Stability analysis of gradient-based training algorithms of discrete-time recurrent neural network

Wu, Yilei

2008

Wu, Y. (2008). Stability analysis of gradient-based training algorithms of discrete-time recurrent neural network. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/13326>

<https://doi.org/10.32657/10356/13326>

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

**STABILITY ANALYSIS OF GRADIENT-BASED TRAINING
ALGORITHMS OF DISCRETE-TIME RECURRENT
NEURAL NETWORK**

**WU YILEI
SCHOOL OF ELECTRICAL AND ELECTRONIC
ENGINEERING
2008**

STABILITY ANALYSIS OF GRADIENT-BASED TRAINING ALGORITHMS OF DISCRETE-TIME RECURRENT NEURAL NETWORK

WU YILEI

School of Electrical and Electronic Engineering

A thesis submitted to the Nanyang Technological University
in fulfillment of the requirement for the degree of
Doctor of Philosophy

2008

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research and has not been submitted for a higher degree to any other University or Institution.

.....

Date

.....

WU YILEI

*To my parents, and dear yuanyuan
for their encouragement and love.*

Acknowledgements

The author would like to express his sincere gratitude to his supervisor, Dr. Song Qing, for his strong guidance and great encouragement in the author's pursuit of a PhD in Nanyang Technological University (NTU). The same gratitude goes to Assistant Professor Shi Daming, School of Computer Engineering, NTU, for the database of handwritten characters and the kind help he has provided. In NTU, the author benefited from the interactions with Dr Liu Sheng, Dr Yang Xulei, Dr Li Jun, Dr Zhou Xin among many other EEE Ph.D. candidates. Special thanks also go to NTU for her financial support during the author's first three years in NTU.

Last but not least, the author is indebted to his family members for their support, understanding and love. This thesis is hence dedicated to the author's parents Zhang Huasheng and Wu huafeng, Zhang Haisheng and Zhu Xiaoyun, and dear wife Zhang Yuanyuan.

Table of Contents

Acknowledgements	i
List of Figures	vi
List of Tables	ix
Summary	xi
1 Introduction	1
1.1 Background and Motivations	1
1.2 Main Contributions	3
1.3 Neural Networks and Recurrent Structure	6
1.4 Stability of RNN and Training Algorithms	10
1.4.1 Lyapunov Approach	12
1.4.2 Input-Output Systematic Approach	13
2 Input-Output Stability Theory of Nonlinear Systems	15
2.1 Mathematical Preliminaries	16

2.1.1	Normed Spaces and Inner Products	16
2.1.2	Lebesgue Function Spaces	19
2.1.3	Frequency Domain Transforms	21
2.1.4	Elements of Matrix Theory	22
2.2	Input-Output Stability	25
2.3	Conic Conditions and Input-Output Stability Theorems	26
3	Gradient Descent Training Algorithms	34
3.1	Review of Gradient Descent Training Algorithms	35
3.1.1	Backpropagation Through Time	35
3.1.2	Truncated Backpropagation Through Time	38
3.1.3	Real Time Recurrent Learning	40
3.1.4	Normalized Real Time Recurrent Learning	43
3.2	Normalized Adaptive Recurrent Learning	45
3.3	Simulation Studies of NARL	54
3.4	Summary	59
4	Robust Adaptive Gradient Descent Training Algorithm	60
4.1	RAGD Training for MISO RNN: Output Layer Analysis	61
4.2	RAGD Training for MISO RNN: Hidden Layer Analysis	70
4.3	RAGD Training for MIMO RNN	78
4.4	Summary	88

5 Applications of RAGD in Realtime Signal Processing	89
5.1 Adaptive Filtering	90
5.2 Time Series Prediction (Revisited)	96
5.3 Output Tracking of Hammerstein-Wiener Model	102
5.4 Binary Image Pattern Association	106
5.5 Summary	110
6 Applications of RAGD in Fault Tolerant Control of Robotic System	113
6.1 Dynamics of 7-link Robot Model	114
6.1.1 Physical Structure of the Robot	114
6.1.2 Walking Patterns	115
6.1.3 Digital Controller	117
6.2 Closed Loop Stability with RAGD Training	120
6.3 Experimental Simulations	125
6.4 Summary	129
7 Conclusion and Future Works	135
7.1 Conclusion	135
7.2 Future works	137
Author's Publications	140

Bibliography	141
---------------------	------------

List of Figures

1.1	Structure of a single neuron	7
1.2	Externally feedback RNN	9
2.1	Closed loop feedback system	27
2.2	Loop transformation	32
3.1	Illustration of BPTT algorithm: unfolds a recurrent model back into the past until the initial input values at time $k = 0$ are reached. . . .	38
3.2	Squared training errors of first 100 steps of the NARL and N-RTRL training respectively (transient response)	56
3.3	Squared training errors (in dB) of full 3000 steps of the NARL and N-RTRL training respectively (steady state response)	56
3.4	Traces of augmented residual error derivative $\gamma_1^w(k)$ and learning rate $\alpha^v(k)$ of the NARL	57
3.5	Prediction errors (in dB) of the NARL and N-RTRL with the evalu- ation data set	58
4.1	Closed loop dynamics of output layer training of RNN	63

4.2	Closed loop dynamics of hidden layer training of RNN	72
4.3	Illustration of a local attractor basin of the RNN against a scalar estimated weight $\hat{W}(k)$	76
4.4	Flow chart of the RAGD algorithm for MISO RNN	79
5.1	Time sequences of signal and noise and respective power spectrums .	91
5.2	Squared training errors of the first 100 steps with the same set of random initializations for different algorithms	93
5.3	Squared training errors of full 3000 steps for different algorithms . . .	94
5.4	Traces of normalization factors $\rho^v(k)$ and $\rho^w(k)$	94
5.5	Traces of learning rate $\alpha^v(k)$ and $\alpha^w(k)$	95
5.6	Traces of hybrid adaptive learning rate $\beta^v(k)$ and $\beta^w(k)$	95
5.7	Traces of the Frobenius norms of RNN weights with the RAGD training	96
5.8	Sequences of the time series for training and evaluation	97
5.9	Squared training errors of the first 100 steps with the same set of random initializations for different algorithms	98
5.10	Squared training errors of full 3000 steps for different algorithms . . .	99
5.11	Traces of normalization factors $\rho^v(k)$ and $\rho^w(k)$	99
5.12	Traces of learning rate $\alpha^v(k)$ and $\alpha^w(k)$	100
5.13	Traces of hybrid learning rate $\beta^v(k)$ and $\beta^w(k)$	100
5.14	Traces of the Frobenius norms of RNN weights with the RAGD training	101
5.15	Trace of model input: AMPRP sequence	103

5.16	Squared training errors of the first 1000 steps	104
5.17	Squared training errors of steady state: 1000–5000 steps	104
5.18	Traces of the normalization factors $\rho^v(k)$ and $\rho^w(k)$	105
5.19	Traces of learning rate $\alpha^v(k)$ and $\alpha^w(k)$	105
5.20	Traces of hybrid learning rate $\beta^v(k)$ and $\beta^w(k)$	106
5.21	The binary patterns correspond to the state evolution of RNN during the training process using the RAGD algorithm.	109
5.22	The binary patterns correspond to the state evolution of RNN during the training process using the BPTT algorithm.	109
5.23	Comparison of the squared error curves between the RAGD and BPTT training procedures.	110
5.24	The binary patterns correspond to the state evolution of association pro- cess of RNN trained by the BPTT algorithm.	111
5.25	The binary patterns correspond to the state evolution of association pro- cess of RNN trained by the RAGD algorithm.	111
6.1	Structure of a 7-link rigid biped robot	115
6.2	Block diagram of RNN+PD control scheme	120
6.3	Reference of joint angle signals	130
6.4	Case 1–tracking error in case of no fault (PD control), transient state	130
6.5	Case 1–tracking error in case of no fault (PD control), steady state	131
6.6	Case 2–steady state of the tracking error with pure PD controller, fault occurs at 3.6th second	131

6.7	Case 2—steady state of squared tracking error with hybrid controller scheme (PD+RNN), fault occurs at 3.6th second	132
6.8	Case 3—steady state of the squared tracking error norms with different control schemes, fault occurs at 1.7th second	132
6.9	Case 3—steady state of the squared training error norms of RNN with different training algorithms	133
6.10	Case 3—trajectories of the Frobenius norms of RNN weight matrices with the RAGD training	133

List of Tables

3.1	Statistics of the training errors of the NARL and N-RTRL with different constant C	57
3.2	Statistics of the training errors of the NARL with different λ and neuron number respectively	59
5.1	Statistics of squared training errors of the RAGD with different λ . .	101
5.2	Statistics of squared training errors of the RAGD with different neurons	102
6.1	Calculation of the coefficients of the robot model	127
6.2	Nominal values of physical parameters of the robot model	128

Summary

Recurrent Neural Network (RNN) is a powerful tool for both theoretical modelling and practical applications. To utilize the RNN as a general learning tool, the understanding of its properties, particularly the robustness and stability, are required. In this thesis, we aim at studying the robustness of the gradient-type training algorithms of the RNN via input-output analysis method of nonlinear system theory.

The work in this thesis originates from modern concepts of control theory, especially the techniques that have been developed for the analysis of feedback systems. A number of new results are presented that are able to effectively improve the transient response of RNN training algorithms. Further, the results lead to many new theoretical concepts and offer some practical approaches, which may be useful in a wide range applications, for instance, signal processing and control problems. In addition to the analytic derivations, we also demonstrate how the derived criterion can be evaluated numerically. Several examples of using RNN to learn dynamics in practical systems are given based on computer simulations.

The overall thesis is organized as follows: Chapter 1 introduces the background, motivations, and major contributions of the thesis, as well as the fundamental knowledge of neural networks. Chapter 2 quickly reviews the related mathematical preliminaries of nonlinear system theory. Specifically, Cluett's law is introduced at the end of the Chapter as an extension to the Conic Sector Stability Theory of Safanov, which will be used in the theoretical analysis of the proposed algorithms that followed. In Chapter 3, firstly the shortcomings of the conventional training, e.g., Realtime Recurrent Learning (RTRL) and Normalized RTRL (N-RTRL), are described, and then the Normalized Adaptive Recurrent Learning (NARL) is proposed to overcome the slow convergence of these algorithms. Inspired by the works of the N-RTRL, normalization factors are used in the NARL to speed up the training. In addition, another two new elements are also introduced, namely, adaptive

learning rate and augmented residual error gradient to strengthen the robustness of the training. Analytical analysis is given to compare the performance between the NARL and the other competitors.

However, as shown in the proof of NARL that there are also limitations in training induced by the augmented residual error gradient. In order to address the problems, a novel Robust Adaptive Gradient Descent (RAGD) training algorithm is proposed in Chapter 4. In addition to the adaptive learning rate, normalization factors, and augmented error gradient, a concept so-called hybrid learning is proposed in the RAGD to ensure the convergence of RNN weights. The robust stability of the RAGD is proved via the Lyapunov approach and the Chuetz's Law respectively.

In Chapter 5, numerical simulations in realtime signal processing are carried out to evaluate the proposed algorithms, e.g., online adaptive filtering, time series prediction etc. Other training algorithms are also implemented with the same RNN structure to practically compare their difference with the RAGD. In Chapter 6, a comprehensive case study of Fault Tolerant Control (FTC) for biped robot tracking system is developed on the basis of RNN and the RAGD. Three fault cases are synthesized in the simulation to verify the effectiveness of the proposed schemes. Comparison with the single PD control scheme and other training algorithms are presented. Finally in Chapter 7, we draw the conclusions and give several advices on future works.

Chapter 1

Introduction

1.1 Background and Motivations

In the past decades, Artificial Neural Network (ANN) has attracted extensive interests in various research fields. One important motivation of these investigations is that ANN can be trained to derive meaning intelligently from complicated or imprecise data, which result in the remarkable capability in modeling nonlinear processes. Further, it can also be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. These intellectual characteristics make ANN be successfully applied in a large number of scientific disciplines, to name a few, computational neuroscience, statistical physics, distributed processing, pattern recognition, image analysis and control theory. More importantly, ANN has seen steadily growing industrial applications world widely.

Recurrent neural network (RNN) is a type of ANN that has feedback connections. An explicit advantage of RNN over the other types ANN is the capability of modeling time-behavior of dynamic systems. Theoretically RNN has been proved to be able to map input sequences to output sequences with infinite accuracy re-

regardless underline dynamics with sufficient training samples. Experimentally the characteristic has been verified by many recent works, showing that RNN can map successfully input data into a desired time-varying trajectory for a wide range of applications [6] [27] [54]. The approximation capability renders RNN as a highly competitive candidate for learning dynamic input-output relations. In addition, from biological point of view, RNN is more plausible to the real neural models, with more powerful computation methods compared to other adaptive methods such as Hidden Markov Models (HMM), feed-forward networks and Support Vector Machines (SVM).

However, due to the large number of state variables and nonlinear distributed nature of computation, many properties of RNN dynamics are not well known so far, such as existence of equilibria, convergence, boundedness, and stability etc. In many cases, RNN is designed and tuned by empirical method without any theoretical guidance. Works on these aspects are still largely open. For example, in practical applications, the process of obtaining a mathematical model of a system always involves approximations, which may introduce imprecision during the modeling procedure. Furthermore, realistic processes are frequently subject to external disturbance, and in addition there may exist imperfectly known inputs. Therefore, in these applications robustness is an important issue for RNN training algorithms to behave properly under the presence of uncertainty. On the other hand, the conventional training algorithms of RNN such as the Backpropagation Through Time (BPTT) and the Real Time Recurrent Learning (RTRL) always suffer from slow convergence speed. If a large learning rate is selected to speed up the weight updating, the training process may become unstable. To solve the problem, it is desirable to develop robust learning algorithms with variable or adaptive learning coefficients to obtain a tradeoff between the stability and fast convergence speed.

In literature, robust stability of RNN has been a particular subject of interest in

many works. Most of them focus on the case of zero or constant input with the objective to show either the origin is globally absolutely stable, or to provide the respective local conditions for multi-stable points with many equilibria. In contrast, despite that there are also some investigations in the robustness of RNN training, the existing research results are far less satisfied compared to the former topic. Among the published RNN articles, some of them rely on coarse approximation of the error gradient by ignoring the higher order derivative. These methods could probably result in insufficient criterion of stability analysis, because it is possible that the approximated gradient is not the same sign as the correct one, which will lead to a wrong gradient search direction in updating rule. Others utilize the Lyapunov method and most of the derived conditions are computationally intractable, because the parallel solution of 2^N condition at all vertices of the convex matrix polytope is not valid. From above arguments, we are motivated to extend the existing methodology and knowledge at certain points to get more reasonable results for robust RNN training algorithms.

1.2 Main Contributions

The aim of the thesis is to investigate the robustness of gradient-type training algorithms of RNN in the discrete-time domain. We firstly propose the Normalized Adaptive Recurrent Learning (NARL) intending to overcome the shortcoming of slow convergence of conventional training algorithms. Inspired by the works of the Normalized RTRL (N-RTRL), we use normalization factors in the NARL to speed up the training. In addition, we also introduce another two new elements in the NARL, i.e., adaptive learning rate and augmented residual error gradient, to strengthen the robustness of the training. Analysis shows that the performance of the NARL is improved in terms of transient and steady state response, as compared to the N-RTRL.

In the second part of the thesis, a novel Robust Adaptive Gradient Descent (RAGD) training algorithm of RNN is proposed in Chapter 4. The RAGD uses a specifically designed error derivatives, which are based on the extended recurrent gradient, to approximate the true gradient for realtime learning. This method provides an improved RNN training speed compared to the standard RTRL algorithm, because the latter has to use a small learning rate to slow the weight changes and follow the true gradient of the RNN. The key difference between the RAGD and the standard BPTT, RTRL is that we use the three proposed adaptive parameters in the RAGD to guarantee the weight convergence and L_2 stability of the training. The first parameter is the hybrid adaptive learning rate, which provides a hybrid training fashion and controls the RAGD to change the training patterns between the standard real time Backpropagation (BP) and the RTRL algorithms according to the proved stability and convergence conditions. Adaptive learning rates based on the dead zone concept and adaptive normalization factors are also used in the RAGD, which are similar to the schemes in the classical control systems and adaptive signal processing. The combination of the three adaptive learning parameters allows the RAGD to overcome slow convergence or instability problem of the RTRL algorithm with a maximum effective learning rate and to locate relatively deeper local attractors of the training. Also the RAGD is different from the static BP algorithm in terms that the former uses the extended recurrent gradient to extend the instantaneous squared estimation error minimization into recurrent fashion, while the latter is strictly based on the instantaneous squared estimation error minimization without specifically considering the recurrent signal. Indeed, the idea of RAGD is a generalized form of training of many existing works [78]– [105]. If we calculate the derivative exactly by unfolding the recurrent structure and set the hybrid learning rate to zero, i.e. pursuing all N steps back in the past, then the algorithm will recover the BPTT algorithm [78] [79]. Moreover, if we assume that the model parameters do not change apparently between each iteration, then we can derive the approach of the

RTRL [103] [105].

The overall weight convergence and robust stability of the RAGD is proved by using the Lyapunov function and the conic sector theorem and its extensions respectively. Different from those works based upon the pure Lyapunov method, we employ a number of methods from input-output systematic theory to address the stability problem of the training. This is because among the various approaches to nonlinear system analysis, the input-output theory on basis of functional analysis provides a natural tool to answer questions about neural networks as well as RNN, as we only need to make minimal assumptions about the process which we are controlling or predicting. Further, the input-output framework of RNN in this thesis is developed through nonlinear system analysis approach along with several recently developed numerical methods. Though many of the concepts used in the proof are classical, the connection and joint application in the area of RNN yields several new results. This contributes to enrich the theoretical understanding of the dynamics of RNN training.

The third part of the thesis consists of two chapters of case studies. In the first one, we carry out four computer simulations of realtime signal processing applications, including adaptive filtering, time series prediction etc. The experimental results show that the RAGD outperforms other counterpart algorithms, and in turn justify the theoretical results quantitatively. In the second case study, we provide a more comprehensive example. RNN and the RAGD algorithm is applied in the fault tolerant control of a robotic system. RNN is designed to compensate unknown system faults. The stability of the overall control system is addressed based on the RAGD. Simulations show that the RNN trained by the RAGD algorithm facilitates to improve robustness of the entire closed loop control system. With these case studies, we are able to qualify the effectiveness of the RAGD and hence draw the conclusion.

Finally we point out that the input-output systematic methods developed in this thesis are not only restricted to RNN but hold for other more general system analysis as well. It can be expanded as a generalized method of stability analysis of any RNN related closed loop system, which is already illustrated as an example in Chapter 6. Next, before introducing the proposed robust training algorithm and carrying out the formal theoretical analysis, we will give a brief review on basic knowledge of ANN in this chapter and necessary mathematical preliminaries in Chapter 2.

1.3 Neural Networks and Recurrent Structure

The concept of ANN is inspired by the way how mammalian brains processing information. The mathematical model roots in emulating the properties of biological nervous systems, and drawing on the analogies of biological learning [27]. One of the most remarkable features of ANN is the densely interconnected and parallel distributed architecture. It is actually a collection of a large number of information processing units: neurons, which are analogous to biological neurons and are tied together with weighted connections that are analogous to synapses. In literature, there are many proposals of such neuron models. One of the widely used structures is shown in Figure 1.1, which can be described by the equation below

$$\begin{cases} \hat{x}(k) = \sum_{i=1}^n \hat{w}_i(k) u_i(k) \\ \hat{y}(k) = \phi(\hat{x}(k) + b(k)) \end{cases} \quad (1.1)$$

where $u_i(k)$, $\hat{x}(k)$, $\hat{y}(k)$ and $b(k)$ are neuron's external input, internal state, output and bias respectively, n is number of input nodes, and $\hat{w}_i(k)$ is the synaptic weights of neuron, and $\phi(\cdot)$ denotes the activation function that defines the output of a neuron in terms of the induced local field $\hat{x}(k)$. There are several types of $\phi(\cdot)$ that are frequently used in literature, such as threshold function, piecewise-linear

function etc. In this work, we employ the sigmoidal function with the form

$$\phi(x) = \frac{1}{1 + \exp(-\lambda x)} \quad (1.2)$$

in which λ is a positive number that represents the slope steepness of the transient range of activation function.

Figure 1.1: Structure of a single neuron

There are many prominent advantages of the above described neuron model, such as the adaptive learning ability, self organization behavior, real time operating, redundant information coding etc. These features make the neuron benefit from the advantages of an intelligent and high fault tolerant capability. If we connect the above described neurons each other by synaptic weights, then a classical ANN is formed. Actually, the type of ANN is intimately linked to the manner in which the neurons are structured. In general, it can be categorized into two major types according to network structures: feed-forward network and recurrent network. Among the feed-forward networks, ANN can be further classified into two sub-groups: single-layer and multi-layer. Single-layer feed-forward network is the simplest form of ANN, that has an input layer of source nodes projected onto an output layer of computation neurons. The designation “single-layer” refers to the output layer of computation neurons while the input layer of source nodes isn’t counted because no computation

is performed there. Multi-layer feed-forward network distinguishes itself by the presence of one or more hidden layers, whose computation nodes are correspondingly called hidden neurons. The function of hidden neurons is to intervene between the external input and the network output in an implicit manner. By adding one or more hidden layers, the network is enabled to extract higher-order statistics. This ability is particularly valuable when the size of input layer is large. As for recurrent network, it is one of the most complex types of ANN that is different from a feed-forward network in that it has at least one feedback loop. For example, a RNN may consists of single layer of neurons with each neuron feeding its output signal back to the inputs of all the other neurons, which is so-called Hopfield network.

The presence of feedback loops has a profound impact on the learning capability of the network and on its performance. Moreover, the feedback loops involve the use of particular branches composed of unit-delay elements, which result in a nonlinear dynamic behavior with operations of nonlinear activation functions, and RNN is enabled to extract statistical variations in non-stationary processes such as speech signals, radar signals, signals picked up from the engine of an automobile, and fluctuations in stock market prices, just to mention a few. Again from the structure point of view, by the different manners of delay, and in turn different dynamics, RNN can be grouped into the following three major classes: i) Focused time lagged network, a powerful nonlinear filter consists of a tapped delay line memory of n order and a multi-layer perceptron. ii) External dynamic network, which is a nonlinear autoregressive model with exogenous input (NARX) model. iii) Internal dynamic network, which is based on the extension of static models with internal memory. A particular example is the famous Hopfield network. As for a detail description on the properties of these three type RNN, readers may refer to [27] [97].

Among the three types of RNN, external dynamic network is by far the most frequently applied in practical design, such as nonlinear dynamic system identification,

model prediction etc. The name “external dynamic” stems from the fact that nonlinear dynamic model can be clearly separated into two parts, a nonlinear static approximator and an external dynamic filter bank, see an example of the Single-input-Single-output (SISO) network in Figure 1.2. The discrete-time nonlinear dynamic RNN model at time instant k can be written as

$$\hat{y}(k) = \hat{V}(k)\Phi(\hat{W}(k)\hat{x}(k)) \quad (1.3)$$

where $\hat{V}(k) \in R^{1 \times m}$ and $\hat{W}(k) \in R^{m \times n}$ are output and hidden layer weights

Figure 1.2: Externally feedback RNN

respectively (in matrix form), $\Phi(\cdot)$ is a vector consisting of nonlinear activation functions defined in (1.2), $\hat{y}(k)$ is network output, and $\hat{x}(k)$ is the state vector that consists of 1 external input $u(k)$ and $n - 1$ delayed output feedback entries

$$\hat{x}(k) = [u(k), \hat{y}(k-1), \dots, \hat{y}(k-n+1)]^T \quad (1.4)$$

in which T denotes transpose operation. To simplify the expression, we use notation $\Phi(k)$ instead of $\Phi(\hat{W}(k)\hat{x}(k))$ hereafter. When estimating a command signal $d(k)$, the instantaneous modeling error of RNN is defined by equation (1.5). Note a disturbance term $\varepsilon(k)$ is also taken into account. Without loss of generality, there

is no assumption on the prior knowledge of $\varepsilon(k)$ and its statistics.

$$e(k) = d(k) - \hat{y}(k) + \varepsilon(k) \quad (1.5)$$

A significant property of RNN is the ability to learn dynamics from its environment. Once we have chosen the network architecture and the parameters of the neurons, the second step is to adjust the weight parameters to implement certain functions or make the RNN output best fit into certain curve with the given data, and meanwhile cost function of the modeling error is minimized. There are a number of training algorithms which have been proposed in literature. Most of them belong to one of the following three types: local optimal algorithms, e.g., gradient-based training, semi-global optimal algorithms, such as deterministic annealing, and global optimal, e.g., gene algorithm. So far the gradient-based training is one of the most fully developed and widely used methods due to the ease of implementation and high adaptability. Most importantly, it can be utilized in case of training data set which has non-stationary statistics.

1.4 Stability of RNN and Training Algorithms

Though the gradient-based algorithms have many advantages, one of their common problems is the slow convergence speed. How to choose suitable learning rates or adaptation gains during training is still an active research topic. It is well known that small learning rate may result in excessive number of iterations to reach the optimum. On the contrary, a large learning rate may lead to big steady-state error, even an unstable training. A common sense is to choose a small enough learning rate to ensure the a stable training while sacrifice the convergence speed [66]. This method is applicable in the case that stability is much more important than convergence speed. However, the approach is not acceptable when transient response

speed becomes a critical measurement of performance, such as those realtime applications with tight timing requirement. In this situation, an optimal learning rate is obviously desirable. Thus we expect to find a tradeoff between transient and steady state response, or more preferably, an adaptive learning rate, which varies according to the nature of training data set.

Numerous works have been carried out to address the issue. To name a few, Rupp and Sayed [80] [81] presented a robustness analysis for the training of RNN via small-gain theory. By recasting the RNN training algorithm into a feedback structure, they derived the robust conditions, which enable one to choose a maximum learning rate without breaking the bound of the ratio between weight estimation error and modeling error. In this manner both fast convergence and small steady-state error can be obtained. Liang and Gupta [47] studied the stability of dynamic backpropagation training algorithm by the Lyapunov method under a noiseless condition. By appending an extra term to augment the error gradient, an additional incremental term was introduced in the updating rule. Furthermore according to a set of Gersgorin's theorem-based stability conditions, a novel dynamic backpropagation algorithm with a constrained learning rate was introduced to improve the convergence speed of training. Atiya and Parlos [11] used a generalized gradient descent method to get the the value of error gradient, which unified the five category training algorithms in literature. The new algorithm was improved on basis of the idea that interchanging the roles of the network states and weight matrices. In another word, the weight variation was estimated according to the variation in network states vector, which was actually a perturbation method. Further, both online and off-line version of the method were proposed in that work. For a further review in gradient calculation please refer to [73].

Overall speaking, the stability theory of nonlinear system has followed two main trends: input-output stability and the stability in the sense of Lyapunov. There are

extensive studies on the local and global stability conditions of RNN system using the Lyapunov function approach, while input-output approaches have experienced an increasing development, particularly in the field of control systems. As a matter of fact, there exists a very close relationship between the kinds of stability results that one can get using this two approaches, just as that between the input-output representation and state representation of a given system. While the two approaches are related, they do not give identical results. In analyzing a system, it is advantageous to have as many answers as we can. It would be good to have both the approaches, each yielding its own set of insights and information. Next, we present a detail survey on works of robust stability analysis of RNN classified by this two approaches.

1.4.1 Lyapunov Approach

The concept of Lyapunov stability plays an important role in modern control theory. It is concerned with the trajectories of a system when the initial state is near an equilibrium point. The three basic definitions of the Lyapunov theory are stability, asymptotic stability, and global asymptotic stability. Roughly speaking, stability corresponds to the system trajectories depending continuously on the initial state; asymptotic stability corresponds to trajectories that start sufficiently close to an equilibrium point actually converging to equilibrium state as time sample $k \rightarrow \infty$. The Lyapunov stability has been studied since the early period of research on RNN. Hopfield and Tank studied stability of dynamic networks and showed their practical applicability to optimization approaches [29]– [31]. Cohen and Grossberg gave a more rigorous results on local and global stability on networks using various nonlinear analysis approaches [19]. Matsuoka generalized the results of [28] and [39] and improved the criteria using a new Lyapunov function, so that the absolute stability condition could be easily checked using only the synaptic connection weights of the

network [63]. Forti et al obtained the necessary and sufficient condition for absolute exponential stability of Hopfield neural networks with symmetric connection weight matrix [21]. The results can be stated as: the weight matrix must be negative semi-definite. Laing et al addressed global absolute stability issue for a general class of discrete-time RNN by using Ostrowski's theorem [48]. A series of inequalities were derived as sufficient conditions for absolute stability. Later, Arik and Tavsanoglu derived a more strict condition on weight matrix for global absolute stability of equilibrium point for RNN [8] [9]. The results state that if the activation function are bounded and positive, and if weight matrix is additively diagonal stable, then system is absolutely stable. Recently, Liang et al extended the results to the absolute exponential stability [32], [49]– [51]. The additively diagonally stable connection weight matrices were verified to be able to guarantee the absolute exponential stability of the RNN with locally Lipschitz continuous and monotone non-decreasing activation function. Chen and Amari proposed a unified approach to analyze the stability of asymmetric Hopfield networks [16]. Different from conventional methods that depend on existing theories, authors established some elementary lemmas to avoid sacrificing accuracy in deriving various sufficient conditions for local and global stability. Barabanov and Prokhorov presented a method to analyze the global Lyapunov stability of a most generalized RNN, to the best our knowledge [12]. A transformation method was carried out to convert the state equation of RNN into a form suitable for stability analysis.

