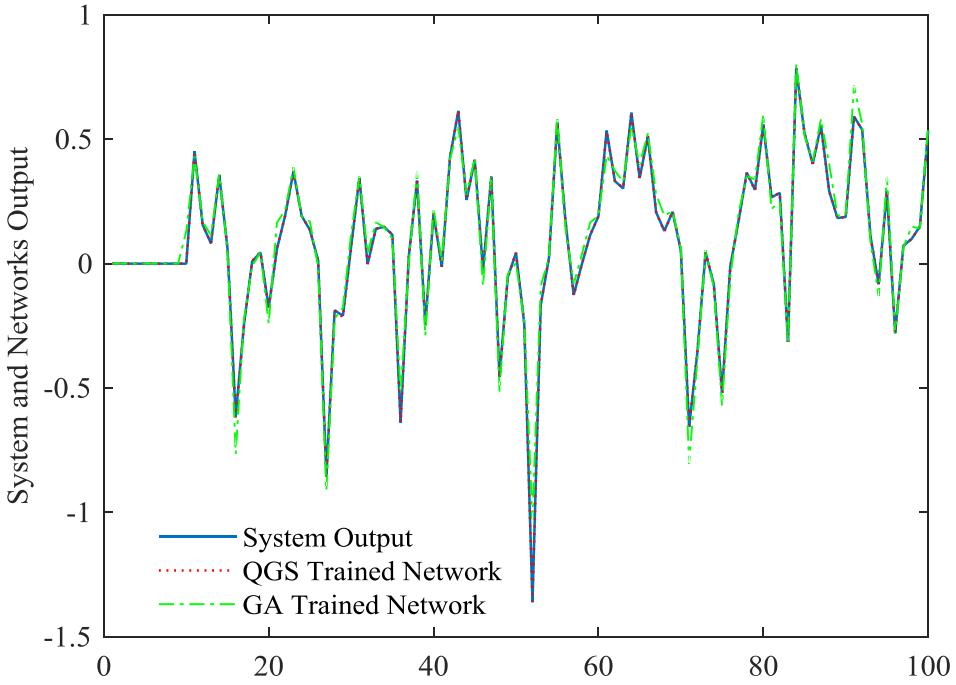

Figure 33. Outputs of the system and the trained neural

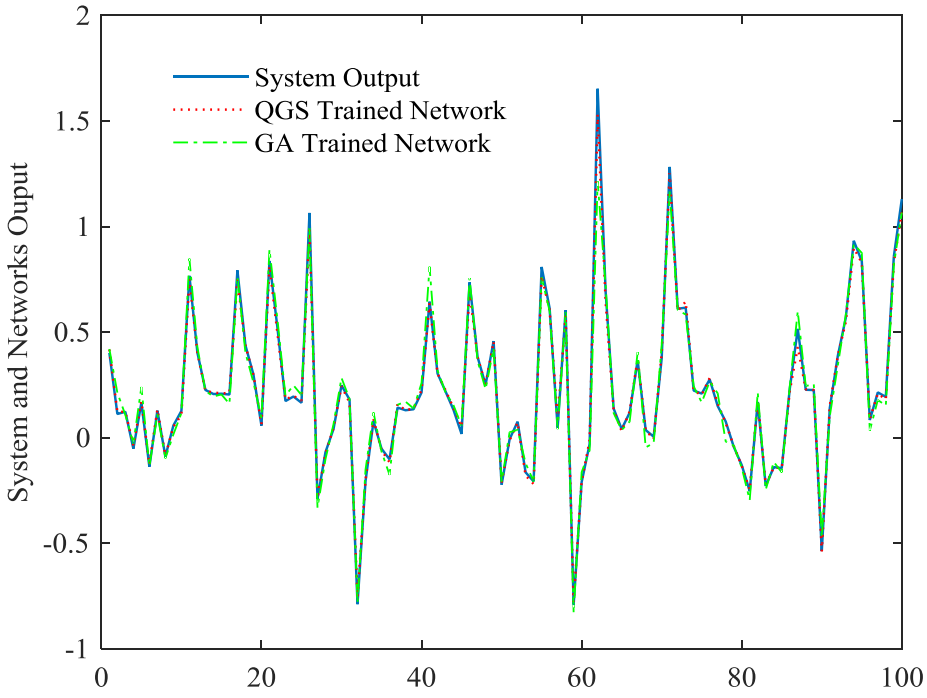

Figure 34. Outputs of the system and the trained neural network

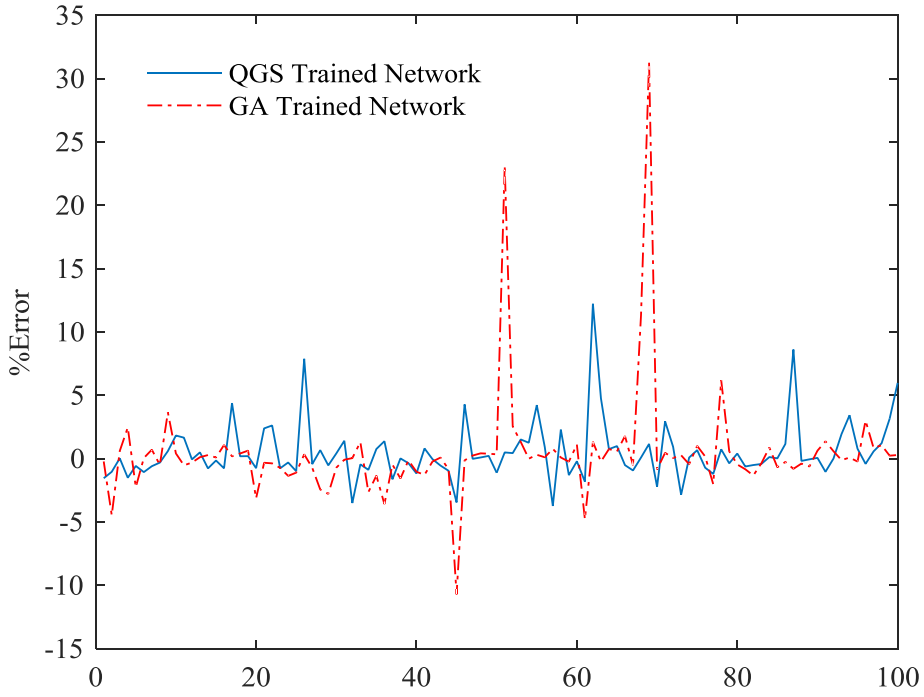

Figure 35. Generalization Error

5.7 Conclusion

In this study, we introduce a new training algorithm for neural network using the QGS. QGS uses trajectories of a nonlinear dynamical system to find a local minima of the optimization problem. The local minimum with the best generalization capability is the global minimum of the optimization problem. Simulation results shows that QGS trained network performs better than networks trained using genetic algorithm and error backpropagation. In particular, QGS networks have better generalization properties, faster training time in comparison to genetic algorithm and are more robust to errors in the inputs.

In contrast to Newton based methods, QGS does not need multiple initial values to find multiple local minima and does not need a huge number of measurements for training. Therefore, QGS is particularly suited to applications with a limited number of available input-output measurements.

Chapter 6. Training Recurrent Neural Networks via Dynamical Trajectory Based Optimization

6.1 Introduction

Neural networks are non-parametric models for function approximation. To provide appropriate models, neural networks must be trained using a suitable training dataset. Training neural networks is by minimization a cost function defined using the output of the network and measurements from the modeled system [170]. Classical training approaches use derivatives of the cost function to update the weights of the neural network [8]. Unfortunately, classical approaches do not guarantee finding the global minimum of the optimization problem and are often trapped in a local minimum. Moreover, classical approaches converge slowly and may have poor generalization capability with noisy data [171].

Researchers have proposed different modifications to the error backpropagation algorithm to improve its performance and reliability [172]-[174]. Jaganathan et al. proposed using delta rule weight tuning to train artificial neural networks [175]. They used passivity theory to show that their weight-tuning algorithm yields a passive neural network.

Multiplier and constrained learning rate algorithm are two other approaches to avoid instability during training neural networks and provide adaptive learning scheme [176],[177]. The algorithms are stable and can be applied adaptive control, image restoration, and visual processing.

The performance of nonlinear system identification with neural networks is dependent on network weights and network structure [178]. Hirose et al. proposed a modified backpropagation algorithm that learns the number of hidden layer nodes and can escape from local minima, unlike Newton-based algorithms [179]. Their simulation results show that their modified error backpropagation algorithm converges faster than standard backpropagation.

Researchers have used model reduction in nonlinear system identification to optimize the internal structure of neural networks [178]. Prasad et al. used singular value decomposition to remove the redundant nodes of a neural network [180]. Their simulation results show that a pruned neural network has equivalent or better generalization capability than over-fitted networks. Karnin used the sensitivity of the error to neural network weights and tracked changes of the network

weights during training to efficiently prune an over-fitted network [181]. His approach is relatively easy to implement and does not significantly increase computational complexity.

Evolutionary algorithms such as genetic algorithm are another approach to train neural networks. It does not rely on calculating derivatives of the cost function and does not get stuck in local minima of the optimization problem [182].

Particle swarm optimization (PSO) is another global optimization approach to train artificial neural networks that was shown to have superior performance to classical optimization approaches [183]. Evolutionary PSO is a derivative of the PSO algorithm that learns the internal structure of neural network as well as internal weights [184]. It has superior performance to the original PSO algorithm. Simulation results show that IPSONet constructs a compact artificial neural network with good generalization capability. Multi-dimensional PSO is another derivative of PSO to obtain the optimal structure of the neural network which has fast convergence rate in lower dimensions [185], [186]. It was shown to perform better than several other general optimization approaches such as basic genetic algorithms and IPSONet [187]. MPANN is a combination of gradient descent and a multi-objective evolutionary algorithm for learning the internal structure and internal weights of artificial neural networks. It has lower training time than gradient based algorithms. PSO-QI combines mechanical quantum concepts with PSO to produce more logical offspring than other evolutionary algorithms and has faster convergence time [188].

Other algorithms have been proposed to train recurrent artificial neural networks. Bounding ellipsoid imposes a bound on the neural network output and the real measurements, and trains the network to satisfy the error bound [189]. The algorithm is a stable alternative to traditional methods [190]. EBP through time is an improved version of traditional EBP for training recurrent neural networks which is faster than evolutionary algorithms such as genetic algorithm but still suffers from local optima issue [163],[164][165][191]. Martens and Sutskever used a combination of Hessian free optimization together with new damping scheme to train recurrent neural networks [167]. Generalized Long-Short Term Memory is an adaptation of Long-Short Term Memory (LSTM) which benefits from the efficient local search of LSTM but is applicable to a larger class of recurrent neural networks [168].

Recursive Bayesian Levenberg-Marquardt was used to train the recurrent neural networks for time-series modeling [192]. The algorithm is numerically stable and has superior performance to

traditional methods, such as extended Kalman training and Newton based methods [193]. Although other approaches, such as Levenberg-Marquadrt [194], multi agent systems [195], dark knowledge transfer, [196], and quotient gradient system , [159], have been applied to train the artificial neural networks, there are other promising optimization approaches in the mathematics literature that can be applied to training neural networks.

In this study, we use a dynamical trajectory based methodology for training the artificial neural networks. The dynamical trajectory based methodology consist of two systems : a Quotient gradient system (QGS) which finds the components of the feasible region of the optimization problem, and a projected gradient system (PGS) which searches the feasible components for local optimal solutions of the optimization problem. By switching between the PGS and QGS phases, the algorithm is able to find multiple local minima of the optimization problem and finally finds the global optimal or a good suboptimal solution of the optimization problem.

Unlike Newton-based methods, trajectory based optimization is a global optimization approach and does not suffer the local minimum issue. Furthermore, trajectory based optimization does not require a huge number of measurements to find the global optimum and does not have user dependent variables. This makes the methodology more desirable than newton based methods whose performance depends on learning rates and initial point, better than approaches such as particle swarm optimization whose performances depends on initial particles and velocities, and better then genetic algorithm whose performance depends on initial population size, mutation and crossover functions.

Although network parameters must remain bounded, training neural networks is usually posed as an unconstrained optimization problem. In this study, we include upper and lower bounds on the neural network parameters with the minimization of the sum of squared errors and pose neural network training as a constrained minimization problem. Then we use trajectory based methodology to find components of the feasible region and search those components for local minima. The upper and lower bounds can be chosen arbitrarily large to avoid the effect of conservative bounds on the optimal solution [252],[253].

6.2 Dynamical Trajectory Based Methodology

Lee and Chiang proposed a dynamical trajectory based methodology for finding multiple optimal solutions of nonlinear programming problems [31]. This section reviews their work. Consider the following optimization problem

$$\min f(\boldsymbol{x}) \ \text{s.t.} \ \boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{0} \tag{6.1}$$

$f(\boldsymbol{x})$ is assumed in $C^2(R^n, R^m)$ to guarantee the existence of the solution and $\boldsymbol{h}(\boldsymbol{x})$ is assumed smooth. The more general case can includes inequality constraints as

$$\begin{aligned} &\min f(\boldsymbol{x}) \\ &\text{s.t. } h_i(\boldsymbol{x}) = 0, \qquad i \in I = \{1,2,\dots,l\} \\ &g_j(\boldsymbol{x}) \le 0, \qquad j \in J = \{1,2,\dots,s\} \end{aligned} \tag{6.2}$$

The inequality constraint can be transformed into equality constraints by introducing positive slack variables such as $g_j(\boldsymbol{x}) + s_j^2 = 0,\ j \in J$. $\bar{\boldsymbol{x}}$ is called the Kuhn-Tucker point of (1) with Lagrange-multipliers $\bar{\boldsymbol{\lambda}} = (\lambda_1, \dots, \lambda_m)$ if it satisfies the Kuhn-Tucker conditions

$$\begin{aligned} &\nabla_x L(\bar{\boldsymbol{x}}, \bar{\boldsymbol{\lambda}}) = \nabla f(\bar{\boldsymbol{x}}) + \sum_{i=1}^{m} \bar{\lambda}_i \nabla \boldsymbol{h}_i(\bar{\boldsymbol{x}}) = 0 \\ &\nabla_\lambda L(\bar{\boldsymbol{x}}, \bar{\boldsymbol{\lambda}}) = \boldsymbol{h}(\boldsymbol{x}) = 0 \end{aligned} \tag{6.3}$$

$L(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \sum_{i \in I} \lambda_i h_i(x)$ is the Lagrangian function of the optimization problem. The feasible region is defined as

$$M := \{x \in R^n : \boldsymbol{h}(\boldsymbol{x}) = 0\} \tag{6.4}$$

The feasible region can be any closed subset of R^{n}. The feasible region is subject to the following assumptions:

Assumption 1:

a) (*Regularity*) At each $\boldsymbol{x} \in M,\ \{\nabla h_i(\boldsymbol{x}),\ i = 1, \dots, m\}$ are linearly independent.

b) (*Nondegeneracy*) At each critical point $\bar{\boldsymbol{x}} \in M,\ \boldsymbol{\ell}^T \nabla^2_{xx} L(\bar{\boldsymbol{x}}, \boldsymbol{\lambda}) \boldsymbol{\ell} \neq 0$ for all $\boldsymbol{\ell} \neq 0$ satisfying $\nabla h_i(\bar{\boldsymbol{x}})^T \boldsymbol{\ell} = 0$ for all $i = 1,2,\dots,m$.

c) (*Finiteness and Separating Property*) f has finitely many critical points in M at which it attains different values of f.

Assumption 1 is generically true and when M is compact, the optimization problem is structurally stable [197],[198].Using the regularity condition together with the implicit function

theorem, it can be shown that M is an $n-m$ dimensional smooth manifold. M may be disconnected and nonconcave and is given by the union

$$M = \bigcup_i M_i \tag{6.5}$$

of disjoint connected components M_i. The dynamical trajectory based method has two main components: the projected gradient system (PGS) and the quotient gradient system (QGS). Starting from an arbitrary initial point, the QGS finds a feasible component of the feasible region, then the PGS finds all the local minima in that component. After finding all the local minima in a particular component, we switch to the QGS to escape from that component and reach another component of the feasible region. By repeating this process, the methodology finds multiple local minima of the minimization problem even if they lie in disjoint regions M_i.

6.2.1 PGS Phase

Escaping from one local optimal solution and moving toward another local optimal solution in the connected component of the feasible region is an essential part of our method. The PGS is a nonlinear dynamical system whose trajectories can be used to escape from one local optimal solution and move toward another solution in the current component of the feasible region. The PGS is defined as

$$\dot{\boldsymbol{x}} = F(\boldsymbol{x}) = -\nabla f_{\text{proj}}(\boldsymbol{x}), \qquad \boldsymbol{x} \in M \tag{6.6}$$

where $\nabla f_{\text{proj}}(\boldsymbol{x})$ is the orthogonal projection of $\nabla f(\boldsymbol{x})$ to the tangent space $T_x M$ of the constraint set M at $\boldsymbol{x}$. When $D\boldsymbol{h}(\boldsymbol{x}) = \partial \boldsymbol{h}(\boldsymbol{x})/\partial \boldsymbol{x}$ is nonsingular, then $\nabla f_{\text{proj}}(\boldsymbol{x}) = P_r(\boldsymbol{x})\nabla f(\boldsymbol{x})$ where $P_r(\boldsymbol{x}) = \left(I - D\boldsymbol{h}(\boldsymbol{x})^T (D\boldsymbol{h}(\boldsymbol{x}) D\boldsymbol{h}(\boldsymbol{x})^T)^{-1} D\boldsymbol{h}(\boldsymbol{x})\right) \in R^{n\times n}$ is the positive semidefinite projection matrix for every $\boldsymbol{x} \in M$. Every local optimal solution of (1) is a stable equilibrium point of the PGS and every trajectory of the PGS converges to one of its equilibrium points. Therefore, starting from any initial point in the stability region of its stable equilibrium, we can reach a local optimal solution. Thus, the problem of escaping from one local optimal solution and moving to another one reduces to escaping from the stability region of one stable equilibrium and entering the stability region of another. To do this, PGS starts from a local optimal solution and proceeds in reverse time until it reaches a decomposition point, which is a saddle point. Starting from the decomposition point, the algorithm moves towards another local optimal solution. Consequently, PGS finds the

multiple local minimums in the connected component of the feasible region. The next step is to move from one connected component of the feasible region to another by invoking the QGS phase.

6.2.2 QGS Phase

To explore all the components of the feasible region, we must be able to approach a connected feasible component then escape from it and approach another one. We use a nonlinear dynamical system whose trajectories can be used to approach and escape from connected components of the feasible region. The nonlinear dynamical system must have stable equilibrium manifolds that are connected feasible components of the feasible region of the optimization problem. The nonlinear system we need is the following quotient gradient system (QGS)

$$\dot{\boldsymbol{x}} = -D\boldsymbol{h}(\boldsymbol{x})^T D\boldsymbol{h}(\boldsymbol{x}) \tag{6.7}$$

where $D\boldsymbol{h}(\boldsymbol{x})$ is the Jacobian of $\boldsymbol{h}$ at $\boldsymbol{x}$. The right hand side of the QGS equation is the gradient of $\|\boldsymbol{h}(\boldsymbol{x})\|^2/2$. Lee and Chang showed that the feasible components of (1) correspond to stable equilibrium manifolds of the QGS and that every trajectory of the QGS converges to one of its equilibrium manifolds [31]. Therefore, a connected component of the feasible region is approached by integrating QGS from any point in the stability region of its stable equilibrium manifold.

Similarly to the PGS phase, escaping from a connected feasible component is by integrating the QGS in reverse time until approaching a point in an unstable manifold on the stability boundary of the stable equilibrium manifold. Then we integrate the QGS forward in time from a point close to the unstable equilibrium manifold until we reach another stable equilibrium manifold. By alternating between the PGS and QGS phases, all the local minima of the optimization problem can be found, and the global minimum can be determined.

6.2.3 Finding Decomposition Points

The decomposition point plays a pivotal role in finding local minima because every two adjacent local optimal solutions of (1) are connected through an unstable manifold of a decomposition point. To define and show how to find a decomposition point, we need several definitions.

A trajectory is the solution of the PGS starting from $x \in M$ at $t = 0$ and is denoted by $\phi(.,\boldsymbol{x})\colon R \to M \subset R^n$. $\boldsymbol{x}^* \in M$ is an equilibrium point if $F(\boldsymbol{x}^*) = 0$. The equilibrium point $\boldsymbol{x}^* \in M$ is said to be hyperbolic if the restriction of Jacobian of $F(.)$ at $\boldsymbol{x}^*$ to the tangent space $T_{\boldsymbol{x}^*}M$ has no eigenvalues with zero real part. If the restricted Jacobian of a hyperbolic equilibrium point has exactly k eigenvalues with positive real parts, it is called a type-k equilibrium point. If all the

eigenvalues of the restricted Jacobian have negative real parts, the equilibrium point is called stable; otherwise it is unstable. The stable and unstable manifolds of $\boldsymbol{x}^*$ are defined as follows

$$W^{\mathrm{s}}(\boldsymbol{x}^*) = \left\{\boldsymbol{x} \in M\colon \lim_{t\to\infty} \boldsymbol{\phi}(t,\boldsymbol{x}) = \boldsymbol{x}^*\right\}$$
$$W^{\mathrm{u}}(\boldsymbol{x}^*) = \left\{\boldsymbol{x} \in M\colon \lim_{t\to-\infty} \boldsymbol{\phi}(t,\boldsymbol{x}) = \boldsymbol{x}^*\right\} \quad (6.8)$$

The stability region of stable equilibrium $\boldsymbol{x}_s$ is defined as

$$A(\boldsymbol{x}_s) \coloneqq \{\boldsymbol{x} \in M\colon \lim_{t\to\infty} \boldsymbol{\phi}(t,\boldsymbol{x}) = \boldsymbol{x}_s\} \quad (6.9)$$

The quasistability region of stable equilibrium $\boldsymbol{x}_s$ is defined as $A_p(\boldsymbol{x}_s) = \mathrm{int}(\overline{A(x_s)})$. $A(\boldsymbol{x}_s)$ and $A_p(\boldsymbol{x}_s)$ are open, connected and invariant sets relative to the manifold M.

A type-one equilibrium point $\boldsymbol{x}_d$ on the quasistability boundary $\partial A_p(\boldsymbol{x}_s)$ of a stable equilibrium $\boldsymbol{x}_s$ is called a decomposition point. However, not all type-one equilibrium points are decomposition points. Assume that $c = f(\boldsymbol{x}_d)$, as the objective function value increases from $c - \varepsilon$ to $c + \varepsilon$, the number of path components of the level set $S_c \coloneqq \{\boldsymbol{x} \in M\colon f(\boldsymbol{x}) < c\}$ decreases by one. For $\boldsymbol{x}_1$ which is a type one equilibrium point but not a decomposition point, when the objective function value increases from $f(\boldsymbol{x}_1) - \varepsilon$ to $f(\boldsymbol{x}_1) + \varepsilon$, the number of path components of S_c remains unchanged (Jongen et al., 1995; Chiang and Ahmed, 1996).

Suppose that $\boldsymbol{x}$ is a local minimum of $f(\boldsymbol{x})$ and $\lambda_1(\boldsymbol{x}) \leq \lambda_2(\boldsymbol{x}), \dots, \lambda_n(\boldsymbol{x})$ be the eigenvalues of $\nabla^2 f(\boldsymbol{x})$ and suppose that $\boldsymbol{v}_j(\boldsymbol{x})$ is normalized eigenvector associated with $\lambda_j(\boldsymbol{x})$. Since $\nabla^2 f(\boldsymbol{x})$ is symmetric, $P_j(\boldsymbol{x}) = \boldsymbol{v}_j(\boldsymbol{x})\boldsymbol{v}_j^T(\boldsymbol{x})$ is the orthogonal projection matrix on the eigenspace associated with $\lambda_j(\boldsymbol{x})$. Using the spectral factorization theorem, $\nabla^2 f(\boldsymbol{x})$ can be rewritten as

$$\nabla^2 f(\boldsymbol{x}) = \sum_{j=1}^{n} \lambda_j(\boldsymbol{x}) P_j(\boldsymbol{x}) \quad (6.10)$$

Define $\Gamma_i(\boldsymbol{x}), i = 1, \dots, n$ as

$$\Gamma_i(\boldsymbol{x}) = \sum_{j=1}^{i} P_j(\boldsymbol{x}) - \sum_{j=i+1}^{n} P_j(\boldsymbol{x}) \quad (6.11)$$

Define the i-th order reflected gradient vector field $\Theta_i(\boldsymbol{x})$

$$\Theta_i(\boldsymbol{x}) = \Gamma_i(\boldsymbol{x})\nabla f(\boldsymbol{x}) \quad (6.12)$$

Next, we present the algorithm to find decomposition points on $\partial \bar{A}(\boldsymbol{x})$ using the reflected gradient method and store them in Ω.

Algorithm for finding decomposition points:

Step 1: Initialization

Step 1.1: Choose q points $\boldsymbol{\varrho}_j,\ j = 1, \dots, q$ from a neighborhood of the local minimum $\boldsymbol{x}$ of $f(\boldsymbol{x})$

Step 1.2: Set a sufficiently small number ϵ

Step 1.3: Set $\Omega = 0$;

Step 2: Find decomposition points

for $j = 1$ to q do:

Step 2.1: Integrate $\Theta_1(\boldsymbol{x})$ with the initial condition $\boldsymbol{\varrho}_j$ until $\|\Theta_1(\boldsymbol{x})\|$ reaches a local minimum, say $\boldsymbol{\varrho}_j^*$, or $\|\Theta_1(\boldsymbol{x})\|$ becomes unbounded.

Step 2.2: Solve the algebraic equation $f(\boldsymbol{x}) = 0$ using initial condition $\boldsymbol{\varrho}_j^*$ and find solution $\boldsymbol{x}_j$.

Step 2.3: Check the type of equilibrium point with respect to $-\nabla f(\boldsymbol{x})$:

if $\boldsymbol{x}_j$ is type-one equilibrium point:

Compute the unstable eigenvector $\boldsymbol{\vartheta}^{\mathrm{u}}$ of $J_{-\nabla f(\boldsymbol{x})}(\boldsymbol{x}_j)$;

Numerically integrate PGS with initial points $\boldsymbol{x}_j - \epsilon\boldsymbol{\vartheta}^{\mathrm{u}}$ and $\boldsymbol{x}_j + \epsilon\boldsymbol{\vartheta}^{\mathrm{u}}$ until it approaches the equilibrium points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$.

if $\boldsymbol{x}_1 \neq \boldsymbol{x}_2$, $\Omega = \Omega \cup \{\boldsymbol{x}_j\}$;

endif;

endif;

end.

The algorithm finds two local minima, $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, associated with $\boldsymbol{x}_j$. The ability of the algorithm depends to the convergence of the nonlinear solver and the number of initial points chosen around the local minimum $\boldsymbol{x}$. When the objective function has certain periodic properties, upper and lower bounds on the number of initial points can be found which are dependent to the number of objective function variables [199].

6.3 Application of Trajectory based Optimization to Neural Network Training

In this study, we use a recurrent neural network with one hidden layer. The structure of the network is shown in Figure 36.

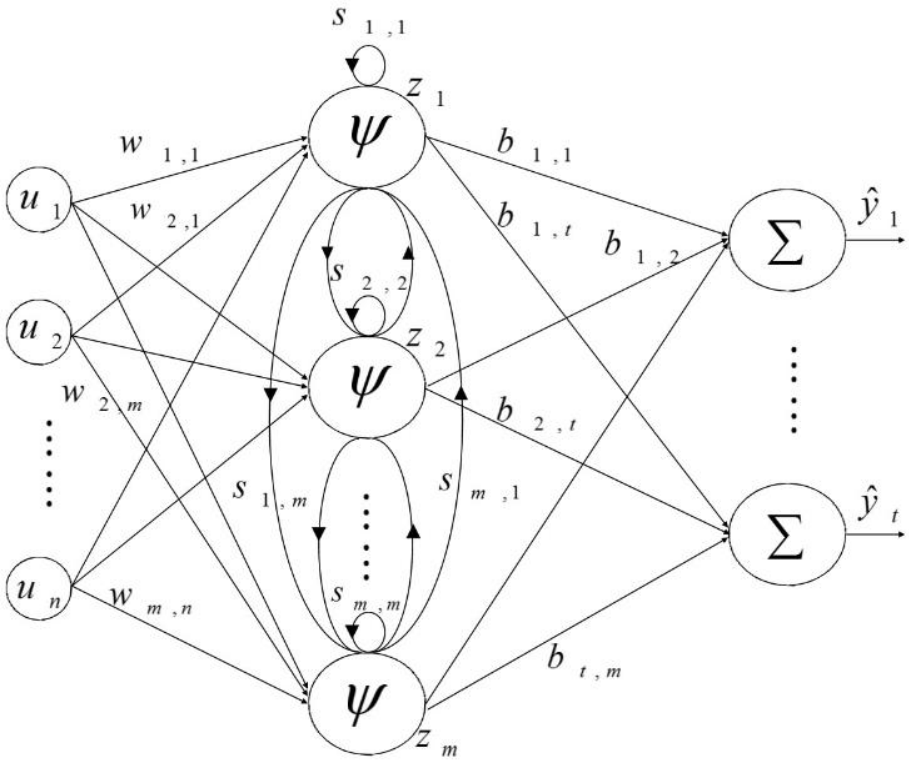

Figure 36. Structure of the recurrent neural network

The input-output relation of the network is

$$\begin{aligned} \boldsymbol{z}(k) &= \boldsymbol{\psi}\big(W\boldsymbol{u}(k) + B\boldsymbol{z}(k-1)\big) \\ \hat{\boldsymbol{y}}(k) &= V\,\boldsymbol{z}(k) \end{aligned} \tag{6.13}$$

where $\boldsymbol{u}(k)$ is the input, $\boldsymbol{z}(k)$ is the internal state, and $\hat{\boldsymbol{y}}(k)$ is the output of the network. W, B and V are neural network weights matrices and $\boldsymbol{\psi}(.)$ is a vector of hyperbolic tangent functions. Assuming that the network has n inputs, m hidden layer nodes and t outputs then $W \in R^{m\times n}$, $B \in R^{m\times m}$, and $V \in R^{t\times m}$. The cost function for finding the optimal network weights is the sum of the squared errors

$$\text{SSE} = \sum_{k=1}^{N} \boldsymbol{e}(k)^T \boldsymbol{e}(k) = \sum_{k=1}^{N} \big(\hat{\boldsymbol{y}}(k) - \boldsymbol{y}(k)\big)^T \big(\hat{\boldsymbol{y}}(k) - \boldsymbol{y}(k)\big) \tag{6.14}$$

where $\boldsymbol{y}(k)$ is the measured output vector, $\hat{\boldsymbol{y}}(k)$ is the neural network output and N is the number of training samples.

Dynamical trajectory based methodology provides a systematic approach to explore components of feasible region of the optimization problem by repeatedly switching between PGS and QGS. To define the PGS and QGS we need to partition the weight matrices as follows

$$V = \begin{bmatrix} \boldsymbol{v}_1^T \\ \vdots \\ \boldsymbol{v}_m^T \end{bmatrix}_{t\times m}, W = \begin{bmatrix} \boldsymbol{w}_1^T \\ \vdots \\ \boldsymbol{w}_m^T \end{bmatrix}_{m\times n}, B = \begin{bmatrix} \boldsymbol{b}_1^T \\ \vdots \\ \boldsymbol{b}_m^T \end{bmatrix}_{m\times m} \tag{6.15}$$

The constraints are defines as the upper and lower bounds on the network parameters. Define $\boldsymbol{o}$ as the vector containing all the elements of W, B and V

$$\boldsymbol{o} = [o_i]_{n_p\times 1} = [\boldsymbol{v}_1, .., \boldsymbol{v}_m, \boldsymbol{w}_1, \dots, \boldsymbol{w}_m, \boldsymbol{b}_1, \dots, \boldsymbol{b}_m]^T$$
$$n_p = m^2 + m \times (n + t) \tag{6.16}$$

The constraints are defined as

$$|o_i| \le l_i, \qquad i = 1, \dots, n_p \tag{6.17}$$

The vector of limits on the parameters is defined as $\boldsymbol{l} = [l_i]_{n_p\times 1}$. By introducing slack variables $\boldsymbol{s}^T = [s_1, \dots, s_{n_p}]$, the constraints can be rewritten as

$$h_i(\boldsymbol{x}) = o_i^2 - l_i^2 + s_i^2 = 0, \qquad i = 1, \dots, n_p \tag{6.18}$$

The vector of optimization problem parameters is defined as

$$\boldsymbol{x} = [\boldsymbol{o}\ \boldsymbol{s}]^T_{(2\times n_p)\times 1} \tag{6.19}$$

Since $\boldsymbol{x}$ contains all the elements of W, B and V, the error, the cost function, and the constraints are functions of $\boldsymbol{x}$

$$\text{SSE} = f(\boldsymbol{x})$$
$$\boldsymbol{h}(x) = [h_i(\boldsymbol{x})]_{n_p\times 1} = x_i^2 - l_i^2 + x_{n_p+i}^2 = 0, i = 1, \dots, n_p \tag{6.20}$$

Then the problem of finding optimal values of network weights can be written as constrained optimization problem

$$\min f(\boldsymbol{x})$$
$$\text{s.t. } \boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{0} \tag{6.21}$$

$D\boldsymbol{h}$ is defined as

$$\boldsymbol{h}(\boldsymbol{x}) = [\partial h_i(\boldsymbol{x})/\partial \boldsymbol{x}]^T_{(n_p)\times(2\times n_p)}$$
$$\partial h_i(\boldsymbol{x})/\partial \boldsymbol{x} = [0, \dots, 0, 2x_i, 0, .. 0, 2x_{i+n_p}, 0, \dots, 0] \tag{6.22}$$

Therefore, all the elements of $[\partial h_i(\boldsymbol{x})/\partial \boldsymbol{x}]$ are zero except for the i^{th} and $\left(n_p + i\right)^{th}$ elements. $D\boldsymbol{h}(\boldsymbol{x})$ is a full rank matrix and the PGS and QGS for training the neural network are defined as

PGS:

$$\dot{\boldsymbol{x}} = -\left(I - D\boldsymbol{h}(\boldsymbol{x})^T (D\boldsymbol{h}(\boldsymbol{x}) D\boldsymbol{h}(\boldsymbol{x})^T)^{-1} D\boldsymbol{h}(\boldsymbol{x})\right) \nabla f(\boldsymbol{x}) \tag{6.23}$$

QGS:

$$\dot{\boldsymbol{x}} = -D\boldsymbol{h}(\boldsymbol{x})^T \boldsymbol{h}(\boldsymbol{x}) \tag{6.24}$$

The process of training neural network using trajectory based method starts with QGS at an arbitrary initial point. QGS finds a feasible component of feasible region, say M_i, and then PGS starts finding local minima in M_i. After finding a set of local minima in M_i, QGS is invoked again to escape from M_i and move towards another feasible component M_j. By repeating this process multiple feasible components of feasible region are located and the algorithm finds a set of local optimal solutions of the optimization problem.

6.4 Stability Analysis

The stability of the training method is a critical issue for any training algorithm. In addition, uncertainties and measurement noise may cause the training algorithm to go unstable and must be considered in stability analysis.

Theorem 1: The equilibrium points of the QGS are asymptotically stable.

Proof: Consider the Lyapunov function $V(\boldsymbol{x}) = \boldsymbol{h}^T(\boldsymbol{x})\boldsymbol{h}(\boldsymbol{x})$. $V(\boldsymbol{x})$ is a locally positive definite function of the state that is equal to zero at global optima of the optimization problem. Thus, $V(\boldsymbol{x})$ is a locally positive definite function in the vicinity of each equilibrium point. The derivative of the Lyapunov function along the system trajectories is

$$\dot{V} = \left(\frac{\partial V}{\partial \boldsymbol{x}}\right)^T \dot{x} = -\boldsymbol{h}^T D\boldsymbol{h} D\boldsymbol{h}^T \boldsymbol{h} = -\|D\boldsymbol{h}^T \boldsymbol{h}\|^2 \tag{6.25}$$

The derivative of the Lyapunov function is negative definite in the vicinity of each equilibrium point of the QGS, which are the local minima of the optimization problem. The Jacobian $D\boldsymbol{h}$ is positive definite in the vicinity of the equilibrium points because they are minima of the cost function. Therefore, all the equilibrium points of the QGS are locally asymptotically stable.

QGS dynamics (6.24) are independent of the neural network input. Measurements errors can perturb a neural network and drive it outside the basin of attraction of a stable equilibrium during training. However, QGS remains stable in the presence of bounded measurement errors.

If measurement errors cause the neural network input to change from $\boldsymbol{u}$ to $\boldsymbol{u} + \Delta\boldsymbol{u}$, then the PGS will terminate at a perturbed point, which is then used to initialize the QGS. The perturbed QGS initial condition can: (i) lie in the stable region of the previous stable equilibrium point, (ii) lie in the stability region of another stable equilibrium point, or (iii) be infeasible. In case (i), the QGS goes to the previous component of the feasible region and the algorithm continues. In case (ii), the QGS goes to another stable equilibrium point which is in another feasible component but will return to the current component in future calls of the QGS. In case (iii), the QGS goes to another feasible component. Therefore, measurement noise only affects the order of exploring the components of the feasible region.

$$P_r = \left(I - D\boldsymbol{h}(\boldsymbol{x})^T (D\boldsymbol{h}(\boldsymbol{x}) D\boldsymbol{h}(\boldsymbol{x})^T)^{-1} D\boldsymbol{h}(\boldsymbol{x})\right)$$

$$P_r = [pr_{i,j}], \qquad pr_{i,j} = \begin{cases} \propto_i, & \text{if } i = j,\ 0 \leq \propto_i \leq 1 \\ \pm\sqrt{1 - \propto_i^2}, & \text{if } |i-j| = n_p \\ 0\text{ , elsewhere} \end{cases} \tag{6.26}$$

Clearly, the norm of the projection matrix is bounded above as $\|P_r\| \leq 1$. The PGS can be rewritten in terms of the projection matrix as

$$\dot{\boldsymbol{x}} = -P_r \frac{\partial}{\partial \boldsymbol{x}} \left(\sum_{k=1}^{N} \boldsymbol{e}(k)^T \boldsymbol{e}(k) \right) = -P_r \sum_{k=1}^{N} \left(\frac{\partial \boldsymbol{e}(k)}{\partial \boldsymbol{x}} \right)^T \boldsymbol{e}(k) \tag{6.27}$$

We now show that the equilibria of the PGS are stable.

Theorem 2: The equilibrium points of the PGS are asymptotically stable.

Proof: Consider the Lyapunov function candidate $V(\boldsymbol{x}) = \sum_{k=1}^{N} \boldsymbol{e}(k)^T \boldsymbol{e}(k)$ where N is the number of inputs. Note that since $\boldsymbol{e}(k) = \hat{\boldsymbol{y}}(k) - \boldsymbol{y}(k)$ and $\hat{\boldsymbol{y}}(k)$ is function of $\boldsymbol{x}$, $\boldsymbol{e}(k)$ is a function of $\boldsymbol{x}$. In addition, $\boldsymbol{e}(k)$ is zero only at equilibrium points of the PGS, which are local minima of the optimization problem and is positive elsewhere. Thus, $V(\boldsymbol{x})$ is a positive definite function of $\boldsymbol{x}$. The derivative of the Lyapunov function along trajectories of PGS is

$$\dot{V} = \dot{\boldsymbol{x}}^T \frac{\partial V}{\partial \boldsymbol{x}} = \dot{\boldsymbol{x}}^T \sum_{k=1}^{N} D\boldsymbol{e}(k)^T \boldsymbol{e}(k) = \left[\sum_{k=1}^{N} D\boldsymbol{e}(k)^T \boldsymbol{e}(k) \right]^T P_r^T \left[\sum_{k=1}^{N} D\boldsymbol{e}(k)^T \boldsymbol{e}(k) \right] \tag{6.28}$$

where $D\boldsymbol{e}(k) = \partial \boldsymbol{e}(k)/\partial \boldsymbol{x}$. Assuming no repeated measurements, $D\boldsymbol{e}(k)^T \boldsymbol{e}(k)$ is a full rank matrix and P_r is positive semidefinite, $\dot{V}$ is negative semi definite function of the states and is zero at local optimal solutions of optimization problem and in the null space of P_r. P_r is projection matrix on the tangent space of the constraint set and since the equilibrium points are in the constraint set, there is no equilibrium point in the null space of P_r. Hence, $\dot{V}$ is negative definite in the feasible region except at the equilibrium points where it is zero. By La Salle's theorem, the equilibrium points of the PGS are locally asymptotically stable. ■

The input to the neural network is a parameter that determines the PGS dynamics, i.e. $\dot{\boldsymbol{x}} = -\boldsymbol{c}(\boldsymbol{x}, \boldsymbol{u})$. Therefore, measurement errors can affect the PGS dynamics and make it unstable. If measurement errors change the neural network input from a nominal value $\boldsymbol{u}^*$ to $\boldsymbol{u}^* + \Delta \boldsymbol{u}$, the Taylor series expansion in terms of $\Delta \boldsymbol{u}$ is

$$\dot{\boldsymbol{x}} = -\boldsymbol{c}(\boldsymbol{x}, \boldsymbol{u}) - \frac{\partial \boldsymbol{c}(\boldsymbol{x}, \boldsymbol{u})}{\partial \boldsymbol{u}} \Delta \boldsymbol{u} + \cdots = -\boldsymbol{c}(\boldsymbol{x}, \boldsymbol{u}) + \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}, \Delta \boldsymbol{u}) \tag{6.29}$$

Suppose that U is subspace of R^n that contains the equilibrium point $\boldsymbol{x}$, i.e. $\boldsymbol{x} \in U \subset R^n$. For a smooth activation function, $\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}^*, \Delta \boldsymbol{u})$ is continuously differentiable and consequently Lipschitz in R^n for all $t \geq 0$. We assume that $\boldsymbol{g}$ satisfies the linear growth bound

$$\|\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}^*, \Delta \boldsymbol{u})\| \leq \gamma \|\boldsymbol{x}\|, \qquad \forall t \geq 0, \qquad \forall \boldsymbol{x} \in U \tag{6.30}$$

The next theorem examines the effect of an input perturbation on the PGS.