1.4.2 Input-Output Systematic Approach

In contrast, input-output representation is different from state representation of looking at the same system – this type of representation is used because it gives a different kind of insight into how the system works. Although stability theories of this kind are fully developed, but the stability analysis for RNN from the input-

output aspect is not so well established as the stability in the sense of Lyapunov. In the beginning, Bode laid fundamental works on the robustness of feedback system [14], which showed that the effects of vanishingly small perturbations in the gains of a single-loop linear feedback system were directly related to the return ratio (or loop gain) and return difference (one minus the return ratio) of a feedback system. Then Zames presented a conic relation stability theory closely related to issue of feedback system robustness [107] [108]. Different from traditional Lyapunov method, this theory utilizes functional analysis, which make useful assessments of qualitative behavior with coarse information about a system. Based on that, Safanov gave a generalized conic sector stability theory on a topological separation [82]. The key idea was to separate the multivariable system into two parts, corresponding to the two elements of feedback structure. In fact, Safanov's theory comprises both Lyapunov theory and Zames theory. Vidyasagar studied the relation between input-output stability and stability in the sense of Lyapunov [99]. A concept so-called "small-gain L_p -stability" is introduced to reveal the conditions for a point to be exponentially stable equilibrium when the system is small-gain L_p stable. There are many other works extending the above results. Ortega et al applied the conic sector theory to analyze the robustness of normalized signals in least square parameter adaptation algorithm [72]. Cluett et al simplified the conic sector theory in mathematics [18]. Song et al further the results to multivariable case in self-tuning control [83]– [88]. For RNN, there are also several investigations on input-output stability, e.g., Steil and Ritter analyzed the RNN stability via circle criterion [92] [93]. Canete et al used sector theory to analyze the static neural network under gradient-type learning algorithm [15]. Chu pushed incremental gain boundary to a more strict conditions for discrete the RNN [17].

Chapter 2

Input-Output Stability Theory of Nonlinear Systems

A large variety of mathematical tools and methods have been used in the analysis of RNN, including such as statistics, group theory, probability theory, linear algebra, Lyapunov functions, dynamic system theory, combinatorics, approximation theory. Because of the cross disciplinary background of RNN, works in different fields may use different mathematical notations in the publications. For instance, there are many kinds of definitions for stability in the control area in the state of arts due to the large amount of publications on the topic. Considering this problem, it is necessary for us present a systematic description on the mathematics and notations that would be employed in later chapters. These basic contents will be introduced in this chapter. In fact, most of them can be found in all kinds of fundamental textbooks of graduate courses. We specially name a few [26] [40] [91] [100] [101]. To avoid over lengthy of the context, proofs of lemmas are not given because they are out of the scope of this thesis. Proofs of theorems in Section 2.3 are outlined, since they are critical to the derivation of theoretical part of the thesis. Next, we start with the introduction of preliminary knowledge such as normed linear spaces,

Banach spaces, inner products, Hilbert spaces, L_p -spaces, the frequency domain transforms and some facts from matrix theory.

2.1 Mathematical Preliminaries

2.1.1 Normed Spaces and Inner Products

Definition 2.1.1. (*Linear Vector Space*) The set L is called a linear vector space over a field K (typically real or complex field) when

i) the addition $+$ is defined on L and with this operation L is an abelian group, that is whenever $x, y, z \in L$

$$(a) \ x + y = y + x;$$

$$(b) \ (x + y) + z = x + (y + z);$$

$$(c) \ \text{There exists an identity } 0 \in L \text{ such that } 0 + x = x + 0 = x;$$

$$(d) \ \text{There exists an } x' \in L \text{ such that } x + x' = 0;$$

ii) a scalar multiplication $\times$ is defined which assigns any pair $(\alpha, x) \in K \times L$ to $\alpha x \in L$; furthermore the following properties are satisfied for all $\alpha, \beta \in K$ and $x, y \in L$

$$(e) \ \alpha(x + y) = \alpha x + \alpha y$$

$$(f) \ (\alpha + \beta)x = \alpha x + \beta x$$

$$(g) \ (1 \cdot x = x) \text{ where } 1 \text{ denotes the unit element of the multiplication in } K.$$

If we omit the definition it is supposed to be the field of real numbers.

Definition 2.1.2. (*Normed Linear Space*) A normed linear space $(L, \|\cdot\|)$ is a linear

vector space L together with a function $\|\cdot\|: L \mapsto R^+$ which satisfies the following axioms

$$i) \|x\| \geq 0, \quad \|x\| = 0 \Leftrightarrow x = 0, \forall x \in L$$

$$ii) \|\alpha x\| = |\alpha| \|x\|, \quad \forall \alpha \in R, x \in L$$

$$iii) \|x + y\| \leq \|x\| + \|y\|, \quad \forall x, y \in L$$

And the function $\|\cdot\|$ is called a norm on L .

The definition of norm is actually a generalization of the length of a multi-dimensional vector. The expression $\|x - y\|$ measures a generalized distance whenever x, y are from some normed vector space. Using this concept we define convergence for any given sequence $\{x_i\}_{i=1}^{\infty}$ of elements in a normed vector space.

Definition 2.1.3. (Convergence) Let $\{x_i\}_{i=1}^{\infty}$ be a sequence of elements in a normed space L . Then the sequence is convergent to $x_0 \in L$ if for all $\varepsilon > 0$, there exists an $N(\varepsilon)$ such that

$$\|x_i - x_0\| < \varepsilon, \quad \forall i > N(\varepsilon) \quad (2.1)$$

Definition 2.1.4. If for a sequence there exists an $N(\varepsilon)$ for all $\varepsilon > 0$ such that $\|x_{i_1} - x_{i_2}\| < \varepsilon$ for all $i_1, i_2 > N(\varepsilon)$, then the sequence is called Cauchy sequence.

A Cauchy sequence is basically a sequence that supposed to be convergent, but may not be so when its anticipated limit is missing from the underlying space. This leads to the definition of a special type space, namely, Banach Space.

Definition 2.1.5. (Banach Space) A normed linear space $(L, \|\cdot\|)$ is complete if every Cauchy sequence converges to an element of L . Such a complete space is also called a Banach Space.

Next, we give the definition of inner products, which can be understood as a measure of the generalized angle between arbitrary two elements of the space. Then based

upon the concept of complete space and inner products, a more structured space, so-called Hilbert space, is introduced.

Definition 2.1.6. (*Inner Product*) Let L be a linear vector space. If a function $\langle \cdot, \cdot \rangle : L \times L \mapsto R$ which satisfies the axioms

$$i) \langle x, y \rangle = \langle y, x \rangle$$

$$ii) \langle x, y + z \rangle = \langle x, y \rangle + \langle x, z \rangle$$

$$iii) \langle x, \alpha y \rangle = \alpha \langle x, y \rangle$$

$$iv) \langle x, x \rangle \geq 0, \quad \langle x, x \rangle = 0 \Leftrightarrow x = 0$$

for all $x, y, z \in L, \alpha \in R$, then the function is called an inner product on L and $(L, \langle \cdot, \cdot \rangle)$ is called an inner product space.

Lemma 2.1.1. (*Schwartz' Inequality*) Let L be an inner product space and define a function $\|\cdot\| : L \times L \mapsto R^+$ as $\|x\| = \langle x, x \rangle^{1/2}$. Then $\|\cdot\|$ is a norm on L and it holds

$$|\langle x, y \rangle| \leq \|x\| \cdot \|y\| \quad (2.2)$$

Definition 2.1.7. (*Hilbert Space*) An inner product space L which is complete with the norm $\|\cdot\| : L \times L \mapsto R^+$ defined by (2.3) is called a Hilbert space.

$$\|v\| = \langle v, v \rangle^{1/2} \quad (2.3)$$

An inner product space can be made a normed space in a natural way. Hilbert space leads to the most structured spaces we will be concerned with. Working with Hilbert spaces is relatively more convenient because convergence can be proved even if there is no known candidate for the limit. Moreover, in Hilbert spaces it is sufficient to prove that the elements of a sequence become arbitrarily close to each other. Now question arises, how a space $L_n[a, b]$, which would be candidates for the input and

output spaces, can be enlarged such that it is complete or even a Hilbert space. To answer the question, we need to introduce a general class of measurable functions and the corresponding concept of Lebesgue integration. This results in a redefinition of the scalar product and provides the basis for the definition of the function space of square (Lebesgue) integrable L_2 -functions.

2.1.2 Lebesgue Function Spaces

In this section, we proceed by introducing the Lebesgue function spaces, which underlay the entire input-output system theory. We give a quick glimpse at Lebesgue theory because only definition of function spaces which are complete with respect to the appropriate scalar products and norms are necessary, and other more sophisticated concepts of Lebesgue and measure theory are out of the scope of this thesis thus we omit. After that we introduce the truncation operator and extended spaces, concepts which are necessary to understand the problems of stability.

Definition 2.1.8. (L_p -space) *The set $L_p[a, b]$, $p \in [1, \infty)$ of functions $f : R \mapsto R$ is called a Lebesgue function space if f satisfies $\int_a^b |f(\xi)|^p d\xi < \infty$*

We denote by $L_\infty[a, b]$ the set of all functions bounded on $[a, b]$. By default, if we omit the interval $[a, b]$ then it means $L_p[0, \infty)$. Now, we define the norm on L_p functions spaces, which for $p = 2$ is consistent with the definition of the norm on $L_n[a, b]$ induced by the common scalar product.

Definition 2.1.9. (L_p -Norm) *The function $\|\cdot\|_p : L_p \mapsto R^+, p \in [1, \infty)$ defined by*

$$\|f(\cdot)\|_p = \left(\int_0^\infty |f(\xi)|^p d\xi \right)^{1/p} \quad (2.4)$$

is a norm on L_p -space.

In particular, the function defined by $\|f(\cdot)\|_\infty = \sup_{\xi \in [0, \infty)} |f(\xi)|$ is a norm on L_∞ . Notice here in definition 2.1.8 and 2.1.9, integral is referred to Lebesgue integral. This has a great advantage that we can investigate also systems or signals in discrete time domain. For example, let $x = \{x(1), x(2), \dots\}$, then the L_2 -norm of $\{x(k)\}$ is defined as $\|x\|_2 = \left\{ \sum_{k=1}^{\infty} x^2(k) \right\}^{1/2}$, and the L_∞ -norm is defined as $\|x\|_\infty = \sup |x(k)|$, $k \geq 1$.

Lemma 2.1.2. *The L_p -spaces are complete normed linear spaces for all $p \in [1, \infty)$ with respect to the norm defined in (2.4).*

The lemma implies that any function in L_p can be approximated arbitrarily close by continuous functions in L_p and every limit of a sequence of continuous functions in L_p belongs to L_p . The case $p = 2$ is special:

Lemma 2.1.3. *The space $(L_2, \|\cdot\|_2)$ together with the inner product*

$$\langle f(\cdot), g(\cdot) \rangle_2 = \int_0^\infty f(\xi)g(\xi)d\xi \quad (2.5)$$

is a Hilbert space.

Definition 2.1.10. (L_p^n -norm) *Consider a vector $\vec{f} = (f_1, \dots, f_n)^T$, $f_i \in L_p$. The set of all such $\vec{f}$ is denoted by L_p^n and the function $\|\vec{f}\|_p = \left(\sum_{i=1}^n \|f_i\|_p^2 \right)^{1/2}$ defines a norm on L_p^n .*

Lemma 2.1.4. *The space $(L_2^n, \|\cdot\|_2)$ together with the L_2^n norm is a Hilbert space.*

Now we have completed the definitions of the relevant $L_p^n(L_2^n)$ space for investigation of multi-dimensional input-output dynamic systems. We still need another two important concepts: truncated functions and extended spaces.

Definition 2.1.11. (*Truncated Functions*) *Let $f : R^+ \mapsto R$ be a function. Then for each $T \in Z^+$ (the set of positive integers) the truncation $\{f\}_T : R^+ \mapsto R$ of f is*

defined by

$$\{f\}_T = \begin{cases} f(\xi) & \xi \leq T \\ 0 & \xi > T \end{cases} \quad (2.6)$$

Definition 2.1.12. (*Extended L_p -spaces*) The set of all functions f with $\{f\}_T \in L_p$, $\forall T \in \mathbb{Z}^+$ is denoted by L_{pe} and called the extended L_p -space.

Clearly $L_p \subset L_{pe}$ but on the contrary, a lot of unbounded functions belong to L_{pe} though not necessarily to L_p . The extended spaces are useful if a function is not a priori known whether it is in L_p or not. For instance we can describe unstable subsystems in control loops by transfer functions which are $L_{pe} \rightarrow L_{pe}$ but not necessarily $L_p \rightarrow L_p$. For this reason the general input-output stability definitions are based on relations on extended spaces.

2.1.3 Frequency Domain Transforms

Definition 2.1.13. (*Fourier Transform*) Let $f \in L_1$. Then the function $F(j\omega) = \int_{-\infty}^{+\infty} f(t)e^{-j\omega t}dt$ is well defined and called the Fourier transform of f , where j denotes the imaginary unit $j = \sqrt{-1}$.

We use the variable $j\omega$ as argument for the transform F , which provides a simple means to express its complex conjugate by changing the argument from $j\omega$ to $-j\omega$: $F^*(j\omega) = F(-j\omega)$. The Fourier transform has the following properties:

- i) $\omega \mapsto F(j\omega)$ is uniformly continuous in $\mathbb{R}$.
- ii) $|F(j\omega)| \rightarrow 0$, as $\omega \rightarrow \infty$
- iii) $F \in L_\infty$
- iv) $f(t) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} F(j\omega)e^{j\omega t}d\omega$

The inverse Fourier transform defined by (iv) is once more defined only almost

everywhere because Lebesgue integrals are involved. The Fourier transform can be extended to functions in L_2 where it is defined as the limit of the Fourier transforms of functions $f_n \in L_1 \cap L_2$ and belongs to L_2 . We omit the details and simply assume that the Fourier transform is defined on L_1 and L_2 and maps L_2 onto L_2 .

Definition 2.1.14. (*Laplace Transform*) Let $f : \mathbb{R} \mapsto \mathbb{R}$ be locally integrable and $f(\xi) = 0$ for all $\xi < 0$. Then for $s = \sigma + j\omega \in \mathbb{C}$, the function $F(s) = \int_0^\infty f(t)e^{-st}dt$ is called Laplace transform of f . Its domain is the domain of convergence of the integral expression.

2.1.4 Elements of Matrix Theory

In this section we deal with matrices in the complex space $\mathbb{C}^{n \times n}$ but keep in mind that the real space $\mathbb{R}^{n \times n}$ is embedded as a subspace in $\mathbb{C}^{n \times n}$. Therefore all the concepts introduced can be given for $\mathbb{R}^{n \times n}$ with the obvious modifications. We start noting that the set $\mathbb{C}^{n \times n}$ of all $n \times n$ matrices with complex elements is a linear space if addition and scalar multiplication are defined point-wise. With the usual matrix vector multiplication any matrix $A \in \mathbb{C}^{n \times n}$ maps vectors from $\mathbb{C}^n$ into $\mathbb{C}^n$ and corresponds to a linear map. We denote the complex conjugated and transpose operation of a matrix by $A^* = \bar{A}^T$. Further we denote by $\lambda_i(A)$ the i th eigenvalue of A . Based on the definition of vector norms in $\mathbb{C}^n$ we have the following definition of induced matrix norms.

Definition 2.1.15. (*Induced Matrix Norm*) Let $\|\cdot\|_q$ a given vector norm on $\mathbb{C}^n$. Then the induced matrix norm with respect to $\|\cdot\|_q$ for any $A \in \mathbb{C}^{n \times n}, x \in \mathbb{C}^n$ is

$$\|A\|_q = \sup_{\|x\|_q=1} \|Ax\|_q = \sup \frac{\|Ax\|_q}{\|x\|_q}$$

For $q = 1, 2, \infty$ and the vector norms defined by $\|x\|_1 = \sum_i |x_i|$, $\|x\|_2 = \sqrt{\langle x, x \rangle}$ and $\|x\|_\infty = \max_i |x_i|$, the induced matrix norms can be derived as

$$\|A\|_1 = \max_j \sum_i |a_{ij}| \quad (\text{column sums})$$

$$\|A\|_2 = \max_i \lambda_i(A^*A)^{1/2} \quad (\text{spectral norm})$$

$$\|A\|_\infty = \max_i \sum_j |a_{ij}| \quad (\text{row sums})$$

where the roots of the eigenvalues of A^*A are denoted by $\sigma_i(A) = \lambda_i(A^*A)^{1/2}$ and named as singular values. Therefore the spectral norm $\|A\|_2$ equals the largest singular value of A . The spectral norm, induced by the Euclidian vector norm, has the special property that it is the smallest norm among all induced norms. It provides the best estimation of the modulus of the eigenvalues of A by a matrix norm, in general it holds for all eigenvalues $\lambda_i(A)$ that

$$\sigma_{\min}(A) \leq |\lambda_i(A)| \leq \sigma_{\max}(A) = \|A\|_2, \quad \forall i$$

This implies that the norm of A is an upper bound for the real parts $\operatorname{Re}(\lambda_i(A)) \leq \|A\|$. Any induced norm satisfies the inequality $\|A\| \geq \rho(A)$, where $\rho(A)$ is spectral radius of A , $\rho(A) = \lim_{r \rightarrow \infty} \|A^r\|^{1/r}$.

In later exposition, we are using $\|\cdot\|_2$ as the default norm for both vectors and matrices, and denote by $\|\cdot\|$ with the subscript 2 ignored so as to differentiate from the L_p function norms defined in 2.1.9 .

Lemma 2.1.5. *Let $\|\cdot\|_q$ an induced matrix norm. Then $\forall A, B \in \mathbb{C}^{n \times n}, x \in \mathbb{C}^n$ we have $\|AB\|_q \leq \|A\|_q \|B\|_q$ and $\|Ax\|_q \leq \|A\|_q \|x\|_q$. More generally, all norms with this property are called compatible, multiplicative, or sub-multiplicative matrix norms.*

In addition to the induced norms, another important method to weight a matrix is to treat it as an $m \times n$ “vector”, such that we can use one of the familiar vector norms. These kinds of norms are classified as entrywise norms. For example

$$\|A\|_{tr} = \operatorname{Trace}\{\sqrt{A^T \cdot A}\} = \sum_{i=1}^{\min\{m,n\}} \sigma_i \quad (\text{trace norm})$$

$$\|A\|_F^2 = \sum_{i=1}^m \sum_{j=1}^n |a_{ij}|^2 = \text{Trace}\{A^T \cdot A\} = \sum_{i=1}^{\min\{m,n\}} \sigma_i^2 \quad (\text{Frobenius norm})$$

$$\|A\|_{\max} = \max\{|a_{ij}|\} \quad (\text{max norm})$$

where the function $\text{Trace}\{\bullet\}$ is the sum of the elements on the main diagonal of the variable matrix, e.g.

$$\text{Trace}\{A\} = a_{11} + a_{22} + \cdots + a_{nn} = \sum_{i=1}^n a_{ii} \quad (2.7)$$

Definition 2.1.16. (*Definiteness*) A hermitian matrix $A \in C^{n \times n}$ is positive (non-negative) definite if for all $x \in C^n, x \neq 0$, $x^T A x > 0 (\geq 0)$. Similarly, the matrix A is said to be negative (non-positive) definite if $-A$ is positive (nonnegative) definite.

Definition 2.1.17. (*Real Part of a Matrix*) The real (hermitian) part of a matrix $A \in C^{n \times n}$ is given by $\text{Re}\{A\} = \text{He}(A) = \frac{1}{2}(A^* + A)$

In the following we write $A > 0$ for “ A is positive definite” and give definiteness results only for positivity if they can be extended to the other cases by simple sign changes. We will use only the symbol $\text{Re}\{\cdot\}$ to indicate the real or hermitian part.

Lemma 2.1.6. Let $A \in C^{n \times n}$. Then $\forall x \in R : \langle x, Ax \rangle \geq 0 \Leftrightarrow \text{Re}\{A\} > 0$

Definition 2.1.18. (*Normality*) A matrix $A \in C^{n \times n}$ is normal if and only if it has a complete set of orthogonal eigenvectors spanning the whole C^n . Then it holds $A^* A = A A^*$.

Equivalently, the definition can be stated in an alternative formulation: A is normal, if there exists a unitary matrix $U \in C^{n \times n}$ such that $A = U^{-1} \text{diag}\{\lambda_i(A)\} U$.

2.2 Input-Output Stability

In the perspective of input-output theory, a system acts as an operator mapping one element of the input space to an element of the output space. To analyze stability issue of this transformation in mathematical terms, we may use norms and inner products in the corresponding function spaces. The traditional approach to stability analysis involves Lyapunov's method. Here it is proposed to take a different course. We stress the relation between input-output behavior and stability. An input-output system is one in which a function of time, called the output, is required to track another function of time, called the input. More generally, the output might be required to track some function of the input. In order to behave properly an input-output system must usually have two properties: i) Bounded inputs must produce bounded outputs, i.e., the system must be nonexplosive. ii) Outputs must not be critical sensitive to small changes in inputs, such as those caused by noise.

Definition 2.2.1. (L_p -Stability) Let H be a map $H : L_p^n \mapsto L_p^n$. Then H is called finite gain L_p -stable if there exist finite constants γ and ζ such that for all pairs $(u(t), y(t))$, the following two inequalities hold

$$i) u(t) \in L_p^n \Rightarrow y(t) \in L_p^n$$

$$ii) \|y(t)\|_p \leq \gamma \|u(t)\|_p + \zeta$$

In general finite gain L_p -stability (for short, L_p -stability) is defined for arbitrary operators mapping certain L_p^n -function into another. In Definition 2.2.1, input-output stability is strongly related to the boundedness properties of the input and output functions. Most important of all, their L_p^n -norms must be finite. As the L_p^n -norm is defined by integration over time, L_p -stability is actually a property of the whole time-development of the entire system. Because in the L_2 -space we employ Lebesgue integration, thus the values of $u(t), y(t)$ at a single time-instant t do not have any influence on the system's input-output stability and could even be

arbitrarily changed. Next, we define the L_p -gain of an operator H as the infimum over all γ such that conditions in definition 2.2.1 is fulfilled for some ζ .

Definition 2.2.2. (*L_p -Gain, Operator Norm*) The number

$$g_p(H) = \inf \{ \gamma \mid \|y(t)\|_p \leq \gamma \|u(t)\|_p + \zeta \} \quad (2.8)$$

is called the L_p gain of H .

Particularly, the L_2 -gain defines at the same time the operator norm induced by the L_2 norm for an operator $H : L_2 \mapsto L_2$, i.e.,

$$g_2(H) = \sup_u \frac{\|H(u)\|_2}{\|u\|_2} = \|H\|_2 \quad (2.9)$$

2.3 Conic Conditions and Input-Output Stability Theorems

In this section, we list a series of the stability criteria which are developed from nonlinear system theorem. Particularly, we are interested in those results of discrete-time systems, such that they can be directly used in theoretical analysis of later chapters.

Generally a system can be defined as acting on inputs u to give outputs y in some arbitrary way, expressed by an operator $H : L_{2e}^n \mapsto L_{2e}^n$. It is given either in the time domain, $y(k) = H(u(k))$ or in the complex domain $y(z) = H(u(z))$, where the argument z indicates that we use the respective z transforms. Specifically, many of these operators H can be transformed into a generalized closed loop structure which consists of a linear feedforward operator H_1 and nonlinear feedback H_2 . In time domain, the linear path is always a convolution operator $y(k) = H_1(e(k))$ and

as the z-transform changes convolution to multiplication we obtain in the complex domain $y(z) = H_1(z)e(z)$. On the other hand, the operator H_2 is restricted to be static nonlinear. In this way, the system H is described in a pure input-output setting, and there is no inherent need to specify internal states of the system. Now consider the following closed loop system (Figure 2.1)

$$\begin{cases} e_0(k) = \varepsilon(k) - \phi(k) \\ e(k) = H_1 e_0(k) \\ \phi(k) = H_2 e(k) \end{cases} \quad (2.10)$$

where operators $H_1, H_2 : L_{2e} \mapsto L_{2e}$, discrete time signals $e_0(k), e(k), \phi(k) \in L_{2e}$ and disturbance terms $\varepsilon(k) \in L_2$.

Figure 2.1: Closed loop feedback system

The boundedness small gain theorem is a very general theorem which gives sufficient conditions for bounded input-output stability. The important questions of existence, uniqueness and continuity of solution are divorced from the stability question and are usually assumed a priori.

Theorem 2.3.1. (*Small Gain Theorem*) Consider the feedback system of equations (2.10). If $g(H_1)g(H_2) < 1$, then the closed loop relations $e_0(k)$ and $e(k)$ are bounded.

Proof: By definition

$$\begin{cases} \|\phi(k)\| \leq g(H_2) \|e(k)\| \\ \|e(k)\| \leq g(H_1) \|e_0(k)\| \end{cases} \quad (2.11)$$

Then by Schwartz inequality

$$\|e_0(k)\| \leq \|\varepsilon(k)\| + \|\phi(k)\| \quad (2.12)$$

Then we can derive

$$\begin{aligned} \|e_0(k)\| &\leq \|\varepsilon(k)\| + \|\phi(k)\| \\ &\leq \|\varepsilon(k)\| + g(H_2) \|e(k)\| \\ &\leq \|\varepsilon(k)\| + g(H_2)g(H_1) \|e_0(k)\| \end{aligned} \quad (2.13)$$

Because $g(H_1)g(H_2) < 1$, thus $1 - g(H_1)g(H_2) > 0$. As a consequence, we can obtain

$$\begin{cases} \|e_0(k)\| \leq \frac{1}{1-g(H_1)g(H_2)} \|\varepsilon(k)\| \\ \|e(k)\| \leq \frac{g(H_1)}{1-g(H_1)g(H_2)} \|\varepsilon(k)\| \end{cases} \quad (2.14)$$

The usefulness of Theorem 2.3.1 is limited by the condition that the open-loop gain product be less than one, a condition seldom met in practice. However, a reduced gain product can often be obtained by transforming the feedback equations. For example, if cI is added to and subtracted from H_2 , then $e_0(k)$ remains unaffected. However, H_1 will be changed into a new relation H'_1 , as in effect $-cI$ appears in feedback around H_1 . Under this transformation, it will appear that a sufficient condition is that the input-output relations of the open loop elements be confined to certain “conic” regions. Next, we will introduce the conic sector stability theorem including some properties of conic and sector relations. The approach is actually more natural and convenient comparing to other counterparts to be used in the analysis of RNN training algorithms.

Definition 2.3.1. *A relation H is interior conic if there are real constants $r \geq 0$*

and c for which the following inequality is satisfied

$$\|H(x) - cx\| \leq r\|x\|, \quad \forall x \in R \quad (2.15)$$

And H is called exterior conic if the inequality sign in (2.15) is reversed. H is conic if it is either exterior conic or interior conic. The constant c will be called the center parameter of H , and r will be called the radius parameter.

Definition 2.3.2. A conic relation H is said to be inside the sector $\{a, b\}$, if $a \leq b$ and if the following inequality holds

$$\langle H(x) - ax, H(x) - bx \rangle \leq 0, \quad \forall x \in R \quad (2.16)$$

H is outside the sector $\{a, b\}$ if $a \geq b$ and if (2.16) holds with the inequality sign reversed.

Definition 2.3.3. A conic relation H is said to be strictly inside the sector $\{a, b\}$, if $a \leq b$ and if the following inequality holds

$$\langle H(x) - ax, H(x) - bx \rangle < 0, \quad x \in R \text{ and } x \neq 0 \quad (2.17)$$

H is strictly outside the sector $\{a, b\}$ if $a \geq b$ and if (2.17) holds with the inequality sign reversed.