Theorem 3: Given a neural network with n inputs, m hidden layer nodes, t outputs and N measurements. Assume that the network input has the bound $\|\boldsymbol{u}\| \leq k_u$ and the network output has the bound $\|\boldsymbol{y}\| \leq k_y$. If the perturbed PGS dynamics satisfies the linear growth constraint (6.29), then the equilibrium of the perturbed PGS is asymptotically stable if

$$\gamma < N\sqrt{Nm} \left[\left(\sqrt{m} + \sqrt{k_y \left(\sqrt{n} k_u + m \right)} \right)^2 \right] \tag{6.31}$$

Proof: Recall that the error $\boldsymbol{e}(k)$ is a function of $\boldsymbol{x}$ and consider the Lyapunov function candidate $V(\boldsymbol{x}) = \boldsymbol{e}(k)^{\mathrm{T}} \boldsymbol{e}(k)$. $V(\boldsymbol{x})$ is a positive definite function of the states and only becomes zero at optimal solutions of the optimization problem. The derivative of $V(\boldsymbol{x})$ including the perturbation is

$$\dot{V} = (-\boldsymbol{e}(k)^T D\boldsymbol{e}(k) P_r^T + \boldsymbol{g}^T) D\boldsymbol{e}(k)^T \boldsymbol{e}(k) \\ \leq -\lambda_{\min}(P_r^T)\, \|D\boldsymbol{e}(k)^T \boldsymbol{e}(k)\|^2 + \|\boldsymbol{g}\| \|D\boldsymbol{e}(k)^T \boldsymbol{e}(k)\| \tag{6.32}$$

For negative definite $\dot{V}$, we need the condition

$$\|\boldsymbol{g}\| < \lambda_{min}(P_r)\, \|D\boldsymbol{e}(k)^T \boldsymbol{e}(k)\| \tag{6.33}$$

The projection matrix P_r satisfies $\lambda(P_r) \in \{0,1\}$ and the trajectories of the PGS do not pass through its null space [31],[160]. Hence, we rewrite (6.32) as

$$\|\boldsymbol{g}\| < \|D\boldsymbol{e}(k)^T \boldsymbol{e}(k)\| \tag{6.34}$$

Using (6.13) and the unity upper bound of hyperbolic functions we have the upper bound

$$\|\hat{\boldsymbol{y}}\| \leq Nm\|\boldsymbol{x}\| \tag{6.35}$$

For any bounded output, we have

$$\|\boldsymbol{e}(k)\| = \|\hat{\boldsymbol{y}} - \boldsymbol{y}\| \leq \|\hat{\boldsymbol{y}}\| + \|\boldsymbol{y}\| \leq N\big(m\|\boldsymbol{x}\| + k_y\big) \tag{6.36}$$

For any bounded input, we have the $\|D\boldsymbol{e}(k)^T\|_{\mathrm{F}}$ upper bound

$$\|D\boldsymbol{e}(k)^T\|_{\mathrm{F}} \leq \sqrt{Nm}\big(1 + \sqrt{n}\|\boldsymbol{u}\|\|\boldsymbol{x}\| + m\|\boldsymbol{x}\|\big) \tag{6.37}$$

We now need the following identity:

$$\forall A \in R^{m\times n}\colon \|A\|_2 \leq \|A\|_F \leq \sqrt{r}\|A\|_2 \tag{6.38}$$

where $\|A\|_{\mathrm{F}}$ is the Frobenius norm of A and r is its rank [200].

Using (6.37) gives

$$\|D\boldsymbol{e}(k)^T\|_2 \leq \sqrt{Nm}\big(1 + \sqrt{n}\|\boldsymbol{u}\|\|\boldsymbol{x}\| + m\|\boldsymbol{x}\|\big) \tag{6.39}$$

By combining (6.39), (6.36) and (6.34), we get

$$\gamma\|\boldsymbol{x}\| \leq \sqrt{Nm}\big(1 + \sqrt{n}k_u\|\boldsymbol{x}\| + m\|\boldsymbol{x}\|\big)N\big(m\|\boldsymbol{x}\| + k_y\big) \tag{6.40}$$

For negative definite $\dot{V}$, we require

$$\gamma < N\sqrt{Nm}\left[\left(\sqrt{m} + \sqrt{K_y\big(K_u\sqrt{n} + m\big)}\right)^2\right] \tag{6.41}$$

■

6.5 Simulation Results

To demonstrate the performance of the dynamical trajectory based methodology, we apply it to the identification of two nonlinear dynamical systems and compare the results to genetic algorithm.

While the results for the training data are similar for both approaches, the generalization capability of the dynamical trajectory-based approach is better than that of genetic algorithm and EBP. The results of EBP from [159] are much worse and are not included for clarity of the figures. We compare to EBP only in the mean squared error tables.

6.5.1 Example 1: NARMA system

The first example is a 10^{th} order NARMA system chosen from [50]. The nonlinear system dynamics is described as

$$y(k+1) = 0.3y(k) + .05y(k)\sum_{i=1}^{9} y(k-i) + 1.5 \times d(k-9)d(k) + 0.1 \tag{6.42}$$

with input $d(k)$ and output $y(k)$. The input is random zero-mean normally distributed with standard deviation $\sigma = 0.5$. Thus, the input signal is persistently exciting. The target value for neural network training is $y(k+1)$ and the neural network input is $\boldsymbol{u}_n(k) = [d(k), \dots, d(k-9), y(k), \dots, y(k-4)]^T$. $N = 150$ training samples were created, of which 100 samples were used as the training data and 50 samples were used to test the generalization capability of the neural network. All the network parameters were initialized randomly with zero-mean normal values of standard deviation $\sigma = 1$. The optimal number of hidden layer nodes was found by plotting the generalization error versus the number of hidden layer nodes and was found to be $m = 6$. The activation function of the hidden layer nodes is

$$\psi(x) = \tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)} \tag{6.43}$$

The weights in V were constrained to the interval $[-10,10]$, while the elements of W and B were in the interval $[-5,5]$. The limits on elements of W and B are smaller to keep the hidden layer neurons active. To find decomposition points, 30 initial points around the local minimum were created by adding a vector of zero-mean normal random values with standard deviation $\sigma = 0.01$ and ϵ for finding the decomposition points is chosen to be 0.01. After invoking the PGS and QGS algorithms repeatedly, we found 6 components of the feasible region and 47 local minima.

Figure 37 shows the output of the system and the networks for training data and Figure 38 shows the output of the system and the networks for test data. Figure 39 shows the generalization error for both neural networks. Both Figure 38 and Figure 39 show that the trajectory based trained network can learn the dynamic of the system better than genetic algorithm trained network. When

the output of the system is very close to zero, a small generalization error leads to a large generalization error percent. This can be seen in time steps such as $k = 10$ and $k = 27$ in Figure 39.

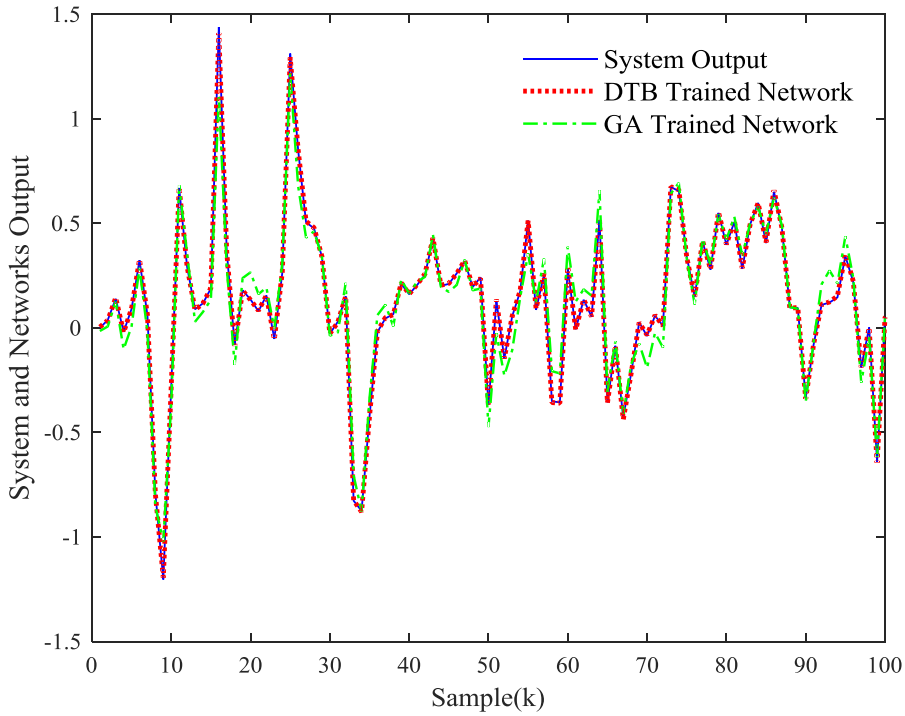

Figure 37. The output of the system and networks for training data. DTB stands for dynamical trajectory based and GA stands for genetic algorithm

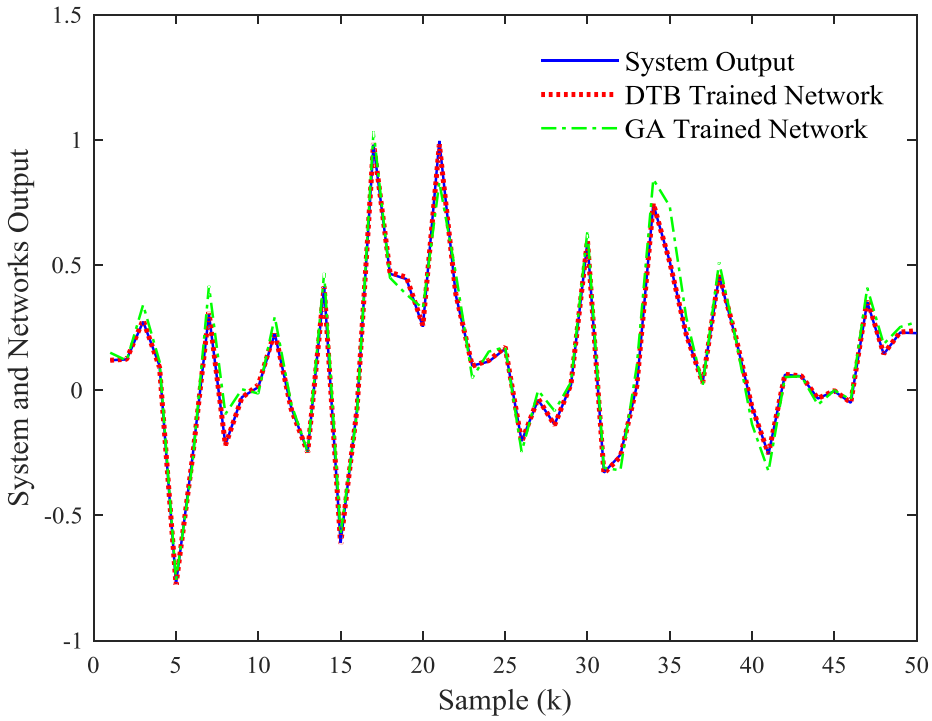

Figure 38. The output of the system and networks for test data

Table 9. Mean squared error

Train Method	DTB	GA	EBP
MSE	0.0461	0.0715	0.1184

Excluding this outlier points, the maximum generalization error for dynamical trajectory base trained network is 7.4% while the maximum generalization error of genetic algorithm trained network is 22% which again indicates the superior performance of this approach for training neural networks. Table 9 shows the average mean squared generalization error for different random test data sets and shows that dynamical trajectory based training outperforms genetic algorithm and EBP trained networks by a large margin.

In [159], another trajectory based method was used to train this recurrent neural network to identify the NARMA system. The method uses QGS trajectories but the QGS dynamics is different from the one used in this study. With the same input and number of nodes and number training samples, the dynamical trajectory based method has better generalization performance than that of [159], and reduces the maximum generalization error from 12.25 to 7.4%.

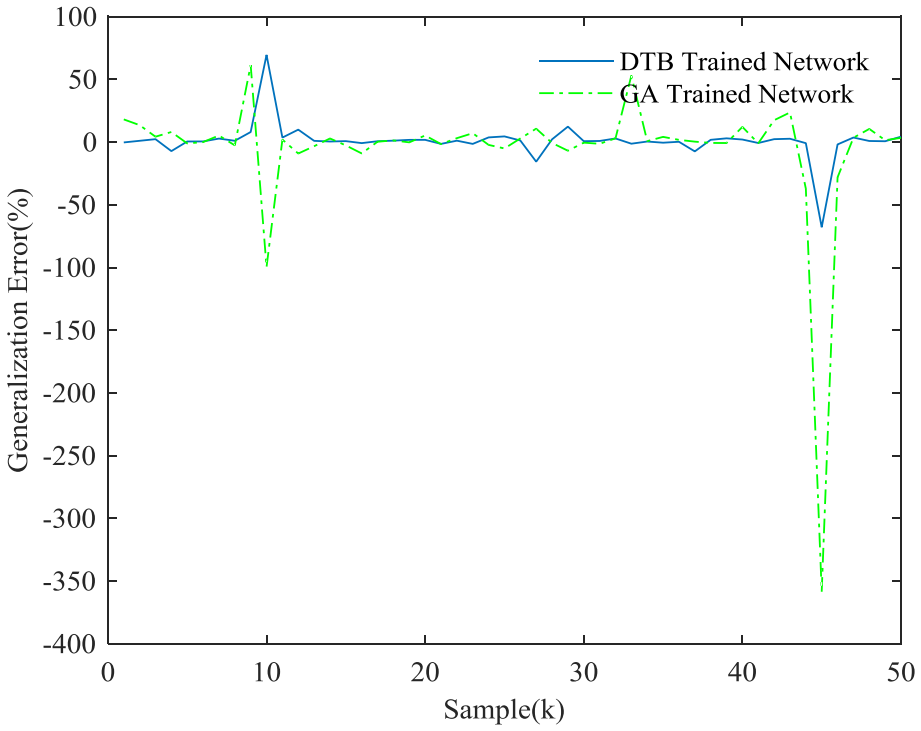

Figure 39. Generalization error for test data

6.5.2 Example 2: Nonlinear Second Order System

Our second example is a second order nonlinear system from [162]. The governing equation of the system is described as

$$y(k+1) = \frac{y(k)y(k-1)(y(k)+0.25)}{1+y(k)^2+y(k-1)^2} + d(k) \tag{6.44}$$

The network input vector is chosen as $\boldsymbol{u}_n(k) = [d(k), d(k-1), y(k), y(k-1)]^T$ and the number of hidden layer nodes is $m = 8$. As in Example 1, the training set is created using 100 zero-mean normal distributed samples and the test data is created with 50 samples from the same distribution.

As in Example 1, the PGS and QGS phases are invoked repeatedly to find local minima of the optimization problem. QGS locates 5 feasible components and PGS locates 41 local minimums in the feasible components. We used the same values as in Example 1 for the number of initial points for finding decomposition points and ϵ. The local minimum with the best generalization capability is chosen as optimal solution of the optimization problem.

Figure 40 shows the output of the system and the networks for training data. It shows that the dynamical trajectory base trained network is able to learn the behavior of the system better than

error genetic algorithm trained network. Figure 41 shows the output of the system and the networks for test data and illustrates the superior generalization performance of the dynamical trajectory base trained network. Figure 42 shows the generalization error for both neural networks. In some samples, the output of the system is close to zero, in such points even small generalization error will lead to big generalization error percentage such as $k = 32$ and $k = 38$ in Figure 42. Excluding outliers, the maximum generalization error of the dynamical trajectory based method is 13.3% while the maximum generalization error of the genetic trained network trained network is 31.2%.

Table 10 shows the average mean squared generalization error for different random test data sets. The table shows that dynamical trajectory based training outperforms genetic algorithm trained network by a large margin. Both the trajectory based approach and genetic algorithm training have superior performance to EBP trained network.

Table 10. Mean squared error

Train Method	DTB	GA	EBP
MSE	0.0523	0.0724	0.1297

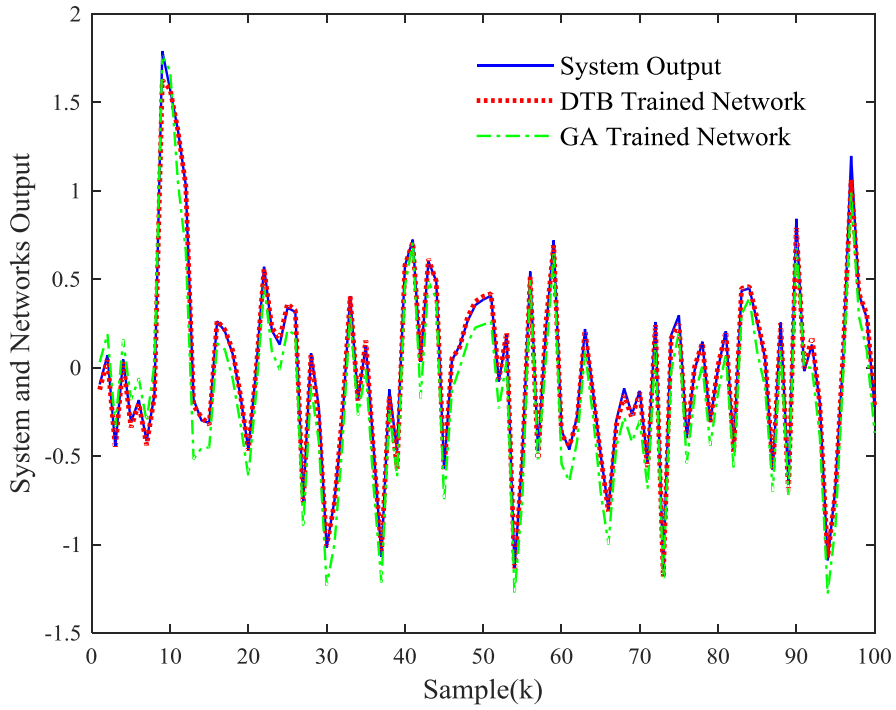

Figure 40. The output of the system and networks for training data. DTB stands for dynamical trajectory based and GA stands for genetic algorithm

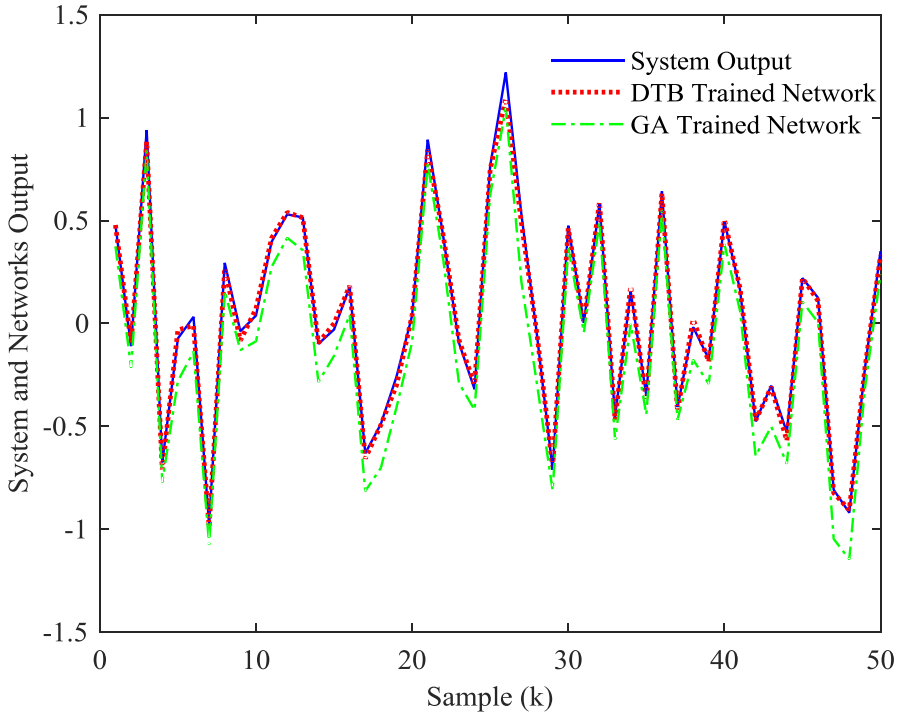

Figure 41. The output of the system and networks for test data

In [159], we used another trajectory based method to train a recurrent neural network to identify the same system. With the same number of hidden nodes and the same input and fewer training samples, the dynamical trajectory based method has better generalization performance and reduces the maximum generalization error from 18.7% to 13.3%.

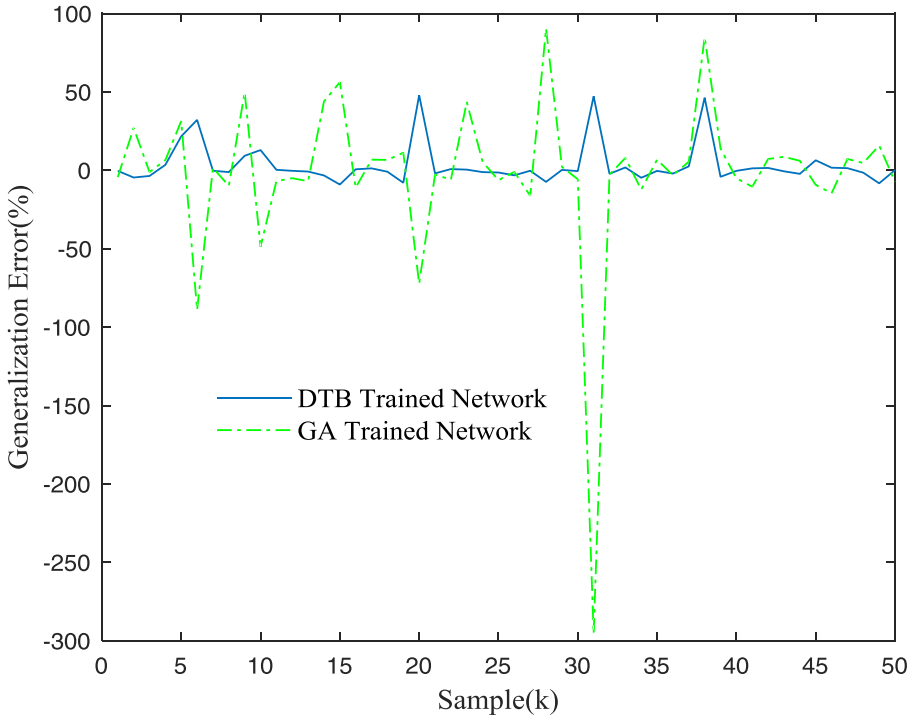

Figure 42. Generalization error for test data

6.6 Conclusion

This study uses a dynamical trajectory based methodology for training artificial neural networks. The methodology provides a systematic approach for finding multiple solutions of general nonlinear optimization problems. By invoking PGS and QGS phases repeatedly, the algorithm is able to locate multiple feasible components of feasible region and locate the local minima in the feasible components. Lyapunov theory was used to prove the stability of the method and its stability in presence of measurement error. Calculating PGS is similar to calculation of the derivative of the cost function with respect to network variables and is therefore similar to calculations of the EBP algorithm. Finding QGS is relatively easy because the constraints on the network weights are simple. Thus, the effort required to implement the algorithm is comparable to

the effort required to implement EBP. Simulation results show that the dynamical trajectory based method provides better performance than genetic algorithm and EBP trained networks.

Chapter 7. Anomaly Classification in Distribution Networks Using a Quotient Gradient System

7.1 Introduction

Due to the frequent disruptive events in power networks, developing a data-driven event diagnostics framework for maintaining the regular operation of the system is of paramount importance. Establishing such a framework, not only assists system operators in extracting useful information such as the cause or location of events, but also aids in other applications, such as preventive maintenance. Preventive maintenance saves time, reduces cost, improves safety, avoids unexpected outages, and reduces maintenance crew utilization.

Although disruptive events may not cause immediate equipment failure, they gradually lead to permanent failure. Hence, a comprehensive study of disruptive event classification in power systems is beneficial and will eventually lead to increasing the life expectancy of critical assets. With the expansion in the number of high-fidelity metering devices (i.e., phasor measurement units (PMUs) and micro-PMUs (μPMUs)) in power systems, data-driven diagnostics frameworks have become feasible for utilities and system operators.

Event classification using the wavelet transform has been extensively investigated in the literature [201]-[204]. In [205], the authors used the wavelet transform and support vector machine (SVM) for feature representation and classification of disruptive events. Masoum et al. proposed a novel approach for the detection and classification of disturbances in power systems [208]. The distorted signal is first denoised using the discrete wavelet transform (DWT), and the dominant features are then fed to a wavelet network classifier. A wavelet-based neural network method for detection and classification of disruptive events in power systems was proposed in [209]. In general, DWT-based methods are an effective technique for dimensionality reduction. They reduce the computational time while preserving the accuracy. The multi-resolution analysis used in DWT is achieves quicker data mining and reduces data storage requirements. However, compared to other methods, DWT based methods do not perform well in the presence of noise.

Another approach to the detection and classification of disruptive events it to use support vector machines (SVM) [210]-[214]. In [215], [215], two different classification methods are compared. The first method represents features using principal component analysis (PCA) to obtain the input to a multi-class SVM. In the second method, feature representation and classification use the

autoencoders and softmax classifiers, respectively. Although the SVM can handle high dimensional data well, it requires a proper choice of the kernel function and suffers from overfitting. In addition, its computational complexity increases drastically with the size of the training set.

Fuzzy-based methods for the classification of power quality disturbances were investigated in [216]-[219]. Manikandan et al. proposed a new method based on a sparse signal decomposition algorithm on hybrid dictionaries for detection and classification of disruptive events [220]. Fuzzy methods are particularly useful for pre and post data analysis. However, their drawback is the high resource consumption involved [221].

Event classification is a significant task that has been extensively investigated within various research areas in the literature. Event detection and classification using the S-transform have been widely studied in [222]-[225]. Event classification based on S-transform and probabilistic neural network (PNN) needs fewer features compared to the wavelet based method and outperforms feedforward multilayer (FFML) and learning vector quantization (LVQ) methods [222]. Unlike the wavelet transform, the features obtained from the S-transform have physical significance and can quantify the disturbances. The time-frequency resolution of S-transform makes it a good candidate for event classification. In addition, it results in better accuracy in the presence of noise. Using a modular neural network yields better accuracy and requires less training time, compared to a single NN [225].

In [226], a new approach for event classification and localization in power system was proposed based on the hyperbolic S-transform (HS) and radial basis function neural network (RBFNN). The HS-transform was applied to the input signal to generate the correspondent time-frequency contours, phase contours, and absolute phase components. The extracted numerical indices then are fed to RBFNN for classification. In [227], the authors presented a method for disruptive event classification using the S-transform and based on genetic algorithm (GA) and PNN. The dominant features of the data captured by the S-transform are fed to the PNN for the automatic classification of disturbances. Finally, GA is used to optimize the smoothing parameters of the PNN and improve the overall classification accuracy. Although this method has some advantages, the performance of the genetic algorithm depends on the proper choice of mutation and crossover methods as well as the proper choice of initial population, and the method has a very slow convergence rate. Hence,

the PNN's are very slow to train and need much more memory than multilayer perceptron networks.

In [228], an autoencoder based neural network is proposed for the classication of the abnormal events in the distribution system,A wide variety of methods are available to train neural networks for classification, such as the scaled complex conjugate algorithm [224], improved generalized adaptive resonance theory [229], Marquardt Levenberg [230], learning vector quantization combined with genetic algorithm [231] and region growing [232], [245]. Nevertheless, there are promising constrained optimization approaches in the mathematics literature that have not been used to train neural networks for classification applications.

In [159], it is shown that an optimization approach, known as the quotient gradient method, offers many advantages in training neural networks over conventional approaches. This stems from the fact that QGS finds the global minimum of the squared error criterion optimized in neural network training rather than the local minima to which other training approaches often converge. The method is a trajectory-based methodology that uses trajectories of a nonlinear dynamical system, the quotient gradient system (QGS), to find feasible solutions of constrained optimization problems [30]. The trajectories of the QGS converge to its equilibrium point, which is also the solutions to the optimization problem.

The quotient gradient method is a systematic approach to find the feasible solutions of the constraint satisfaction problems. It transforms the constraint satisfaction problem into an unconstrained minimization problem that defines the QGS. The equilibrium points of the QGS are local minima of the unconstrained minimization problem as well as the feasible solutions of the constraint satisfaction problem. In [159], the authors used the QGS to train a single stage fully recurrent neural network for nonlinear system identification and compared the results with those of error backpropagation.

QGS does not have user dependent variables, such as learning rate. Unlike Newton-based methods, it does not require a huge number of measurements. Furthermore, QGS is not sensitive to the choice of starting point and does not require a considerable amount of memory. These advantages make QGS an attractive training approach for neural networks.

In this paper, we propose the use of QGS to train a two-stage partially recurrent neural network for event classification and localization in power distribution networks. We compare the results of QGS training with those of genetic algorithm trained networks. Both the underlying theory and

extensive simulation results show that QGS consistently outperforms EBP, even after multiple initialization to improve EBP results. QGS also outperforms networks trained using genetic algorithms. This is because QGS systematically escapes from the stability region of one local minimum and moves toward another and can thus search a larger space for local optimal solutions.

In this study, we propose the use of QGS to train a two-stage partially recurrent neural network for event classification and localization in power distribution networks. We compare the results of QGS training with those of genetic algorithm trained networks. Both the underlying theory and extensive simulation results show that QGS consistently outperforms EBP, even after multiple initialization to improve EBP results. QGS also outperforms networks trained using genetic algorithms. This is because QGS systematically escapes from the stability region of one local minimum and moves toward another and can thus search a larger space for local optimal solutions

This study proposes a novel disruptive events classification and localization using PMU data in distribution grids. The paper considers four different events: (1) malfunctioned capacitor bank switching; (2) malfunctioned regulator on-load tap changer (OLTC) switching; (3) grid reconfiguration; and (4) normal abrupt load change. The proposed classification algorithm is developed using neural networks trained with the quotient gradient method. The contributions of the approach are as follows:

It provides a single framework for event classification and localization in distribution grids using PMU data. This is a major simplification over other data-driven methods that separate the two steps.

The proposed classifier is quite resilient in the presence of measurements noise and can distinguish classes better than genetic algorithm and EBP trained neural networks. In the presence of large measurement noise, the QGS trained network outperforms GA trained network and error backpropagation trained network by a significant margin.

The proposed method is based on a two-stage neural network that improves the classification accuracy compared to state-of-the-art NN-based classifiers. The first layer distinguishes the malfunctioned capacitor bank switching and reconfiguration events. The second layer distinguishes between events that have similar signatures and are much more difficult to separate; namely, malfunctioned OLTC and abrupt load changing events. [251].

7.2 QGS-based Neural Network Methodology for Event Classification

7.2.1 Normal vs Abnormal Events in Distribution Grids

Disruptive events occur intermittently in power systems interrupting normal operation. Finding a mechanism to detect, classify, and localize the source and location of these events prevents further damage to equipment and power outages.

Figure 43 illustrates the whole picture of the proposed event diagnostics framework. High-resolution metering devices, such as PMUs, are installed at several nodes in distribution systems. The data are measured and then transmitted to the data storage and archiving center via communication links, such as LTE networks [233]. Finally, post-event processing, including event classification and localization, is performed.

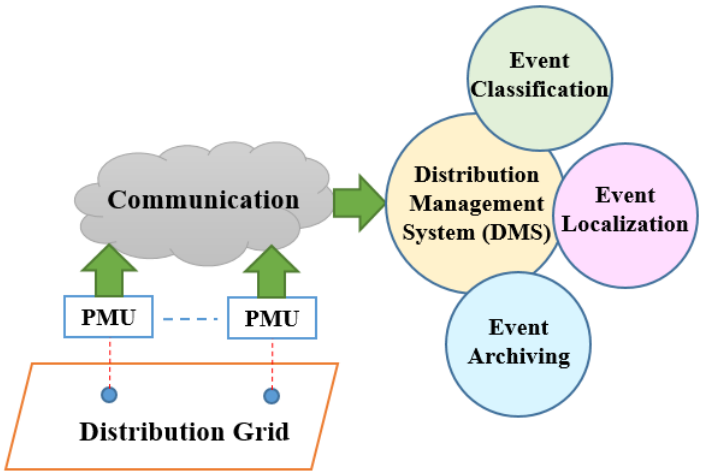

Figure 43. PMU data-driven event classification in distribution systems

Four different events are studied in this paper: i) malfunctioned capacitor bank switching; ii) malfunctioned regulator OLTC switching; iii) grid reconfiguration; and iv) normal abrupt load change. The first two classes are disruptive events, and the last two are normal events. Malfunctioned capacitor bank switching events occur because of failure in the mechanical switches of transformers. It takes about one cycle, i.e., 16.67 *ms*, for a capacitor bank to switch [234]. Malfunctioned regulator OLTC switching occurs when the tap changer is dislocated to a position and then relocated to its original position. This disturbance can result from aging and degradation of the selector switches in regulators. The on-load tap changer switching takes about 30-200 *ms* [235]. In the grid reconfiguration event, one recloser opens at one part of the network, and another one closes. Opening and closing distribution reclosers take about five cycles, i.e., about 83 *ms* [236]. The last class is the abrupt load changing occurs due to an increase or decrease

in demand in some nodes of the network. In this paper, events are first categorized based on their types and then based on their locations. Therefore, each event with a specific type and at a particular location is assigned to a class.

PMUs measure voltage magnitudes (pu), voltage angles (degree), current magnitudes (pu), and current angles (degree). For classification, we calculate: (i) the difference between two successive PMU samples, i.e., the change in voltage magnitude between two successive samples ($|v(n+1)| - |v(n)|$), (ii) the change in the voltage angle between two successive samples ($\delta v(n+1) - \delta v(n)$), (iii) the change in the current magnitude between two successive samples ($|i(n+1)| - |i(n)|$), and (iv) the change in the current angle between two successive samples ($\delta i(n+1) - \delta i(n)$). It is assumed that the sequence of events, i.e., the pre- and the during-event sequence is identified before initiating the classification process with an algorithm such as [246],[247]. Next, the feature matrix is formed using the current magnitude (pu), and current angle (degree) of the pre-event and during-event PMU samples along with the difference between the successive pre-event and during-event samples, as depicted in Figure 44. The feature matrix is the input to the neural network for classification

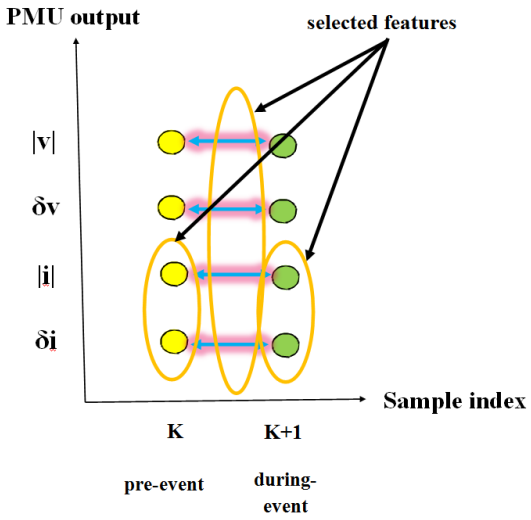

Figure 44. Feature selection process

7.2.2 QGS-based NN Classification Methodology

A systems of linearly or nonlinearly constrained equations appears in many fields of engineering and science. Lee and Chiang used the trajectories of a nonlinear dynamical system, the QGS, to find feasible solutions of the constraint satisfaction problem [30]. This section reviews the work of Lee and Chiang.

Consider the following constraint satisfaction problem

$$C_I(\boldsymbol{y}) < 0$$
$$C_E(\boldsymbol{y}) = 0, \boldsymbol{y} \in R^{n-l} \tag{7.1}$$

where $C_I(\boldsymbol{y})$ is the set of inequality constraints and $C_E(\boldsymbol{y})$ is the set of equality constraints and $\boldsymbol{y}$ is the vector of unknown variables to be found. To guarantee the existence of the solution, $C_I = (c_1, \dots, c_l)^T : R^{n-l} \to R^l$ and $C_E = (c_{l+1}, \dots, c_m)^T : R^{n-l} \to R^{m-l}$ must be smooth. Lee and Chiang argued that the constraint satisfaction problem can be rewritten as the unconstrained minimization problem

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{h}(\boldsymbol{x})\|^2, \qquad \boldsymbol{x} = (\boldsymbol{y}, \boldsymbol{s}) \in R^n \tag{7.2}$$

$$\boldsymbol{h}(\boldsymbol{x}) = \begin{bmatrix} C_I(\boldsymbol{y}) + \hat{\boldsymbol{s}}^2 \\ C_E(\boldsymbol{y}) \end{bmatrix} \in R^m, \hat{\boldsymbol{s}}^2 = (s_1^2, \dots, s_l^2)^T \tag{7.3}$$

where $\hat{\boldsymbol{s}}$ is set of introduced slack variables needed to transform the inequality constraints to equality constraints. Local minima of unconstrained minimization problem are possible feasible solutions of the original constrained satisfaction problem. The QGS is a nonlinear dynamical system based on the constraints given by

$$\dot{\boldsymbol{x}} = F(\boldsymbol{x}) = -\nabla \boldsymbol{f}(\boldsymbol{x}) \coloneqq -D_{\boldsymbol{x}}\boldsymbol{h}(x)^T \boldsymbol{h}(\boldsymbol{x}) \tag{7.4}$$

The stable equilibrium points of (4) are local minima of the unconstrained minimization problem, which are the possible feasible solutions of the constraint satisfaction problem [30]. To characterize the feasible points we need the following definitions.

A solution of the QGS starting from $\boldsymbol{x}(0)$ at $t = 0$ is called a trajectory ($\phi(., \boldsymbol{x}) : R \to R^n$). A path connected component of the $F^{-1}(0)$ is called an equilibrium manifold. Every stable equilibrium manifold has a stability region defined as

$$A(\sum_s) = \{x \in R^n : \lim_{t \to \infty} \phi(t, x) \in \sum_s\} \tag{7.5}$$

The stability boundary, denoted by $\partial A(\sum_s)$, is the boundary of the stability region of stable equilibrium manifold. Moreover, stable equilibrium manifolds of the QGS are path components of the solution set of the constraint satisfaction problem. Note that the stable equilibrium manifolds of the GQS are not necessarily in the feasible region of the constraint satisfaction problem. In such cases, the QGS must escape from the region of attraction of the stable equilibrium manifold and enter another stability region. This process must be repeated until the QGS enters the stability region

of all stable equilibrium point or until it reaches the stopping criterion. The QGS can reach the stability region of the stable equilibrium point by integrating from an initial point, which can be infeasible. The QGS can escape from the stability region by integrating backward in time. The eigenvalues of the Jacobian matrix of the QGS are a stability measure. Escaping from the stability region of local optimal solution and moving toward another local optimal solution enables QGS to search a larger space for local optimal solutions without passing one solution twice. Hence, the algorithm can find more solutions in comparison to multiple initialization and training with EBP algorithm [30],[159].

Neural networks have been successfully used in many engineering applications, including system identification and pattern recognition among the others. In this study, we use a three-layer recurrent neural network with one hidden layer. Figure 45 depicts the internal structure of the neural network. $\boldsymbol{u}(k)$, $\boldsymbol{z}(k)$ and $\widehat{\boldsymbol{y}}(k)$ are the input, internal state and output vectors of the network and are respectively defined as

$$\begin{aligned} \boldsymbol{u}(k) &= [u_1(k) \quad \dots \quad u_n(k)]^T \\ \boldsymbol{z}(k) &= [z_1(k) \quad \dots \quad z_m(k)]^T \\ \widehat{\boldsymbol{y}}(k) &= [\hat{y}_1(k) \quad \dots \quad \hat{y}_q(k)]^T \end{aligned} \tag{7.6}$$

The governing equation of the network is

$$\begin{aligned} \boldsymbol{z}(k) &= \boldsymbol{\psi}\big(W\boldsymbol{u}(k) + P\boldsymbol{z}(k-1)\big) \\ \widehat{\boldsymbol{y}}(k) &= V\,\boldsymbol{z}(k) \end{aligned} \tag{7.7}$$

where $\boldsymbol{\psi}$ is the activation function of the hidden layer nodes. The activation function is the tangent hyperbolic function

$$\psi(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{7.8}$$

For a network with n inputs, m hidden layer nodes and q outputs, W and V are $m \times n$ and $q \times m$ matrices respectively and P is, a $m \times m$ diagonal matrix. The cost function for training network s the Sum of Squared Errors (SSE)

$$SSE = \sum_{k=1}^{N} \boldsymbol{e}(k)^T \boldsymbol{e}(k) = \sum_{k=1}^{N} \big(\widehat{\boldsymbol{y}}(k) - \boldsymbol{y}(k)\big)^T \big(\widehat{\boldsymbol{y}}(k) - \boldsymbol{y}(k)\big) \tag{7.9}$$

where $\boldsymbol{e}(k)$ is the error between network output, $\widehat{\boldsymbol{y}}(k)$, the target output $\boldsymbol{y}(k)$, and N is the number of training samples.