Lemma 2.3.1. Assume H_1 and H_2 are conic relations, that H_1 is inside the sector $\{a_1, b_1\}$ with $b_1 > 0$, that H_2 is inside $\{a_2, b_2\}$ with $b_2 > 0$. And $k \geq 0$ is a constant, then we have

i) kH_1 is inside $\{ka_1, kb_1\}$. $-H_1$ is inside $\{-b_1, -a_1\}$.

ii) $H_1 + H_2$ is inside $\{a_1 + a_2, b_1 + b_2\}$. (sum rule)

iii) If $a_1 > 0$, then H_1^{-1} is inside $\{1/b, 1/a\}$; if $a_1 < 0$, then H_1^{-1} is outside

$\{1/a, 1/b\}$; if $a_1 = 0$, then $H_1^{-1} - \frac{1}{b}I > 0$. (inverse rule)

iv) Above three properties remains valid with the terms “inside” and “outside” interchanged throughout.

v) $g(H_1) \leq \max\{|a_1|, |b_1|\}$; specially if H_1 is inside $\{-r, r\}$ then $g(H) \leq r$.

Proof: Properties i) and ii) can be immediately drawn from the definition of sector.

For iii), assume $y = H_1^{-1}(x)$, i.e., $x = H_1(y)$. When $a_1 \neq 0$, we can derive

$$\begin{aligned} \langle H_1^{-1}(x) - \frac{1}{b_1}x, H_1^{-1}(x) - \frac{1}{a_1}x \rangle &= \langle y - \frac{1}{b_1}H_1(y), y - \frac{1}{a_1}H_1(y) \rangle \\ &= \frac{1}{a_1 b_1} \langle H_1(y) - a_1 y, H_1(y) - b_1 y \rangle \end{aligned} \quad (2.18)$$

Since H_1 is inside $\{a_1, b_1\}$ and $b_1 > 0$, the sign of the last expression is opposite to that of a_1 . Thus the inverse rule is obtained. When $a_1 = 0$, the property is implied by the inequality

$$\langle x, H_1^{-1}(x) - \frac{1}{b_1}x \rangle = \frac{1}{b_1} \langle H_1(y), b_1 y - H_1(y) \rangle \geq 0 \quad (2.19)$$

Property iv) Simply reverse all the inequality signs.

Property v)

$$\begin{aligned} \|H_1(x)\| &\leq \|H_1(x) - \frac{1}{2}(a+b)x\| + \|\frac{1}{2}(a+b)x\| \quad (\text{triangle inequality}) \\ &\leq \frac{1}{2}|b-a| \cdot \|x\| + \frac{1}{2}|b+a| \cdot \|x\| \\ &= \max\{|a|, |b|\} \cdot \|x\| \end{aligned} \quad (2.20)$$

Thus $g(H_1) \leq \max\{|a_1|, |b_1|\}$.

Theorem 2.3.2. (Conic Sector Theorem) *The feedback system of (2.10) is finite gain stable if there exist $0 < a \leq b$ such that*

(i) H_1^{-1} lies outside the sector $(-b, -a)$;

(ii) H_2 lies strictly inside the sector (a, b) .

Proof: To evaluate the stability of entire system, it is necessary to transform the loop (H_1, H_2) into another form (H'_1, H'_2) by scaling and addition of linear auxiliary operators, which is equivalent with respect to stability in the sense that any solution of latter is also a solution of former. That means that boundedness of (H_1, H_2) will imply boundedness of (H'_1, H'_2) . Hence we may derive the stability conditions of (H_1, H_2) by applying small gain theorem to (H'_1, H'_2) instead. For this purpose, define the new elements

$$\begin{cases} e'_0(k) = e_0(k) - c \cdot e(k) \\ \varphi(k) = \phi(k) + c \cdot e(k) \end{cases} \quad (2.21)$$

where $c = -(b + a)/2$ and $r = (b - a)/2$. Then we construct the following closed loop system.

$$\begin{cases} e'_0(k) = \varepsilon(k) - \varphi(k) \\ e(k) = H'_1(e'_0(k)) \\ \varphi(k) = H'_2(e(k)) \end{cases} \quad (2.22)$$

in which

$$\begin{cases} H'_1 = (H_1^{-1} - cI)^{-1} \\ H'_2 = H_2 + cI \end{cases} \quad (2.23)$$

The new structure is displayed in Figure 2.2

Since H_2 is inside $\{a, b\}$, then H'_2 should be inside $\{-\frac{b-a}{2}, \frac{b-a}{2}\}$, or equivalently, $\{-r, r\}$. By property (v) in lemma 2.3.1, we have $g(H'_2) \leq r$. Similarly, because H_1^{-1} is outside $\{-b, -a\}$, which means $H_1^{-1} + cI$ is outside $\{-r, r\}$. Thus, by the inverse rule, $H'_1 = (H_1^{-1} + cI)^{-1}$ is inside $\{-\frac{1}{r}, \frac{1}{r}\}$, i.e., $g(H'_1) \leq \frac{1}{r}$. To this end, we can obtain

$$g(H'_1)g(H'_2) \leq 1 \quad (2.24)$$

Figure 2.2: Loop transformation

By the small gain theorem, the transformed system (2.22) is stable. Moreover, the solution of (2.10) is contained in (2.22) such that we conclude the boundedness of original system.

The following theorems are necessary extensions to the conic sector stability theory for discrete control systems and useful in the analysis of many adaptive control systems [18].

Theorem 2.3.3. (*Cluett's Law-1*) *If the following two conditions hold*

- i) $H_1 : e_0(k) \rightarrow e(k)$ satisfies $\sum_{k=0}^N [e^2(k) + \alpha e_0(k)e(k) + \beta e_0^2(k)] \geq -\gamma, \forall N \in \mathbb{Z}^+$
- ii) $H_2 : e(k) \rightarrow \phi(k)$ satisfies $\sum_{k=0}^N [\beta \phi^2(k) - \alpha \phi(k)e(k) + e^2(k)] \leq -\eta \{ \|\phi(k), e(k)\|_2^2 \}_N,$
 $\forall N \in \mathbb{Z}^+$

for some $\alpha, \beta \in \mathbb{R}$, which are independent of k and N , and $\gamma \geq 0, \eta > 0$, which are independent of N , then the closed loop feedback system of (2.10) is stable in the sense of $e(k), \phi(k) \in L_2$.

Proof: By the inequality i) and using $e_0(k) = \varepsilon(k) - \phi(k)$

$$\sum_{k=0}^N [\beta \phi^2(k) - \alpha \phi(k)e(k) + e^2(k)] + \sum_{k=0}^N [\alpha \varepsilon(k)e(k) - 2\beta \varepsilon(k)\phi(k) + \beta \varepsilon^2(k)] \geq -\gamma \quad (2.25)$$

Combining inequality ii) and equation (2.25)

$$-\eta \{ \|(\phi(k), e(k))\|_2^2 \}_N + \sum_{k=0}^N [\alpha \varepsilon(k)e(k) - 2\beta \varepsilon(k)\phi(k) + \beta \varepsilon^2(k)] \geq -\gamma \quad (2.26)$$

Using the Schwartz inequality

$$\begin{aligned} \eta \{ \|(\phi(k), e(k))\|_2^2 \}_N - |\alpha| \cdot \{ \|\varepsilon(k)\|_2 \}_N \cdot \{ \|e(k)\|_2 \}_N - 2|\beta| \cdot \{ \|\varepsilon(k)\|_2 \}_N \cdot \{ \|\phi(k)\|_2 \}_N \\ \leq \gamma + |\beta| \cdot \{ \|\varepsilon(k)\|_2^2 \}_N \end{aligned} \quad (2.27)$$

Assume $\{ \|(\phi(k), e(k))\|_2^2 \}_N \rightarrow \infty$ as $N \rightarrow \infty$, then from equation (2.27) we derive $\eta \leq 0$. This is a contradiction. Therefore $\{ \|(\phi(k), e(k))\|_2^2 \}_N$ is bounded for all $N \in Z^+$, i.e., $\phi(k), e(k) \in L_2$.

Lemma 2.3.2. (*Cluett's Law-2*) Consider the feedback system of (2.10), if

i) $H_1 : e_0(k) \rightarrow e(k)$ satisfies $\sum_{k=1}^N (e_0(k)e(k) + \sigma e_0(k)^2/2) \geq -\gamma, \forall N \in Z^+$

ii) $H_2 : e(k) \rightarrow \phi(k)$ satisfies $\sum_{k=1}^N (\sigma \phi(k)^2/2 - \phi(k)e(k)) \leq -\eta \{ \|\phi(k), e(k)\|_2^2 \}_N, \forall N \in Z^+$

for some $\gamma \geq 0, \eta > 0$, which are independent of N , and $\sigma \in (0, 1]$, which is independent of k and N , then the closed loop signals $e(k), \phi(k) \in L_2$.

Proof: The proof follows the approach taken for Theorem 2.3.3.

Chapter 3

Gradient Descent Training Algorithms

In this chapter, we turn to the issues of RNN training. The objective of RNN training can be stated as updating the weights of each layer recursively to minimize certain cost function $f(e)$, such that network output is best fitted into the given data. There are basically two modes of training an ordinary RNN: batch mode and sequential mode [27]. In the batch mode, the sensitivity of the network is computed for the entire training set before adjusting the weights. In the sequential mode, on the other hand, weight adjustments are made after the presentation of each pattern in the training set. Specifically, in an environment of time-varying signal statistics, gradient type algorithms are often used to reduce $f(e)$ by estimating the weight at each time instant. In this work, we will focus on the sequential type training and its realtime applications.

Generally, for realtime (sequential) gradient based training procedure, the transient response is always a major concern because of the tight timing requirement. In adaptive filters, it is well known that a small step size may result in excessive number of iterations to reach the optimum, i.e., a slow convergence speed. Conversely, a

large one may lead to big steady-state error, or even an unstable training procedure [42] [62]. Thus an effective tradeoff between transient and steady state response is desirable. The problem has been extensively studied for feedforward type ANN, e.g. Jagannathan proposed several novel training algorithms of ANN and derived the selection criterions of adaptive learning parameters to guarantee the control system performance via the Lyapunov approach [33] [34]. However, this topic is still open for RNN training algorithms. This is because weight adjustments of RNN may affect the entire state variables during the network evolution due to the inherent feedback and distributive parallel structure. Hence it is often difficult to obtain the error derivative for gradient type updating rules [58] [104] and in turn, difficult to analyze the underlying dynamics of training.

In this chapter, a robust training of RNN named Normalized Adaptive Recurrent Learning (NARL) is developed in the first place. Several modifications based upon conventional algorithms are proposed to improve the convergence speed and the robustness of the training. The strategies include normalization factors, adaptive learning rate and augmented residual error gradient. In order to provide a better understanding on the proposed algorithm, we start with the introduction of conventional gradient-based training algorithms of RNN, such as the Backpropagation Through Time (BPTT) and Real Time Recurrent Learning (RTRL) etc.

3.1 Review of Gradient Descent Training Algorithms

3.1.1 Backpropagation Through Time

In principle, the gradient type algorithms are based on calculation or approximation of the error gradients. A typical example of RNN training is BPTT. BPTT is

developed by Werbos [102] and Rumelhart et al [78] [79] as an extension of the standard backpropagation algorithm, which was originally introduced for multi-layer networks. It may be derived by unfolding the temporal operation of RNN into a layered feedforward network, the topology of which grows by one layer at every time step. To be specific, consider the model of external feedback RNN in equation (1.3), in the sequential training mode, if using the squared error $f(e) = e^2(k)/2$ as cost function, then the updating rule for output layer weight in BPTT is expressed by

$$\hat{V}(k+1) = \hat{V}(k) + \alpha e(k) \frac{\partial \hat{y}(k)}{\partial \hat{V}(k)} \quad (3.1)$$

$$\frac{\partial \hat{y}(k)}{\partial \hat{V}(k)} = \Phi(k)^T + \hat{V}(k) \cdot \frac{\partial \Phi(k)}{\partial \hat{y}(k-1)} \cdot \frac{\partial \hat{y}(k-1)}{\partial \hat{V}(k)} \quad (3.2)$$

where α is learning rate, $e(k)$ is instantaneous modeling error defined in (1.5). In (3.2), the first term is corresponding to the model output gradient with respect to the parameters, which also appears in static backpropagation. The second term, however, arises from the feedback component. If the model input $\hat{y}(k-1)$ was an external signal, e.g., the measured process output $y(k-1)$ as in the case for the series-parallel configuration, then no dependency on the model parameters would exist and this derivative would be equal to zero. For recurrent models, however, $\hat{y}(k-1)$ is the previous model output, which depends on network weights. In fact, for the higher order recurrent models and internal dynamic state space models, an expression like the second term in (3.2) always appears for each state. The evaluation of $\partial \hat{y}(k-1)/\partial \hat{V}(k)$ requires the derivative of the previous model output with respect to the current parameters. It can be calculated from

$$\hat{y}(k-1) = \hat{V}(k) \Phi(\hat{W}(k) \hat{x}(k-1)) \quad (3.3)$$

which corresponds to (1.3) shifted one step into the past. Note that (3.3) was written with the actual parameters $\hat{V}(k)$, which differs from the evaluation of the model output carried out one time instant before since that was based on the previous parameters $\hat{V}(k-1)$. With (3.3) the derivative (3.2) becomes

$$\frac{\partial \hat{y}(k-1)}{\partial \hat{V}(k)} = \Phi(\hat{W}(k)\hat{x}(k-1))^T + \hat{V}(k) \cdot \frac{\partial \Phi(\hat{W}(k)\hat{x}(k-1))}{\partial \hat{y}(k-2)} \cdot \frac{\partial \hat{y}(k-2)}{\partial \hat{V}(k)} \quad (3.4)$$

Again, the second term in (3.4) requires the derivative of the model output one time instant before. This procedure can be carried out until time $k=0$ with the initial value of RNN output $\hat{y}(0)$, which does not depend on the parameters, i.e.,

$$\frac{\partial \hat{y}(0)}{\partial \hat{V}(k)} = 0 \quad (3.5)$$

In fact, the idea of BPTT can be generalized to apply to any kind of recurrent model with any number of layers, just by unfolding the recurrent structure of RNN straightforward. For example, for hidden layer weight of (1.3), we can derive

$$\hat{W}_i(k+1) = \hat{W}_i(k) + \alpha e(k) \frac{\partial \hat{y}(k)}{\partial \hat{W}_i(k)} \quad (3.6)$$

$$\frac{\partial \hat{y}(k)}{\partial \hat{W}_i(k)} = \text{diag}\{\Phi(k)\} \hat{V}(k)^T \hat{x}(k)^T + \hat{W}_i(k) \cdot \frac{\partial \Phi(k)}{\partial \hat{y}(k-1)} \cdot \frac{\partial \hat{y}(k-1)}{\partial \hat{W}_i(k)} \quad (3.7)$$

$$\begin{aligned} \frac{\partial \hat{y}(k-1)}{\partial \hat{W}_i(k)} &= \text{diag}\{\Phi(\hat{W}(k)\hat{x}(k-1))\} \hat{V}(k)^T \hat{x}(k-1)^T \\ &\quad + \hat{W}_i(k) \cdot \frac{\partial \Phi(\hat{W}(k)\hat{x}(k-1))}{\partial \hat{y}(k-2)} \cdot \frac{\partial \hat{y}(k-2)}{\partial \hat{W}_i(k)} \end{aligned} \quad (3.8)$$

$$\begin{aligned} &\vdots \\ \frac{\partial \hat{y}(0)}{\partial \hat{W}_i(k)} &= 0 \end{aligned} \quad (3.9)$$

where $\hat{W}_i(k)$ is the i th row of hidden layer weight matrix, with $i = 1, 2, \dots, m$. The entire recursive substitution procedure described by (3.60)–(3.9) is illustrated in Figure 3.1.

Figure 3.1: Illustration of BPTT algorithm: unfolds a recurrent model back into the past until the initial input values at time $k = 0$ are reached.

We can see that the BPTT algorithm calculates the exact model derivatives by pursuing all k steps back into the past. This means that for the last training data sample, derivatives have to be calculated for each parameter. Thus, the gradient calculations with BPTT is exact for batch mode. It should be pointed out that because the whole procedure of unfolding must be iterated through entire training process, since the number of training data samples is generally quite large, the computational effort and memory requirement of BPTT are usually unacceptable in practice.

3.1.2 Truncated Backpropagation Through Time

In the applications with realtime training, if we permit BPTT to go back to the start point of training data, the computation time and storage requirement would grow linearly with time as the network runs, eventually reaching a point where the whole learning process becomes impractical. As a matter of fact, during the training of BPTT, since the contributions of the former model derivatives become smaller as the algorithm goes into the past because of recursive multiplication within each step, hence any information older than some k_t time steps into the past can be considered irrelevant, and may therefore be ignored. This phenomena is also named

as vanishing gradient. With this feature, we can employ an approximate version of error gradient by truncating it at an early stage, i.e., only the past k_t steps are unfolded [27]. Now, the local gradient is defined by

$$\begin{cases} \hat{V}(k+1) = \hat{V}(k) + \alpha e(k) \frac{\partial \hat{y}(k)}{\partial \hat{V}(k)} \\ \hat{W}_i(k+1) = \hat{W}_i(k) + \alpha e(k) \frac{\partial \hat{y}(k)}{\partial \hat{W}_i(k)} \end{cases} \quad (3.10)$$

$$\begin{cases} \frac{\partial \hat{y}(k)}{\partial \hat{V}(k)} = \Phi(k)^T + \hat{V}(k) \cdot \frac{\partial \Phi(k)}{\partial \hat{y}(k-1)} \cdot \frac{\partial \hat{y}(k-1)}{\partial \hat{V}(k)} \\ \frac{\partial \hat{y}(k)}{\partial \hat{W}_i(k)} = \text{diag}\{\Phi(k)\} \hat{V}(k)^T \hat{x}(k)^T + \hat{W}_i(k) \cdot \frac{\partial \Phi(k)}{\partial \hat{y}(k-1)} \cdot \frac{\partial \hat{y}(k-1)}{\partial \hat{W}_i(k)} \end{cases} \quad (3.11)$$

$$\begin{cases} \frac{\partial \hat{y}(k-1)}{\partial \hat{V}(k)} = \Phi(\hat{W}(k) \hat{x}(k-1))^T + \hat{V}(k) \cdot \frac{\partial \Phi(\hat{W}(k) \hat{x}(k-1))}{\partial \hat{y}(k-2)} \cdot \frac{\partial \hat{y}(k-2)}{\partial \hat{V}(k)} \\ \frac{\partial \hat{y}(k-1)}{\partial \hat{W}_i(k)} = \text{diag}\{\Phi(\hat{W}(k) \hat{x}(k-1))\} \hat{V}(k)^T \hat{x}(k-1)^T \\ \quad + \hat{W}_i(k) \cdot \frac{\partial \Phi(\hat{W}(k) \hat{x}(k-1))}{\partial \hat{y}(k-2)} \cdot \frac{\partial \hat{y}(k-2)}{\partial \hat{W}_i(k)} \end{cases} \quad (3.12)$$

$$\begin{cases} \vdots \\ \frac{\partial \hat{y}(k-k_t)}{\partial \hat{V}(k)} = 0 \\ \frac{\partial \hat{y}(k-k_t)}{\partial \hat{W}_i(k)} = 0 \end{cases} \quad (3.13)$$

where $0 \leq k_t \leq k$. Once the computation of BPTT has been performed back to time $k - k_t$, then the adjustment is applied to the weights. This second form of the algorithm is called the Truncated BPTT (T-BPTT) [27], with k_t names as truncation depth. For different choices of k_t , a tradeoff exists between the computational effort and the accuracy of the derivative calculation. In the extreme case $k_t = 0$ the static backpropagation is recovered since the second term in partial derivative is neglected. This method is called ordinary truncation; For $0 < k_t < k$, the method is called multi-step truncation; when $k_t = k$, a full BPTT is then recovered.

In real-life applications of T-BPTT, the use of truncation is not as artificial as it may sound. Unless the RNN is unstable, there should be a convergence of the derivative $\partial \hat{y}(k)/\partial \hat{V}(k)$ and $\partial \hat{y}(k)/\partial \hat{W}_i(k)$, because computations farther back in time correspond to higher powers of feedback strengths (roughly equal to sigmoid slopes multiplied by weights). In any events, the truncation depth k_t must be large enough

to produce derivative that closely approximate the actual values. This requirement places a lower bound on the value of k_t . For example, in the application of dynamically driven RNN to engine idle-speed control, the value $k_t = 30$ is considered to be reasonably conservative choice for that learning task to accomplished.

Compare BPTT with T-BPTT, we found that the unfolding procedures of both algorithms provide a useful tool for picturing them in terms of a cascade of similar layers progressing forward in time, thereby helping us to develop an understanding of how the procedures function. However, this strong point is unfortunately also the cause of the weakness of BPTT. Though BPTT works perfectly fine for relatively simple RNN consisting of a few neurons, the underlying formulas become unwieldy when the unfolding procedure is applied to more general architectures that are typical of those encountered in practice. In situations of this kind, the preferred procedure is to use the T-BPTT as described above.

3.1.3 Real Time Recurrent Learning

In this section, we describe another learning algorithm referred as RTRL. The algorithm derives its name from the fact that adjustments are made to the synaptic weights of a fully connected RNN in real time. RTRL algorithm has been proposed by Williams et al to overcome the shortcoming of BPTT [103] [104]. It is more efficient since it doesn't unfold the model into the past, but instead assume that the model parameters do not change during a specific time interval, i.e.,

$$\begin{aligned}\hat{V}(k) &= \hat{V}(k-1) = \dots = \hat{V}(1) \\ \hat{W}(k) &= \hat{W}(k-1) = \dots = \hat{W}(1)\end{aligned}\tag{3.14}$$

This assumption is exactly fulfilled for batch training mode, where the weight parameters are updated only after a full sweep through the training data. On the

other hand, even for sequential training in which the weights are updated at each time instant, RTRL can still be applied in case the learning rate is small enough such that the parameter changes can be negligible, i.e.,

$$\begin{aligned}\hat{V}(k) &\approx \hat{V}(k-1) \approx \dots \approx \hat{V}(1) \\ \hat{W}(k) &\approx \hat{W}(k-1) \approx \dots \approx \hat{W}(1)\end{aligned}\tag{3.15}$$

With this assumption, we can derive the following relationship

$$\frac{\partial \hat{y}(k-1)}{\hat{V}(k)} = \frac{\partial \hat{y}(k-1)}{\hat{V}(k-1)}, \quad \frac{\partial \hat{y}(k-1)}{\hat{W}_i(k)} = \frac{\partial \hat{y}(k-1)}{\hat{W}_i(k-1)}, \quad i = 1, 2, \dots, n$$

Following the rule, the derivative in (3.2) and (3.7) can be greatly simplified as

$$\frac{\partial \hat{y}(k)}{\partial \hat{V}(k)} = \Phi(k)^T + \hat{V}(k) \cdot \frac{\partial \Phi(k)}{\partial \hat{y}(k-1)} \cdot \frac{\partial \hat{y}(k-1)}{\partial \hat{V}(k-1)}\tag{3.16}$$

$$\frac{\partial \hat{y}(k)}{\partial \hat{W}_i(k)} = \text{diag}\{\Phi(k)\} \hat{V}(k)^T \hat{x}(k)^T + \hat{W}_i(k) \cdot \frac{\partial \Phi(k)}{\partial \hat{y}(k-1)} \cdot \frac{\partial \hat{y}(k-1)}{\partial \hat{W}_i(k-1)}\tag{3.17}$$

An explicit advantage of RTRL is that $\partial \hat{y}(k-1)/\partial \hat{V}(k-1)$ and $\partial \hat{y}(k-1)/\partial \hat{W}_i(k-1)$ are the previous model gradient and already available. Thus, calculation of RTRL requires only one step substitution, no need to carry out all the derivatives along the time axis.

The use of instantaneous gradient means that the RTRL described here deviates from BPTT, which is based on the true error gradient. The RTRL is not guaranteed to follow the precise negative gradient of the error function with respect to weight matrices. Compared with BPTT, RTRL is simpler and faster, it requires less memory, and most importantly its complexity doesn't depend on the size of training duration. The practical difference between the RTRL and non-realtime versions are often slight. These two versions become nearly identical as the learning rate parameter α is reduced. The most severe potential consequence of this deviation from the true gradient is that the observed trajectory may itself depend on the

weight changes produced by the algorithm, which may be viewed as another source of feedback and therefore a cause of instability in the system. This effect can be avoided by using a learning-rate parameter small enough to make the time scale of the weight changes much smaller than the time scale of the network operation.

Teacher Forcing

A strategy that is frequently used in the RTRL is teach forcing [105]. In adaptive filtering, teacher forcing is known as the equation error method [64]. Basically, teacher forcing involves replacing the actual output of a neuron, during training of the RNN, with the corresponding desired response, i.e., target signal, in subsequent computation of the dynamics behavior of RNN, whenever that desired response is available. Although teacher forcing is being described in the section under RTRL algorithm, its use actually applied to any other learning algorithm. For it to be applicable, however, the neuron in question must feed its output back to the network. Beneficial effects of teacher forcing include [105]

- i) Teacher forcing may lead to faster training. The reason for this improvement is the use of teacher forcing amounts to the assumption that the RNN has correctly learned all the earlier parts of the task that pertain to the neurons where teacher forcing has been applied.
- ii) Teacher forcing may serve as a corrective mechanism during training. For example, the synaptic weights of the network may have the correct values, but somehow the network is currently operating in the wrong region of the state space. Clearly, adjusting the synaptic weights is the wrong strategy in such a situation.

A gradient-based learning that uses teacher forcing is in actual fact optimizing a cost function different from its unforced counterpart. The teacher forced and unforced versions of the algorithm may therefore yield different solutions, unless the pertinent error signals are zero, in which case learning is unnecessary.

3.1.4 Normalized Real Time Recurrent Learning

A common problem of both BPTT and RTRL is that they typically suffer from slow convergence when dealing with statistically non-stationary input. Inspired by the work of normalized least mean square (NLMS) in the area of linear adaptive filters [90], Mandic and Chambers presented a normalized RTRL (N-RTRL) algorithm on training of RNN [59]. Similar to the derivations in [90], they start from the Taylor series expansion of the instantaneous output error. For example, consider the output layer training of RNN

$$\begin{cases} e(k) = d(k) - \hat{V}(k)\Phi(k) \\ \hat{V}(k+1) = \hat{V}(k) + \frac{\alpha}{\rho^v(k)} \cdot e(k) \frac{\partial \hat{y}(k)}{\partial \hat{V}(k)} \end{cases} \quad (3.18)$$

where $e(k)$ is the instantaneous modeling error and $\rho^v(k)$ is the normalization factor of output layer. If the modeling error is expanded using a Taylor series expansion

$$e(k+1) = e(k) + \sum_{i=1}^m \frac{\partial e(k)}{\partial \hat{V}_i(k)} \cdot \Delta \hat{V}_i(k) + \frac{1}{2!} \sum_{i=1}^m \sum_{j=1}^m \frac{\partial^2 e(k)}{\partial \hat{V}_i(k) \partial \hat{V}_j(k)} \cdot \Delta \hat{V}_i(k) \Delta \hat{V}_j(k) + \dots \quad (3.19)$$

From equation (3.18), the elements in (3.19) are

$$\begin{cases} \frac{\partial e(k)}{\partial \hat{V}_i(k)} = -\Phi'(\hat{W}_i(k)\hat{x}(k)) \\ \Delta \hat{V}_i(k) = \frac{\alpha}{\rho^v(k)} \Phi'(\hat{W}_i(k)\hat{x}(k)) \end{cases} \quad (3.20)$$

Ignoring the higher order derivatives and substituting (3.20) into (3.19), a truncated Taylor series expansion gives

$$\begin{aligned} e(k+1) &= e(k) \left(1 - \frac{\alpha}{\rho^v(k)} \sum_{i=1}^m \left| \Phi'(\hat{W}_i(k)\hat{x}(k)) \right|^2 \right) \\ &= e(k) \left(1 - \frac{\alpha}{\rho^v(k)} \|\Phi'(k)\|^2 \right) \end{aligned} \quad (3.21)$$

The aim of training is for the error $e(k+1)$ in (3.21) to vanish, which is the case for nontrivial solution (with $\alpha = 1$)

$$\rho^v(k) = \|\Phi'(k)\|^2 \quad (3.22)$$

Take into account the bounds on the values of higher order derivatives in (3.19), for a contractive activation function we may adjust the derived learning rate with a positive constant C , as

$$\rho^v(k) = C + \|\Phi'(k)\|^2 \quad (3.23)$$

Thus the magnitude of $\rho^v(k)$ varies in time with the power of the first order derivative of activation function, which provides a normalization of the algorithm. Actually, the training of (3.18) and (3.23) impose similar stabilization and convergence effects as the idea of NLMS. On the other hand, in the derivation it assumes statistical independence between the weights, input vector, teaching signal and learning rate, which is often not the case in practical applications. There is another reason to use $\rho^v(k)$ since there is a need to add a positive constant C to the denominator of normalization factor. Following the same method, we can derive the N-RTRL for hidden layer of RNN as

$$\left\{ \begin{array}{l} \hat{W}_i(k+1) = \hat{W}_i(k) + \frac{\alpha}{\rho_i^w(k)} \cdot \frac{\partial \hat{y}(k)}{\partial \hat{W}_i(k)} \\ \rho_i^w(k) = C + \|\hat{V}(k)\|^2 \|\Phi'(k)\|^2 \|\hat{x}(k)\|^2 \\ i = 1, \dots, m \end{array} \right. \quad (3.24)$$

where α is a constant in the interval $(0, 1)$ and $\rho_i^w(k)$ are normalization factors employed to speed up the training.