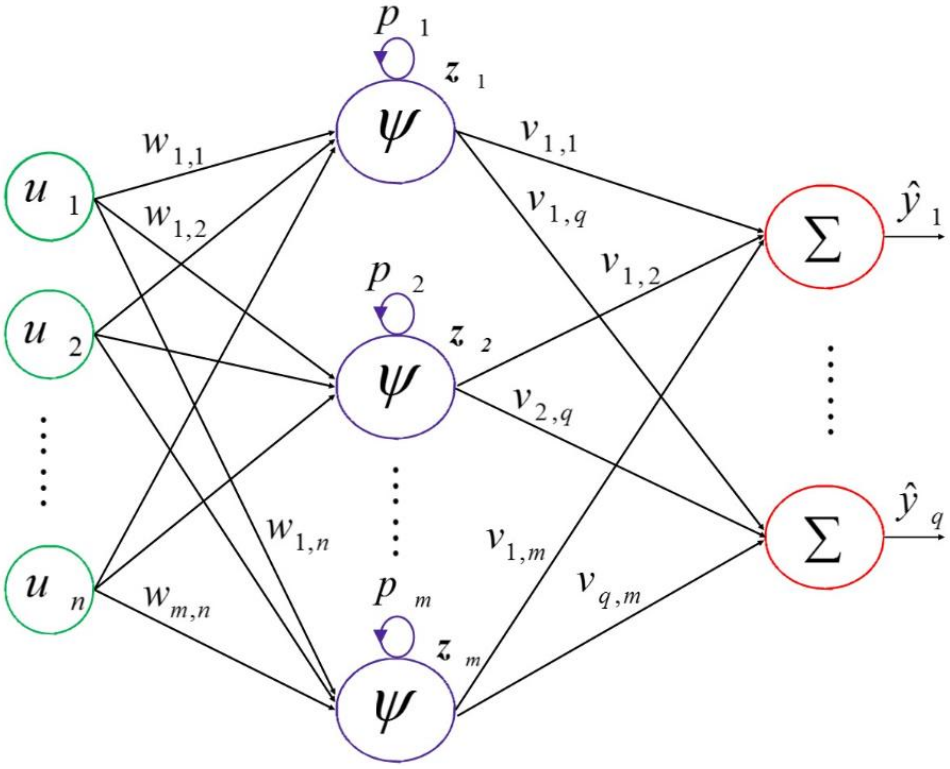

Figure 45. Structure of the neural network

QGS is used to minimize the SSE to find the optimal values of the network weights. QGS provides a systematic search method to find the local minima of the unconstrained minimization problem of (7.2). To train the neural network using the QGS, we write the training set as the equality constraints of (7.1). The resulting minimization problem is to minimize the sum of the squared errors and is solved using the QGS. The QGS finds the set of local minima of the optimization problem and the local minimum with the lowest cost is the global minimum of the minimization problem. If N measurement samples are available, the constraints are

$$\begin{aligned} &\boldsymbol{h}(\boldsymbol{x}) = [h_i(\boldsymbol{x})], i = 1,2,\dots,N \\ &h_i(x) = V\boldsymbol{\psi}\big(W\boldsymbol{u}(i) + P\boldsymbol{z}(i-1)\big) - y(i) \end{aligned} \tag{7.10}$$

where $\boldsymbol{x}$ is the vector of network parameters comprising all the elements of V and W and nonzero elements of P with

$$= \begin{bmatrix} \boldsymbol{v_1^T} \\ \vdots \\ \boldsymbol{v_m^T} \end{bmatrix}_{q\times m} \quad W = \begin{bmatrix} \boldsymbol{w_1^T} \\ \vdots \\ \boldsymbol{w_m^T} \end{bmatrix}_{m\times n} \quad P = diag(\boldsymbol{p})_{m\times m} \tag{7.11}$$

In (7.11), $diag(\boldsymbol{p})$ denotes a diagonal matrix with the elements of $\boldsymbol{p}$ as its diagonal elements. Using (7.11), $\boldsymbol{x}$ can be expressed as

$$\boldsymbol{x} = [x_i]_{n_p \times 1} = [\boldsymbol{v_1}, .., \boldsymbol{v_m}, \boldsymbol{w_1}, \dots, \boldsymbol{w_m}, \boldsymbol{p}]^{\boldsymbol{T}}, \boldsymbol{n_p} = \boldsymbol{m} \times (\boldsymbol{n} + \boldsymbol{q} + \boldsymbol{1}) \tag{7.12}$$

In addition to the constraints of (7.1), upper and lower bounds are imposed on the network parameters. The upper and lower bounds are inequality constraints that can be transformed into equality constraints by introducing slack variables. The QGS for training neural network is written as

$$\dot{\boldsymbol{x}} = -\boldsymbol{f}(\boldsymbol{x}) = -D_x \boldsymbol{h}(\boldsymbol{x})^T \boldsymbol{h}(\boldsymbol{x}) \tag{7.13}$$

Where

$$D_x \boldsymbol{h}(\boldsymbol{x}) = \begin{bmatrix} \dfrac{\partial h_1(\boldsymbol{x})}{\partial \boldsymbol{x}} \\ \vdots \\ \dfrac{\partial h_N(\boldsymbol{x})}{\partial \boldsymbol{x}} \end{bmatrix}_{N \times n_p} \tag{7.14}$$

To train neural network using QGS, the feature vector and target values are used to construct the constraint set (7.10), then the constraint set is transformed into an unconstrained minimization problem. Equation (7.13) defines the QGS for finding local optimal solutions of the unconstrained minimization problem. Using the fact that the equilibrium points of QGS are local minima of the unconstrained minimization problem, the algorithm finds a QGS equilibrium point and then escapes from it and moves towards another QGS equilibrium point. The first step is done by integrating the QGS from a starting point, which need not be feasible, until QGS reaches one of its equilibrium points. Next, we escape from the stability region of the stable equilibrium point to an unstable point with backward integration of QGS in time. After reaching an unstable point, we integrate QGS forward in time to find another stable equilibrium point. The eigenvalues of the Jacobian matrix can be used as a measure of stability. The algorithm continues until it cannot find any new equilibrium point or until it satisfies the stopping criterion.[159].

7.3 Simulation Results

7.3.1 IEEE 123 Bus Test System

To validate the performance of the QGS-trained neural network, we use it to classify events in the IEEE 123-bus test systems. The modified IEEE 123-bus system is shown in Fig.4. The network is composed of

i) four three-phase capacitor banks at buses 51, 57, 83, and 108,

ii) four voltage regulators at bus 149-150, bus 9-14, bus 25-26, and bus 67-160,

iii) 91 loads at different buses,

iv) six normally closed and six normally opened reclosers at several buses.

The line model used for this system is a "Pi" model with shunt capacitance. The line parameters, R, X, and C matrices are properly chosen for the unbalanced system. The loads in the network are defined with their nominal active and reactive powers. The loads are modeled with one of the following three implementations

1) Constant active power P and constant reactive power Q

2) Constant impedance Z

3) Constant current magnitude I

The voltage regulator has a control, which can change the line drop compensator setting by adjusting R, X, the primary of CT, and the PT ratio. There are 33 different tap positions on the regulator that allows for -10% to 10% variation from the nominal value.

There are five PMUs located on five buses, 1, 13, 18, 60, and 97 for streaming data. The PMU located at bus 1 serves as the angle reference for other PMUs. The other four PMUs measure voltage at bus 13, 18, 60, and 97 respectively, and current from bus 152 to 13, bus 135 to 13, bus 160 to 60, and bus 197 to 97, respectively. It is assumed that PMUs only stream the steady-state signals and transient states are ignored. Therefore, all the simulations are performed in the steady-state mode using OpenDSS, a comprehensive electrical power system simulation tool primarily developed for distribution systems [238]. The PMUs used in this paper have two reporting rates: i) 60 sample per second (SPS), such as SEL 651 [239]; and ii) 120 SPS, such as the μPMUs developed at the University of California, Berkeley [240].

Figure 47 depicts the PMU voltage magnitude of phase *a* at bus 60 over one second, 60 samples, corresponding to four events, i) a malfunctioned capacitor bank switching at bus 57, ii) a malfunctioned OLTC switching of the voltage regulator between bus 149 and 150, iii) an abrupt load change at bus 71, and iv) a line reconfiguration due to opening of S_2 and closing S_5. In the malfunctioned capacitor switching, the capacitor switches off and then switches back on. The switching changes the amount of reactive power in the system and, subsequently, causes voltage variations in the capacitor bank substation and neighboring substations. In the malfunctioned OLTC switching, the tap changer dislocates to an unwanted position and then returns to its original

position. Shifting the position of the tap changer causes a change in the resistance and reactance of the regulator and changes the turn ratio of the regulator and the Y-bus of the system. In the reconfiguration event, opening one switch and closing another changes the system topology and the system Y-bus. This changes the voltage and current in the system.

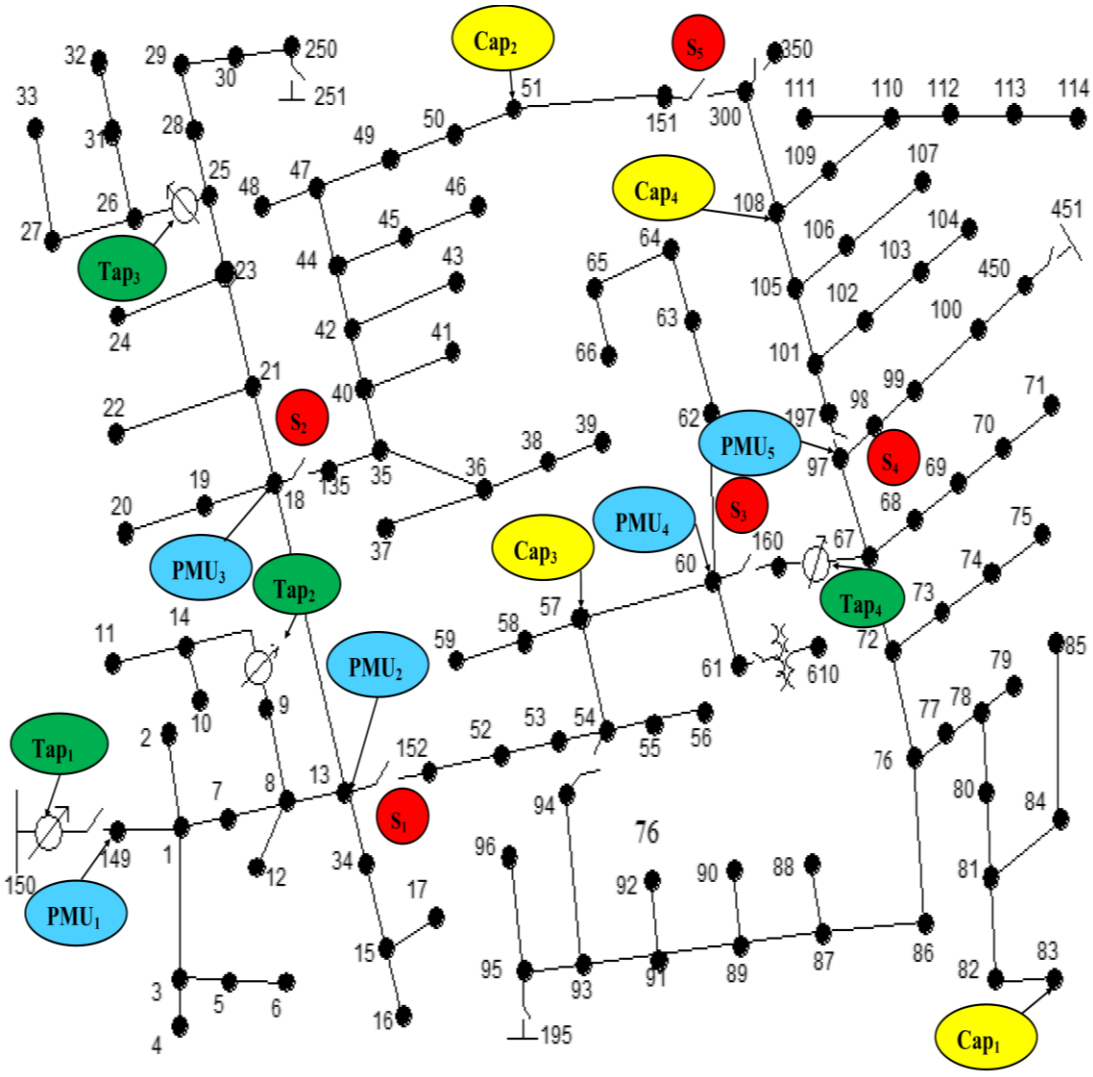

Figure 46. The modified IEEE 123-bus system

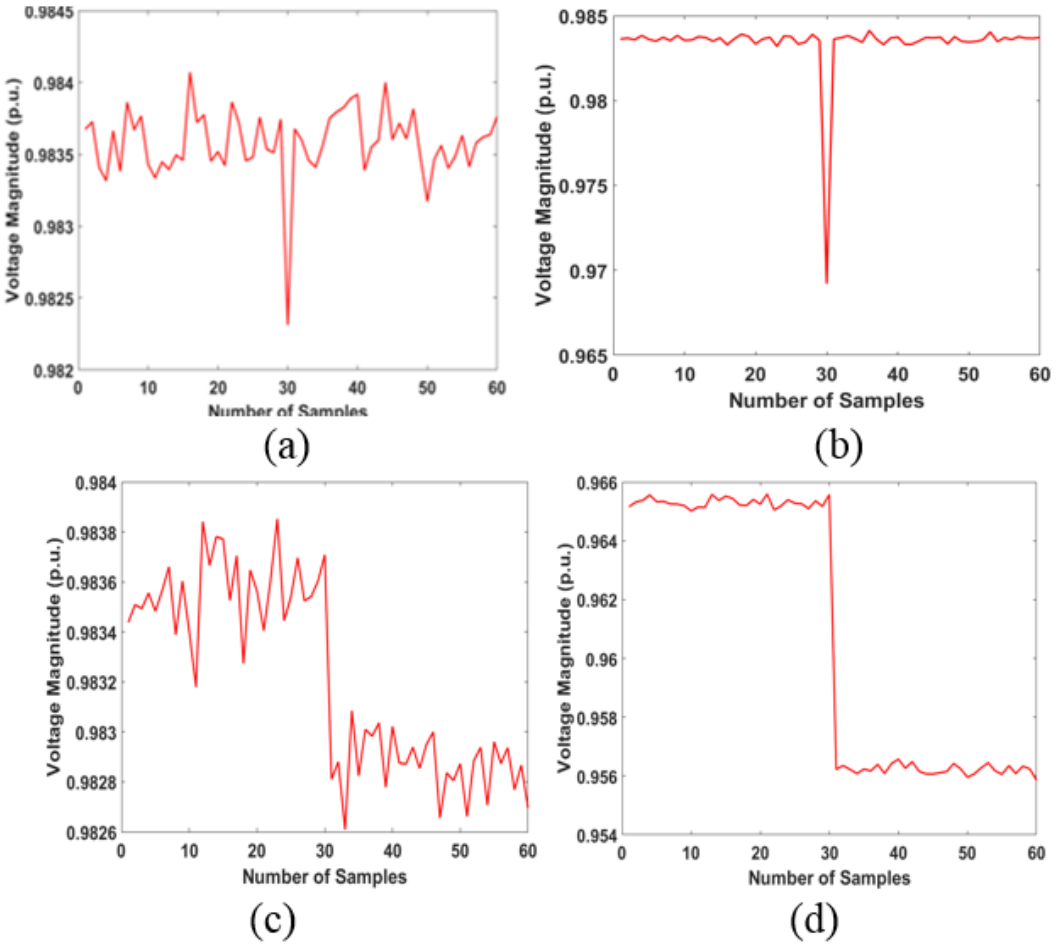

Figure 47. PMU voltage magnitude of phase a at bus 60 over one second a) malfunctioned capacitor bank switching, b) malfunctioned OLTC switching c) abrupt load changing c) reconfiguration

Data-driven event diagnostics methods rely on the availability of historical datasets with event information. However, the number of abnormal events with known class type is limited in practice. Three main solutions are used to overcome this challenge [241],[242],[243]. The first solution is to assume all unknown events as a single class and conduct a supervised event classification based on available data. The second is to apply an unsupervised clustering algorithm to detect all the events that fall under the same group and then use a supervised classification on the available data. The third solution is to simulate all the classes in the power system software, train the data using the classifier, then apply it to the test data. Because of insufficient field data for events, this paper adopts the third solution.

To classify the events according to their types and locations, events are divided into 13 different classes where each class is labeled with a tag indicating its types and location. For example, if an

event is classified as class 2, it is concluded that a malfunctioned capacitor switching occurred at bus 51, as shown in Figure 47. If an event is classified as class 7, it is concluded that a malfunctioned OLTC switching occurred between buses 25 and 26. This procedure is followed for other classes to label all the classes.

To create enough experiments for malfunctioned three-phase capacitor bank switching events (classes 1 to 4), the capacitor switching are simulated at several loading conditions. There are total of 91 different loads in the network, ten different loading levels ranging from 50% to 140% of the average level with 10% increment are simulated. This results in an overall of 910 experiments for the capacitor banks switching class. The same number of experiments are obtained for malfunctioned OLTC switching events (labeled as classes 5 to 8). For a normal abrupt load change (class 9), only one load suddenly changes at a every instant. The abrupt load change is simulated based on 5%, 10%, 15%, 20%, and 25% increase or decrease in its power. Therefore, the overall number of experiments in class 9 is 910. Finally, for the reconfiguration events, it is assumed that one recloser opens and another one closes. We have simulated four different reconfiguration events as: (1) S_1 opens & S_5 closes; (2) S_2 opens & S_5 closes; (3) S_3 opens and S_5 closes; and (4) S_4 opens and S_5 closes. An overall number of 910 experiments is simulated for the reconfiguration events (classes 10 to 13). The accuracy of the classification is assessed using the following index:

$$Accuracy = \frac{number\ of\ accurate\ classification}{total\ number\ of\ test\ cases} \tag{7.15}$$

Preliminary results revealed that classifying between classes 6, 7, 8 and 9 (OLTC switching of regulators two, three, four and abrupt load changing) is difficult due to the similarity of their PMU data. Therefore, a two-stage neural network for event classification is proposed and shown in Figure 48. The extracted feature vector is first fed into the neural network in the first stage. This stage distinguishes the capacitor bank switching and the recloser actions versus the remaining classes. The second stage classifies the OLTC switching versus the abrupt load changes. The optimal number of hidden layer nodes is $m = 8$ for the first network and $m = 6$ for the second neural network. All the parameters of networks are initialized with random values from a zero-mean normal distribution with standard deviation $\sigma = 0.5$. We have 900 samples for each type of event. We randomly choose 100 samples from each type of events to train the neural networks and use the remaining samples as the evaluation dataset. The QGS finds 15 different local minima for the optimization problem. We select the parameter values corresponding to the local minimum

with the best generalization capability as the optimal parameters of the neural network. The neural network with the optimal parameters is used to classify new events.

Table 11 shows the confusion matrix where the performance of the trained neural networks is evaluated. Class 6, the OLTC switching of the regulator between buses 9 and 14 is correctly classified with 88% accuracy and 10% misclassified as class 7 and 2% misclassified as class 5. Class 7, the OLTC switching of the regulator between buses 25 and 26 is classified with 84% accuracy, while 16% of the events are misclassified as class 6. The classification accuracy of class 8 is 98% with 2% misclassification as class 9. The classification accuracy of class 9 is 94% with 6% misclassification as class 8. The classification accuracies for the remaining classes of 100% validates the acceptable performance of the proposed two-layer neural network for event classification in distribution grids. The overall classification accuracy is 96%.

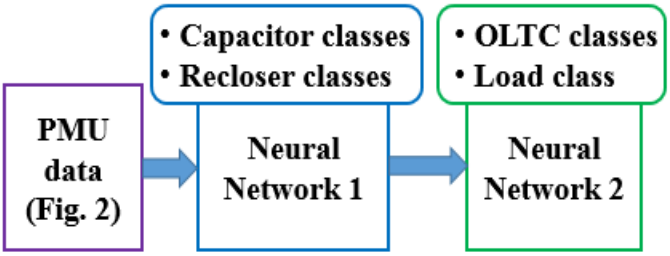

Figure 48. Two-layer NN-based events classification

Table 11. Classification confusion matrix

		Output Class												
Target Class		1	2	3	4	5	6	7	8	9	10	11	12	13
	1	100 %	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	2	0%	100 %	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	3	0%	0%	100 %	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	4	0%	0%	0%	100 %	0%	0%	0%	0%	0%	0%	0%	0%	0%
	5	0%	0%	0%	0%	100 %	0%	0%	0%	0%	0%	0%	0%	0%
	6	0%	0%	0%	0%	2%	88%	10%	0%	0%	0%	0%	0%	0%
	7	0%	0%	0%	0%	0%	16%	84%	0%	0%	0%	0%	0%	0%
	8	0%	0%	0%	0%	0%	0%	0%	98%	2%	0%	0%	0%	0%
	9	0%	0%	0%	0%	0%	0%	0%	6%	94%	0%	0%	0%	0%

	10	0%	0%	0%	0%	0%	0%	0%	0%	0%	100 %	0%	0%	0%
	11	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100 %	0%	0%
	12	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100 %	0%
	13	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100 %

7.3.2 PMU Reporting Rate Analysis

To assess the performance of the proposed method with respect to the reporting rate of PMU, the number of hidden layer nodes of the first and second neural networks must first be determined. In this scenario, the number of hidden layer nodes is 10 for the first neural network and 4 for the second. Increasing the PMU reporting rate does not have a significant impact on the classification rate but improves the overall classification accuracy by 0.66%. This is because a 60 SPS reporting rate is fast enough to capture the events in the grid. Table 12 shows the overall classification accuracies for 60 and 120 SPS reporting rates.

Table 12. Overall classification accuracies for 60 and 120 SPS

PMU reporting rate	60 sps	120 sps
Accuracy	96%	96.66%

7.3.3 Measurement Noise Analysis

PMU data may include errors or measurement noise. To analyze the effect of measurement noise, Gaussian white noise is added to the PMU data stream. The measurement noises with standard deviations of $\sigma^2 = 0.005$, $\sigma^2 = 0.01$, $\sigma^2 = 0.02$, and $\sigma^2 = 0.05$ of the reported phasor values are added to the PMU data, and the noisy data is then used to train and evaluate the neural networks. Figure 49 shows the overall classification accuracies with noisy data. As it can be seen from figure 49, increasing the noise level, decreases the classification accuracy.

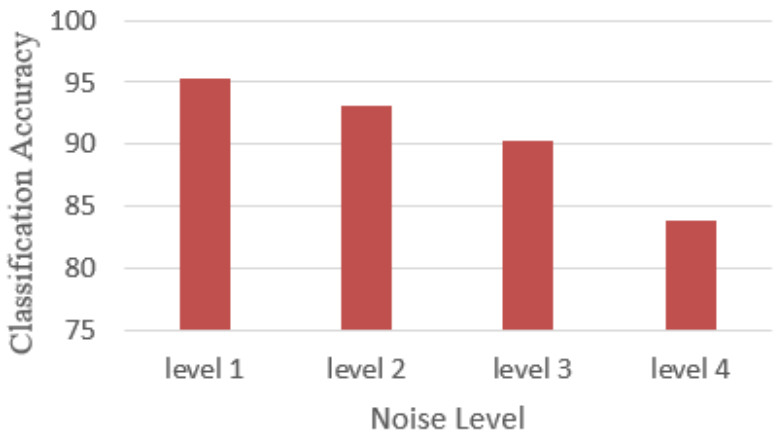

Figure 49. Overall classification accuracies with different level of noise

7.3.4 Number of PMUs

To analyze the impacts of number of installed PMUs on the performnace of the proposed method, the neural networks are trained using the data stream from 3 PMUs (PMU 2, 3, 4), 2 PMUs (PMU 3, 4), and only PMU 4, respectively. Figure 50 shows the overall classification accuracies for different numbers of PMUs. The classification accuracy increases with the number of PMUs. Fewer PMUs result in fewer features from the grid and decreases the classification accuracy. However, with even one PMU the proposed method achieves a limited accuracy level. This shows that the proposed data-driven approach can be useful as utilities install PMUs over an extended period.

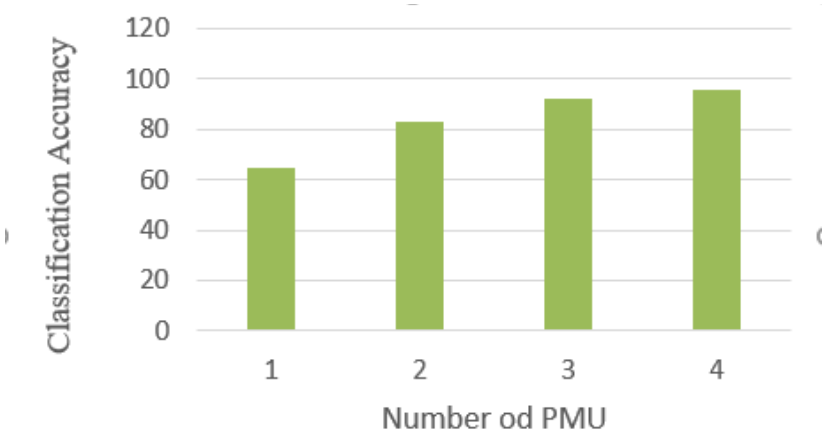

Figure 50. Classification accuracy with different number of installed PMU's

7.3.5 Boosting Scenario

In the boosting scenario, we add the misclassified data to the training set and retrain the neural networks. After training with the new data set, a new set of test data is fed to the neural network, the misclassified data is added to training data, and the network is trained again. This process is repeated three times and then we test the network with a new test data set. Boosting does not appear to have a significant effect on the anomaly classification results and only improves the overall accuracy by a modest 0.44%. Table 13 shows the overall classification accuracy with the boosting scenario.

Table 13. Classification accuracy with boosting scenario

Training scenario	Normal	Boosting
Accuracy	96%	96.44%

7.3.6 Comparison with Traditional Neural Networks

To show the effectiveness of the QGS in training neural networks for event classification and localization, we compare the overall classification accuracy with those obtained by neural networks trained using the classical error backpropagation (EBP) and the genetic algorithm (GA). The genetic algorithm training uses the MATLAB optimization toolbox. An initial population size of 10000, with top fitness scaling, Roulette selection, adaptive feasible mutation and scattered crossover resulted in the best performance for the GA-trained neural network.

Table 14 shows the results of the three different training methods. The overall classification accuracies are calculated with measurement noise variance $\sigma^2 = 0.05$. The QGS and GA result in superior performance compared to backpropagation. This is expected since both methods are global optimization approaches. Although, the QGS- and GA-trained network errors are within the acceptable range, the QGS-trained network has better generalization capability and outperforms the GA-trained network by 6.5%.

Training with EBP was repeated multiple times to find the network with the best performance. Although multiple initialization of EBP can improve the accuracy of the network, both the underlying theory and extensive simulation results show that QGS consistently outperforms EBP. This is because QGS systematically escapes from the stability region of one local minimum and moves toward another and can thus search a larger space for local optimal solutions.

Table 14. Classification accuracy comparison with GA- and EBP-based NN

NN Training method	QGS	GA	EBP
Accuracy	83.77%	77.33%	70.44%

7.4 Conclusion

This study proposed the use of the quotient gradient system (QGS) to train recurrent neural networks (NNs) for classifying and localizing events in power distribution networks. The proposed algorithm is developed to distinguish two classes of events, malfunctioned capacitor bank switching and malfunctioned on-load tap changer (OLTC) switching versus normal recloser switching and abrupt load changing. To enhance the accuracy of state-of-the-art events classifiers, NNs reformulate a constraint satisfaction problem as an unconstrained minimization problem to be solved using the QGS approach. The input data are the phasor measurement units (PMUs) data from the grid. The performance of the proposed algorithm is evaluated using the simulation results in the IEEE 123-bus system. The sensitivity analysis with respect to the reporting rate and number of PMUs, noise level, boosting scenario, and the comparison with genetic algorithm- and error backpropagation-based NNs validate the superior performance of the proposed event classification and localization method.

Chapter 8. Conclusion and Future Work

This dissertation proposes several applications and training methods for artificial neural networks. These methods reduce the number of hidden layer nodes in neural networks, which makes them more desirable for online identification and control applications. In addition, this dissertation proposes using two trajectory based optimization approaches for training neural networks. The trajectory-based optimization makes neural network training a systematic and simple procedure.

Chapter 2 proposes the idea of parallel implementation of different neural networks with feedforward component for nonlinear system identification. The method is tested on two nonlinear systems. Simulation results show that the feedforward component can drastically reduce the number of hidden layer nodes without significant increase of the generalization error and in some cases; it even reduces the generalization error. Future work on this chapter can be proofing the convergence of the echo state network with feedforward component and studying the effect of feedforward component on the short time memory of the echo state network.

In chapter 3, we used wavelet neural network with feedforward component to identify the model of 5 story benchmark structure and designed a model predictive controller based on the wavelet neural network model. Simulation results show that wavelet neural network based model predictive controller can effectively reduce the acceleration and displacements of the base and different stories of the structure during near fault and far field ground motions and outperforms the traditional PID and LQG controllers. Future work on this subject is using echo state network with feedforward component to identify the model of the structure and implementation of the model and controller on the test structure.

Chapter 4 proposes networked control of unmanned vehicle using wavelet neural network and model predictive controller. Wavelet neural network is used to identify the model of unmanned vehicle in presence fixed and random network delay and packet loss and model predictive controller uses the wavelet neural network model to find the future control actions by optimizing the controller cost function. The model predictive controller uses extended prediction horizon to cope with the network delay and packet loss in the network. Simulation results shows that model predictive controller with extended prediction horizon can effectively control the unmanned vehicle and calculates the control action to track the desired path. Lyapunov theory is used to prove

the convergence of the wavelet neural network and prove the stability of the wavelet neural network based model predictive controller. Using different identification method with model predictive controller with extended prediction horizon may reduce the tracking error at the end. In addition, proving the stability of the controller with random network delay and packet loss is a challenging problem.

Training neural networks is a non-convex optimization problem. During past decades, various local and global optimization approaches is used for training artificial neural networks. Chapter 5 proposes using quotient gradient method for training artificial neural network. Quotient gradient method is a systematic approach for finding possible feasible solution of the constraint satisfaction problems. This chapter uses the quotient gradient method to find the local optimal solutions of the neural network optimization problem and compares the results with those of Newton based optimization methods and genetic algorithm. Simulation results shows that quotient gradient method can efficiently train the neural networks and outperforms the gradient descent and genetic algorithm for training neural networks. Lyapunov theory is used to prove the stability of the quotient gradient system and its stability in presence of the measurement noise. Future work on this subject includes application of the methodology for training other types of neural networks such as wavelet networks and deep networks. Furthermore, adapting the methodology to learn the internal structure of the neural network while training the neural network is interesting subject.

In most of the optimization problems, the feasible region of the optimization problem is union of multiple disjoint components. In such a cases, the optimization approach needs to locate disjoint components and search the components for local optimal solutions. Chapter 6 proposes using dynamical trajectory based approach for training artificial neural networks. Dynamical trajectory based methodology uses trajectories of two nonlinear dynamical systems to locate the disjoint components of the feasible region and searching feasible components for local optimal solutions. Simulation results shows that trajectory based optimization approach is able to find multiple components of the feasible region and finds the local optimal solutions in the disjoint components. Furthermore, it shows that trajectory based optimization can effectively train the artificial neural network for nonlinear system identification and outperforms the genetic algorithm. Lyapunov theory is used to prove the stability of the nonlinear dynamical systems and their stability in presence of the measurement noise. Future work on this subject is to use the methodology for training other neural networks, adapting methodology to learn the number of hidden layer nodes

while training and combination of the methodology with pruning algorithms to prune the neural network after training.

Anomaly classification and detection is essential for maintenance of the power networks. Chapter 7 proposes using quotient gradient system for training of two stage partially recurrent neural network for anomaly classification in power networks. The high fidelity data from micro-phasor measurement units is fed to the neural network as the feature vector and the neural network detects the anomaly class based on the feature vector. Simulation results shows that the quotient gradient system can efficiently train the neural network for anomaly classification and outperforms the gradient descent and genetic algorithm trained networks with significant margin. Future work on this subject is to uses quotient gradient system for training feedforward and fully recurrent neural networks for anomaly classification. In addition, combination of quotient gradient system with fully recurrent neural network and pruning algorithms can be subject of another research on this subject.

Chapter 9. References

[1] G. Cybenko, “Approximation by Superposition of a Sigmoidal Function”, Mathematics Control Signals Systems, Vol. 2, pp303-314, 1989.

[2] K. Hornik, M. Stinchcombe and H. White., “Multilayer Feedforward Networks are Universal Approximator”, Neural Networks, Vol. 2, pp 359-3661., 1989.

[3] J. H. Garrett Jr., M. P. Case, J. W. Hall, S. Yerramareddy, A. Herman, R. Sun, S. Ranjithan, and J. Westervelt, “Engineering applications of neural networks,” Journal of Intelligent Manufacturing, Vol. 4, pp. 1-21, 1993.

[4] J.E. Steck, B.M. McMillin, K. Krishnamurthy and M.R. Ashouri, “Parallel implementation of a recursive least squares neural network training method on the Intel iPSC/2”, IJCNN, vol. 1, pp 631-636 San Diego, CA, June `1990.

[5] Q. Zhang and A. Benveniste, “Wavelet networks, “IEEE Trans. Neural Networks, Vol 3, pp. 889–898, 1992.

[6] I.E. Livieris and P. Pintelas, “A survey on algorithms for training artificial neural networks,” Tech. Rep, Department of Math., University of Patras, Patras, Greece, 2008.

[7] D.E. Rumelhart, G.E. Hinton and R.J. Williams, ”Learning internal representations by error propagation” In D. Rumelhart and J. McClelland, editors, Parallel Distributed Processing: Explorations in the Microstructure of Cognition, chapter 8, MIT press, cambridge, MA 1986.

[8] M. Gori and A. Tesi, “On the problem of local minima in backpropagation,” IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol. 14, Issue. 1, pp. 76-86, 1992.

[9] C.L.P. Chen and J. Luo, “Instant learning for supervised learning neural networks: a rank-expansion algorithm,” IEEE International Conference on neural networks, 28 June-2 July, 1994.

[10] P.J. Werbos, “Backpropagation: past and future,” In Proceedings ICNN-88, pp. 343-353, San Diego, 1988.

[11] R.A. Jacobs, “Increased rates of convergence through learning rate adaptation,” Neural Networks, Vol. 1, pp. 295-307, 1988.

[12] G.D. Magoulas, M.N. Vrahatis and G.S. Androulakis, “Effective backpropagation with variable step size,” Neural Networks, Vol. 10, pp. 69-82, 1997.

[13] V.P. Plagianakos, D.G. Sotiropoulos and M.N. Vrahatis, "Automatic adaptation of learning rate for backpropagation neural networks," In N. E. Mastorakis, editor, Recent Advantages in Circuits and Systems, pp. 337-341, Singapore, 1998.

[14] A. Ribert, E. Stocker, Y. Lecourtier and A. Ennaji, "A Survey on Supervised Learning by Evolving Multi-Layer Perceptrons," IEEE International conference on Computational Intelligence and Multimedia Applications, pp. 122-126, 1999.

[15] C. Charalambous, "Conjugate gradient algorithm for effcient training of artificial neural networks," IEEE Proceedings G-circuits,Devices and Systems, Vol. 139, Issue. 3, pp. 301-310, 1992.

[16] M.F. Moller, "A scaled conjugate gradient algorithm for fast supervised learning," Neural Networks, Vol. 6, pp. 525-533, 1993.

[17] J. Nocedal and Y. Yuan, "Analysis of a self-scaling quasi-Newton method. Mathematical Programming," Vol. 61, pp. 19-37, 1993.

[18] S.S. Oren, "Self-scaling variable metric algorithms for unconstrained minimization," PhD thesis, Stanford University, California, USA, 1972.

[19] J.W. Gardner and E.L. Hines, "Pattern analysis techniques," Handbook of Biosensors and Electronic Noses: Medicine, Food and the Environment, (Edited by E. Kress-Rogers), CRC Press, pp. 633–652, 1997.

[20] R. Battiti and G. Tecchiolli, "Training neural nets with the reactive tabu search," IEEE Trans. Neural Netw., Vol. 6, Issue. 5, pp. 1185–1200, Sep 1995.

[21] M. Rocha, P. Cortez and J. Neves, "Evolutionary neural network learning," LNAI 2902, Springer-verlag, pp. 740–741, 2003.

[22] J. Martens and I. Sutskever, "Deep learning with Hessian-free optimization," In Proceedings of the 28th International Conference on Machine Learning (ICML), 2011.

[23] S.M.R. Tousi and S. Aznavi, "Performance optimization of a STATCOM based on cascaded multi-level converter topology using multi-objective Genetic Algorithm," Electrical Engineering (ICEE), 2015 23rd Iranian Conference on, 1688-1693, 2015.

[24] S. Aznavy and A. Deihimi, "Determining Switching Angles And Dc Voltages In Cascaded Multilevel Inverter For Optimizing Thd In Different Output Voltage Levels Using Genetic Algorithm," 6th International Conference on "Technical and Physical Problems of Power Engineering, 2010.

[25] W. Duch and J. Korczak, "Optimization and global minimization methods suitable for neural networks", Dept. of Informatics, NCU technical report 1998.

[26] M.R. Bastian, J.H. Gunther and T.K. Moon, "A simplified natural gradient learning algorithm," Advances in Artificial Neural Systems, Vol. 2011, Article ID 407497, 9 pages, 2011.

[27] L. Bottou, "Large-scale machine learning with stochastic gradient descent," In Proceedings of COMPSTAT, pp. 177–186, Springer, 2010.

[28] A. Georgieva and I. Jordanov, "A hybrid meta-heuristic for global optimization using low-discrepancy sequences of points, Computers and Operations Research," Vol. 37, No. 3, pp. 456-469, March, 2010.

[29] A. Abraham, "Optimization of evolutionary neural networks using hybrid learning algorithms," Journal of Information Technology Education, vol. 4, pp. 20-23, 2002.

[30] J. Lee and H.D. Chiang, "Quotient gradient methods for solving constraint satisfaction problems," IEEE Int. Symp on Circuits and Systems, Sidney, Austrarlia, May 2001.

[31] J. Lee and H.D. Chiang, "A dynamical trajectory-based methodology for systematically computing multiple optimal solutions of general nonlinear programming problems," *IEEE Trans. Autom. Control*, vol. 49, no. 6, pp. 888-899, Jun. 2004.

[32] K.J. Astrom and B. Wittenmark, "Adaptive Control", Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 1994.J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., Vol. 2, pp.68–73. Oxford: Clarendon, 1892.

[33] I.D. Landau, R. Lozano, M. M'Saad and A. Karimi, "Adaptive Control, Algorithms, Analysis, and Applications", 2nd Ed. London,UK, Springer-Verlag, 2011.

[34] F. Allgöwer and Zheng "Nonlinear Model Predictive Control", Birkhäuser, Basel, 2000.

[35] R. Findeisen F. Allgöwer, "An Introduction to Nonlinear Model Predictive Control", 21st Benelux Meeting on Systems and Control, Veidhoven, Netherlands, 2002.

[36] D. Gu and H. Hu, "Wavelet Neural Network based Predictive Control for Mobile Robots",IEEE international Conference on Systems,Man and cybernetics, Vol. 5,pp. 3544-3549, Nashville, TN,2000.

[37] C. De Sousa, E.M. Hemerly and R.K.H Galvao, "Adaptive control for mobile robot using wavelet networks," IEEE Trans on Systems, Man, and Cybernetics, Vol. 32, pp. 493-504, Aug 2002.

[38] S.J. Yoo, Y.H. Choi and J.B. Park, "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-I, Vol. 53, No. 6, pp.1381-1394, Jun 2006.

[39] J. Hespanha, P. Naghshtabrizi and Y. Xu, "A survey of recent results in networked control systems," Proc. IEEE, Vol. 95, No. 1, pp. 138–162, January 2007.

[40] K. Li and G.P. Liu, "A Simplified GPC Algorithm of Networked Control Systems," IEEE International Conference on Networking, Sensing and Control, London, pp. 58-63, April 2007.

[41] Y. Xue and K. Liu, "Variable-sampling networked control systems based on neural network prediction control", WCICA, pp. 8128-8133, Chongqing, China, June 2008.

[42] Z. Jinhui and X. Yuanqing, "Networked predictive output feedback control of networked control systems," proceeding of the 30th Chinese control conference, pp. 4098-4703, Yantai, China, July 2011.

[43] D. Bianchi, A. Ferrara and M.D. Di Benedetto, "Adaptive networked model predictive control of freeway traffic systems", ACC 2013, pp 413-418, Washington, DC USA , June 2013

[44] R. Yang, G.P. Liu, P. Shi, C. Thomas and M.V. Basin, "Predictive Output Feedback Control for Networked Control Systems," IEEE Trans. Industrial Electronics, Vol. 61, No. 1, pp. 512-520 ,Jan 2014.

[45] H. Jaeger, "The 'echo state' approach to analyzing and training recurrent neural networks," German Nat. Res. Center Inform. Technol., St. Augustin, Germany, Tech. Rep. GMD 148, 2001.

[46] C. Hartland N. Bredeche, "Using Echo State Networks for Robot Navigation Behavior Acquisition," IEEE Trans on Robotics andBiomimetics,pp. 201-206,S anya, China, 2007.

[47] S-F. Kazemi, A.J. Hand, E.Y. Hajj, P.E. Sebaaly and R.V. Siddharthan, "Modeling Interface Debonding between Asphalt Layers under Dynamic Aircraft Loading," In Airfield and Highway Pavements 2017 (pp. 71-81).

[48] H. Jaeger and H. Hass, "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless telecommunication," Science, Vol. 304, No. 5667, pp. 78–80, Apr. 2004.

[49] K. Bush and C. Anderson, "Modeling reward functions for incompletestate representations via echo state networks," in Proc. Int. Joint Conf. Neural Netw., pp. 2995–3000, Montreal, QC, Canada, Jul. 2005.

[50] H. Jaeger, “Adaptive nonlinear systems identification with echo statenetwork,” in Advances in Neural Information Processing Systems, vol.15. Cambridge, MA: MIT Press, 2003

[51] M. Salmen and P.G. Ploger, “Echo state networks used for motor control”, In In Proceedings of the 2005 IEEE international conferenceon robotics and automation, pp. 1953–1958,April 2005.

[52] M.H. Tong and A. Bicket, Christiansen, E., & Cottrell, G., “Learning grammatical structure with echo state network,” Neural Networks, Vol. 20, No. 3, pp. 424–432, Apr. 2007.

[53] M. Cernansky and A. Tino, “Predictive modelling with echo state networks,” in Proc. 18th Int. Conf. Artif. Neural Netw., pp. 778–787 , Prague, Czech Republic, 2008.

[54] A. Rodan and P. Tino, “Minimum complexity echo state network,” IEEE Trans. Neural Netw., Vol. 22, No. 1, pp. 131–144, January 2011.

[55] E. De la Rosa, W. Yu and X. Li, “Nonlinear system identification using deep learning and randomized algorithms. 2015 IEEE International Conference on Information and Automation, China, 2015.

[56] K. Binaee, A. Starynska, R. Kothari, C. Kanan, J. Pelz, and G. Diaz, “Modeling Hand-Eye Movements in a Virtual Ball Catching Setup using Deep Recurrent Neural Network,” Journal of Vision, Vol. 17, Issue. 10, pp.17-17, 2017.