In [59], several computer simulation experiments were carried out to show that the N-RTRL algorithms of (3.18) and (3.24) outperform standard RTRL with fixed learning rate. Further, simulations show that the performance of N-RTRL is highly

dependent on the choice of the constant C . Empirically, $C < 1$ is a sufficiently good range for the N-RTRL in many practical cases.

3.2 Normalized Adaptive Recurrent Learning

Though a factor C is employed in N-RTRL to compensate the gradient approximation error, however, there are no explicit rules on how to choose a suitable C . Further, C is given as a constant value, which may not be “smart” enough to be automatically adjusted when training data statistics is varying. In such cases, the stability sufficiency of training is sacrificed. Hence, we expect to use adaptive coefficients to obtain a more accurate model of error gradient. This will be helpful to improve the overall robustness of training. Inspired by this idea, a modification based on the N-RTRL is proposed in this section, namely, Normalized Adaptive Recurrent Learning (NARL). A new augmented residual error gradient is introduced with stability proof of the training algorithm. The analysis shows that the proposed algorithm can effectively improve the robustness of training compared to the N-RTRL and others. Now consider the RNN model in equation (1.3) with m hidden layer neurons and the definition of instantaneous error in (1.5).

$$\begin{cases} e(k) = d_n(k) - \hat{y}(k) \\ d_n(k) = d(k) + \varepsilon(k) \end{cases} \quad (3.25)$$

where $d_n(k)$ is a command signal corrupted by the noise signal $\varepsilon(k)$. In a multi-layered RNN, it may not be possible to update all the estimated weights with a single gradient approximation function. We shall partition the training into different layers. Thus, in the proposed NARL algorithm, output layer weight $\hat{V}(k)$ and hidden layer weight $\hat{W}(k)$ are to be updated separately using different learning parameters

as follows

$$\begin{cases} \hat{V}(k+1) = \hat{V}(k) + \frac{\alpha^v(k)e(k)}{\rho^v(k)}\Phi(k)^T \\ \hat{W}(k+1) = \hat{W}(k) + \frac{\alpha^w e(k^+)}{\rho^w(k)}(\hat{x}(k)\hat{V}(k+1)\text{diag}\{\Phi'(k)\} + \gamma^w(k))^T \end{cases} \quad (3.26)$$

where $x(k)$ is $n \times 1$ dimensional state vector, variable $\alpha^v(k)$ and constant α^w are learning rates, $\rho^v(k)$ and $\rho^w(k)$ are normalization factors, $\Phi'(\cdot)$ is the first order derivative of activation function $\Phi(k)$ on the input space, and $\text{diag}\{\Phi'(\cdot)\}$ means a diagonal matrix which has each entry of $\Phi'(\cdot)$ on the main diagonal, and $\gamma^w(k)$ is so-called augmented residual error gradient that compensates the gradient approximation error. For the convenience of exposition, some notations are introduced as below

$$\begin{aligned} \tilde{V}(k) &= V^* - \hat{V}(k) & \Delta\Phi(k+1) &= \Phi(k+1) - \Phi(k) \\ \tilde{W}(k) &= W^* - \hat{W}(k) & \Delta\hat{x}(k+1) &= \hat{x}(k+1) - \hat{x}(k) \\ \Delta\hat{W}(k+1) &= \hat{W}(k+1) - \hat{W}(k) & \Delta d_n(k+1) &= d_n(k+1) - d_n(k) \\ \tilde{\Phi}(k) &= \Phi^* - \Phi(k) = \Phi(W^*x^*) - \Phi(\hat{W}(k)\hat{x}(k)) \end{aligned} \quad (3.27)$$

where V^* and W^* are optimal weights of output and hidden layer respectively, x^* is ideal attractor of RNN corresponding to the optimal signal $d(k)$. To simplify the derivation, we assume x^* , V^* and W^* is time-invariant. This assumption is true in case we choose the sampling time small enough such that the background signal $d(k)$ can be regarded as constant d temporarily. Notice neither d nor $\varepsilon(k)$ are directly measurable except $d_n(k)$.

Theorem 3.2.1. *If the learning rate α^w of the NARL in equation (3.26) is chosen in the interval $(0, 1)$, and $\alpha^v(k)$ is calculated by*

$$\alpha^v(k) = \begin{cases} 1 + \frac{\Delta d_n(k+1)}{e(k)} & |e(k)| > \xi \\ 1 & |e(k)| \leq \xi \end{cases} \quad (3.28)$$

and the normalization factors $\rho^v(k)$, $\rho^w(k)$ and the augmented residual error gradient $\gamma^w(k) \in R^{n \times m}$ are determined as

$$\rho^v(k) = \|\Phi(k)\|^2 \quad (3.29)$$

$$\rho^w(k) = \phi_d^2 \cdot \|\hat{V}(k+1)\|^2 \|\hat{x}(k)\|^2 \quad (3.30)$$

$$\begin{aligned} \gamma^w(k) = & -\frac{\hat{x}(k)}{\hat{x}(k)^T \hat{x}(k+1)} \left(\frac{\rho^w(k)}{\alpha^w e(k^+)} \hat{W}(k) \Delta \hat{x}(k+1) \right. \\ & \left. + \text{diag}\{\Phi'(k)\} \hat{V}(k+1) \hat{x}(k)^T \Delta \hat{x}(k+1) \right)^T \end{aligned} \quad (3.31)$$

where ξ is a small positive number, ϕ_d is the maximum value of the first order derivative of RNN activation function, which is defined by

$$\phi_d = \max\{|\Phi'_i(\cdot)|\} \quad (3.32)$$

Then the training is stable in the sense that $e(k)$ is bounded (convergent) as long as $\Delta d_n(k)$ is bounded (convergent).

Proof: To study the stability of the algorithm, we start with establishing the error dynamics of the training. Because the training is separated into output and hidden layer respectively. We should also partition the analysis into this two subsystems.

(i) Output Layer Training

Expanding $e(k)$ as

$$\begin{aligned} e(k) &= d - \hat{V}(k)\Phi(k) + \varepsilon(k) \\ &= V^*\Phi^* - \hat{V}(k)\Phi(k) + \varepsilon(k) \\ &= [V^*\Phi^* + \hat{V}(k)\Phi(k) - \hat{V}(k)\Phi^* - V^*\Phi(k)] \\ &\quad + [V^*\Phi(k) - \hat{V}(k)\Phi(k)] + [\hat{V}(k)\Phi^* - \hat{V}(k)\Phi(k)] + \varepsilon(k) \\ &= [V^* - \hat{V}(k)][\Phi^* - \Phi(k)] + [V^*\Phi(k) - \hat{V}(k)\Phi(k)] \\ &\quad + [\hat{V}(k)\Phi^* - \hat{V}(k)\Phi(k)] + \varepsilon(k) \end{aligned}$$

$$= \tilde{V}(k)\tilde{\Phi}(k) + \tilde{V}(k)\Phi(k) + \hat{V}(k)\tilde{\Phi}(k) + \varepsilon(k) \quad (3.33)$$

If define

$$\begin{cases} e^\Delta(k) = \tilde{V}(k)\tilde{\Phi}(k) \\ e^v(k) = \tilde{V}(k)\Phi(k) \\ e^w(k) = \hat{V}(k)\tilde{\Phi}(k) \end{cases} \quad (3.34)$$

Then $e(k)$ can be expressed in a simplified form

$$e(k) = e^v(k) + e^w(k) + e^\Delta(k) + \varepsilon(k) \quad (3.35)$$

Similarly, the posterior error $e(k^+)$ can be decomposed as

$$\begin{aligned} e(k^+) &= d - \hat{V}(k+1)\Phi(k) + \varepsilon(k) \\ &= [V^*\Phi^* - V^*\Phi(k) - \hat{V}(k+1)\Phi^* + \hat{V}(k+1)\Phi(k)] \\ &\quad + [V^*\Phi(k) - \hat{V}(k+1)\Phi(k)] + [\hat{V}(k+1)\Phi^* - \hat{V}(k+1)\Phi(k)] + \varepsilon(k) \\ &= \tilde{V}(k+1)\tilde{\Phi}(k) + \tilde{V}(k+1)\Phi(k) + \hat{V}(k+1)\tilde{\Phi}(k) + \varepsilon(k) \\ &= e^\Delta(k^+) + e^v(k^+) + e^w(k^+) + \varepsilon(k) \end{aligned} \quad (3.36)$$

where

$$\begin{cases} e^\Delta(k^+) = \tilde{V}(k+1)\tilde{\Phi}(k) \\ e^v(k^+) = \tilde{V}(k+1)\Phi(k) \\ e^w(k^+) = \hat{V}(k+1)\tilde{\Phi}(k) \end{cases} \quad (3.37)$$

Subtracting V^* by both sides of output layer training equation in (3.26)

$$\tilde{V}(k+1) = \tilde{V}(k) - \frac{\alpha^v(k)e(k)}{\rho^v(k)}\Phi(k)^T \quad (3.38)$$

Substituting (3.38) into (3.37)

$$e^\Delta(k^+) = \tilde{V}(k)\tilde{\Phi}(k) - \frac{\alpha^v(k)e(k)}{\rho^v(k)}\Phi(k)^T\tilde{\Phi}(k)$$

$$= e^\Delta(k) - \frac{\alpha^v(k)e(k)}{\rho^v(k)}\Phi(k)^T\tilde{\Phi}(k) \quad (3.39)$$

$$\begin{aligned} e^v(k^+) &= \tilde{V}(k)\Phi(k) - \frac{\alpha^v(k)e(k)}{\rho^v(k)}\Phi(k)^T\Phi(k) \\ &= e^v(k) - \frac{\alpha^v(k)e(k)}{\rho^v(k)}\Phi(k)^T\Phi(k) \end{aligned} \quad (3.40)$$

$$\begin{aligned} e^w(k^+) &= \hat{V}(k)\tilde{\Phi}(k) + \frac{\alpha^v(k)e(k)}{\rho^v(k)}\Phi(k)^T\tilde{\Phi}(k) \\ &= e^w(k) + \frac{\alpha^v(k)e(k)}{\rho^v(k)}\Phi(k)^T\tilde{\Phi}(k) \end{aligned} \quad (3.41)$$

Because of the fact

$$\rho^v(k) = \|\Phi(k)\|^2 = \Phi(k)^T\Phi(k) \quad (3.42)$$

Substituting (3.42) into equation (3.39)–(3.41) respectively

$$\begin{cases} e^\Delta(k^+) = e^\Delta(k) - \alpha^v(k)e(k) \\ e^v(k^+) = e^v(k) - \alpha^v(k)e(k) \\ e^w(k^+) = e^w(k) + \alpha^v(k)e(k) \end{cases} \quad (3.43)$$

Now summing up all the three equation, we can obtain the following equality

$$\begin{aligned} e(k^+) &= e^\Delta(k^+) + e^v(k^+) + e^w(k^+) + \varepsilon(k) \\ &= e^\Delta(k) + e^v(k) + e^w(k) + \varepsilon(k) - \alpha^v(k)e(k) \\ &= (1 - \alpha^v(k))e(k) \end{aligned} \quad (3.44)$$

(ii) Hidden Layer Training

Following the same route as the output layer analysis, the modeling error of RNN at the $k + 1$ time sample can be decomposed as

$$\begin{aligned} e(k+1) &= d - \hat{V}(k+1)\Phi(k+1) + \varepsilon(k+1) \\ &= [d - \hat{V}(k+1)\Phi(k) + \varepsilon(k)] - [\hat{V}(k+1)\Phi(k+1)] \end{aligned}$$

$$\begin{aligned}
& -\hat{V}(k+1)\Phi(k)] + \varepsilon(k+1) - \varepsilon(k) \\
& = e(k^+) - \hat{V}(k+1)\Delta\Phi(k+1) + \Delta\varepsilon(k+1)
\end{aligned} \tag{3.45}$$

By the mean value theorem

$$\begin{aligned}
& \Delta\Phi(k+1) \\
& = \Phi(k+1) - \Phi(k) \\
& = \Phi(\hat{W}(k+1)\hat{x}(k+1)) - \Phi(\hat{W}(k)\hat{x}(k)) \\
& = \text{diag}\{\Phi'(\mu(k+1))\}(\hat{W}(k+1)\hat{x}(k+1) - \hat{W}(k)\hat{x}(k)) \\
& = \text{diag}\{\Phi'(\mu(k+1))\}(\hat{W}(k+1)\Delta\hat{x}(k+1) + \Delta\hat{W}(k+1)\hat{x}(k))
\end{aligned} \tag{3.46}$$

where $\mu(k+1)$ is an unknown vector with all the entry values between those of $\hat{W}(k+1)\hat{x}(k+1)$. Further, by the training equation (3.26), we have

$$\begin{aligned}
& \Delta\hat{W}(k+1)\hat{x}(k) \\
& = \frac{\alpha^w e(k^+)}{\rho^w(k)}(\hat{x}(k)\hat{V}(k+1)\text{diag}\{\Phi'(k)\} + \gamma^w(k))^T \hat{x}(k) \\
& = \frac{\alpha^w e(k^+)}{\rho^w(k)}\text{diag}\{\Phi'(k)\}\hat{V}(k+1)^T \hat{x}(k)^T \hat{x}(k) + \frac{\alpha^w e(k^+)}{\rho^w(k)}\gamma^w(k)^T \hat{x}(k)
\end{aligned} \tag{3.47}$$

Substituting (3.46) and (3.47) into (3.45), then the error dynamics of hidden layer is obtained

$$\begin{aligned}
e(k+1) & = -\frac{\alpha^w e(k^+)}{\rho^w(k)}\hat{V}(k+1)\text{diag}\{\Phi'(\mu(k+1))\}\text{diag}\{\Phi'(k)\}\hat{V}(k+1)^T \hat{x}(k)^T \hat{x}(k) \\
& \quad -\hat{V}(k+1)\text{diag}\{\Phi'(\mu(k+1))\}(\hat{W}(k+1)\Delta\hat{x}(k+1) \\
& \quad + \frac{\alpha^w e(k^+)}{\rho^w(k)}\gamma^w(k)^T \hat{x}(k)) + e(k^+) + \Delta\varepsilon(k+1)
\end{aligned} \tag{3.48}$$

Substituting $\hat{W}(k+1)$ of (3.26) into the last term on the right side of formula (3.48)

$$\hat{W}(k+1)\Delta\hat{x}(k+1) + \frac{\alpha^w e(k^+)}{\rho^w(k)}\gamma^w(k)^T \hat{x}(k)$$

$$\begin{aligned}
&= \hat{W}(k)\Delta\hat{x}(k+1) + \frac{\alpha^w e(k^+)}{\rho^w(k)}(\hat{x}(k)\hat{V}(k+1)\text{diag}\{\Phi'(k)\} + \gamma^w(k))^T \Delta\hat{x}(k+1) \\
&\quad + \frac{\alpha^w e(k^+)}{\rho^w(k)}\gamma^w(k)^T \hat{x}(k) \\
&= \hat{W}(k)\Delta\hat{x}(k+1) + \frac{\alpha^w e(k^+)}{\rho^w(k)}\text{diag}\{\Phi'(k)\}\hat{V}(k+1)^T \hat{x}(k)^T \Delta\hat{x}(k+1) \\
&\quad + \frac{\alpha^w e(k^+)}{\rho^w(k)}\gamma^w(k)^T \hat{x}(k+1)
\end{aligned} \tag{3.49}$$

According to the definition of $\gamma^w(k)$ in (3.31) and by trivial computations

$$\begin{aligned}
&\frac{\alpha^w e(k^+)}{\rho^w(k)}\gamma^w(k)^T \hat{x}(k+1) \\
&= -\frac{\alpha^w e(k^+)}{\rho^w(k)\hat{x}(k)^T \hat{x}(k+1)}\left(\frac{\rho^w(k)}{\alpha^w e(k^+)}\hat{W}(k)\Delta\hat{x}(k+1) \right. \\
&\quad \left. + \text{diag}\{\Phi'(k)\}\hat{V}(k+1)^T \hat{x}(k)^T \Delta\hat{x}(k+1)\right)\hat{x}(k)^T \hat{x}(k+1) \\
&= -\hat{W}(k)\Delta\hat{x}(k+1) - \frac{\alpha^w e(k^+)}{\rho^w(k)}\text{diag}\{\Phi'(k)\}\hat{V}(k+1)^T \hat{x}(k)^T \Delta\hat{x}(k+1)
\end{aligned} \tag{3.50}$$

Thus by the results of (3.49) and (3.50), the following condition can be derived

$$\hat{W}(k+1)\Delta\hat{x}(k+1) + \frac{\alpha^w e(k^+)}{\rho^w(k)}\gamma^w(k)^T \hat{x}(k) = 0 \tag{3.51}$$

Substituting (3.50) into (3.48), the error dynamics of hidden layer training becomes

$$\begin{aligned}
e(k+1) &= \Delta\varepsilon(k+1) + e(k^+)(1 - \frac{\alpha^w}{\rho^w(k)}\hat{V}(k+1)\text{diag}\{\Phi'(\mu(k+1))\} \cdot \\
&\quad \text{diag}\{\Phi'(k)\}\hat{V}(k+1)^T x(k)^T x(k))
\end{aligned} \tag{3.52}$$

(iii) *stability of the NARL training algorithm*

Defining

$$\Lambda(k) = \frac{\alpha^w}{\rho^w(k)}\hat{V}(k+1)\text{diag}\{\Phi'(\mu(k+1))\}\text{diag}\{\Phi'(k)\}\hat{V}(k+1)^T \hat{x}(k)^T \hat{x}(k) \tag{3.53}$$

Consequently (3.52) can be simplified to

$$e(k+1) = \Delta\varepsilon(k+1) + (1 - \Lambda(k))e(k) \quad (3.54)$$

Because

$$\rho^w(k) = \phi_d^2 \cdot \|\hat{V}(k+1)\|^2 \|\hat{x}(k)\|^2 \quad (3.55)$$

then by the definition of ϕ_d in (3.32)

$$0 < \Lambda(k) \leq \frac{\alpha^w \hat{V}(k+1) \hat{V}(k+1)^T \hat{x}(k)^T \hat{x}(k)}{\|\hat{V}(k+1)\|^2 \|x(k)\|^2} < 1 \quad (3.56)$$

Substituting (3.44) into (3.54)

$$e(k+1) = (1 - \Lambda(k))(1 - \alpha^v(k))e(k) + \Delta\varepsilon(k+1) \quad (3.57)$$

Equation (3.57) establishes the error dynamics of the entire NARL training algorithm including both output and hidden layer. It reflects the propagation of the modeling error between each full iteration step. Next, dividing both sides of equation (3.57) by $\Delta\varepsilon(k+1)$

$$\frac{e(k+1)}{\Delta\varepsilon(k+1)} = (1 - \Lambda(k))(1 - \alpha^v(k)) \frac{e(k)}{\Delta\varepsilon(k+1)} + 1 \quad (3.58)$$

Above equality actually reflects the dependence of the modeling error on external disturbances by the NARL. Further, by the assumption of time invariant d , we have $\Delta d_n(k) = \Delta\varepsilon(k)$. Then by the condition of (3.28), we may derive that

$$-1 \leq (1 - \alpha^v(k)) \frac{e(k)}{\Delta d_n(k+1)} \leq 0 \quad (3.59)$$

which actually ensures $\left| \frac{e(k+1)}{\Delta d_n(k+1)} \right| \leq 1$, and in turn the boundedness of the modeling error of the NARL. From above arguments, we can draw that the modeling error of the NARL algorithm will always be bounded (convergent) as long as $\Delta d_n(k)$ is

bounded (convergent). Thus we conclude the proof.

Indeed the above proposed NARL is a generalized form of training and many of existing algorithms can be derived as its special forms. For example, if we set $\rho^v = \rho^w = 1$ and $\gamma^w(k) = 0$, then the algorithm becomes the RTRL introduced by Williams and Zipser [103]. If we take into account the normalization factor, then it will become a N-RTRL algorithm. The most remarkable feature of the NARL is the improved robustness comparing to these conventional training algorithms. In loose terms, robustness implies that the ratio of modeling error with respect to the variation of external input is guaranteed to be bounded, no matter what the nature and the statistics of the disturbances are [81]. Under this framework, we may compare the robustness between the NARL and N-RTRL by looking at the ratio $\frac{e(k+1)}{\varepsilon(k+1)} = \frac{e(k+1)}{\Delta d_n(k+1)} \cdot \frac{\Delta d_n(k+1)}{\varepsilon(k+1)}$, which is desired to be as small as possible. When a constant learning rate is employed, say, $0 < \alpha^v < 1$, then the term $\frac{e(k)}{\Delta d_n(k+1)}(1 - \alpha^v)$ could probably larger than zero if $e(k)$ and $\Delta d_n(k+1)$ has the same sign. In this case, the ratio of $\frac{e(k+1)}{\Delta d_n(k+1)}$ will be larger than 1. In another word, there is no rejection of the input disturbance, or even the influence of the disturbance variation is amplified. With the proposed adaptive learning rate, the sensitivity can be ensured to be less than unit. Hence in this manner, the robustness is improved.

Another important advantage of the NARL is that only limited (one step) past value of state vector is involved by $\gamma^w(k)$, which provides an affordable computation load compared to the BPTT, which requires to fully unfold the derivative back ward to the initial state. Intuitively speaking, the term $\gamma^w(k)$ reveals the recurrent nature of RNN that different from other feed forward neural network training algorithms. The calculation of $\gamma^w(k)$ in (3.31) involves the variation of RNN output in previous iteration steps, which reflects the adaptation partially depends on the training in the past steps. This is caused by internal feedback structure. In many of previous works [59] [60], the residual error derivative $\gamma^w(k)$ is ignored in order to reduce

the computation efforts, means using $x(k)\hat{V}(k+1)\text{diag}\{\Phi'(k)\}$ in the hidden layer updating rule. However, this approximation could probably result in insufficient criterion of stability analysis. Because when $\hat{x}(k)\hat{V}(k+1)\text{diag}\{\Phi'(k)\}$ is not the same sign with $\hat{x}(k)\hat{V}(k+1)\text{diag}\{\Phi'(k)\} + \gamma^w(k)$, then the gradient search direction of the approximated updating rule will be opposite to the correct one and potentially cause instability.

There are also some limitations of the NARL. One obvious disadvantage is the employment of $\hat{x}(k+1)$ in $\gamma^w(k)$, which requires the value of future input. From practical concern, though we may use delayed data set or independent training data set to address the problem, however those methods will definitely restrict the applicability of the NARL. Finally, we acknowledge that the stability of the NARL is not fully addressed yet, because the convergence of $\hat{V}(k)$ and $\hat{W}(k)$ still remains unsolved. Though the proof of the NARL is incomplete regarding to this point, but it is still presented in the thesis, because we believe the works do present a in-depth study on the gradient type training and the findings provide valuable guidance for future research.

3.3 Simulation Studies of NARL

In this section, we investigate the NARL algorithm via simulations on a time series prediction problem, and carry out the comparison with the N-RTRL. The objective is to use RNN to predict the next sample of a real sequence $\{y(k)\}$, which is generated by the following process

$$y(k+1) = \frac{y(k)}{1+y^2(k)} + u^3(k) \quad (3.60)$$

where $u(k)$ is white Gaussian input sequence. The model of (3.60) is chosen from the benchmark problem in [59] (pp.159). Two data groups are generated in simulations.

One is the training data set, and the other is for evaluation purpose. The RNN is constructed by 50 neurons and both hidden layer and output layer weights are random initialized in the range of $(-1, 1)$. The input vector is 5 dimensional, which consists of this and last sample of time sequence $u(k)$ and RNN output feedback with 1 to 3 steps delay respectively. Sigmoid function is chosen as activation function, which is monotonic increasing, and both first and second order differentiable. The function and its first order derivative are given in equation (3.61) including the boundaries.

$$-1 < y = \frac{1}{1 + e^{-\lambda x}} < 1 \quad 0 < \frac{dy}{dx} = \frac{\lambda e^{-\lambda x}}{(1 + e^{-\lambda x})^2} \leq \frac{\lambda}{4} \quad (3.61)$$

Firstly, the influence of $\alpha^v(k)$ on the training performance of the NARL is studied. To provide a comparative idea, we have also implemented the N-RTRL ($\beta = 1$) in simulations with constant $C = 0$ and $C = 0.2$ respectively. The training results are presented in Figure 3.2–3.3. In order to present a clear illustration on both transient and steady state performance of each training algorithm, the first 100 steps and full 3000 steps of training error are shown separately. Moreover, the squared training errors of first 100 steps are plot in logarithmic format to provide a further better comparison. The steady state training errors in Figure 3.3 are expressed in dB (20 times the logarithm of the amplitude ratio between error and signal) such that performance difference between the NARL and N-RTRL can be more explicit. The statistics of the training data, including the mean of the training error of steady state as well as the standard deviations, are listed in table 3.1.

The two figures shows that the convergent speeds of the N-RTRL and NARL are almost same (both within 20 steps). On the contrary, the NARL can achieve better steady state response than the N-RTRL. The mean of squared training error of the NARL is -29.80dB while the N-RTRL is -26.99dB at best, which is improved about 10%. The data tallies with the theoretical results. The traces of learning rate $\alpha^v(k)$ is provided in Figure 3.4 for reader's reference. We also shown the first entry

Figure 3.2: Squared training errors of first 100 steps of the NARL and N-RTRL training respectively (transient response)

Figure 3.3: Squared training errors (in dB) of full 3000 steps of the NARL and N-RTRL training respectively (steady state response)

NARL		N-RTRL		
Mean(dB)	Std	C	Mean(dB)	Std
−29.80	9.60	1.0	−20.19	5.92
		0.8	−21.13	6.05
		0.5	−22.81	6.11
		0.2	−24.67	6.07
		0.0	−26.99	6.15

Table 3.1: Statistics of the training errors of the NARL and N-RTRL with different constant C

of the augmented residual error derivative term $\gamma^w(k)$, which is arbitrarily chosen and the other entries are omitted to avoid over plotting.

Figure 3.4: Traces of augmented residual error derivative $\gamma_1^w(k)$ and learning rate $\alpha^v(k)$ of the NARL

Now with the successfully trained RNN weights of each algorithm, we investigate how well the training is by the evaluation data set. The prediction results are displayed in Figure 3.5. There are no apparent difference between the two performance. This is because we are using the same RNN structure.

Figure 3.5: Prediction errors (in dB) of the NARL and N-RTRL with the evaluation data set

In the second simulation, we are interested in how the training is affected by the number of hidden layer neurons and the exponential factor of activation functions. We list the statistics of the training errors with various values of this two parameters in table 3.2. The data are obtained by averaging the squared error of the steady states through 50 runs. All simulations start with same initial weights, which can make a same starting point of training error such that we can make a convincing comparison. The results indicate that the steady state performance is slightly improved as the λ increases. A possible reason is that transition slope of linear region of activation function becomes higher (faster) with larger λ , which will result in more possibility of making $\Lambda(k)$ in (3.58) towards 0. A similar phenomena is also observed in [81] (pp.617). In contrast, there is no obvious influence of the neuron number on the training performance.

exponential factor λ	NARL		N-RTRL		number of nodes	NARL		N-RTRL	
	Mean	Std	Mean	Std		Mean	Std	Mean	Std
1	-29.73	9.61	-26.90	6.15	10	-29.79	9.58	-26.96	6.11
4	-29.80	9.60	-26.99	6.10	20	-29.81	9.60	-26.99	6.15
8	-29.85	9.58	-27.05	6.12	50	-29.80	9.60	-26.99	6.10
12	-29.87	9.53	-27.06	6.09	80	-29.81	9.66	-27.01	6.21

Table 3.2: Statistics of the training errors of the NARL with different λ and neuron number respectively

3.4 Summary

In this chapter, we briefly describe the conventional gradient descent training algorithms of RNN, including the BPTT, T-BPTT, RTRL, N-RTRL. Then we compare the advantages and disadvantages of these algorithms with each other. Based on that, we formulate the motivations for developing new algorithms, which should be able to reduce or eliminate the risk of instability, meanwhile improve or at least maintain the training performance. Along the line of this thought, the NARL training algorithm of RNN is proposed. Though the algorithm does not completely realize a robust training independent of external disturbance and system uncertainties, it does effectively reduce the influence of them on steady state performance, by using the augmented residual gradient error and adaptive learning rate. Another advantage of the NARL is that it employs normalization factors to improve the transient response, which is similar to the N-RTRL. All these points are justified by theoretical analysis, as well as the stability proof of the NARL. Simulations are presented to compare the NARL against the N-RTRL on time series prediction problems and hence verify the theoretical results.

Chapter 4

Robust Adaptive Gradient Descent Training Algorithm

There are some limitations of the NARL algorithm proposed in last chapter. For example, the involvement of $\hat{x}(k+1)$ requires the value of future input, which may sacrifice the applicability of the algorithm. Also the stability of the weights have not been proved yet. To address these problems, in this chapter, we introduce a Robust Adaptive Gradient Descent (RAGD) algorithm and apply the Cluett's law to derive the stability conditions of the training. The main focus of the RAGD is to introduce an improved RNN training speed over the standard N-RTRL algorithm in terms of less discrete time steps. The RNN model is still based on the external dynamics of (1.3). In addition to the Multi-Input Single-Output (MISO) RNN, results of Multi-Input Multi-Output (MIMO) RNN are given as well. The main feature of the RAGD algorithm is to use three adaptive parameters to adjust the effective adaptive learning rate. The first parameter is the hybrid learning rate, which provides a hybrid training fashion and controls the RAGD to change the training patterns between the real time Backpropagation (BP) [59] and the RTRL [103]. Adaptive dead zone learning rates and normalization factors are also used in the RAGD, which

are similar to the classical control systems and adaptive signal processing [84] [87].

4.1 RAGD Training for MISO RNN: Output Layer Analysis

We start with the output layer analysis of MISO type RNN. Considering the RNN in equation (1.3) with m hidden layer neuron, 1 output layer neuron, and n dimensional state vector, for the convenience of exposition, we rewrite it as follows

$$\hat{y}(k) = \hat{V}(k)\Phi(\hat{W}(k)x(k)) \quad (4.1)$$

where $\hat{V}(k) \in R^{1 \times m}$, $\hat{W}(k) \in R^{m \times n}$, and $\Phi(\cdot)$ is the vector of nonlinear activation functions. The state vector is defined by

$$\hat{x}(k) = [u(k), u(k-1), \dots, u(k-k_1+1), \hat{y}(k-1), \dots, \hat{y}(k-k_2)]^T \in R^{n \times 1} \quad (4.2)$$

in which k_1 is the number of the tapped delay of input, k_2 is the largest delay of output feedback, and they satisfy $n = k_1 + k_2$. When using the standard gradient training algorithm, the output layer updating is expressed by

$$\hat{V}(k+1) = \hat{V}(k) + \alpha e(k) \frac{\partial \hat{y}(k)}{\partial \hat{V}(k)} \quad (4.3)$$

In order to analyze the dynamics of this training equation via input-output approach, the first step is to restructure (4.3) into an error feedback loop, which should be the same as that in Figure 2.1. Further, the weight estimation error must be referred as the output signal. For this purpose, define the estimation error

$$e^v(k) = \hat{V}(k)\Phi(k) - V^*\Phi(k) = \tilde{V}(k)\Phi(k) \quad (4.4)$$

where $V^* \in R^{1 \times m}$ and $\tilde{V}(k) = V(k) - V^*$ are the ideal weight vector and estimation error vector of output layer respectively. Similar to $\Phi(k)$, we define

$$\Phi^*(k) = [\phi(W_1^* x^*(k)) \quad \phi(W_2^* x^*(k)) \quad \cdots \quad \phi(W_m^* x^*(k))]^T \quad (4.5)$$

where $x^*(k) \in R^{n \times 1}$ is the ideal input state, $W^* \in R^{m \times n}$ is the ideal weight matrix of hidden layer of the RNN, and subscript i denotes i th row for matrices or i th entry for vectors. For example, $e_i(k)$ is the i th entry of training error vector $e(k) = [e_1(k) \quad e_2(k) \cdots e_l(k)]^T$, and $W_i^*(k) \in R^{1 \times m}$ denotes the i th row of output layer weight matrix $W^*(k)$. Under this framework, the training error of RNN can be expanded as

$$\begin{aligned} e(k) &= d(k) - \hat{y}(k) + \varepsilon(k) \\ &= V^* \Phi^*(k) - \hat{V}(k) \Phi(k) + \varepsilon(k) \\ &= [V^* \Phi^*(k) - V^* \Phi(k)] - [\hat{V}(k) \Phi(k) - V^* \Phi(k)] + \varepsilon(k) \end{aligned} \quad (4.6)$$

Because the term $V^* \Phi^*(k) - V^* \Phi(k)$ is temporarily constant in case of output layer training, we can define $\tilde{\varepsilon}^v(k) = \varepsilon(k) + V^* \Phi^*(k) - V^* \Phi(k)$. Then the training error can be rewritten in the following form

$$\tilde{\varepsilon}^v(k) - e^v(k) = e(k) \quad (4.7)$$

Equation (4.7) has a similar form as the feedback path of the system (2.10), with $e^v(k)$ and $e(k)$ corresponding to $e(k)$ and $e_0(k)$ in Figure 2.1 respectively, and here the feedback gain is unity, i.e., $H_2 = 1$. To provide a better explanation, we illustrate the closed loop dynamics of RNN output layer training in Figure 4.1.

There is an important implication in the relation of (4.7). The $e^v(k)$, $e(k)$ and $\tilde{\varepsilon}^v(k)$ correspond to the weight estimation error, the RNN modeling error and the distur-

Figure 4.1: Closed loop dynamics of output layer training of RNN

bance, respectively. Hence the training error is directly linked to the disturbance, and in turn, the parameter estimating error of the RNN output layer. If we further establish a nonlinear mapping from the original disturbance $\tilde{\varepsilon}^v(k)$ to the resulting parameter estimation error $e^v(k)$, the relationship between L_2 -stability of the output layer training and the learning parameters can subsequently be studied by imposing the conditions of Lemma 2.3.2.

Now we are in the position to propose the RAGD training algorithm for the output layer of the RNN. The key idea is to design an adaptive hybrid learning algorithm which uses both the standard BP and the N-RTRL modes. That is,

$$\hat{V}(k+1) = \hat{V}(k) - \frac{\alpha^v(k)}{\rho^v(k)} \frac{df(e(k))}{d\hat{V}(k)} = \hat{V}(k) + \frac{\alpha^v(k)}{\rho^v(k)} e(k) \frac{d\hat{y}(k)}{d\hat{V}(k)} \quad (4.8)$$

The estimated derivative of the RNN output with respect to the output layer weight can be calculated as

$$\frac{d\hat{y}(k)}{d\hat{V}(k)} = \frac{\partial \hat{y}(k)}{\partial \hat{V}(k)} + \beta^v(k) \frac{\partial \hat{y}(k)}{\partial \hat{x}(k)} \frac{\partial \hat{x}(k)}{\partial \hat{V}(k)} = \Phi(k)^T + \beta^v(k) \hat{V}(k) \text{diag}\{\Phi'(k)\} \hat{W}(k) \hat{D}^v(k) \quad (4.9)$$

For the convenience of presentation, we define an auxiliary notation $\hat{A}(k) \in R^{1 \times m}$ as

$$\hat{A}(k) = \hat{V}(k) \text{diag}\{\Phi'(k)\} \hat{W}(k) \hat{D}^v(k) \quad (4.10)$$

Subsequently equation (4.9) is simplified as $d\hat{y}(k)/d\hat{V}(k) = \Phi(k)^T + \beta^v(k) \hat{A}(k)$. In (4.8), $\beta^v(k)$ is a hybrid adaptive learning rate

$$\begin{cases} \beta^v(k) = 1 & \text{if } \Phi(k)^T(\delta I + \Phi(k)\Phi(k)^T)^{-1}\hat{A}(k)^T \geq 0 \\ \beta^v(k) = 0 & \text{if } \Phi(k)^T(\delta I + \Phi(k)\Phi(k)^T)^{-1}\hat{A}(k)^T < 0 \end{cases} \quad (4.11)$$

where I is the unit matrix and δ is a small constant to guarantee a full rank of the matrix $\delta I + \Phi(k)\Phi(k)^T$ [89].

Vector $\Phi'(k) \in R^{m \times 1}$ and Jacobian $D^v(k) \in R^{n \times m}$ are defined as

$$\Phi'(k) = \begin{bmatrix} \phi'(\hat{W}_1(k)\hat{x}(k)) & \phi'(\hat{W}_2(k)\hat{x}(k)) & \cdots & \phi'(\hat{W}_m(k)\hat{x}(k)) \end{bmatrix}^T \quad (4.12)$$

$$\hat{D}^v(k) = \begin{bmatrix} \frac{\partial u(k)}{\partial \hat{V}(k)} & \cdots & \frac{\partial u(k-k_1+1)}{\partial \hat{V}(k)} & \frac{\partial \hat{y}(k-1)}{\partial \hat{V}(k)} & \cdots & \frac{\partial \hat{y}(k-k_2)}{\partial \hat{V}(k)} \end{bmatrix}^T \quad (4.13)$$

in which $\phi'(\bullet)$ is the derivative of the activation function. $\alpha^v(k)$ is an adaptive learning rate similar to the dead zone approach in the classical RNN and adaptive control systems [84], i.e.,

$$\begin{cases} \alpha^v(k) = 1 & \text{if } |e(k)| \geq \varepsilon_{max}^v / \sqrt{1 - \frac{\|\Phi(k)^T + \beta^v(k)\hat{A}(k)\|^2}{\rho^v(k)}} \\ \alpha^v(k) = 0 & \text{if } |e(k)| < \varepsilon_{max}^v / \sqrt{1 - \frac{\|\Phi(k)^T + \beta^v(k)\hat{A}(k)\|^2}{\rho^v(k)}} \end{cases} \quad (4.14)$$

with $\varepsilon_{max}^v = \max\{|\tilde{\varepsilon}^v(k)|\}$, and $\rho^v(k)$ is the normalization factor to prevent the vanishing radius problem [18] as determined by

$$\rho^v(k) = \nu \rho^v(k-1) + \max\{\bar{\rho}^v, \|\Phi(k)^T + \beta^v(k)\hat{A}(k)\|^2\} \quad (4.15)$$

and $\nu < 1$ and $\bar{\rho}^v$ are positive constants.

The RAGD algorithm uses the specific designed derivative in (4.9), which includes the state estimator in the second term of the partial derivative. To make the proposed algorithm adaptive in a realtime and recurrent fashion, the approximation of $\hat{D}^v(k)$ in (4.13) is calculated following the same idea of the RTRL in [103], such that the partial derivatives between the output and weight can be obtained from the previous steps (normally, the input $u(k)$ is independent from the weight so the

first k_1 rows of $\hat{D}^v(k)$ are zero vectors). To speed up the calculation of the RAGD, we use components of both the standard BP and the N-RTRL for the RAGD when the hybrid adaptive learning rate $\beta^v(k) = 1$ meets the convergence and stability requirements in Theorem 4.1.1 as given below. Furthermore, since we have estimated the best available gradient at each step k , the combined weight and state estimates in (4.9) should provide a relatively deeper local attractor of the nonlinear iteration with the suggested adaptive learning rates.

Theorem 4.1.1. *If the output layer of the RNN is trained by the adaptive normalized gradient algorithm (4.8), the weight $\hat{V}(k)$ is guaranteed to be stable in the sense of Lyapunov*

$$\|\tilde{V}(k+1)\|^2 - \|\tilde{V}(k)\|^2 \leq 0, \quad \forall k$$

with $\tilde{V}(k) = V(k) - V^*$. Also the training will be L_2 -stable in the sense of $e^v(k) \in L_2$ if $\alpha^v(k) \neq 0$ for all $k \in \mathbb{Z}^+$.

Proof: Subtracting V^* and squaring both sides of the equation (4.8)

$$\begin{aligned} & \|\tilde{V}(k+1)\|^2 - \|\tilde{V}(k)\|^2 \\ = & \frac{2\alpha^v(k)e(k)}{\rho^v(k)} \cdot \tilde{V}(k)(\Phi(k)^T + \beta^v(k)\hat{A}(k))^T + \left(\frac{\alpha^v(k)e(k)}{\rho^v(k)}\right)^2 \|\Phi(k)^T + \beta^v(k)\hat{A}(k)\|^2 \\ = & \frac{2\alpha^v(k)e(k)}{\rho^v(k)} \cdot \tilde{V}(k)(\Phi(k) + \beta^v(k)\hat{A}(k)^T) + \left(\frac{\alpha^v(k)e(k)}{\rho^v(k)}\right)^2 \|\Phi(k)^T + \beta^v(k)\hat{A}(k)\|^2 \end{aligned} \quad (4.16)$$

Regarding the first term on the right side of (4.16), we find that it may be easily associated with the term $e^v(k)$ due to the explicit appearance of $\tilde{V}(k)$ and $\Phi(k)$. Following this idea, we need to apply certain transformation to the term $\beta^v(k)\hat{A}(k)^T$ in (4.16), such that $\Phi(k)$ can be extracted from the summation. When it comes to this point, our first thought is to insert the term $\Phi(k)\Phi(k)^T(\Phi(k)\Phi(k)^T)^{-1}$. However, the transformation is not valid because $\Phi(k)\Phi(k)^T$ is not an invertible matrix ($\Phi(k)$ is a column vector). Fortunately, inspired by the approximation method of classical

Gauss-Newton iteration algorithm [27] (pp.126-127), we can employ the approximation $(\delta I + \Phi(k)\Phi(k)^T)^{-1}$, which uses a small positive constant δ to ensure that $\delta I + \Phi(k)\Phi(k)^T$ is positive definite for all k . On this basis, we have the following

$$\begin{aligned}
& \|\tilde{V}(k+1)\|^2 - \|\tilde{V}(k)\|^2 \\
&= \frac{2\alpha^v(k)e(k)}{\rho^v(k)} \cdot \tilde{V}(k)(\Phi(k) + \beta^v(k)\Phi(k)\Phi(k)^T(\delta I + \Phi(k)\Phi(k)^T)^{-1}\hat{A}(k)^T) \\
&\quad + \left(\frac{\alpha^v(k)e(k)}{\rho^v(k)}\right)^2 \|\Phi(k)^T + \beta^v(k)\hat{A}(k)\|^2 \\
&= \frac{2\alpha^v(k)e(k)}{\rho^v(k)} \cdot \tilde{V}(k)\Phi(k)(1 + \beta^v(k)\Phi(k)^T(\delta I + \Phi(k)\Phi(k)^T)^{-1}\hat{A}(k)^T) \\
&\quad + \left(\frac{\alpha^v(k)e(k)}{\rho^v(k)}\right)^2 \|\Phi(k)^T + \beta^v(k)\hat{A}(k)\|^2 \\
&= \frac{2\alpha^v(k)e(k)}{\rho^v(k)} e^v(k)(1 + \beta^v(k)\Phi(k)^T(\delta I + \Phi(k)\Phi(k)^T)^{-1}\hat{A}(k)^T) \\
&\quad + \left(\frac{\alpha^v(k)e(k)}{\rho^v(k)}\right)^2 \|\Phi(k)^T + \beta^v(k)\hat{A}(k)\|^2 \tag{4.17}
\end{aligned}$$

$$\begin{aligned}
&= \frac{2\alpha^v(k)(\tilde{\varepsilon}^v(k)e(k) - e^2(k))}{\rho^v(k)} (1 + \beta^v(k)\Phi(k)^T(\delta I + \Phi(k)\Phi(k)^T)^{-1}\hat{A}(k)^T) \\
&\quad + \left(\frac{\alpha^v(k)e(k)}{\rho^v(k)}\right)^2 \|\Phi(k)^T + \beta^v(k)\hat{A}(k)\|^2 \tag{4.18}
\end{aligned}$$

where (4.18) is obtained by substituting (4.7) into (4.17). Then based on the triangular inequality $2\tilde{\varepsilon}^v(k)e(k) \leq (\tilde{\varepsilon}^v(k))^2 + e^2(k)$, (4.18) can be further deducted as

$$\begin{aligned}
& \|\tilde{V}(k+1)\|^2 - \|\tilde{V}(k)\|^2 \\
&\leq \frac{\alpha^v(k)((\tilde{\varepsilon}^v(k))^2 - e^2(k))}{\rho^v(k)} (1 + \beta^v(k)\Phi(k)^T(\delta I + \Phi(k)\Phi(k)^T)^{-1}\hat{A}(k)^T) \\
&\quad + \left(\frac{\alpha^v(k)}{\rho^v(k)}\right)^2 e^2(k) \|\Phi(k)^T + \beta^v(k)\hat{A}(k)\|^2 \\
&= \frac{\alpha^v(k)}{\rho^v(k)} (1 + \beta^v(k)\Phi(k)^T(\delta I + \Phi(k)\Phi(k)^T)^{-1}\hat{A}(k)^T) ((\tilde{\varepsilon}^v(k))^2 \\
&\quad - (1 - \frac{\alpha^v(k)\|\Phi(k)^T + \beta^v(k)\hat{A}(k)\|^2}{\rho^v(k)(1 + \beta^v(k)\Phi(k)^T(\delta I + \Phi(k)\Phi(k)^T)^{-1}\hat{A}(k)^T)}) e^2(k))
\end{aligned}$$

By the definition of $\beta^v(k)$, we may derive that $\beta^v(k)\Phi(k)^T(\delta I + \Phi(k)\Phi(k)^T)^{-1}\hat{A}(k)^T \geq 0$. Furthermore, because that $\rho^v(k) \geq \|\Phi(k)^T + \beta^v(k)\hat{A}(k)\|^2$ as defined in (4.15) which lead to $1 - \frac{\|\Phi(k)^T + \beta^v(k)\hat{A}(k)\|^2}{\rho^v(k)} > 0$, and by the definition of $\alpha^v(k)$, the convergence of $\tilde{V}(k)$ can be derived

$$\begin{aligned}
& \|\tilde{V}(k+1)\|^2 - \|\tilde{V}(k)\|^2 \\
& \leq \frac{\alpha^v(k)}{\rho^v(k)} (1 + \beta^v(k)\Phi(k)^T(\delta I + \Phi(k)\Phi(k)^T)^{-1}\hat{A}(k)^T) ((\tilde{\varepsilon}^v(k))^2 \\
& \quad - (1 - \frac{\alpha^v(k)\|\Phi(k)^T + \beta^v(k)\hat{A}(k)\|^2}{\rho^v(k)}) e^2(k)) \\
& \leq \frac{\alpha^v(k)}{\rho^v(k)} (1 + \beta^v(k)\Phi(k)^T(\delta I + \Phi(k)\Phi(k)^T)^{-1}\hat{A}(k)^T) ((\varepsilon_{max}^v)^2 \\
& \quad - (1 - \frac{\alpha^v(k)\|\Phi(k)^T + \beta^v(k)\hat{A}(k)\|^2}{\rho^v(k)}) e^2(k)) \\
& \leq 0
\end{aligned} \tag{4.19}$$

Next considering the case that the assumption $\alpha^v(k) \neq 0$ holds for all $k \in Z^+$, we can divide both sides of (4.17) by $2\alpha^v(k)(1 + \beta^v(k)\Phi(k)^T(\delta I + \Phi(k)\Phi(k)^T)^{-1}\hat{A}(k)^T)$ and then sum up to N steps

$$\begin{aligned}
-\Delta V &= \sum_{k=1}^N \left(\frac{e(k)e^v(k)}{\rho^v(k)} + \frac{\alpha^v(k)\|\Phi(k)^T + \beta^v(k)\hat{A}(k)\|^2}{2(\rho^v(k))^2(1 + \beta^v(k)\Phi(k)^T(\delta I + \Phi(k)\Phi(k)^T)^{-1}\hat{A}(k)^T)} e^2(k) \right) \\
&\leq \sum_{k=1}^N (\bar{e}(k)\bar{e}^v(k) + \frac{1}{2}\bar{\sigma}^v(\bar{e}(k))^2) \quad \forall k \in \{k \mid \alpha^v(k) \neq 0\}
\end{aligned} \tag{4.20}$$

where the normalized error signals are defined as

$$\bar{e}(k) = \frac{e(k)}{\sqrt{\rho^v(k)}}, \quad \bar{e}^v(k) = \frac{e^v(k)}{\sqrt{\rho^v(k)}}$$

and the cone satisfies

$$\bar{\sigma}^v = \sup_k \left\{ \frac{\|\Phi(k)^T + \beta^v(k)\hat{A}(k)\|^2}{\rho^v(k)} \right\} < 1$$

which prevents the vanishing radius problem, i.e., $\bar{\sigma}^v$ is strictly smaller than one [18]. Because for each k the Lyapunov function (4.19) is guaranteed smaller or equal to zero, we have

$$\begin{aligned}
0 &\leq \Delta V = -\sum_{k=1}^N \frac{\|\tilde{V}(k+1)\|^2 - \|\tilde{V}(k)\|^2}{2(1 + \beta^v(k)\Phi(k)^T(\delta I + \Phi(k)\Phi(k)^T)^{-1}\hat{A}(k)^T)} \\
&\leq -\frac{1}{2}\sum_{k=1}^N (\|\tilde{V}(k+1)\|^2 - \|\tilde{V}(k)\|^2) \\
&= \frac{1}{2}(\|\tilde{V}(1)\|^2 - \|\tilde{V}(N+1)\|^2)
\end{aligned}$$

Due to the specific selection of the normalization factor in (4.15), the normalized error signals guarantee that the original signals $e(k)$ and $e^v(k)$ are bounded according to the original operators H_1^v and H_2 [18] [87]. Now the operator H_1^v represented by (4.20) satisfies the condition (i) of Lemma 2.3.2, and condition (ii) is guaranteed to hold due to $H_2 = 1$. Thus we conclude that $e^v(k) \in L_2$.

Remark 1. We recognize that the approximation taken in (4.17) sacrifices the accuracy of the error gradient, because $\Phi(k)\Phi(k)^T(\delta I + \Phi(k)\Phi(k)^T)^{-1}$ does not strictly equal, but approach 1. However, the final inequality of (4.19) still holds with the approximation, because we are using the dead zone learning rate as shown in (4.14), which can absorb the approximation error implicitly in the second inequality of (4.19). Also we acknowledge that though such approximation method is inspired by the one of Gauss-Newton iteration algorithm as in [27] (pp.126-127), nevertheless the technical issues in this two cases are not entirely identical, since the matrix $\Phi(k)\Phi(k)^T$ here is always singular and δI must be added all the time, not merely in case of the rank deficiency of the associated matrix. Note the comments are also applicable to the cases of the hidden layer training of the RAGD, as well as the RAGD of MIMO RNN (refer to later sections, see (4.30) (4.50) and (4.51) for example).

Remark 2. The condition of $\alpha^v(k) \neq 0$ for all k in Theorem 4.1.1 is actually a pretty extreme case of the RAGD. It is required because the division of (4.17) by

$2\alpha^v(k)(1 + \beta^v(k)\Phi(k)^T(\delta I + \Phi(k)\Phi(k)^T)^{-1}\hat{A}(k)^T)$ would be invalid for the cases of $\alpha^v(k) = 0$. On the other hand, the problem cannot be solved by simply ignoring those $e^v(k)$ when $\alpha^v(k) = 0$. Because the first term on the right side of (4.20) is not sign definite, hence the summation of the right side of (4.20) cannot yield the final inequality under this framework. Another possible solution of this problem is to project the $e^v(k)$ into a new signal as

$$\begin{cases} e^v(k) & \text{if } \alpha^v(k) \neq 0 \\ 0 & \text{if } \alpha^v(k) = 0 \end{cases}$$

In the mean time, apply the same projection on $e(k)$ and $\tilde{\varepsilon}(k)$ as well, and rewrite (4.20), which represents the nonlinear operator H_1^v , to exclude the cases of $\alpha^v(k) = 0$, then the feedback path H_2 will remain 1 and we can follow similar derivations to obtain the L_2 stability of the new projected closed loop. Subsequently the results in Theorem 4.1.1 can be extended to a more general one. Note the projected closed-loop structure described above is different from the original one in Figure 4.1 in terms that the projected input and output signals belong to subsets of the original ones.

Remark 3. According to the theoretical analysis, the three adaptive parameters $\alpha^v(k)$, $\beta^v(k)$ and $\rho^v(k)$ play important roles in the design of the RAGD. The adaptive learning rate $\alpha^v(k)$ is based on the standard adaptive control system to solve the weight drift problem [84]. The normalization factor $\beta^v(k)$ prevents the so-called vanishing cone problem of the conic sector theorem [18], which has a similar role to the local stability condition as in [47] to bound the gradient in (4.8). The specific designed hybrid adaptive learning rate $\beta^v(k)$ can be further interpreted as activating the recurrent learning mode in case $\Phi(k)^T(\delta I + \Phi(k)\Phi(k)^T)^{-1}\hat{A}(k)^T \geq 0$. It implies that the recurrent training of the RAGD will be active only if the second term of the derivative of (4.9) gives the negative gradient direction, i.e., a relatively deeper local attractor, otherwise the RAGD training procedure will be the same as a static BP algorithm and likely escape this undesired local attractor since it is unfavorable in

the recurrent training. This design is especially effective for accelerating the training of the RNN when the iteration is near the bottom of basin of a local attractor, where the derivatives are changed slowly.

Remark 4. The idea of RAGD is similar to the existing works [69] [78] [103]. If we calculate the derivative in (4.9) exactly by unfolding the recurrent structure and force $\beta^v(k) = 0$, i.e, pursuing all N steps back in the past, then the algorithm will recover the BPTT [78] [102]. Moreover, based on the assumption that the model parameters do not change apparently between each iteration [69], then we can derive a similar approach as the N -RTRL [103]. However, the key difference between the RAGD and the N -RTRL is that we use the hybrid learning rate $\beta^v(k)$ to guarantee the weight convergence and system stability.

4.2 RAGD Training for MISO RNN: Hidden Layer Analysis

Next we present the stability analysis for the hidden layer training of the RAGD for MISO RNN. Apparently the analysis of the hidden layer is more difficult than that of the output layer, because the dynamics between the weight and modeling error is nonlinear due to the activation function. In turn, the derivation of error gradient must be carried out through one layer backward, which involves the derivative of activation function. In fact, this nonlinearity can be avoided by using the mean value theorem as will be shown in this section.

In the proof, we use the the Frobenius norm as weight matrix norm $\|\hat{W}(k)\|_F$, which is defined in Section 2.1.4. A direct benefit of this expression is that the proof and the training equation can be presented in matrix forms, while not in a manner of row by row. However question arises, how to calculate the Jacobian if using the

Frobenius norm. We show in the proof that the problem can be solved by extending the Jacobian into a long vector form on the row basis as defined in (4.27) and (4.28). Now we begin with the expanding of the modeling error around the hidden layer weight as

$$\begin{aligned}
e(k) &= d(k) - \hat{y}(k) + \varepsilon(k) \\
&= V^* \Phi^*(k) - \hat{V}(k) \Phi(k) + \varepsilon(k) \\
&= V^* \Phi^*(k) - \hat{V}(k) \Phi(W^* \hat{x}(k)) + \hat{V}(k) \Phi(W^* \hat{x}(k)) - \hat{V}(k) \Phi(\hat{W}(k) \hat{x}(k)) + \varepsilon(k) \\
&= \hat{V}(k) \Phi(W^* \hat{x}(k)) - \hat{V}(k) \Phi(\hat{W}(k) \hat{x}(k)) + \tilde{\varepsilon}^w(k) \\
&= -\hat{V}_1(k) \mu_1(k) \tilde{W}_1(k) \hat{x}(k) - \hat{V}_2(k) \mu_2(k) \tilde{W}_2(k) \hat{x}(k) - \dots \\
&\quad - \hat{V}_m(k) \mu_m(k) \tilde{W}_m(k) \hat{x}(k) + \tilde{\varepsilon}^w(k) \\
&= -\sum_{i=1}^m \hat{V}_i(k) \mu_i(k) \tilde{W}_i(k) \hat{x}(k) + \tilde{\varepsilon}^w(k) \\
&= -\hat{V}(k) \text{diag}\{\Psi(k)\} \tilde{W}(k) \hat{x}(k) + \tilde{\varepsilon}^w(k)
\end{aligned} \tag{4.21}$$

where $\tilde{\varepsilon}^w(k) = V^* \Phi^*(k) - \hat{V}(k) \Phi(W^* \hat{x}(k)) + \varepsilon(k)$, $\tilde{W}_i(k) \in R^{1 \times n}$ is the vector difference between the i th row of the estimated $W(k)$ and the ideal weight W^* , $\mu_i(k)$ is the mean value of the i th nonlinear activation function, and $\Psi(k)$ is

$$\Psi(k) = [\mu_1(k), \mu_2(k), \dots, \mu_m(k)]^T$$

Defining

$$e^w(k) = \hat{V}(k) \text{diag}\{\Psi(k)\} \tilde{W}(k) \hat{x}(k) \tag{4.22}$$

then equation (4.21) can be simplified as

$$e(k) = -e^w(k) + \tilde{\varepsilon}^w(k) \tag{4.23}$$

Note that we always update the output layer first as similar to the BPTT algorithm and the weight $\hat{V}(k)$ is bounded as proved in last section for the RAGD such that the

error signal $\tilde{\varepsilon}^w(k)$ is also bounded for every step k . Similar to the output layer case, we formulate the closed loop feedback system of hidden layer training via equation (4.21). To differentiate with output layer, the nonlinear dynamics is plotted in Figure 4.2.