[57] S-F. Kazemi, A. Bozorgzad and F. Moghadas Nejad, “Finite-Element Modeling and Laboratory Validation of Evaporation-Induced Moisture Damage to Asphalt Mixtures. In. 97th Transportation Research Board Annual Meeting: National Academy of Science, 2018.

[58] H. Nejib, O. Taouali and N. Bouguila, “Identification of nonlinear systems with kernel methods”, IEEE International Conference on Systems, Man, and Cybernetics (SMC), Hungary, 2016.

[59] M.R. Kandroodi B. Moshiri, “Identification and model predictive control of continuous stirred tank reactor based on artificial neural networks,” ICCIA 2011, pp. 338-343 , Shiraz, Iran, Dec 2011.

[60] A. Ghadirian and M. Zakerim, “MIMO nonlinear dynamic systems identification using fully recurrent wavelet neural network,” ICCIA 2011, pp. 1113-1118, Shiraz, Iran, Dec 2011.

[61] H. Khodabandehlou and M.S. Fadali, "Networked control of unmanned vehicle using wavelet-based generalized predictive controller," Neural Networks (IJCNN), 2016 International Joint Conference on, pp. 5226-5233, Canada, 2016.

[62] I.G. Buckle and R.L. Mayes, "Seismic isolation: History, application, and performance - a world view," Earthquake Spectra, Vol. 6, pp. 161-201, 1990.

[63] F. Naeim and J.M. Kelly, "Design of Seismic Isolated Structures: From Theory to Practice," John Wiley & Sons, Canada, 1999.

[64] J.F. Hall, T.H. Heaton, M.W. Halling and D.J. Wald, "Near-source ground motion and its effects on flexible buildings," Earthquake Spectra, Vol. 11, pp. 569-605, 1995.

[65] R.S. Jangid and J.M. Kelly, "Base isolation for near-fault motions," Earthquake Engineering and Structural Dynamics, Vol. 30, pp. 691-707, 2001.

[66] I.G. Buckle, "New Zealand seismic base isolation concepts and their application to nuclear engineering," Nuclear Engineering and Design, pp. 313-326, 1985.

[67] S-F. Kazemi, P.E. Sebaaly, R.V. Siddharthan, E.Y. Hajj, A.J.T. Hand and M. Ahsanuzzaman, "Dynamic pavement response coefficient to estimate the impact of variation in dynamic vehicle load," Paper presented at the Tenth International Conference on the Bearing Capacity of Roads, Railways and Airfields, Athens, Greece, 2017.

[68] M.J. Kelly, "The role of damping in seismic isolation," Earthquake Engineering and Structural Dynamics, pp. 3-20, 1999.

[69] M. R. Elahifard, S. Ahmadvand, A. Mirzanejad, "Effects of Ni-doping on the photo-catalytic activity of TiO 2 anatase and rutile: Simulation and experiment," Mater. Sci. Semicond. Process, Vol. 84, pp. 10–16, 2018. DOI: 10.1016/j.mssp.2018.05.001.

[70] M. Padervand, H. Salari, S. Ahmadvand, M. R. Gholami, "Removal of an organic pollutant from waste water by photocatalytic behavior of AgX/TiO_2 loaded on mordenite nanocrystals," Res. Chem. Intermed, 38 (8), 2012. DOI: 10.1007/s11164-012-0519-8.

[71] M.J. Kelly, G. Leitmann and A.G. Soldatos, "Robust control of base-isolated structures under earthquake excitation," Journal of Optimization Theory and Applications, Vol. 53, Issue. 2, pp. 159-180, 1987.

[72] J.A. Inaudi and M.J Kelly, "Hybrid isolation systems for equipment protection," Earthquake Engineering and Structural Dynamics, Vol. 22, pp. 297–313, 1993.

[73] S. Nagarajaiah, M.A. Riley and A.M. Reinhorn, "Control of sliding isolated bridge with absolute acceleration feedback," Journal of Engineering Mechanics, Vol. 119, Issue. 11, pp. 2317-2332, 1993.

[74] A.H. Barbat J. Rodellar E.P. Ryan and N. Molinares, "Active control of nonlinear base-isolated buildings," Engineering Mechanics, pp. 676-684, 1995.

[75] N. Makris, "Rigidity-plasticity-viscosity: Can electrorheological dampers protect base-isolated structures from near-source ground motions?," Earthquake Engineering and Structural Dynamics, pp. 571-591, 1997.

[76] M.D. Symans, G.L. Madden A.N. Wongprasert, "Experimental study of an adaptive base isolation system for buildings," Proceedings of Twelfth World Conference on Earthquake Engineering, Auckland, New Zealand, 1965.

[77] G.J. Madden, A.N. Wongprasert and M.D. Symans, "Analytical and numerical study of a smart sliding base isolation system for seismic protection of buildings," Computer-Aided Civil and Infrastructure Engineering, pp. 19-30, 2003.

[78] M. Usman, S.H. Sung, D.D. Jang, H.J. Jung and J.H. Koo, "Numerical investigation of smart base isolation system employing MR elastomer," Proceedings of the 11th Conference on Electrorheological Fluids and Magnetorheological Suspensions, 2009.

[79] M.M.A. Salem, G. Pekcan and A. Itani, "Seismic response control of structures using semi-active and passive variable stiffness devices," Technical Report No: CCEER 14-01, Center for Civil and Earthquake Engineering, University of Nevada Reno, 2014

[80] M. Behrooz, S. Yarra, D. Mar, N. Pinuelas, B. Muzinich, N.G. Publicover, G. Pekcan, A. Itani and G. Gordaninejad, "A self-sensing magnetorheological elastomer-based adaptive bridge bearing with a wireless data monitoring system," Proc. SPIE 9803, Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems, 2016, 98030D (April 20, 2016); doi:10.1117/12.2218691.

[81] R.M. Hasany, Y. Shafahi and S-F. Kazemi, "A Comprehensive Formulation For Railroad Blocking Problem," In ECMS, pp. 758–763, 2013.

[82] H. Chen, G. Tsai, G., J. Qi, F. Yang and F. Amini, "Neural network for structure control," Journal of Computing in Civil Engineering," Vol. 9, Issue. 2, pp. 168–176, 1995.

[83] J. Ghaboussi and A. Joghataie, "Active control of structures using neural networks," Journal of Engineering Mechanics, Vol. 121, Issue. 4, pp.555–567, 1995.

[84] K. Bani-Hani and J. Ghaboussi, "Nonlinear structural control using neural networks," Journal of Engineering Mechanics, Vol. 124, Issue. 2, pp. 319–327, 1998.

[85] D.A. Liut, E. Matheu, M. Singh and D. Mook, "Neural-network of building structures by a force-matching training scheme," Earthquake Engineering and Structural Dynamics, Vol. 28, pp. 1601–1620, 1999.

[86] H.J. Lee, G. Yang, H. Jung, B. Spencer and I. Lee, "Semi-active neurocontrol of a base-isolated benchmark structure," Structural Control and Health Monitoring, Vol. 13, pp. 682–692, 2006.

[87] S. Suresh, S. Narasimhan, S. Nagarajaiah and N. Sundararajan, "Fault-tolerant adaptive control of nonlinear base isolated buildings using EMRAN," Engineering Structures, Vol. 32, Issue. 8, pp. 2477–2487, 2010.

[88] S. Suresh, S. Narasimhan and S. Nagarajaiah, "Direct adaptive neural controller for the active control of earthquake-excited nonlinear base-isolated buildings," Structural Control and Health Monitoring, Vol. 19, pp. 370–384, 2012.

[89] Y. Ohtori, R.E. Christenson, B.F. Spencer and S.J. Dyke, "Benchmark control problems for seismically excited nonlinear buildings," Journal of Engineering Mechanics ASCE, Vol. 130, Issue. 4, pp.366-385, 2004.

[90] M. Han, W. Guo and J.C. Wang, "Predictive control based on feed forward neural network for strong nonlinear system," IEEE International Joint Conference on Neural Networks, pp. 2266-2271, 2005.

[91] M. Zayeni and H. Ahmadi Noubari., "Nonlinear System Identification Using Radial Wavelet Networks," Paper presented at SIAM meeting, St. Louis, Miss, USA, 1995.

[92] T.E. Saaed, G. Nikolakopoulos, J.E. Jonasson and H. Hedlund, "A state-of-the-art review of structural control systems," Journal of Vibration and Control, Vol. 21, Issue. 5, 919-937, 2013.

[93] B. Mullany, H. Shahinian, J. Navare, F. Azimi, E. Fleischhauer, P. Tkacik and R. Keanini, "The application of computational fluid dynamics to vibratory finishing processes," CIRP Annals, Vol. 66 , Issue. 1, pp. 309-312, 2017.

[94] S. Kozák, "State-of-the-art in control engineering," Journal of Electrical Systems and Information Technology, Vol. 1, pp. 1–9, 2014.

[95] G.J. Sirca H. Adeli, “System identification in structural engineering,” Scienta Iranica Transactions A: Civil Engineering, Vol. 19, Issue. 6, pp. 1355-1364, 2006.

[96] S-L. Hung, C.S. Huang, C.M. Wen and Y.C. Hsu, “Nonparametric Identification of a Building Structure from Experimental Data Using Wavelet Neural Network,” Computer-Aided Civil and Infrastructure Engineering, Vol. 18, pp. 356–368, 2003.

[97] B.A. Anderson and J.B. Moore, “Optimal Control: Linear Quadratic Methods,” Prentice-Hall, 1990.

[98] J. Kelly and H.C. Tsai, “Seismic response of light internal equipment in base-isolated structures,” Earthquake Engineering and Structural Dynamics, Vol. 13, pp. 711-732, 1985.

[99] J.,C. Ramallo, E.A. Johnson and B.F. Spencer, “Smart base isolation systems,” Engineering Mechanics , Vol. 128, Issue. 10, pp. 1088-1100, 2002.

[100] F. Ikhouane and J. Rodellar, “Systems with hysteresis, analysis, identification and control using the Bouc-Wen Model,” John Wiley & Sons, Ltd, 2007.

[101] MATLAB. Version 8.3.0.532 (R2014a). The MathWorks Inc., Natick, Massachusetts, 2014.

[102] FEMA. “Quantification of building seismic performance factors,” Technical report, FEMA P695. Prepared by Applied Technology Council for the Federal Emergency Management Agency, Washington, D.C, 2009.

[103] T.C. Yang, “ Networked control system: a brief survey,” IEE Proc. Contr. Theory and Appl., Vol. 153, pp. 403–412, 2006.

[104] T.C. Yang, “On computational delay in digital and adaptive controllers”, Proc. Int. Conf. Control, pp. 21-24, 1994.

[105] J. Chen, K. Gu V. Kharitono, “Stability of time-delay systems,” Birkha¨user, Boston, Massachusetts, 2002.

[106] M.Z. Manutch and M. Jamshidi, “Time-delay systems: analysis, optimization and applications,” Elsevier Science Inc. New York, NY, USA, 1987.

[107] K.J. Astrom and B. Wittenmark, “Computer controlled systems: theory and design,” Prentice-Hall, 1997.

[108] D. Nesic and A.R. Teel, “Input–output stability properties of networked control systems,” IEEE Trans. Automat. Control, Vol. 49, pp. 1650–1667, 2004.

[109] G.C. Walsh H. Ye and L.G. Bushnell, "Stability analysis of networked control systems," *IEEE Trans. Contr. Syst. Technol.*, Vol. 10, pp. 438–446, 2002.

[110] G. Diaz, K. Binaee and F. Phillips, "Predictive movements of the hands and eyes to a target that disappears briefly when moving in depth," Journal of Vision, Vol. 16. Issue. 12, pp. 1349-1349, 2016.

[111] Y.L. Wang, W.T. Liu, X.L. Zhu and Z.P. Du, "A survey of networked control systems with delay and packet dropout", Chineese Control and Decision Conference, pp. 2342-2346 2011.

[112] M. Tatari, M. Fard, N. Nasrolahzadeh, M. Mahjoob, "Characterization of the automotive seat structural dynamics," Proceedings of the FISITA 2012 World Automotive Congress, pp. 541-552, 2013.

[113] M. Akraminia, M. J. Mahjoob, and M. Tatari, "Active noise control using adaptive POLYnominal Gaussian WinOwed wavelet networks," Journal of Vibration and Control 2015, Vol 21, Issue 15, pp. 3020–3033.

[114] G. Xie and L. Wang, "Stabilization of networked control systems with time-varying network-induced delay", Proc. 43rd IEEE Conf. Decision Control, pp. 3551-3556, 2004.

[115] Y.L. Wang and G.H. Yang, "H_∞ control of networked control systems with time delay and packet disordering", IET Control Theory & Applications, Vol. 1, No. 5, pp. 1344-1354, 2007.

[116] P. Fajri, S. Heydari, and N. Lotfi, "Optimum low speed control of regenerative braking for electric vehicles," in 2017 IEEE 6th International Conference on Renewable Energy Research and Applications (ICRERA), San Diego, CA, pp. 875-879, 2017.

[117] Y.L. Wang and G.H. Yang, "State feedback control synthesis for networked control systems with packet dropout", Asian Journal of Control, Vol. 11, No. 1, pp. 49-58, 2009.

[118] K.J. Astrom and B. Bernhardsson, "Comparison of periodic and event based sampling for first-order stochastic systems", Proceedings of the 14th IFAC World congress, Vol. 11, pp. 301-306, 1999.

[119] D. Yue, E.G. Tian and Q.L. Han, "A delay system method for designing event-triggered controllers of networked control systems", IEEE Trans. on Aut. Control, Vol. 58, No. 2, pp. 475-481, 2013.

[120] S.L. Hu, "Analysis and synthesis of networked control systems with event-triggering scheme", Huazhong University of Science and Technology, 2012.

[121] P.G. Otanez, J.R. Moyne D.M. Tilbury, "Using deadbands to reduce communication in networked control systems", in Proc. of American Control Conference, Vol. 4, pp. 3015-3020, 2002.

[122] X. Wang M.D. Lemmon, "Self-triggered feedback control systems with finite-gain stability," IEEE Trans. on Aut. Control, Vol. 54, No. 3, 452-467, 2009.

[123] S. Ahmadvand, R. R. Zaari, S. A. Varganov, "Spin-forbidden and spin-allowed cyclopropenone (C-H_2C_3O) formation in interstellar medium," Astrophys. J. 2014, 795 (2) DOI: 10.1088/0004-637X/795/2/173.

[124] H. Salari, S. Ahmadvand, A. R. Harifi-Mood, M. Padervand, M. R. Gholami, "Molecular-Microscopic Properties and Preferential Solvation in Protic Ionic Liquid Mixtures," J. Solution Chem. 2013, 42 (9), 1757–1769 DOI: 10.1007/s10953-013-0079-6.

[125] M. Miskowicz, "Efficiency of event-based sampling according to error energy criterion", Sensors, Vol. 10, No. 3, pp. 2242-2261, 2010.

[126] W. Heemels, M.C.F. Donkers A.R. Teel, "Periodic event-triggered control based on state feedback," in Proc. IEEE Joint Conf. Decision & Control and European Control Conf., 2011.

[127] X. Wang, E. Kharisov and N. Hovakimyan, "Real-time L1 adaptive control algorithm in uncertain networked control systems," IEEE Trans. on Aut. Control, Vol. 60, No.9, pp. 2500-2505, 2011.

[128] C. Peng and Q.L. Han, "Output-based event-triggered H-infinity control for sampled-data control systems with nonuniform sampling," in Proc. of American Control Conference (ACC), pp. 1727-1732, 2013.

[129] Y. Wang, G. Zhuang, S. Zhang and Z. Shan, "Reliable H∞ Control for a Class of Nonlinear Networked Control Systems with Random Delays," Asian journal of control, Vol. 17, No. 5 , pp. 1821–1830, 2015.

[130] T.B. Wang, Y.L. Wang, H. Wang and J. Zhang, "Observer-Based H_∞ control for continuous-time networked control systems," Asian Journal of Control, Vol. 18, No. 2, pp. 581–594, 2016.

[131] Y. Guan, Q.L. Han and C. Peng, "Decentralized event-triggered control for sampled-data systems with asynchronous sampling," in Proc. Of American Control Conference (ACC), pp. 6565-6570, 2013.

[132] Q. Wang, Y. Zou and Y. Niu, "Event-triggered model predictive control for wireless networked control systems with packet losses," 2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER),Shenyang, China, 2015.

[133] W. Yao, L. Jiang, L.Y. Wen, Q.H. Wu and S.J Cheng, "Wide-area damping controller for power system interarea oscillations: a networked predictive control approach," IEEE Trans. ControL Syst. TechnoLogy, Vol. 23, No. 1, pp. 27-36. 2015.

[134] H. Li, Z. Sun, H. Liu and M.Y. Chow, "Predictive observer-based control for networked control systems with network induced delay and packet dropout," Asian Journal of Control , Vol. 10, No. 6, pp. 638–650, 2008.

[135] G. Cao, E.M. Lai and F. Alam, "Gaussian process model predictive control of unmanned quadrotors," 2016 2nd InternationalConference on Control Automation and Robotics (ICCAR), pp. 200-206, 2016.

[136] C. Lehnert and G. Wyeth, "Locally weighted learning model predictive control for nonlinear and time varying dynamics," Proc. of the Int. Conf. on Robotics and Automation, pp. 2604-2610, 2013.

[137] H. Khodabandehlou and M.S. Fadali, "Echo State versus Wavelet Neural Networks: Comparison and Application to Nonlinear System Identification," IFAC-PapersOnLine, Vol. 50, Issue. 1, pp. 2800-2805, doi: https://doi.org/10.1016/j.ifacol.2017.08.630

[138] M. Minsky and S. Papert, "Perceprrons," Cambridge: MIT Press, 1969.

[139] D.E. Rumelhart, G.E. Hinton and R.J. Williams, "Learning internal representations by error propagation," In D. Rumelhart and J. McClelland, editors, Parallel Distributed Processing: Explorations in the Microstructure of Cognition, chapter 8, MIT press, cambridge, MA, 1986.

[140] C.L.P. Chen and J. Luo, "Instant learning for supervised learning neural networks: a rank-expansion algorithm," IEEE International Conference on neural networks, 1994.

[141] V.P. Plagianakos, D.G. Sotiropoulos and M.N. Vrahatis, "Automatic adaptation of learning rate for backpropagation neural networks," In N.E. Mastorakis, editor, Recent Advantages in Circuits and Systems, pp. 337-341, 1998.

[142] M. Mezard and J.P. Nadal, Learning in feedforward layered networks : the tiling algorithm, Journal of Physics, A.22, pp. 2191-2204, 1989.

[143] S.E. Fahlman snd C. Lebiere, "The cascade-corelation learning architecture," In D. S. Toureski et al, Advances in Neural Information Processing Systems, Vol. 2, pp. 524-532, 1989.

[144] S. Knerr, L. Personnaz and G. Dreyfus, "Single-layer learning revisited: a stepwise procedure for building and training a neural network," In Neurocomputing, NATO ASI Series, Series F, 68, Springer, pp. 41-50, 1990.

[145] L. A. Duffaut Espinosa, M. Almassalkhi, P. Hines, S. Heydari, and J. Frolik, "Towards a Macromodel for Packetized Energy Management of Resistive Water Heaters," in Conference on Information Sciences and Systems, Mar. 2017.

[146] Z. Michalewicz, "Genetic algorithms+data structures=evolution programs," 3rd ed, Springer, 1996.

[147] S. Kirkpatrick, C.D. Gellat Jr and M.P. Vecchi, "Optimization by simulated annealing," Science, Vol. 220, pp. 671-680, 1983.

[148] K.P Unnikrishnan and K.P. Venugopal, "Alopex, a correlation-based learning algorithm for feedforward and recurrent neural networks," Neural Computations,Vol. 6, pp 469-490, 1994.

[149] D. Cvijovic and J. Klinowski, "Taboo search: an approach to the multiple ´ minima problem," Science, Vol. 267,pp. 664-666, 1995.

[150] R. Battiti and G. Tecchiolli, "The reactive tabu search," ORSA Journal on Computing, Vol. 6, No. 2, pp. 126-140, 1992.

[151] R. Battiti and G. Tecchiolli, "Reactive search, a history-sensitive heuristics for MAX-SAT," ACM Journal of Experimental Algorithmics,Vol. 2, Article 2, 1994.

[152] D.E. Goldberg, "A note on Boltzmann tournament selection for genetic algorithms and population oriented simulated annealing," Complex Sys, Vol. 4, pp. 445-460, 1994.

[153] M.K. Sen and P.L. Stoffa, "Comparative analysis of simulated annealing and genetic algorithms: Theoretical aspects and asymptotic convergence," Geophys, 1993.

[154] Y. Shang and B.W. Wah, "Global optimization for neural network training," IEEE Computer, Vol 29,Issue 3, pp. 45-54, 1996.

[155] Sanaye S, Nasab AM. Modeling and optimizing a CHP system for natural gas pressure reduction plant. Energy 2012; 40(1):358–69.

[156] Nasab AM, Sabzehzar A, Tatari M, Majidi C, Shan W. A Soft Gripper with Rigidity Tunable Elastomer Strips as Ligaments. Soft Robotics 2017; 4(4):411-420.

[157] Nasab AM, Wang D, Chen Z, Shan W. Buckling shape transition of an embedded thin elastic rod after failure of surrounding elastic medium. Extreme Mechanics Letters 2017; 15:51-56.

[158] H.Th. Jongen ,P. Jonker and F. Twilt, "Nonlinear Optimization in Finite Dimensions: Morse Theory, Chebyshev Approximation, Transversality, Flows, Parametric Aspects," Kluwer Academic, 2000.

[159] H. Khodabandehlou and M.S. Fadali, "A Quotient Gradient Method to Train Artificial Neural Networks," In Proc. Int. Joint Conf. Neural Networks (IJCNN), Anchorage, USA, 2017.

[160] J. Lee and H.D. Chiang, "Stability regions of non-hyperbolic dynamical systems: Theory and optimal estimation," IEEE International Symposium on Circuits and Systems, pp. 28-31, 2001.

[161] L. Ljung, "Identification of nonlinear systems," IEEE 9th International Conference on Control, Automation,Robotics and Vision (ICARCV), 2006.

[162] A.F. Atiya A.G. Parlos, "New results on recurrent network training: Unifying the algorithms and accelerating convergence," IEEE Trans. Neural Netw., Vol. 11, No. 3, pp. 697–709, May 2000.

[163] P.J. Werbos, "Generalization of backpropagation with application to a recurrent gas market model," Neural Networks, vol. 1, 339-356, 1998.

[164] M. Hermans, J. Dambre and P. Bienstman, "Optoelectronic systems trained with backpropagation through time," IEEE Trans. Neural Netw. Learn. Syst., Vol. 26, No. 7, 1545-1550, 2015.

[165] A.A. Sharma, "Univariate short term forecasting of solar irradiance using modified online backpropagation through time," 2016 International Computer Science and Engineering Conference (ICSEC), 1-6, 2016.

[166] M. Lukoševičius and H. Jager, "Survey: Reservoir computing approaches to recurrent neural network training," Computer science review, Vol. 3, Issue. 3, pp. 127-149, 2009.

[167] J. Martens and I. Sutskever, "Learning recurrent neural networks with hessian-free optimization," In Proceedings of the 28th international conference on machine learning (ICML-11), pp. 1033–1040, 2011.

[168] D.D. Monner and J.A. Reggia, "A generalized LSTM-like training algorithm for second-order recurrent neural networks," Neural Networks, Vol. 25, pp. 70-83, 2012.

[169] Z. Lu., V. Sindhwani and N.S. Tara, "Learning compact recurrent neural networks," *IEEE International Conference on Acoustics, Speech and Signal Processing* (ICASSP), 5960-5964.

[170] L. Zhang and P. N. Suganthan, "A survey of randomized algorithms for training neural networks," Information Sciences, Vol. 364–365, pp. 146–155, Oct. 2016.

[171] K. Hornik, "Approximation capabilities of multilayer feedforward networks," Neural Networks, Vol. 4, No. 2, pp. 251–257, 1991.

[172] Y.Q. Zhu, W.F. Xie and J. Yao, "Nonlinear system identification using genetic algorithm based recurrent neural networks," Canadian Conference on Electrical and Computer Engineering (CCECE), 2006.

[173] A. Ghasemkhani and Lei Yang, "Reinforcement Learning based Pricing for Demand Response," 2018 IEEE International Conference on Communications Workshops (ICC Workshops), 2018.

[174] M. Jafari, V. Sarfi, A. Ghasemkhani, H. Livani, L. Yang, H. Xu and R. Koosha, "Adaptive Neural Network Based Intelligent Secondary Control for Microgrids," in Power and Energy Conference (TPEC), IEEE Texas, 2018.

[175] S. Jagannathan and F.L. Lewis, "Identification of nonlinear dynamical systems using multilayered neural networks," Automatica, Vol. 32, No. 12, pp. 1707-1712, Dec. 1996.

[176] L. Jin and M.M. Gupta, "Stable dynamic backpropagation learning in recurrent neural networks," IEEE Trans. Neural Netw., Vol. 10, No. 6, pp. 1321 –1334, 1999.

[177] F. Azimi, B. Young and B. Mullany, "Statistical Analysis of Surface Measurements and Images," ASPE 32nd annual meeting, Charlotte, NC, USA.

[178] Y. Zhu, "Nonlinear system identification using a genetic algorithm and recurrent artificial neural networks," Master's thesis, Concordia University, Canada, 2006.

[179] Y. Hirose, Yamashita, K., & Hijiya, S., "Backpropagation algorithm which varies the number of hidden units," Int. Joint Conf. Neural Networks (IJCNN), 1989.

[180] V. Prasad and B.W. Bequette, "Nonlinear system identification and model reduction using artificial neural networks," Computers & Chemical Engineering, Vol. 27, No. 12, pp. 1741–1754, Dec. 2003.

[181] E.D. Karnin, "A simple procedure for pruning back-propagation trained neural networks," IEEE Trans. Neural Netw., Vol. 1, No. 2, pp. 239-242, Jun. 1990.

[182] M. Georgiopoulos, C. Li and T. Kocak, "Learning in the feed-forward random neural network: A critical review," Performance Evaluation, Vol. 68, No. 4, pp. 361–384, April. 2011.

[183] J. Salerno, "Using the particle swarm optimization technique to train a recurrent neural model," in Proc. of 9th International Conference on Tools with Artificial Intelligence, IEEE Press, pp. 45–49, 1997.

[184] Y. Yu, L. Xi and S. Wang, "An improved particle swarm optimization for evolving feed-forward artificial neural networks," Neural Processing Letters, Vol. 26, pp. 217–231, 2007.

[185] S. Kiranyaz, I. Turker, A. Yildirim and M. Gabbouj, "Evolutionary artificial neural networks by multi-dimensional particle swarm optimization," Neural Networks, Vol. 22, pp. 1448–1462, 2009.

[186] S-F. Kazemi and Y. Shafahi, "An Integrated Model Of Parallel Processing And PSO Algorithm For Solving Optimum Highway Alignment Problem," Paper presented at the ECMS, 2013.

[187] H.A. Abass, "Speeding up back-propagation using multi-objective evolutionary algorithms," *Neural Computation*, Vol. 15, pp. 2705–2726, 2003.

[188] B. Luitel and G.K. Venayagamoorthy, "A PSO with quantum infusion algorithm for training simultaneous recurrent neural networks," Int. Joint Conf. Neural Networks (IJCNN), 2009.

[189] W. Yu and J.D.J. Rubio, "Recurrent neural networks training with stable bounding ellipsoid algorithm," IEEE Trans. Neural Netw., Vol. 20, No. 6, pp. 983–991, 2009.

[190] J.D.J. Rubio and W. Yu, "Neural networks training with optimal bounded ellipsoid algorithm," in Proc. of the 4th international symposium on Neural Networks: Advances in Neural Networks, Nanjing, China, 2007.

[191] S. Stroeve, "An analysis of learning control by backpropagation through time," Neural Networks, Vol. 11, Issue. 4, pp. 709-721, 1998.

[192] D.T. Mirikitani and N. Nikolaev, "Recursive Bayesian recurrent neural networks for time-series modeling. IEEE Trans. Neural Netw., Vol. 21, No. 2, pp. 262-274, 2010.

[193] D.T. Mirikitani and N. Nikolaev, "Recursive Bayesian Levenberg-Marquardt training of recurrent neural networks," In Proc. Int. Joint Conf. Neural Networks (IJCNN), pp. 1089–1098, 2007.

[194] X. Fu, S. Li, M. Fairbank, D.C. Wunsch and E. Alonso, "Training recurrent neural networks with the Levenberg-Marquardt algorithm for optimal control of a grid-connected converter. IEEE Trans. Neural Netw. Learn. Syst., Vol. 26, No. 9, pp. 1900-1912, 2015.

[195] A. Ghazikhani, T.M.R. Akbarzadeh and R. Monsefi, "Genetic regulatory network inference using recurrent neural networks trained by a multi agent system. 1st International eConference on Computer and Knowledge Engineering (ICCKE), 2011.

[196] Z. Tang, D. Wang and Z. Zhang, "Recurrent neural network training with dark knowledge transfer," IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2016.

[197] J. Guddat, F.G. Vazquez and H.T. Jongen, "Parametric optimization: singularities, path following and jumps," New York, Wiley, 1990.

[198] H.T. Jongen, P. Jonker and F. Twilt, "Nonlinear Optimization in R^n," Germany, Frankfurt: Peter Lang Verlag, 1995.

[199] H.D. Chiang and C.C. Chu, "A systematic search method for obtaining multiple local optimal solutions of nonlinear programming problems. IEEE International Symposium on Circuits and Systems (ISCAS 94), 1994.

[200] C. Meyer, "Matrix Analysis and Applied Linear Algebra," Society for Industrial and Applied Mathematics, Philadelphia, PA, 2000.

[201] W.M. Lin, C.H. Wu, C.H. Lin and F.S. Cheng, "Detection and classification of multiple power-quality disturbances with wavelet multiclass SVM," IEEE Transactions on Power Delivery, Vol. 23, No. 4, pp. 2575-2582, 2008.

[202] S. Santoso, E.J. Powers, W.M. Grady and P. Hofmann, "Power quality assessment via wavelet transform analysis," IEEE Transactions on Power Delivery, Vol. 11, No. 2, pp. 924-930, 1996.

[203] A.M. Gaouda, M.M.A Salama, M.R. Sultan and A.Y. Chikhani, “Power quality detection and classification using wavelet-multiresolution signal decomposition,” IEEE Transactions on Power Delivery, Vol. 14, No. 4, pp.1469-1476, 1999.

[204] M. Khoshdeli, I. Niazazari, R.J. Hamidi, H. Livani, H. and B. Parvin, “Electromagnetic Transient Events (EMTE) Classification in Transmission Grids,” In Proc. Power and Energy Society General Meeting (PESGM), pp. 1-5, 2017.

[205] Z. Moravej, Z., Abdoos, A. A. and M. Pazoki, “Detection and classification of power quality disturbances using wavelet transform and support vector machines,” Electric Power Components and Systems, Vol. 38, No. 2, pp. 182-196, 2009.

[206] S. Sharifi, A. Tivay, S.M. Rezaei, M. Zareinejad, B. Mollaei-DarianiLeakage fault detection in electro-hydraulic servo systems using a nonlinear representation learning approach. ISA Trans 2018. pp. 154-164

[207] S. Sharifi, S. M. Rezaei, A. Tivay, F. Soleymani and M. Zareinejad, "Multi-class fault detection in electro-hydraulic servo systems using support vector machines," *2016 4th International Conference on Robotics and Mechatronics (ICROM)*, Tehran, 2016, pp. 252-257.

[208] M.A.S. Masoum, S. Jamali and N. Ghaffarzadeh, “Detection and classification of power quality disturbances using discrete wavelet transform and wavelet networks,” IET Science, Measurement & Technology, Vol. 4, No. 4, pp. 193-205, 2010.

[209] Z.L. Gaing, “Wavelet-based neural network for power disturbance recognition and classification,” IEEE Transactions on Power Delivery, Vol. 19, No. 4, pp.1560-1568, 2004.

[210] P. Janik and T. Lobos, “Automated classification of power-quality disturbances using SVM and RBF networks,” IEEE Transactions on Power Delivery, Vol. 21, No. 3, pp.1663-1669, 2006.

[211] H. Erişti and Y. Demir, “A new algorithm for automatic classification of power quality events based on wavelet transform and SVM,” Expert systems with applications, Vol. 37, No. 6, pp. 4094-4102, 2010.

[212] M. Parvizimosaed, A. Anvari-Moghaddam, A. Ghasemkhani and A. Rahimi-Kian, “Multi-objective dispatch of distributed generations in a grid-connected micro-grid considering demand response actions,” 22nd International Conference and Exhibition on Electricity Distribution (CIRED 2013), Stockholm, pp. 1-4, 2013.

[213] P.G. Axelberg, I.Y.H. Gu and M.H. Bollen, "Support vector machine for classification of voltage disturbances," IEEE Transactions on Power Delivery, Vol. 22, No. 3, pp. 1297-1303, 2007.

[214] X.F. Song and J.C. Chen, "Classification method of dynamic power quality disturbances based on SVM," Electric Power Automation Equipment, Vol. 26, No. 4, pp. 39-42, 2006.

[215] I. Niazazari and H. Livani, "Disruptive Event Classification using PMU Data in Distribution Networks," In Proc. Power and Energy Society General Meeting (PESGM), pp. 1-5, 2017 .

[216] B.K. Panigrahi and V.R. Pandi, "Optimal feature selection for classification of power quality disturbances using wavelet packet-based fuzzy k-nearest neighbour algorithm," IET Generation, Transmission & Distribution, Vol. 3, No. 3, pp. 296-306, 2009.

[217] M.V. Chilukuri P.K. Dash, "Multiresolution S-transform-based fuzzy recognition system for power quality events," IEEE Transactions on Power Delivery, Vol. 19, No. 1, pp. 323-330, 2004.

[218] G.S. Hu, J. Xie and F.F. Zhu, "Classification of power quality disturbances using wavelet and fuzzy support vector machines," In Proc. of International Conference on Machine Learning and Cybernetics, Vol. 7, pp. 3981-3984, Aug. 2005.

[219] J. Huang, M. Negnevitsky and D.T. Nguyen, "A neural-fuzzy classifier for recognition of power quality disturbances," IEEE Transactions on Power Delivery, Vol. 17, No. 2, pp. 609-616, 2002.

[220] M.S. Manikandan, S.R. Samantaray and I. Kamwa, "Detection and classification of power quality disturbances using sparse signal decomposition on hybrid dictionaries," IEEE Transactions on Instrumentation and Measurement, Vol. 64, No. 1, pp. 27-38, 2015.

[221] H. Kaur, G. Singh and J. Minhas, "A review of machine learning based anomaly detection techniques," International Journal of Computer Applications Technology and Research, Vol. 2, No. 2, pp. 185-187, 2013.

[222] S. Mishra, C.N. Bhende and B.K. Panigrahi, "Detection and classification of power quality disturbances using S-transform and probabilistic neural network," IEEE Transactions on Power Delivery, Vol. 23, No. 1, pp. 280-7, Jan 2008.

[223] F. Zhao and R. Yang, "Power-quality disturbance recognition using S-transform," IEEE Transactions on Power Delivery, Vol. 22, No. 2, pp. 944-950, Apr. 2007.

[224] R. Kumar, B. Singh, D.T. Shahani, A. Chandra and K. Al-Haddad, "Recognition of power-quality disturbances using S-transform-based ANN classifier and rule-based decision tree," IEEE Transactions on Industry Applications, Vol. 51, No. 2, pp. 1249-1258, Mar./Apr. 2015.

[225] C.N. Bhende, S. Mishra and B.K. Panigrahi, "Detection and classification of power quality disturbances using S-transform and modular neural network," Electric Power Systems Research, Vol. 78, No. 1, pp. 122-128, 2008.

[226] S.R. Samantaray, P.K. Dash and G. Panda, "Power system events classification using pattern recognition approach," International Journal of Emerging Electric Power Systems, Vol. 6, No.1, 2006.

[227] K. Manimala and K. Selvi, "Power Disturbances Classification Using S-Transform Based GA–PNN," Journal of The Institution of Engineers (India), Vol. 96, No.3, pp. 283-295, 2015.

[228] I. Niazazari and H. Livani, "A PMU-data-driven disruptive event classification in distribution systems," Electric Power Systems Research, Vol. 157, pp. 251-260, 2018.

[229] K.S. Yap, C.P. Lim and M.T. Au, "Improved GART neural network model for pattern classification and rule extraction with application to power systems," IEEE Transaction on Neural Netw., Vol. 22, No. 12, pp. 2310-2323, Dec. 2011.

[230] D. Biswas, P.M. Adhikari and A. De, "An artificial neural network based power swing classification technique," India Conference (INDICON), 2014 Annual IEEE, 11-13 Dec. 2014

[231] O. Sen, S. Zhengxiang, W. Jinhua and C. Degui, "Application of LVQ neural networks combined with genetic algorithm in power quality signals classification," International Conference on Power System Technology, pp. 491-495, 2002.

[232] R. Kothari, K. Binaee, R. Bailey, C. Kanan, G. Diaz and J. Pelz, "Gaze-in-World movement Classification for Unconstrained Head Motion during Natural Tasks," Journal of Vision, Vol. 17, Issue. 10, pp. 1156-1156, 2017.

[233] M. Elattar and J. Jasperneite, "Using LTE as an access network for internet-based cyber-physical systems," 11th IEEE World Conference on Factory Communication Systems (WFCS), pp. 1-7, May 2015.

[234] L.P. Hayes, "First Energy Corp, Akron, OH," [Online], 2005, Available: http://castlepowersolutions.biz/Components/Joslyn%20Hi-

Voltage/Technical%20Papers/First_Energy_Corporation_Paper_Doble_Conference_2005_VBM_Failures.pdf

[235] R. Jongen, E. Gulski, K. Siodła, J. Parciak and J. Erbrink, "Diagnosis of degradation effects of on-load tap changer in power transformers," ICHVE International Conference on High Voltage Engineering and Application, Poznan, pp. 1-4, 2014.

[236] Siemens Vacuum Recloser 3AD. [Available online] https://w3.siemens.com/powerdistribution/global/SiteCollectionDocuments/en/mv/outdoor-devices/catalogue-vaccum-recloser-3AD_en.pdf

[237] Y. Zhou, R. Arghandeh, I. Konstantakopoulos, S. Abdullah, A.V. Meier and C.J. Spanos, "Abnormal event detection with high resolution micro-PMU data," Power Systems Computation Conference (PSCC), pp. 1-7, 2016.

[238] EPRI, "Simulation Tool – OpenDSS", [online] Available http://smartgrid.epri.com/SimulationTool.aspx.

[239] SEL 651 relay. [Online] Available: https://selinc.com/

[240] A.V. Meier, D. Culler, A. McEachern and R. Arghandeh, "Micro-synchrophasors for distribution systems," In. Proc. Innovative Smart Grid Technologies Conference (ISGT), pp. 1-5, 2014.

[241] S. Brahma, R. Kavasseri, H. Cao, N.R. Chaudhuri, T. Alexopoulos and Y. Cui, "Real-Time Identification of Dynamic Events in Power Systems Using PMU Data, and Potential Applications—Models, Promises, and Challenges," IEEE Transactions on Power Delivery, Vol. 32, No.1, pp. 294-301, 2017.

[242] S.M.M. HN, S. Heydari, H. Mirsaeedi, A. Fereidunian, and A.R. Kian, "Optimally operating microgrids in the presence of electric vehicles and renewable energy resources," in 2015 Smart Grid Conference (SGC), pp. 66–72, 2015.

[243] S. Heydari, S. M. Mohammadi-Hosseininejad, H. Mirsaeedi, A. Fereidunian, and H. Lesani, "Simultaneous placement of control and protective devices in the presence of emergency demand response programs in smart grid," International Transactions on Electrical Energy Systems, 2018; e2537. doi: https://doi.org/10.1002/etep.2537.

[244] H. Khodabandehlou, G. Pekcan, M.S. Fadali, Mohamed and M.A. Salem, "Active neural predictive control of seismically isolated structures," Structural Control and Health Monitoring, Vol. 25, Issue. 1, 2017, doi: https://doi.org/10.1002/stc.2061

[245] S. Aznavi, P. Fajri, M. Ben-Idris and B. Falahati, "Hierarchical Droop Controlled Frequency Optimization and Energy Management of a Grid-Connected Microgrid," 2017 IEEE Conference on Technologies for Sustainability, (SusTech), Phoenix, AZ, pp. 1-7, 2017.

[246] R. Jalilzadeh Hamidi, H. Khodabandehlou, H. Livani, and M. Sami Fadali, "Application of distributed compressive sensing to power system state estimation," NAPS, pp. 1-6, Oct. 2015.

[247] R. Jalilzadeh Hamidi, H. Khodabandelou, H. Livani, and M. Sami-Fadali, "Hybrid state estimation using distributed Compressive Sensing," Power and Energy Society General Meeting (PESGM), Boston, Massachusetts, pp. 1-5, Jul. 2016.

[248] H. Khodabandehlou and M. Sami Fadali, "Training Recurrent Neural Networks as a Constraint Satisfaction Problem," arXiv:1803.07200v6 [cs.LG], 2018.

[249] H. Khodabandehlou, "Adaptive Matching Pursuit based Online Identification and Control Scheme for Nonlinear Systems," arXiv:1803.01897v4 [eess.SP], 2018.

[250] H. Khodabandehlou and M. Sami Fadali, "Networked Model Predictive Control Using a Wavelet Neural Network," Paper under review.

[251] H. Khodabandehlou, I. Niazazari, H. Livani and M.S. Fadali, "Anomaly Classification in Distribution Networks Using a Quotient Gradient System," Paper under review

[252] H. Khodabandehlou and M. Sami Fadali, "Training Recurrent Neural Networks via Dynamical Trajectory-Based Optimization," arXiv:1805.04152v1 [eess.SP], 2018.

[253] H. Khodabandehlou and M. Sami Fadali, "Nonlinear System Identification using Neural Networks and Trajectory-Based Optimization," arXiv:1804.10346v2 [eess.SP], 2018.