Figure 4.2: Closed loop dynamics of hidden layer training of RNN

Since $H_2 = 1$ is inside any cone, thus we only need to study the operator H_1 to analyze the stability of the hidden layer training. Defining the Jacobian $\hat{D}_i^w(k) \in R^{n \times n}$ for the hidden layer

$$\hat{D}_i^w(k) = \begin{bmatrix} \frac{\partial u(k)}{\partial \hat{W}_i(k)} & \cdots & \frac{\partial u(k - k_1 + 1)}{\partial \hat{W}_i(k)} & \frac{\partial \hat{y}(k - k_2)}{\partial \hat{W}_i(k - 1)} & \cdots & \frac{\partial \hat{y}(k - l)}{\partial \hat{W}_i(k - l)} \end{bmatrix}^T \quad (4.24)$$

Then the adaptive normalized gradient training algorithm of the hidden layer is as follows

$$\hat{W}(k+1) = \hat{W}(k) - \frac{\alpha^w(k)}{\rho^w(k)} \cdot \frac{df(e(k))}{d\hat{W}(k)} = \hat{W}(k) + \frac{\alpha^w(k)}{\rho^w(k)} \cdot e(k) \frac{d\hat{y}(k)}{d\hat{W}(k)} \quad (4.25)$$

where the estimated derivative is defined by

$$\begin{aligned} \frac{d\hat{y}(k)}{d\hat{W}(k)} &= \frac{\partial \hat{y}(k)}{\partial \hat{W}(k)} + \beta^w(k) \frac{\partial \hat{y}(k)}{\partial \hat{x}(k)} \frac{\partial \hat{x}(k)}{\partial \hat{W}(k)} \\ &= \text{diag}\{\Phi'(k)\} \hat{V}(k)^T \hat{x}(k)^T + \beta^w(k) \text{diag}\{\Phi'(k)\} \hat{V}(k)^T \underline{\hat{W}}(k) \hat{D}^w(k) \\ &= \text{diag}\{\Phi'(k)\} \hat{V}(k)^T \left(\hat{x}(k)^T + \beta^w(k) \underline{\hat{W}}(k) \hat{D}^w(k) \right) \end{aligned} \quad (4.26)$$

in which $\Phi'(k) \in R^{m \times 1}$, and $\underline{\hat{W}}(k) \in R^{1 \times (m \times n)}$ is a long vector version of the weight

matrix $\hat{W}(k)$

$$\begin{aligned}\Phi'(k) &= [\phi'(\hat{W}_1(k)\hat{x}(k)) \quad \phi'(\hat{W}_2(k)\hat{x}(k)) \quad \cdots \quad \phi'(\hat{W}_m(k)\hat{x}(k))]^T \\ \underline{\hat{W}}(k) &= [\hat{W}_1(k) \quad \hat{W}_2(k) \quad \cdots \quad \hat{W}_m(k)]\end{aligned}\quad (4.27)$$

and the extended Jacobian $D^w(k) \in R^{(m \times n) \times n}$ is

$$\hat{D}^w(k) = [\hat{D}_1^w(k) \quad \hat{D}_2^w(k) \quad \cdots \quad \hat{D}_m^w(k)]^T \quad (4.28)$$

Similarly to the training of output layer, we use notation $\hat{B}(k) \in R^{1 \times n}$ to represent the partial derivative term $\underline{\hat{W}}(k)\hat{D}^w(k)$ to avoid over length of equations, i.e.

$$\hat{B}(k) = \underline{\hat{W}}(k)\hat{D}^w(k) \quad (4.29)$$

The hybrid adaptive learning rate of hidden layer is defined as

$$\begin{cases} \beta^w(k) = 1 & \text{if } \hat{x}(k)^T(\delta I + \hat{x}(k)\hat{x}(k)^T)^{-1}\hat{B}(k)^T \geq 0 \\ \beta^w(k) = 0 & \text{if } \hat{x}(k)^T(\delta I + \hat{x}(k)\hat{x}(k)^T)^{-1}\hat{B}(k)^T < 0 \end{cases} \quad (4.30)$$

where δI is a small perturbation to ensure the matrix $\delta I + \hat{x}(k)\hat{x}(k)^T$ full rank, just same as the approximation method of the output layer training. The normalization factor of hidden layer is defined by

$$\rho^w(k) = \nu \rho^w(k-1) + \max\{\bar{\rho}^w, \frac{\mu_{max} \|\text{diag}\{\Phi'(k)\} \hat{V}(k)^T (\hat{x}(k)^T + \beta^w(k)\hat{B}(k))\|_F^2}{\phi'_{min}(k)}\} \quad (4.31)$$

where $\varepsilon_{max}^w = \max\{|\tilde{\varepsilon}^w(k)|\}$, $\nu < 1$ and $\bar{\rho}^w < 1$ are positive constants, μ_{max} is the maximum value of the activation function, and

$$\phi'_{min}(k) = \min\{\Phi'_1(k) \cdots \Phi'_m(k)\}$$

The adaptive dead zone learning rate $\alpha^w(k)$ of hidden layer is defined by

$$\begin{cases} \alpha^w(k) = 1 & \text{if } |e(k)| \geq \varepsilon_{max}^w / \sqrt{1 - \frac{\mu_{max} \|\text{diag}\{\Phi'(k)\} \hat{V}(k)^T (\hat{x}(k)^T + \beta^w(k) \hat{B}(k))\|_F^2}{\rho^w(k) \phi'_{min}(k)}}} \\ \alpha^w(k) = 0 & \text{if } |e(k)| < \varepsilon_{max}^w / \sqrt{1 - \frac{\mu_{max} \|\text{diag}\{\Phi'(k)\} \hat{V}(k)^T (\hat{x}(k)^T + \beta^w(k) \hat{B}(k))\|_F^2}{\rho^w(k) \phi'_{min}(k)}}} \end{cases} \quad (4.32)$$

Theorem 4.2.1. *The weight matrix $\hat{W}(k)$ is guaranteed to be stable in the sense of Lyapunov*

$$\|\tilde{W}(k+1)\|_F^2 - \|\tilde{W}(k)\|_F^2 \leq 0, \quad \forall k$$

with $\tilde{W}(k) = \hat{W}(k) - W^*$. Also the hidden layer training of the RAGD will be L_2 -stable in the sense of $e^w(k) \in L_2$ if $\alpha^w(k) \neq 0$ for all $k \in Z^+$.

Proof: Subtracting W^* from both sides of (4.25)

$$\tilde{W}(k+1) = \tilde{W}(k) + \frac{\alpha^w(k)}{\rho^w(k)} \cdot e(k) \frac{d\hat{y}(k)}{d\hat{W}(k)} \quad (4.33)$$

Squaring both sides of above equation

$$\begin{aligned} & \tilde{W}(k+1)^T \tilde{W}(k+1) \\ &= \left(\tilde{W}(k) + \frac{\alpha^w(k)}{\rho^w(k)} \cdot e(k) \frac{d\hat{y}(k)}{d\hat{W}(k)} \right)^T \left(\tilde{W}(k) + \frac{\alpha^w(k)}{\rho^w(k)} \cdot e(k) \frac{d\hat{y}(k)}{d\hat{W}(k)} \right) \\ &= \tilde{W}(k)^T \tilde{W}(k) + \frac{\alpha^w(k) e(k)}{\rho^w(k)} \tilde{W}(k)^T \cdot \frac{d\hat{y}(k)}{d\hat{W}(k)} + \frac{\alpha^w(k) e(k)}{\rho^w(k)} \frac{d\hat{y}(k)}{d\hat{W}(k)^T} \cdot \tilde{W}(k) \\ & \quad + \frac{(\alpha^w(k))^2 e^2(k)}{(\rho^w(k))^2} \frac{d\hat{y}(k)}{d\hat{W}(k)^T} \cdot \frac{d\hat{y}(k)}{d\hat{W}(k)} \end{aligned} \quad (4.34)$$

By the definition of the Frobenius norm

$$\begin{cases} \text{Trace}\{\tilde{W}(k+1)^T \tilde{W}(k+1)\} = \|\tilde{W}(k+1)\|_F^2 \\ \text{Trace}\{\tilde{W}(k)^T \tilde{W}(k)\} = \|\tilde{W}(k)\|_F^2 \\ \text{Trace}\left\{\frac{d\hat{y}(k)}{d\hat{W}^T(k)} \cdot \frac{d\hat{y}(k)}{d\hat{W}(k)}\right\} = \left\|\frac{d\hat{y}(k)}{d\hat{W}(k)}\right\|_F^2 = \|\text{diag}\{\Phi'(k)\} \hat{V}(k)^T (\hat{x}(k)^T + \beta^w(k) \hat{B}(k))\|_F^2 \\ \text{Trace}\left\{\tilde{W}(k)^T \cdot \frac{d\hat{y}(k)}{d\hat{W}(k)}\right\} = \text{Trace}\left\{\frac{d\hat{y}(k)}{d\hat{W}(k)^T} \cdot \tilde{W}(k)\right\} \end{cases}$$

where $Trace\{\bullet\}$ function is defined in equation (2.7), then we can derive the following equation

$$\begin{aligned} \|\tilde{W}(k+1)\|_F^2 - \|\tilde{W}(k)\|_F^2 &= \frac{2\alpha^w(k)e(k)}{\rho^w(k)} Trace\left\{\frac{d\hat{y}(k)}{d\hat{W}(k)^T}\tilde{W}(k)\right\} \\ &+ \frac{(\alpha^w(k))^2 e^2(k)}{(\rho^w(k))^2} \cdot \|diag\{\Phi'(k)\}\hat{V}(k)^T(\hat{x}(k)^T + \beta^w(k)\hat{B}(k))\|_F^2 \end{aligned} \quad (4.35)$$

Using the trace properties, the first term on the right side of (4.35) can be transformed as

$$\begin{aligned} &e(k)Trace\left\{\frac{d\hat{y}(k)}{d\hat{W}(k)^T}\tilde{W}(k)\right\} \\ &= e(k)Trace\left\{\left(\hat{x}(k) + \beta^w(k)\hat{B}(k)^T\right)\hat{V}(k)diag\{\Phi'(k)\}\tilde{W}(k)\right\} \\ &= e(k)Trace\left\{\left(\hat{x}(k)\hat{V}(k)diag\{\Phi'(k)\} + \beta^w(k)\hat{B}(k)^T\hat{V}(k)diag\{\Phi'(k)\}\right)\tilde{W}(k)\right\} \\ &= e(k)Trace\{\hat{x}(k)\hat{V}(k)diag\{\Phi'(k)\}\tilde{W}(k)\} \\ &\quad + e(k)\beta^w(k)Trace\{\hat{B}(k)^T\hat{V}(k)diag\{\Phi'(k)\}\tilde{W}(k)\} \\ &= e(k)Trace\{\hat{V}(k)diag\{\Phi'(k)\}\tilde{W}(k)\hat{x}(k)\} \\ &\quad + e(k)\beta^w(k)Trace\{\hat{V}(k)diag\{\Phi'(k)\}\tilde{W}(k)\hat{B}(k)^T\} \end{aligned}$$

By adding a small constant diagonal matrix δI to $\hat{x}(k)\hat{x}(k)^T$ to make it invertible, which is similar to the perturbation method in output layer training (see page 65–66), we are able to derive the following equations

$$\begin{aligned} &e(k)Trace\left\{\frac{d\hat{y}(k)}{d\hat{W}(k)^T}\tilde{W}(k)\right\} \\ &= e(k)\hat{V}(k)diag\{\Phi'(k)\}\tilde{W}(k)\hat{x}(k) \\ &\quad + e(k)\beta^w(k)V(k)diag\{\Phi'(k)\}\tilde{W}(k)\hat{x}(k)\hat{x}(k)^T(\delta I + \hat{x}(k)\hat{x}(k)^T)^{-1}\hat{B}(k) \\ &= e(k)\hat{V}(k)diag\{\Phi'(k)\}\tilde{W}(k)\hat{x}(k)(1 + \beta^w(k)\hat{x}(k)^T(\delta I + \hat{x}(k)\hat{x}(k)^T)^{-1}\hat{B}(k)^T) \\ &= e(k)\left(\sum_{i=1}^m \hat{V}_i(k)\Phi'_i(k)\tilde{W}_i(k)\hat{x}(k)\right)(1 + \beta^w(k)\hat{x}(k)^T(\delta I + \hat{x}(k)\hat{x}(k)^T)^{-1}\hat{B}(k)^T) \end{aligned} \quad (4.36)$$

Before proceeding, let's consider a RNN with scalar weight $\hat{W}(k)$, the relation of

the local attractor basin of the instantaneous square error against the $\tilde{W}(k)$ can be presented by $-\frac{df(e(k))}{\tilde{W}(k)}\tilde{W}(k) \leq 0, \forall k$, as illustrated in Figure 4.3 [84]. Extending this result to the RNN with a matrix weight $\hat{W}(k)$, we have a similar presentation by the local attractor basin concept

$$-\frac{df(e(k))}{\hat{W}_i(k)^T}\tilde{W}_i(k) \leq 0, \quad \forall k, i \quad (4.37)$$

Figure 4.3: Illustration of a local attractor basin of the RNN against a scalar estimated weight $\hat{W}(k)$

Because of the local attractor basin properties as stated in (4.37)

$$\begin{aligned} & e(k)(\hat{V}_i(k)\Phi'_i(k)\tilde{W}_i(k)\hat{x}(k))(1 + \beta^w(k)\hat{x}(k)^T(\delta I + \hat{x}(k)\hat{x}(k)^T)^{-1}\hat{B}(k)^T) \\ &= e(k)\frac{d\hat{y}(k)}{\hat{W}_i(k)^T}\tilde{W}_i(k) = -\frac{df(e(k))}{\hat{W}_i(k)^T}\tilde{W}_i(k) \leq 0 \end{aligned} \quad (4.38)$$

Thus we can enlarge the right side of (4.36) as

$$\begin{aligned} & e(k)\left(\sum_{i=1}^m \frac{\Phi'_i(k)}{\mu_i(k)} \cdot \hat{V}_i(k)\mu_i(k)\tilde{W}_i(k)\hat{x}(k)\right)(1 + \beta^w(k)\hat{x}(k)^T(\delta I + \hat{x}(k)\hat{x}(k)^T)^{-1}\hat{B}(k)^T) \\ & \leq e(k)\left(\frac{\phi'_{min}(k)}{\mu_{max}}\right) \cdot \left(\sum_{i=1}^m \hat{V}_i(k)\mu_i(k)\tilde{W}_i(k)\hat{x}(k)\right)(1 + \beta^w(k)\hat{x}(k)^T(\delta I + \hat{x}(k)\hat{x}(k)^T)^{-1}\hat{B}(k)^T) \\ & = \frac{\phi'_{min}(k)}{\mu_{max}} \cdot e(k)e^w(k)(1 + \beta^w(k)\hat{x}(k)^T(\delta I + \hat{x}(k)\hat{x}(k)^T)^{-1}\hat{B}(k)^T) \end{aligned} \quad (4.39)$$

Substituting (4.39) into (4.35)

$$\begin{aligned}
& \|\tilde{W}(k+1)\|_F^2 - \|\tilde{W}(k)\|_F^2 \\
& \leq \frac{2\alpha^w(k)\phi'_{min}(k)e(k)e^w(k)}{\rho^w(k)\mu_{max}}(1 + \beta^w(k)\hat{x}(k)^T(\delta I + \hat{x}(k)\hat{x}(k)^T)^{-1}\hat{B}(k)^T) \\
& \quad + \frac{(\alpha^w(k))^2 e^2(k)}{(\rho^w(k))^2} \cdot \|diag\{\Phi'(k)\}\hat{V}(k)^T(\hat{x}(k)^T + \beta^w(k)\hat{B}(k))\|_F^2 \quad (4.40)
\end{aligned}$$

Next, similar to the output layer analysis, substituting (4.23) into (4.40)

$$\begin{aligned}
& \|\tilde{W}(k+1)\|_F^2 - \|\tilde{W}(k)\|_F^2 \\
& \leq \frac{2\alpha^w(k)\phi'_{min}(k)(\tilde{\varepsilon}^w(k)e(k) - e^2(k))}{\rho^w(k)\mu_{max}}(1 + \beta^w(k)\hat{x}(k)^T(\delta I + \hat{x}(k)\hat{x}(k)^T)^{-1}\hat{B}(k)^T) \\
& \quad + \frac{(\alpha^w(k))^2 e^2(k)}{(\rho^w(k))^2} \cdot \|diag\{\Phi'(k)\}\hat{V}(k)^T(\hat{x}(k)^T + \beta^w(k)\hat{B}(k))\|_F^2 \\
& \leq \frac{\alpha^w(k)\phi'_{min}(k)((\tilde{\varepsilon}^w(k))^2 - e^2(k))}{\rho^w(k)\mu_{max}}(1 + \beta^w(k)\hat{x}(k)^T(\delta I + \hat{x}(k)\hat{x}(k)^T)^{-1}\hat{B}(k)^T) \\
& \quad + \frac{(\alpha^w(k))^2 e^2(k)}{(\rho^w(k))^2} \cdot \|diag\{\Phi'(k)\}\hat{V}(k)^T(\hat{x}(k)^T + \beta^w(k)\hat{B}(k))\|_F^2 \\
& = \frac{\alpha^w(k)\phi'_{min}(k)}{\rho^w(k)\mu_{max}}(1 + \beta^w(k)\hat{x}(k)^T(\delta I + \hat{x}(k)\hat{x}(k)^T)^{-1}\hat{B}(k)^T)((\tilde{\varepsilon}^w(k))^2 \\
& \quad - (1 - \frac{\alpha^w(k)\mu_{max}\|diag\{\Phi'(k)\}\hat{V}(k)^T(\hat{x}(k)^T + \beta^w(k)\hat{B}(k))\|_F^2}{\rho^w(k)\phi'_{min}(k)(1 + \beta^w(k)\hat{x}(k)^T(\delta I + \hat{x}(k)\hat{x}(k)^T)^{-1}\hat{B}(k)^T)})e^2(k)) \\
& \leq \frac{\alpha^w(k)\phi'_{min}(k)}{\rho^w(k)\mu_{max}}(1 + \beta^w(k)\hat{x}(k)^T(\delta I + \hat{x}(k)\hat{x}(k)^T)^{-1}\hat{B}(k)^T)((\varepsilon_{max}^w)^2 \\
& \quad - (1 - \frac{\alpha^w(k)\mu_{max}\|diag\{\Phi'(k)\}\hat{V}(k)^T(\hat{x}(k)^T + \beta^w(k)\hat{B}(k))\|_F^2}{\rho^w(k)\phi'_{min}(k)})e^2(k)) \quad (4.41)
\end{aligned}$$

By the definition of $\rho^w(k)$ and $\alpha^w(k)$ in (4.31) and (4.32) respectively, we can draw that

$$\|\tilde{W}(k+1)\|_F^2 - \|\tilde{W}(k)\|_F^2 \leq 0 \quad (4.42)$$

Again, consider the extreme case with the assumption of nonzero $\alpha^w(k)$. If dividing both sides of (4.40) by

$$\frac{2\alpha^w(k)\phi'_{min}(k)}{\mu_{max}}(1 + \beta^w(k)\hat{x}(k)^T(\delta I + \hat{x}(k)\hat{x}(k)^T)^{-1}\hat{B}(k)^T)$$

and then summing up to N steps

$$\begin{aligned}
-\Delta W &\leq \sum_{k=1}^N \left(\frac{e(k)e^w(k)}{\rho^w(k)} + \frac{\alpha^w(k)\mu_{max}\|diag\{\Phi'(k)\}\hat{V}(k)^T(\hat{x}(k)^T + \beta^w(k)\hat{B}(k))\|_F^2}{2(\rho^w(k))^2\phi'_{min}(k)(1+\beta^w(k)\hat{x}(k)^T(\delta I + \hat{x}(k)\hat{x}(k)^T)^{-1}\hat{B}(k)^T)}e^2(k) \right) \\
&\leq \sum_{k=1}^N \{\bar{e}(k)\bar{e}^w(k) + \frac{1}{2}\bar{\sigma}^w\bar{e}^2(k)\}
\end{aligned} \tag{4.43}$$

where the normalized error signals are $\bar{e}(k) = e(k)/\sqrt{\rho^w(k)}$, $\bar{e}^w(k) = e^w(k)/\sqrt{\rho^w(k)}$, and the cone is

$$\bar{\sigma}^w = \sup_k \left\{ \frac{\mu_{max}\|diag\{\Phi'(k)\}\hat{V}(k)^T(\hat{x}(k)^T + \beta^w(k)\hat{B}(k))\|_F^2}{\rho^w(k)\phi'_{min}(k)} \right\} < 1 \tag{4.44}$$

and ΔW is greater than zero because for each k the Lyapunov function (4.42) is guaranteed smaller than or equal to zero, i.e.

$$\begin{aligned}
0 &\leq \Delta W = -\sum_{k=1}^N \frac{\mu_{max}(\|\tilde{W}(k+1)\|_F^2 - \|\tilde{W}(k)\|_F^2)}{2\phi'_{min}(k)(1 + \beta^w(k)\hat{x}(k)^T(\delta I + \hat{x}(k)\hat{x}(k)^T)^{-1}\hat{B}(k)^T)} \\
&\leq \frac{\mu_{max}}{2\min\{\phi'_{min}(k)\}} \sum_{k=1}^N (\|\tilde{W}(k)\|_F^2 - \|\tilde{W}(k+1)\|_F^2) \\
&= \frac{\mu_{max}}{2\min\{\phi'_{min}(k)\}} (\|\tilde{W}(1)\|_F^2 - \|\tilde{W}(N+1)\|_F^2)
\end{aligned} \tag{4.45}$$

Due to the specific selection of the normalization factor in (4.31), the original signals $e(k)$ and $e^w(k)$ are guaranteed to be bounded according to the original operators H_1^w and H_2 [18] [87]. Now the operator H_1^w represented by (4.35) satisfies the condition (i) of Lemma 2.3.2. Thus we conclude $e^w(k) \in L_2$ in case of $\alpha^w(k) \neq 0, \forall k \in Z^+$.

The flow chart of the overall RAGD training procedure for MISO RNN is summarized in Figure 4.4.

4.3 RAGD Training for MIMO RNN

In this section, we discuss the RAGD training for Multi-Input Multi-Output (MIMO) types of RNN. A MIMO RNN has a same form as MISO RNN in mathematics, as

Figure 4.4: Flow chart of the RAGD algorithm for MISO RNN

shown in (4.1). However, the two types of RNN have different expressions in the output layer weight. Assume there are l output neurons in MIMO RNN, then output layer weight should be a $l \times m$ dimensional matrix. As a matter of fact, a MIMO RNN can be regarded as consisting of l MISO RNN. Thus the training of MIMO RNN can be studied by decomposition. In detail, for the output layer training, we may calculate the gradient of each output neuron with respect to weight parameters, and then obtain the total weight updating by summing these individual gradient. As for the hidden layer, we also use this method to take into account the influence of multi-output neurons on total weight updating. Following the idea, the extension from MISO to MIMO is straight forward. Now, consider the RNN of (4.1) with

$\hat{V}(k) \in R^{l \times m}$ and $\hat{W}(k) \in R^{m \times n}$. We introduce the following RAGD algorithm

$$\begin{cases} \hat{V}(k+1) = \hat{V}(k) + \frac{\alpha^v(k)}{\rho^v(k)} e(k) (\Phi(k)^T + \beta^v(k) \hat{A}(k)) \\ \hat{W}(k+1) = \hat{W}(k) + \frac{\alpha^w(k)}{\rho^w(k)} \text{diag}\{\Phi'(k)\} \hat{V}(k)^T e(k) (\hat{x}(k)^T + \beta^w(k) \hat{B}(k)) \end{cases} \quad (4.46)$$

where $\Phi'(k)$ is the derivative of activation function vector, $\alpha^v(k)$, $\alpha^w(k)$ are adaptive dead zone learning rates, $\beta^v(k)$, $\beta^w(k)$ are hybrid learning rates, $\rho^v(k)$, $\rho^w(k)$ are normalization factors, and $\hat{A}(k)$, $\hat{B}(k)$ are residual error gradients. These variables are determined as follows

(a) $\Phi'(k) \in R^{m \times 1}$

$$\Phi'(k) = [\phi'(\hat{W}_1(k)\hat{x}(k)) \quad \phi'(\hat{W}_2(k)\hat{x}(k)) \quad \cdots \quad \phi'(\hat{W}_m(k)\hat{x}(k))]^T \quad (4.47)$$

(b) $\hat{A}(k) \in R^{1 \times m}$ and $\hat{B}(k) \in R^{1 \times n}$

$$\hat{A}(k) = \underline{\hat{V}}(k) \cdot [\text{diag}\{\Phi'(k)\}]_l \cdot [\hat{W}(k)]_l \cdot \hat{D}^v(k) \quad (4.48)$$

$$\hat{B}(k) = \underline{\hat{W}}(k) \hat{D}^w(k) \quad (4.49)$$

where $[\text{diag}\{\Phi'(k)\}]_l \in R^{(l \times m) \times (l \times m)}$ and $[\hat{W}(k)]_l \in R^{(l \times m) \times (l \times n)}$ are block diagonal matrices with sub-matrix $\text{diag}\{\Phi'(k)\}$ and $\hat{W}(k)$ on the diagonal respectively

$$[\text{diag}\{\Phi'(k)\}]_l = \begin{bmatrix} \text{diag}\{\Phi'(k)\} & & & 0 \\ & \text{diag}\{\Phi'(k)\} & & \\ & & \ddots & \\ 0 & & & \text{diag}\{\Phi'(k)\} \end{bmatrix}$$

$$[\hat{W}(k)]_l = \begin{bmatrix} \hat{W}(k) & & & 0 \\ & \hat{W}(k) & & \\ & & \ddots & \\ 0 & & & \hat{W}(k) \end{bmatrix}$$

and the $\underline{\hat{V}}(k) \in R^{1 \times (l \times m)}$ and $\underline{\hat{W}}(k) \in R^{1 \times (m \times n)}$ are long vector versions of the weight matrices $\hat{V}(k)$ and $\hat{W}(k)$ respectively

$$\begin{aligned}\underline{\hat{V}}(k) &= [\hat{V}_1(k) \quad \hat{V}_2(k) \quad \cdots \quad \hat{V}_l(k)] \\ \underline{\hat{W}}(k) &= [\hat{W}_1(k) \quad \hat{W}_2(k) \quad \cdots \quad \hat{W}_m(k)]\end{aligned}$$

and the Jacobian $\hat{D}^v(k) \in R^{(l \times n) \times m}$ and $\hat{D}^w(k) \in R^{(m \times n) \times n}$

$$\begin{cases} \hat{D}^v(k) = [\hat{D}_1^v(k)^T \quad \hat{D}_2^v(k)^T \quad \cdots \quad \hat{D}_l^v(k)^T]^T \\ \hat{D}^w(k) = [\hat{D}_1^w(k)^T \quad \hat{D}_2^w(k)^T \quad \cdots \quad \hat{D}_m^w(k)^T]^T \end{cases}$$

in which $\hat{D}_i^v(k) = \frac{\partial \hat{x}(k)}{\partial \hat{V}_i(k)} \in R^{n \times m}$, $\hat{D}_i^w(k) = \frac{\partial \hat{x}(k)}{\partial \hat{W}_i(k)} \in R^{n \times n}$ are sub-matrices.

(c) $\beta^v(k)$ and $\beta^w(k)$

$$\beta^v(k) = \text{sgn}\{\Phi(k)^T(\delta I + \Phi(k)\Phi(k)^T)^{-1}\hat{A}(k)^T\} \quad (4.50)$$

$$\beta^w(k) = \text{sgn}\{\hat{x}(k)^T(\delta I + \hat{x}(k)\hat{x}(k)^T)^{-1}\hat{B}(k)^T\} \quad (4.51)$$

where δ is a small positive constant and I is the identity matrix of the same dimension as $\Phi(k)\Phi(k)^T$ in $\beta^v(k)$ and $\hat{x}(k)\hat{x}(k)^T$ in $\beta^w(k)$ respectively.

(d) $\rho^v(k)$ and $\rho^w(k)$

$$\rho^v(k) = \nu \rho^v(k-1) + \max\{\bar{\rho}^v, \|\Phi(k)^T + \beta^v(k)\hat{A}(k)\|^2\} \quad (4.52)$$

$$\rho^w(k) = \nu \rho^w(k-1) + \max\{\bar{\rho}^w, \frac{\mu_{max} \|\text{diag}\{\Phi'(k)\}\hat{V}(k)^T\|_F^2 \cdot \|\hat{x}(k)^T + \beta^w(k)\hat{B}(k)\|^2}{\phi'_{min}(k)}\} \quad (4.53)$$

where $\nu < 1$, $\bar{\rho}^v$ and $\bar{\rho}^w < 1$ are positive constants, μ_{max} is the maximum value of the activation function, and $\phi'_{min}(k) = \min\{\Phi'_1(k), \dots, \Phi'_m(k)\}$.

(e) $\alpha^v(k)$ and $\alpha^w(k)$

$$\alpha^v(k) = \text{sgn}\{\|e(k)\| - \varepsilon_{max}^v / \sqrt{1 - \frac{\|\Phi(k)^T + \beta^v(k)\hat{A}(k)\|^2}{\rho^v(k)}}\} \quad (4.54)$$

$$\alpha^w(k) = \text{sgn}\{\|e(k)\| - \varepsilon_{max}^w / \sqrt{1 - \frac{\mu_{max}\|\text{diag}\{\Phi'(k)\}\hat{V}(k)^T\|_F^2 \cdot \|\hat{x}(k)^T + \beta^w(k)\hat{B}(k)\|^2}{\phi'_{min}(k) \cdot \rho^w(k)}}\} \quad (4.55)$$

where $\varepsilon_{max}^v = \max\{\|\tilde{\varepsilon}^v(k)\|\}$, $\varepsilon_{max}^w = \max\{\|\tilde{\varepsilon}^w(k)\|\}$, and $\text{sgn}(\bullet)$ function is defined by

$$\text{sgn}(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases} \quad (4.56)$$

Theorem 4.3.1. *If the RNN is trained by the adaptive normalized gradient algorithm (4.46)-(4.55), then the weight $\hat{V}(k)$ and $\hat{W}(k)$ are guaranteed to be stable in the sense of Lyapunov.*

Proof: (i) *Output layer analysis:* To study the stability of the RAGD, we need to establish the error dynamics of the training algorithm. First of all, define the estimation error

$$e^v(k) = \hat{V}(k)\Phi(k) - V^*\Phi(k) = \tilde{V}(k)\Phi(k) \quad (4.57)$$

where $V^* \in R^{l \times m}$ is the ideal output layer weight, and

$$\begin{cases} \tilde{V}(k) = V(k) - V^* \\ \Phi^*(k) = [\cdots h(W_i^*(k)x^*(k)) \cdots]^T \end{cases} \quad (4.58)$$

Expanding $e(k) \in R^{l \times 1}$ with respect to the output layer weight as

$$\begin{aligned} e(k) &= d(k) - \hat{y}(k) + \varepsilon(k) \\ &= [V^*\Phi^*(k) - V^*\Phi(k)] - [\hat{V}(k)\Phi(k) - V^*\Phi(k)] + \varepsilon(k) \\ &= \tilde{\varepsilon}^v(k) - e^v(k) \end{aligned} \quad (4.59)$$

with $\tilde{\varepsilon}^v(k) = V^* \Phi^*(k) - V^* \Phi(k) + \varepsilon(k)$. In (4.59), we restructure the output layer training of the RAGD algorithm into a closed loop form same as that of (2.10) , by which the weight estimation error $e^v(k)$ is referred as the output signal. Next subtracting V^* and squaring both sides of the output layer training equation in (4.46)

$$\begin{aligned} \|\tilde{V}(k+1)\|_F^2 &= \|\tilde{V}(k)\|_F^2 + \frac{2\alpha^v(k)}{\rho^v(k)} \text{Trace}\{(\Phi(k)^T + \beta^v(k)\hat{A}(k))^T e(k)^T \tilde{V}(k)\} \\ &\quad + \left(\frac{\alpha^v(k)}{\rho^v(k)}\right)^2 \|e(k)\Phi(k)^T + e(k)\beta^v(k)\hat{A}(k)\|_F^2 \end{aligned} \quad (4.60)$$

Note the matrix norm is referred as the Frobenius norm similar to that in Section 4.2. By the matrix trace properties

$$\begin{aligned} &\text{Trace}\{(\Phi(k)^T + \beta^v(k)\hat{A}(k))^T e(k)^T \tilde{V}(k)\} \\ &= \text{Trace}\{\Phi(k)e(k)^T \tilde{V}(k)\} + \beta^v(k) \text{Trace}\{\hat{A}(k)^T e(k)^T \tilde{V}(k)\} \\ &= \text{Trace}\{e(k)^T \tilde{V}(k)\Phi(k)\} + \beta^v(k) \text{Trace}\{e(k)^T \tilde{V}(k)\hat{A}(k)^T\} \\ &= e(k)^T \tilde{V}(k)\Phi(k) + \beta^v(k)e(k)^T \tilde{V}(k)\hat{A}(k)^T \end{aligned}$$

Again, we employ the customary practice by using a small positive perturbation constant δ to make $\delta I + \Phi(k)\Phi(k)^T$ full rank and then apply the approximation as

$$\begin{aligned} &\text{Trace}\{(\Phi(k)^T + \beta^v(k)\hat{A}(k))^T e(k)^T \tilde{V}(k)\} \\ &= e(k)^T \tilde{V}(k)\Phi(k) + \beta^v(k)e(k)^T \tilde{V}(k)\Phi(k)\Phi(k)^T (\delta I + \Phi(k)\Phi(k)^T)^{-1} \hat{A}(k)^T \\ &= e(k)^T e^v(k) + \beta^v(k)e(k)^T e^v(k)\Phi(k)^T (\delta I + \Phi(k)\Phi(k)^T)^{-1} \hat{A}(k)^T \\ &= e(k)^T e^v(k) (1 + \beta^v(k)\Phi(k)^T (\delta I + \Phi(k)\Phi(k)^T)^{-1} \hat{A}(k)^T) \end{aligned} \quad (4.61)$$

Substituting (4.59) and (4.61) into (4.60)

$$\|\tilde{V}(k+1)\|_F^2 - \|\tilde{V}(k)\|_F^2 = \frac{2\alpha^v(k)}{\rho^v(k)} e(k)^T e^v(k) (1 + \beta^v(k)\Phi(k)^T (\delta I + \Phi(k)\Phi(k)^T)^{-1} \hat{A}(k)^T)$$

$$\begin{aligned}
& + \left(\frac{\alpha^v(k)}{\rho^v(k)} \right)^2 \|e(k)\Phi(k)^T + e(k)\beta^v(k)\hat{A}(k)\|_F^2 \\
\leq & \frac{2\alpha^v(k)}{\rho^v(k)} e(k)^T e^v(k) (1 + \beta^v(k)\Phi(k)^T(\delta I + \Phi(k)\Phi(k)^T)^{-1}\hat{A}(k)^T) \\
& + \left(\frac{\alpha^v(k)}{\rho^v(k)} \right)^2 \cdot \|e(k)\|^2 \cdot \|\Phi(k)^T + \beta^v(k)\hat{A}(k)\|^2
\end{aligned} \tag{4.62}$$

$$\begin{aligned}
= & \frac{2\alpha^v(k)}{\rho^v(k)} (e(k)^T \tilde{\varepsilon}^v(k) - \|e(k)\|^2) (1 + \beta^v(k)\Phi(k)^T(\delta I + \Phi(k)\Phi(k)^T)^{-1}\hat{A}(k)^T) \\
& + \left(\frac{\alpha^v(k)}{\rho^v(k)} \right)^2 \cdot \|e(k)\|^2 \cdot \|\Phi(k)^T + \beta^v(k)\hat{A}(k)\|^2
\end{aligned} \tag{4.63}$$

$$\begin{aligned}
\leq & \frac{\alpha^v(k)}{\rho^v(k)} (\|\tilde{\varepsilon}^v(k)\|^2 - \|e(k)\|^2) (1 + \beta^v(k)\Phi(k)^T(\delta I + \Phi(k)\Phi(k)^T)^{-1}\hat{A}(k)^T) \\
& + \left(\frac{\alpha^v(k)}{\rho^v(k)} \right)^2 \cdot \|e(k)\|^2 \cdot \|\Phi(k)^T + \beta^v(k)\hat{A}(k)\|^2
\end{aligned} \tag{4.64}$$

$$\begin{aligned}
= & \frac{\alpha^v(k)}{\rho^v(k)} (1 + \beta^v(k)\Phi(k)^T(\delta I + \Phi(k)\Phi(k)^T)^{-1}\hat{A}(k)^T) (\|\tilde{\varepsilon}^v(k)\|^2 \\
& - (1 - \frac{\alpha^v(k)\|\Phi(k)^T + \beta^v(k)\hat{A}(k)\|^2}{\rho^v(k)(1 + \beta^v(k)\Phi(k)^T(\delta I + \Phi(k)\Phi(k)^T)^{-1}\hat{A}(k)^T)}) \|e(k)\|^2) \\
\leq & \frac{\alpha^v(k)}{\rho^v(k)} (1 + \beta^v(k)\Phi(k)^T(\delta I + \Phi(k)\Phi(k)^T)^{-1}\hat{A}(k)^T) (\|\tilde{\varepsilon}^v(k)\|^2 \\
& - (1 - \frac{\alpha^v(k)\|\Phi(k)^T + \beta^v(k)\hat{A}(k)\|^2}{\rho^v(k)}) \|e(k)\|^2) \\
\leq & \frac{\alpha^v(k)}{\rho^v(k)} (1 + \beta^v(k)\Phi(k)^T(\delta I + \Phi(k)\Phi(k)^T)^{-1}\hat{A}(k)^T) ((\varepsilon_{max}^v)^2 \\
& - (1 - \frac{\alpha^v(k)\|\Phi(k)^T + \beta^v(k)\hat{A}(k)\|^2}{\rho^v(k)}) \|e(k)\|^2)
\end{aligned} \tag{4.65}$$

where (4.63) is obtained by substituting (4.59), (4.64) is because of the triangular inequality $\tilde{\varepsilon}^v(k)^T e(k) \leq (\|\tilde{\varepsilon}^v(k)\|^2 + \|e(k)\|^2)/2$, and (4.65) is due to

$$\beta^v(k)\Phi(k)^T(\delta I + \Phi(k)\Phi(k)^T)^{-1}\hat{A}(k)^T \geq 0$$

which can be derived by the definition of $\beta^v(k)$ in (4.50). Now combining the inequality (4.65) with the definition of $\rho^v(k)$ and $\alpha^v(k)$ in (4.54) and (4.52) respectively, the Lyapunov equation of output layer estimation error can be derived

$$\|\tilde{V}(k+1)\|_F^2 - \|\tilde{V}(k)\|_F^2 \leq 0 \tag{4.66}$$

(ii) *Hidden layer analysis:* Expanding $e(k)$ with respect to the estimation error of hidden layer weight

$$\begin{aligned} e^w(k) &= \hat{V}(k)\Phi(\hat{W}(k)\hat{x}(k)) - \hat{V}(k)\Phi(W^*\hat{x}(k)) \\ &= \sum_{i=1}^l \sum_{j=1}^m e_i(k)\hat{V}_{i,j}(k)\mu_j(k)\tilde{W}_j(k)\hat{x}(k) \end{aligned} \quad (4.67)$$

where $W^*(k) \in R^{m \times n}$ is the ideal hidden layer weight matrices, $x^*(k) \in R^{n \times 1}$ is the ideal state vector, $\mu_i(k)$ is the mean value of the i th nonlinear activation function at instant k , and $\tilde{W}_j(k) = \hat{W}_j(k) - W_j^*$. Using the local attractor basin concept that similar to (4.37), we have

$$\begin{aligned} & \text{Trace}\left\{\left(\hat{x}(k)^T + \beta^w(k)\hat{B}(k)\right)^T e(k)^T \hat{V}(k) \text{diag}\{\Phi'(k)\} \tilde{W}(k)\right\} \\ &= \text{Trace}\{\hat{x}(k)e(k)^T \hat{V}(k) \text{diag}\{\Phi'(k)\} \tilde{W}(k)\} \\ & \quad + \beta^w(k) \text{Trace}\{\hat{B}(k)^T e(k)^T \hat{V}(k) \text{diag}\{\Phi'(k)\} \tilde{W}(k)\} \\ &= \text{Trace}\{e(k)^T \hat{V}(k) \text{diag}\{\Phi'(k)\} \tilde{W}(k) \hat{x}(k)\} \\ & \quad + \beta^w(k) \text{Trace}\{e(k)^T \hat{V}(k) \text{diag}\{\Phi'(k)\} \tilde{W}(k) \hat{B}(k)^T\} \\ &= e(k)^T \hat{V}(k) \text{diag}\{\Phi'(k)\} \tilde{W}(k) \hat{x}(k) \\ & \quad + \beta^w(k) e(k)^T \hat{V}(k) \text{diag}\{\Phi'(k)\} \tilde{W}(k) \hat{B}(k)^T \end{aligned}$$

By adding a small positive constant δ , the matrix $\delta I + \hat{x}(k)\hat{x}(k)^T$ is guaranteed to be full rank such that

$$\begin{aligned} & \text{Trace}\left\{\left(\hat{x}(k)^T + \beta^w(k)\hat{B}(k)\right)^T e(k)^T \hat{V}(k) \text{diag}\{\Phi'(k)\} \tilde{W}(k)\right\} \\ &= e(k)^T \hat{V}(k) \text{diag}\{\Phi'(k)\} \tilde{W}(k) \hat{x}(k) \\ & \quad + \beta^w(k) e(k)^T \hat{V}(k) \text{diag}\{\Phi'(k)\} \tilde{W}(k) \hat{x}(k) \hat{x}(k)^T (\delta I + \hat{x}(k)\hat{x}(k)^T)^{-1} \hat{B}(k)^T \\ &= e(k)^T \hat{V}(k) \text{diag}\{\Phi'(k)\} \tilde{W}(k) \hat{x}(k) \left(1 + \beta^w(k) \hat{x}(k)^T (\delta I + \hat{x}(k)\hat{x}(k)^T)^{-1} \hat{B}(k)^T\right) \\ &= \left(\sum_{i=1}^l \sum_{j=1}^m e_i(k) \hat{V}_{i,j}(k) \Phi'_j(k) \tilde{W}_j(k) \hat{x}(k)\right) \left(1 + \beta^w(k) \hat{x}(k)^T (\delta I + \hat{x}(k)\hat{x}(k)^T)^{-1} \hat{B}(k)^T\right) \end{aligned}$$

$$\begin{aligned}
&= \left(\sum_{i=1}^l \sum_{j=1}^m \frac{\Phi'_j(k)}{\mu_j(k)} e_i(k) \hat{V}_{i,j}(k) \mu_j(k) \tilde{W}_j(k) \hat{x}(k) \right) \left(1 + \beta^w(k) \hat{x}(k)^T (\delta I + \hat{x}(k) \hat{x}(k)^T)^{-1} \hat{B}(k)^T \right) \\
&\leq \frac{\phi'_{min}(k)}{\mu_{max}} \left(\sum_{i=1}^l \sum_{j=1}^m e_i(k) \hat{V}_{i,j}(k) \mu_j(k) \tilde{W}_j(k) \hat{x}(k) \right) (1 + \beta^w(k) \hat{x}(k)^T (\delta I + \hat{x}(k) \hat{x}(k)^T)^{-1} \hat{B}(k)^T) \\
&= \frac{\phi'_{min}(k)}{\mu_{max}} e^w(k)^T e(k) \left(1 + \beta^w(k) \hat{x}(k)^T (\delta I + \hat{x}(k) \hat{x}(k)^T)^{-1} \hat{B}(k)^T \right) \quad (4.68)
\end{aligned}$$

Substituting W^* and squaring both sides of hidden layer training equation of the RAGD, we can derive the Lyapunov function of the hidden layer weight of MIMO RNN based upon (4.46) and (4.68) as

$$\begin{aligned}
&\|\tilde{W}(k+1)\|_F^2 - \|\tilde{W}(k)\|_F^2 \\
&= \frac{2\alpha^w(k)}{\rho^w(k)} \text{Trace}\{(\hat{x}(k)^T + \beta^w(k) \hat{B}(k))^T e(k)^T \hat{V}(k) \text{diag}\{\Phi'(k)\} \tilde{W}(k)\} \\
&\quad + \left(\frac{\alpha^w(k)}{\rho^w(k)} \right)^2 \|\text{diag}\{\Phi'(k)\} \hat{V}(k)^T e(k) (\hat{x}(k)^T + \beta^w(k) \hat{B}(k))\|_F^2 \\
&\leq \frac{2\alpha^w(k) \phi'_{min}(k)}{\rho^w(k) \mu_{max}} e^w(k)^T e(k) (1 + \beta^w(k) \hat{x}(k)^T (\delta I + \hat{x}(k) \hat{x}(k)^T)^{-1} \hat{B}(k)^T) \\
&\quad + \left(\frac{\alpha^w(k)}{\rho^w(k)} \right)^2 \|\text{diag}\{\Phi'(k)\} \hat{V}(k)^T\|_F^2 \cdot \|e(k)\|^2 \cdot \|\hat{x}(k)^T + \beta^w(k) \hat{B}(k)\|^2 \quad (4.69) \\
&= \frac{2\alpha^w(k) \phi'_{min}(k)}{\rho^w(k) \mu_{max}} (1 + \beta^w(k) \hat{x}(k)^T (\delta I + \hat{x}(k) \hat{x}(k)^T)^{-1} \hat{B}(k)^T) (\tilde{\varepsilon}^w(k)^T e(k) - e(k)^T e(k)) \\
&\quad + \left(\frac{\alpha^w(k)}{\rho^w(k)} \right)^2 \|\text{diag}\{\Phi'(k)\} \hat{V}(k)^T\|_F^2 \cdot \|e(k)\|^2 \cdot \|\hat{x}(k)^T + \beta^w(k) \hat{B}(k)\|^2 \\
&\leq \frac{\alpha^w(k) \phi'_{min}(k)}{\rho^w(k) \mu_{max}} (1 + \beta^w(k) \hat{x}(k)^T (\delta I + \hat{x}(k) \hat{x}(k)^T)^{-1} \hat{B}(k)^T) (\|\tilde{\varepsilon}^w(k)\|^2 - \|e(k)\|^2) \\
&\quad + \left(\frac{\alpha^w(k)}{\rho^w(k)} \right)^2 \|\text{diag}\{\Phi'(k)\} \hat{V}(k)^T\|_F^2 \cdot \|e(k)\|^2 \cdot \|\hat{x}(k)^T + \beta^w(k) \hat{B}(k)\|^2 \\
&= \frac{\alpha^w(k) \phi'_{min}(k)}{\rho^w(k) \mu_{max}} (1 + \beta^w(k) \hat{x}(k)^T (\delta I + \hat{x}(k) \hat{x}(k)^T)^{-1} \hat{B}(k)^T) (\|\tilde{\varepsilon}^w(k)\|^2 \\
&\quad - (1 - \frac{\alpha^w(k) \mu_{max} \|\text{diag}\{\Phi'(k)\} \hat{V}(k)^T\|_F^2 \cdot \|\hat{x}(k)^T + \beta^w(k) \hat{B}(k)\|^2}{\rho^w(k) \phi'_{min}(k) (1 + \beta^w(k) \hat{x}(k)^T (\delta I + \hat{x}(k) \hat{x}(k)^T)^{-1} \hat{B}(k)^T)}) \|e(k)\|^2) \\
&\leq \frac{\alpha^w(k) \phi'_{min}(k)}{\rho^w(k) \mu_{max}} (1 + \beta^w(k) \hat{x}(k)^T (\delta I + \hat{x}(k) \hat{x}(k)^T)^{-1} \hat{B}(k)^T) ((\varepsilon_{max}^w)^2 \\
&\quad - (1 - \frac{\alpha^w(k) \mu_{max} \|\text{diag}\{\Phi'(k)\} \hat{V}(k)^T\|_F^2 \cdot \|\hat{x}(k)^T + \beta^w(k) \hat{B}(k)\|^2}{\rho^w(k) \phi'_{min}(k)}) \|e(k)\|^2) \\
&\leq 0 \quad (4.70)
\end{aligned}$$

Summarizing (4.66) and (4.70), we can conclude the proof.

Theorem 4.3.2. *If a MIMO RNN is trained by the adaptive normalized gradient algorithm (4.46)-(4.55), and $\alpha^v(k), \alpha^w(k)$ are nonzero for all $k \in Z^+$, then the training will be L_2 -stable in the sense of $e^v(k), e^w(k) \in L_2$.*

Proof: Respectively, dividing both sides of (4.62) and (4.69) by the following two factors (since $\alpha^v(k), \alpha^w(k) \neq 0$)

$$\left\{ \begin{array}{l} 2\alpha^v(k)(1 + \beta^v(k)\Phi(k)^T(\delta I + \Phi(k)\Phi(k)^T)^{-1}\hat{A}(k)^T) \\ \frac{2\alpha^w(k)\phi'_{min}(k)}{\mu_{max}}(1 + \beta^w(k)\hat{x}(k)^T(\delta I + \hat{x}(k)\hat{x}(k)^T)^{-1}\hat{B}(k)^T) \end{array} \right\} \quad (4.71)$$

and then summing both inequalities up to N steps, we have for the output layer

$$\begin{aligned} -\Delta V_N &= \sum_{k=1}^N \left\{ \frac{e^v(k)^T e(k)}{\rho^v(k)} + \frac{\alpha^v(k)\|\Phi(k)^T + \beta^v(k)\hat{A}(k)\|^2}{2(\rho^v(k))^2(1 + \beta^v(k)\Phi(k)^T(\delta I + \Phi(k)\Phi(k)^T)^{-1}\hat{A}(k)^T)} \|e(k)\|^2 \right\} \\ &\leq \sum_{k=1}^N \left\{ \bar{e}^v(k)^T \bar{e}_v(k) + \frac{1}{2}\bar{\sigma}^v \|\bar{e}_v(k)\|^2 \right\} \end{aligned} \quad (4.72)$$

and for the hidden layer

$$\begin{aligned} -\Delta W_N &\leq \sum_{k=1}^N \left\{ \frac{e^w(k)^T e(k)}{\rho^w(k)} + \frac{\mu_{max}\alpha^w(k)\|\text{diag}\{\Phi'(k)\}\hat{V}(k)^T\|_F^2 \cdot \|\hat{x}(k)^T + \beta^w(k)\hat{B}(k)\|^2}{2(\rho^w(k))^2\phi'_{min}(k)(1 + \beta^w(k)\hat{x}(k)^T(\delta I + \hat{x}(k)\hat{x}(k)^T)^{-1}\hat{B}(k)^T)} \|e(k)\|^2 \right\} \\ &\leq \sum_{k=1}^N \left\{ \bar{e}^w(k)^T \bar{e}_w(k) + \frac{1}{2}\bar{\sigma}^w \|\bar{e}_w(k)\|^2 \right\} \end{aligned} \quad (4.73)$$

where

$$\begin{aligned} 0 \leq \Delta V_N &\leq \frac{1}{2}(\|\tilde{V}(1)\|_F^2 - \|\tilde{V}(N+1)\|_F^2) \\ 0 \leq \Delta W_N &\leq \frac{\mu_{max}}{2\min\{\phi'_{min}(k)\}}(\|\tilde{W}(1)\|_F^2 - \|\tilde{W}(N+1)\|_F^2) \end{aligned}$$

and the normalized signals are defined by

$$\bar{e}_v(k) = \frac{e(k)}{\sqrt{\rho^v(k)}} \quad \bar{e}_w(k) = \frac{e(k)}{\sqrt{\rho^w(k)}}$$

$$\begin{aligned}
\bar{e}^v(k) &= \frac{e^v(k)}{\sqrt{\rho^v(k)}} & \bar{e}^w(k) &= \frac{e^w(k)}{\sqrt{\rho^w(k)}} \\
\bar{\sigma}^v(k) &= \sup_k \left\{ \frac{\|\Phi(k)^T + \beta^v(k)\hat{A}(k)\|^2}{\rho^v(k)(1 + \beta^v(k)\Phi(k)^T(\delta I + \Phi(k)\Phi(k)^T)^{-1}\hat{A}(k)^T)} \right\} \\
\bar{\sigma}^w(k) &= \sup_k \left\{ \frac{\mu_{max}\|diag\{\Phi'(k)\}\hat{V}(k)^T\|_F^2 \cdot \|\hat{x}(k)^T + \beta^w(k)\hat{B}(k)\|^2}{\rho^w(k)\phi'_{min}(k)(1 + \beta^w(k)\hat{x}(k)^T(\delta I + \hat{x}(k)\hat{x}(k)^T)^{-1}\hat{B}(k)^T)} \right\}
\end{aligned}$$

Due to the specific selection of the normalization factor $\rho^v(k)$ and $\rho^w(k)$ as in (4.52) and (4.53), the normalized error signals $\bar{e}_v(k)$, $\bar{e}^v(k)$, $\bar{e}_w(k)$, and $\bar{e}^w(k)$ are guaranteed to be bounded. Now, for each $\hat{V}(k)$ and $\hat{W}(k)$, applying the Cluett's law, we find that the operator H_1^v and H_1^w represented by (4.72) and (4.73) satisfy the condition (i). Further, $H_2 = 1$ ensures condition (ii) holds, thus $e^v(k)$ and $e^w(k)$ are L_2 stable with $\alpha^v(k), \alpha^w(k) \neq 0, \forall k \in Z^+$.

4.4 Summary

In this chapter, the RAGD training algorithm of RNN is investigated. Because conventional gradient type algorithms most likely suffer from slow convergence when dealing with statistically non-stationary inputs, the RAGD introduces a series of new elements in the training to overcome the shortcomings. These modifications include hybrid learning rates, normalization factors, and adaptive dead zone learning rate. Moreover, the robust local stability of the RAGD has been addressed for the three layer MISO RNN based upon the Cluett's law. Then the results are extended to the MIMO type RNN. Theoretical analysis shows that the proposed adaptive learning parameters improve the training performance in terms of a deeper gradient descent direction updating, which leads to a better transient response. Further, compared to BPTT, the RAGD algorithm requires limited backward unfolding, which reduces the computational complexity.

Chapter 5

Applications of RAGD in Realtime Signal Processing

In this chapter, we carry out quantitative studies on the RAGD algorithm via computer simulations. In literature, numerous applications of RNN can be found in a wide range of disciplines due to its promising capability in modeling nonlinear dynamics. See [22] [60] [76] [96] for examples. Among these applications, we choose four of the most representative ones to verify the effectiveness of the RAGD algorithm. In the first application, online adaptive filtering problem is considered. The background signal is corrupted by noise with non-stationary statistics and RNN is employed to recover the original signal by the RAGD training algorithm. Feed-forward Neural Network (FNN) is also implemented in the simulations to provide comparisons with RNN. In the second example we revisit the problem of predicting nonlinear dynamic process. The performance of RNN and the RAGD are evaluated based on one-step prediction in this work, which in fact can be generalized to other sophisticated cases. In the third example, the problems of dynamic system identification in a noisy environment are synthesized. The RAGD is utilized to track the output of a time-varying Hammerstein-Wiener model with the measurements of in-

put and output signals. In the last simulation, we investigate the application of RNN in the area of pattern association. The RAGD is employed to train image patterns as dynamic attractors. In all of these simulations in this chapter, we consider only MISO type RNN. As for MIMO RNN, we will present a comprehensive case study of closed loop control problem in the next chapter. By default, RNN is constructed with 50 hidden neurons, 5 input nodes, and 1 output node. The 5-dimensional input vector consists of current and last sample of time sequence $u(k)$ and RNN output feedback with 1 to 3 steps delay respectively. Both hidden and output layer weights are initialized as uniformly distributed in the interval of $(-1, 1)$. Sigmoid function is chosen as activation function, which is monotonic increasing, and both first and second order differentiable. The function and its first order derivative are given in equation (5.1), including the boundaries

$$\begin{cases} 0 \leq \phi(x) = \frac{1}{1+e^{-\lambda x}} \leq 1 \\ 0 \leq \phi'(x) = \frac{\lambda e^{-\lambda x}}{(1+e^{-\lambda x})^2} \leq \frac{\lambda}{4} \end{cases} \quad (5.1)$$

For the purpose of comparison, in most of the simulations we also provide the results of other training algorithms, such as T-BPTT, RTRL etc. For details, please see the rest sections.

5.1 Adaptive Filtering

In the first example, adaptive filtering problem of RNN is investigated. The signal $d(k)$ is assumed to be corrupted by noise $\varepsilon(k)$ as shown in Figure 5.1, where $d(k)$ consists of two frequency components as 20Hz and 500Hz with amplitudes 1.1 and 0.5 respectively, which can be expressed by

$$d(k) = 1.1 \sin\left(\frac{2\pi f_1}{F_s} k\right) + 0.5 \sin\left(\frac{2\pi f_2}{F_s} k\right) \quad (5.2)$$

in which $f_1 = 20$, $f_2 = 500$ and F_s is sampling frequency. $\varepsilon(k)$ is noise with the statistics varied at time steps $k = 1000$ and $k = 2000$ respectively. The input of RNN is expressed by

$$u(k) = d(k) + \varepsilon(k) \quad (5.3)$$

The design objective is to suppress $\varepsilon(k)$ and recover the original $d(k)$. From the figure, we may see that the spectrum of $d(k)$ is mixed up with noise in the mid frequency range, hence it is difficult to realize the extraction with conventional filter. Further, due to the varying noise statistics, an adaptive filter is preferred rather than a fixed one.

Figure 5.1: Time sequences of signal and noise and respective power spectrums

The simulation runs for 3000 steps and the squared filtering error is chosen as the measure of training performance. To obtain a comparative idea, the filtering results of RTRL algorithm with various fixed learning rate α are also provided. In addition, a FNN is implemented to compare the performance with that of RNN. In analogous to RNN, FNN consists of 50 hidden layer neurons, 5 input nodes, and 1 output node. Differently, the input vector of FNN consists of the current value of the time series

and its 1 to 4 steps delay respectively. A general Backpropagation (BP) algorithm is employed to train the FNN, with the learning parameters carefully tuned such that an optimized learning result is obtained. In order to provide a clear comparison on both transient and steady state performance, the first 100 steps and full 3000 steps of training error are shown separately in Figure 5.2 and 5.3. Moreover, the transient state are plotted in two groups, which correspond to the average over 25 best and worst runs respectively, while the steady state are plotted from the average over 50 runs. The traces of normalization factors $\rho^v(k)$ and $\rho^w(k)$ are displayed in Figure 5.4.

From above figures, we found that the steady state error of FNN is obviously greater than that of RNN. This is possibly because FNN is a static type network, which may not be able to follow the statistics variation in noise as good as RNN. Further, for recurrent type network, the training errors of both the RAGD and the RTRL with small learning rates can achieve convergence. But the transient response speed of RTRL with a small learning rate is poorer than the RAGD. Further, it is worth noting with a smaller learning rate α , the training error of RTRL is likely to be oscillating in steady state, which could be caused by the slow instantaneous response speed under the affect of noise. In contrast, the filtering error of the RAGD tends to be more smooth. This is due to the fact that the RAGD can be trained in a relatively faster speed with deeper local attractors to reduce the transient error.

For readers' reference, learning rates $\alpha^v(k), \alpha^w(k)$ and hybrid learning rates $\beta^v(k), \beta^w(k)$ of the RAGD are displayed in Figure 5.5–5.6 respectively to show the effectiveness of the proposed adaptive learning parameters. The traces of the Frobenius norms of the weight matrices are displayed in Figure 5.7 to show the convergence of the weights with the RAGD training algorithm.

5.2.1: Average of the 25 best transient responses

5.2.2: Average of the 25 worst transient responses

Figure 5.2: Squared training errors of the first 100 steps with the same set of random initializations for different algorithms

Figure 5.3: Squared training errors of full 3000 steps for different algorithms

Figure 5.4: Traces of normalization factors $\rho^v(k)$ and $\rho^w(k)$

Figure 5.5: Traces of learning rate $\alpha^v(k)$ and $\alpha^w(k)$

Figure 5.6: Traces of hybrid adaptive learning rate $\beta^v(k)$ and $\beta^w(k)$

Figure 5.7: Traces of the Frobenius norms of RNN weights with the RAGD training

5.2 Time Series Prediction (Revisited)

In this section, we consider again the simulation of time series prediction, which we have investigated in Section 3.3 using the NARL. Here we are going to present alternative solutions to this same problem by using the RAGD algorithm. Same as that in Section 3.3, the model in (5.4) is employed to generate $\{y(k)\}$

$$y(k+1) = \frac{y(k)}{1+y^2(k)} + u^3(k) \quad (5.4)$$

where $\{u(k)\}$ is white Gaussian input sequence. The traces of the time series for training and evaluation are displayed in Figure 5.8.

We also implement the NARL and N-RTRL ($C=0$) in this simulation for comparison. All the simulations run for 10000 steps. And the training errors are displayed in first 100 steps and full 10000 steps separately as shown in Figure 5.9 and 5.10. Again,

Figure 5.8: Sequences of the time series for training and evaluation

we show the transient state as the average over 25 best and worst runs respectively, and the steady state error is obtained from the average over 50 runs. The traces of the normalization factors, learning rates and hybrid learning rates of the RAGD are shown in Figure 5.11–5.13. The trajectories of the Frobenius norms of RNN weights with the RAGD training are displayed in Figure 5.14.

The results show that the RAGD algorithm is successfully stabilized in the sense that the Frobenius norms of the weights converge. There is no explicit difference between the RAGD and the NARL in convergence speed. Both of them are faster than N-RTRL. Moreover, the RAGD can achieve better steady state error (mean squared training error $5.79e - 3$) than both the NARL and N-RTRL with mean squared training errors $6.67e - 3$ and $8.28e - 3$ respectively.

In this simulation, we also investigate how the training is affected by the number of hidden layer neurons and the exponential factor of activation functions. The statistics with respect to various value of this two parameters are given in table 5.1 and 5.2 respectively. The data are obtained by averaging the results of 50 runs

5.9.1: Average of 25 best transient responses

5.9.2: Average of 25 worst transient responses

Figure 5.9: Squared training errors of the first 100 steps with the same set of random initializations for different algorithms

Figure 5.10: Squared training errors of full 3000 steps for different algorithms

Figure 5.11: Traces of normalization factors $\rho^v(k)$ and $\rho^w(k)$

Figure 5.12: Traces of learning rate $\alpha^v(k)$ and $\alpha^w(k)$

Figure 5.13: Traces of hybrid learning rate $\beta^v(k)$ and $\beta^w(k)$

Figure 5.14: Traces of the Frobenius norms of RNN weights with the RAGD training

(each have 10000 steps). All simulations start with same initial weights, which can make a same starting point of the training error such that we can make a convincing comparison. Same results as those in Section 3.3 (pp.58) can be drawn: the steady state performance may be improved a little bit as the λ increases, while it doesn't change apparently with the neuron number.

exponential factor λ	RAGD		NARL		N-RTRL($C = 0$)	
	Mean	Std	Mean	Std	Mean	Std
1	5.72e-3	1.0e-2	6.61e-3	1.2e-2	8.24e-3	1.4e-2
4	5.79e-3	1.0e-2	6.67e-3	1.3e-2	8.28e-3	1.4e-2
8	5.82e-3	1.0e-2	6.73e-3	1.2e-2	8.29e-3	1.5e-2
12	5.82e-3	1.1e-2	6.75e-3	1.3e-2	8.31e-3	1.2e-2

Table 5.1: Statistics of squared training errors of the RAGD with different λ

number of nodes	RAGD		NARL		N-RTRL($C = 0$)	
	Mean	Std	Mean	Std	Mean	Std
10	5.78e-3	1.0e-2	6.69e-3	1.3e-2	8.30e-3	1.3e-2
20	5.77e-3	1.0e-2	6.64e-3	1.3e-2	8.28e-3	1.4e-2
50	5.79e-3	1.0e-2	6.67e-3	1.3e-2	8.28e-3	1.4e-2
80	5.81e-3	1.0e-2	6.68e-3	1.3e-2	8.29e-3	1.3e-2

Table 5.2: Statistics of squared training errors of the RAGD with different neurons

5.3 Output Tracking of Hammerstein-Wiener Model

In this section, the RAGD is evaluated using a system identification problem. Before proceed to study, we give a brief introduction on the model used to describe the “unknown” plant. It consists of a dynamic linear block followed by a static nonlinearity, which is a so-called Hammerstein-Wiener model. Such a cascaded structure has been successfully applied in modeling various systems, e.g., the sensor saturation, control valves, and channel equalization etc. In this simulation, the model dynamics is supposed to vary with time in terms of the time-varying coefficients of linear part, which can be expressed in a polynomial form as [71]

$$\begin{cases} x(k) = 0.3 \tanh(0.5u(k)) + 0.5 \tanh(0.8u(k)) \\ d(k) = 0.7680x(k) + 0.2872e^{-0.006k}x(k-1) - 0.1147e^{-0.03k}x(k-2) + 0.2140x(k-3) \\ \quad - 0.6435\sin(0.008k)u(k-4) + 0.3548x(k-5) + 0.0197x(k-6) \\ \quad - 0.0201x(k-7) + 0.0954x(k-8) \end{cases} \quad (5.5)$$

The objective of the simulation is to model the plant’s input-output behavior by the RNN. The command signal was given by $u(k)$, and the RNN attempts to emulate the plant output $d(k)$ as close as possible. The estimation error between actual plant output and reference signal $e(k) = d(k) - y(k)$ is fed back to RNN to adjust the weight parameters. One of the most crucial tasks in system identification is

the design of appropriate excitation signals. It is important that the training data cover the entire range of plant operation due to non accurate extrapolation of RNN. In this simulation, Amplitude Modulated Pseudo Random Permutation (AMPRP) sequence are generated as training set, with the data uniformly distributed in the range of $(0,1)$, see Figure 5.15. We have also implemented the Truncated BPTT (T-BPTT) algorithm in simulations. The learning rate $\alpha = 0.05$ (tuned by trial-and-error) was used for T-BPTT. We present the squared training error of first 1000 (transient) and 1000–5000 (steady state) steps separately in Figure 5.16 and 5.17. Results show that the RAGD converges within 200 steps while T-BPTT takes around 1000 steps. In addition, the steady state error of the RAGD is smaller than T-BPTT. Hence we say the RAGD is capable of providing a faster response to the changes of system dynamics. The traces of various parameters of the RAGD are provided in Figure 5.18–5.20.

Figure 5.15: Trace of model input: AMPRP sequence

Figure 5.16: Squared training errors of the first 1000 steps

Figure 5.17: Squared training errors of steady state: 1000–5000 steps

Figure 5.18: Traces of the normalization factors $\rho^v(k)$ and $\rho^w(k)$

Figure 5.19: Traces of learning rate $\alpha^v(k)$ and $\alpha^w(k)$

Figure 5.20: Traces of hybrid learning rate $\beta^v(k)$ and $\beta^w(k)$

5.4 Binary Image Pattern Association

In the fourth simulation, we study the problem of stable equilibrium point learning associated with discrete-time RNN using the RAGD algorithm. In the applications of visual processing and pattern recognition, RNN plays an important role due to the feature of associative memory. Generally, RNN dealing with a static target pattern can be divided into two classes according to how the pattern in the network is expressed [4] [5] [7] [20]: i) the target pattern (input pattern) is given as an initial state of the network or ii) the target pattern is given as a constant input to the network. In both the cases, RNN must be designed such that the state of the network converges ultimately to a locally or globally stable equilibrium point which depends only on the target pattern [25] [44].

The earliest work on neural associative memory was proposed by Hopfield with a well-known RNN for a binary vector pattern [29] [30]. In that model, every mem-

ory vector is an equilibrium point of the dynamic network, and the stability of the equilibrium point is guaranteed by the stable learning process. A dynamic learning algorithm was then proposed by Pineda [74] for continuous time RNN where the analog target pattern is directly stored at an equilibrium point of the network. At the same time, another dynamic learning algorithm was described by Almeida [3]. In order to improve the capability of storing multiple patterns in such an associative memory, a modified algorithm for the dynamic learning process was later developed by Pineda [75]. Two dynamic phenomena in the dynamic learning process were isolated into primitive architectural components which perform the operations of continuous nonlinear transformation and auto-associative recall. The dynamic learning techniques for programming the architectural components were presented in a formalism appropriate for a RNN [75]. This dynamic learning process was actually based on a modification of BPTT due to the involvement of the gradient descent method [67] [68]. Many alternative techniques for storing binary vectors using continuous-time dynamic networks have appeared since then [2] [1]. For example, Sudharsanan and Sunareshan [94] developed a systematic synthesis procedure for constructing a continuous time dynamic neural network in which a given set of analog vectors can be stored as the stable equilibrium points. Marcus et al [61] discussed an associative memory in a so-called analog iterated-map neural network using both the Hebb rule and the pseudo inverse rule. Atiya and Abu-Mostafa [10] recently proposed a new method using the Hopfield continuous-time network, and a set of static weight learning formulations was developed in their paper. For a detail survey of some previous work on the design of associative memories similar to Hopfield continuous-time model, readers may refer to [65].

In this section we discuss the problem of image pattern storage by discrete-time RNN as one of the applications of the RAGD. The work is inspired by an earlier paper of Liang and Gupta [47]. In [47], the authors considered absolute stability of BPTT for a general class of discrete time RNN by the Lyapunov first method. In this work the

RAGD will be incorporated in place of BPTT to develop a stable learning process. To present comparison with precedent works, we implement a similar simulation case of binary pattern association as well as BPTT algorithm, where the target pattern is a 10×10 binary image as shown in the first picture of Figure 5.21. The training of RNN is to store the target pattern directly as a local attractor, i.e., an equilibrium point of RNN. Since the state vector is 100 dimensional (number of pixels in target pattern) and there are no external inputs, RNN is configured with 100 inputs and outputs. As a matter of fact, this structure is analogous to the conventional Hopfield type network. RNN is utilized as an auto-associator and we aim at studying self-organizing behavior with the RAGD training algorithm. In order to demonstrate the changing of the binary image corresponding to the state of RNN during learning process, a filter layer based on sign function is added to observe the RNN output pattern, which represents the binary image at the iterative instant. The training process of the RAGD is shown in Figure 5.21. As mentioned, we also implement the BPTT algorithm to provide comparison. The learning rate for the BPTT is 0.028. Similar to previous sections, this value is obtained by trial-and-error tuning method without violating stability constraint. The changing process of the binary image corresponding to the state vector of RNN is shown in Figure 5.22.

From Figure 5.21 and 5.22, we see that using the RAGD training method, the dynamic learning process is completed within 300 steps, which is superior to the 500 steps of the BPTT algorithm. Further, we provide the squared error during the dynamic learning process of the RAGD and BPTT in Figure 5.23. The results indicate that the convergent process of the BPTT (about 450 iterations) is longer than the RAGD (about 280 iterations).

With these training results, we evaluate the association performance upon a distorted test pattern. The target image pattern is assumed to be disturbed by a white Gaussian noise with the noise level about 40% pixels, as shown in the first picture

Figure 5.21: The binary patterns correspond to the state evolution of RNN during the training process using the RAGD algorithm.

Figure 5.22: The binary patterns correspond to the state evolution of RNN during the training process using the BPTT algorithm.

Figure 5.23: Comparison of the squared error curves between the RAGD and BPTT training procedures.

of Figure 5.24. This image is utilized as initial state of RNN to test the capability of recalling the associative memory. The recovered binary images at each time instant during recalling procedure of the two RNN trained by the RAGD and BPTT are given in Figure 5.24 and 5.25 respectively. The results show that the 10×10 binary pattern is successfully stored as a stable equilibrium point of the RNN by both algorithms. And there is no obvious difference of recall duration between two schemes (both within 10 iterations).

5.5 Summary

We have presented quantitative studies of the proposed RAGD algorithm in this chapter. Computer simulations are synthesized to justify the effectiveness of the RAGD. We give four examples which are the most frequent application areas of RNN: i) Adaptive filtering of colored noise across with target signal; ii) One-step prediction of non-statistical time series, which is generated by benchmark process

Figure 5.24: The binary patterns correspond to the state evolution of association process of RNN trained by the BPTT algorithm.

Figure 5.25: The binary patterns correspond to the state evolution of association process of RNN trained by the RAGD algorithm.

model; iii) Identification of a nonlinear dynamic plant and the training data set is generated by a time-varying Hammerstein-Wiener model; iv) Pattern association of binary images. Further, we provide comprehensive comparisons between the RAGD and various of other algorithms such as the RTRL, the N-RTRL, the T-BPTT, and the BPTT. Specially we compare the performance between RNN and FNN in the first simulation. In most results of these simulations, RNN trained by the RAGD demonstrates explicit advantages in the transient response speed, e.g., see Figure 5.2, 5.9 and 5.16. Some of the results also indicates that the RAGD can achieve better steady state responses, such as those in Figure 5.3 and 5.17. Hence by these experiment results, we conclude that the performance of the RAGD training algorithm of RNN is improved.

Chapter 6

Applications of RAGD in Fault Tolerant Control of Robotic System

In this chapter, we investigate the RAGD algorithm in the application of a 7-link rigid robot tracking control system. A hybrid controller that consists of PD and RNN is proposed, which simultaneously uses a linear controller that provides the basic tracking performance as well as a RNN that compensates uncertainties. Since the robot model has 6 degree-of-freedom (DOF), a MIMO type RNN is employed. Because of the remarkable capability of the RNN in modeling time-behavior of dynamics processes, it is expected that RNN compensation is able to facilitate robotic system to improve the adaptability to the change of working environments [57] [106]. Indeed, this combination method of RNN and conventional linear controller is becoming one of the major research directions in control system design, namely, Fault Tolerant Control (FTC) [13] [85] [95]. In [24] [46] feedforward ANN was integrated into PID controller to improve the tracking performance. In [55] fuzzy neural network was used and robustness was guaranteed by H_∞ design methods. In [33] [34],

ANN is employed in the control of a general class of discrete-time nonlinear systems. The training algorithms are designed to guarantee the robustness of the overall system via the Lyapunov approach.

In our approach, the RNN is utilized in a realtime plug-and-play fashion, i.e., the control effort of RNN is kept standby when there is no fault and turned on when fault occurs. A 7-link robot model is employed as control objective in theoretical analysis. Robust stability of the overall closed loop system is addressed based upon the RAGD algorithm, which is developed in Chapter 4. In consideration of practical applications, all the algorithms are developed in the discrete-time domain, which can benefit from an easy implementation on digital signal processor (DSP). Simulations are presented to justify the advantages of the proposed control scheme. Various comparisons are given, e.g., the standard PD control versus the proposed RNN FTC method, performance of RNN trained by the RAGD versus that by the RTRL, etc. The results show that the RNN trained by the RAGD algorithm can provide the best compensation to unknown system faults, and hence overall closed loop performance can be effectively improved in terms of both transient and steady state response.

6.1 Dynamics of 7-link Robot Model

6.1.1 Physical Structure of the Robot

The rigid biped robot under consideration is shown in Figure 6.1. A detailed description of its structure can be found in [52] [53] [55]. For later reference we would like to give an immediate brief review. The robot consists of 7 links, which represent trunk, thigh, shin, metatarsal respectively. The links are joined together at 6 pin joints: two hip joints, two knee joints and two ankle joints. The six joints

are ideally rotational. Each of them has one degree of freedom and is driven by an independent electric DC motor. For each segment, there are four parameters: the mass of the link— m_i , moment of inertia about the center of gravity (COG)— I_i , the length of the link— l_i , and the distance between the COG and the lower joint— a_i , where $i = 1, \dots, 6$. The motion of the robot is constrained on the sagittal plane such that the total number of degrees of freedom will be limited enough.

Figure 6.1: Structure of a 7-link rigid biped robot

6.1.2 Walking Patterns

In literature the most popular analytical model of human walking can be described as it is performed so as to have the least expenditure of energy. Under this framework, the walking pattern of a biped robot can be roughly divided into three phases: single support phase, double support phase and transition phase. Among them, the single support phase is a predominant portion that its duration is longer than the other two's and the robot body is statically unstable in this period. We focus on this phase in the simulation and use the set of the joint angles θ of each link with the vertical

as controlling objectives. In mathematics, the dynamic equations of non-kick action of above described single-leg-supporting phase can be expressed in an analog model by [45]

$$M(\theta)\ddot{\theta} + C(\theta, \dot{\theta})\dot{\theta} + G(\theta) + D(\theta, \dot{\theta}) + F(\theta, \dot{\theta}, t) = \tau \quad (6.1)$$

where $\dot{\theta}$ and $\ddot{\theta}$ are joint angle velocities and accelerations respectively, $M(\theta) \in R^{6 \times 6}$ is the inertia matrix, $C(\theta, \dot{\theta}) \in R^{6 \times 6}$ is the coriolis/centripetal torque matrix, $G(\theta) \in R^{6 \times 1}$ is gravity vector, $\tau \in R^{6 \times 1}$ denotes the input torque vector, $D(\theta, \dot{\theta}) \in R^{6 \times 1}$ is static and dynamic friction and other disturbance torques, and $F(\theta, \dot{\theta}, t)$ stands for the unknown fault that occurs in robot manipulator. The parameters $M(\theta)$, $C(\theta, \dot{\theta})$, and $G(\theta)$ of robot dynamics can be calculated by

$$\begin{cases} M(\theta) = \{c_{ij} \cos(\theta_i - \theta_j)\} \\ C(\theta) = \{c_{ij} \sin(\theta_i - \theta_j)\} \\ G(\theta) = \{-h_i \sin \theta_i\} \end{cases} \quad (6.2)$$

in which the values of c_{ij} and h_i are defined in Section 6.3. The following assumptions of biped robot dynamics are necessary [88]

Assumption 6.1.1.

(i) The inertia matrix $M(\theta)$ is symmetric, positive definite and both $M(\theta)$ and $M^{-1}(\theta)$ are uniformly upper and lower bounded.

(ii) The inertia matrix $M(\theta)$ is also kinetic energy matrix of the manipulator and the kinetic energy can be written as $\dot{\theta}^T M(\theta) \dot{\theta} / 2$, where superscript T denotes the transpose of the vectors and matrices.

(iii) The matrix $\dot{M}(\theta) - 2C(\theta, \dot{\theta})$ is skew-symmetric.

(iv) The unknown system uncertainty is bounded.

6.1.3 Digital Controller

In this section, we introduce a hybrid controller that integrates both PD and RNN. In functionality, the PD is utilized as basic feedback control for trajectory following and the RNN facilitates to compensate the nonlinearity and system faults [98]. In order to achieve the desired fault accommodation, the controller module must be able to meet the following two requirements: i) In a fault-free condition, the computed torque controller can drive the joint position to track the desired position as closely as possible. ii) When a fault occurs, the RNN controller should be able to accommodate the faults via controller reconfiguration even in the presence of model uncertainties. Bear this in mind, we firstly transform the robot dynamics of equation (6.1) into a nominal form as [45]

$$\ddot{\theta} = M^{-1}(\theta)[\tau - C(\theta, \dot{\theta})\dot{\theta} - G(\theta) - D(\theta, \dot{\theta}) - F(\theta, \dot{\theta}, t)] \quad (6.3)$$

Because the RNN (6.16) is a digital system while the robot (6.3) is modelled in continuous-time domain, we must digitize the robot model before carrying out formal analysis. For this purpose, a sampler is placed on θ to define

$$\theta(k) = \theta(kT_s) \in R^{6 \times 1} \quad (6.4)$$

where T_s represents the sampling period. Following the similar rule, we have

$$M(k) = M(\theta(kT_s)), \quad C(k) = C(\theta(kT_s), \dot{\theta}(kT_s)) \quad (6.5)$$

However, there is no convenient way to digitize the nonlinear dynamics in (6.3), such as derivatives. One of the most popular approaches is to use Euler's rule. By this method, the discrete-time model of the angular speed and angular acceleration

of joint angles can be obtained as

$$\begin{cases} \dot{\theta}(k)T_s = \theta(k) - \theta(k-1) \\ \ddot{\theta}(k)T_s = \dot{\theta}(k) - \dot{\theta}(k-1) \end{cases} \quad (6.6)$$

Substituting (6.6) into (6.3), the robot dynamics is transformed to an input-output framework

$$\begin{aligned} \theta(k+1) = & 2\theta(k) - \theta(k-1) + T_s^2 M^{-1}(\theta(k))[\tau_0(k) + \tau_c(k) \\ & - C(\theta(k), \theta(k-1)) \frac{\theta(k) - \theta(k-1)}{T_s} - G(\theta(k), \theta(k-1)) \\ & - D(\theta(k), \theta(k-1)) - F(\theta(k), \theta(k-1))] \end{aligned} \quad (6.7)$$

Now we can proceed to derive the error dynamics with (6.7). With the feedback signal $e_r(k) = \theta_d(k) - \theta(k)$, the output of PD controller is

$$y_{pd}(k) = (K_P + K_D \frac{z-1}{zT_s})e_r(k) \quad (6.8)$$

Consequently, the computed torque in discrete-time domain can be expressed by

$$\begin{aligned} \tau_0(k) = & M(\theta(k))T_s^{-2}[\theta_d(k+1) - 2\theta_d(k) + \theta_d(k-1) + K_P e_r(k) \\ & + K_D \frac{e_r(k) - e_r(k-1)}{T_s}] + G(\theta(k)) \\ & + C(\theta(k), \theta(k-1)) \frac{\theta(k) - \theta(k-1)}{T_s} \end{aligned} \quad (6.9)$$

$$\tau_c(k) = M(\theta(k))T_s^{-2}\hat{y}_{rnn}(k) \quad (6.10)$$

where $\tau_c(k)$ is the RNN estimation of uncertainties $D(\theta(k), \theta(k-1))$ and $F(\theta(k), \theta(k-1))$. Substituting (6.9) (6.10) into (6.7), the closed loop error dynamics of robot control system become

$$T_s^2 M^{-1}(\theta(k))(D(\theta(k), \theta(k-1)) + F(\theta(k), \theta(k-1))) - \hat{y}_{rnn}(k)$$

$$\begin{aligned}
&= e_r(k+1) - 2e_r(k) + e_r(k-1) + K_P e_r(k) + K_D \frac{e_r(k) - e_r(k-1)}{T_s} \\
&= e_r(k+1) * [1 + (K_P + \frac{K_D}{T_s} - 2)z^{-1} + (1 - \frac{K_D}{T_s})z^{-2}]
\end{aligned} \tag{6.11}$$

where $*$ means convolution. As we can see, in a fault free environment, $D(\theta(k), \theta(k-1))$ and $F(\theta(k), \theta(k-1))$ remains zero. The PD controller alone can provide sufficient control for the robot. However, if faults occur, i.e., the term on the right side of (6.11) is no longer zero, there may not exist suitable K_P and K_D making $e_r(k)$ converge. In this case, we employ RNN to reduce the impact of disturbances and improve the fault tolerance. If we regard $T_s^2 M^{-1}(\theta(k))[D(\theta(k), \theta(k-1)) + F(\theta(k), \theta(k-1))]$ as the desired output $d(k)$ of RNN, then the RNN estimation error $e(k) \in R^{6 \times 1}$ can be defined by

$$e(k) = d(k) - \hat{y}_{rnn}(k) \tag{6.12}$$

Then (6.11) can be transformed as

$$e(k) = e_r(k+1) * [1 + (K_P + \frac{K_D}{T_s} - 2)z^{-1} + (1 - \frac{K_D}{T_s})z^{-2}] \tag{6.13}$$

Further define the operator

$$G(z) = 1 + (K_P + \frac{K_D}{T_s} - 2)z^{-1} + (1 - \frac{K_D}{T_s})z^{-2} \tag{6.14}$$

Then equation (6.13) can be simplified as

$$e_r(k+1) = G^{-1}(z) * e(k) \tag{6.15}$$

To this end, we establish the closed loop error dynamics already. In fact, the operator $G^{-1}(z)$ is an IIR Filter, thus as long as we can ensure all the poles of $G^{-1}(z)$ located in the unit circle in complex z -plane through controller design, and the RNN training error to be convergent with the RAGD training algorithm, then the closed loop

system will also be guaranteed to be convergent. Along the line of this thought, we will present the controller design in the next section as well as the stability analysis. The block diagram of the entire closed loop control system is displayed in Figure 6.2.

Figure 6.2: Block diagram of RNN+PD control scheme

6.2 Closed Loop Stability with RAGD Training

Given a RNN with 24 input nodes (structure shown in Figure 1.2), 6 output nodes, and 50 hidden layer neurons, the output and hidden layer weights of the RNN can be expressed (in matrix form) by $\hat{V}(k) \in R^{6 \times 50}$ and $\hat{W}(k) \in R^{50 \times 24}$ respectively. Upon an input vector $u(k)$, the corresponding RNN output can be expressed as

$$\hat{y}_{rnn}(k) = \hat{V}(k) \Phi(\hat{W}(k) \cdot x(k)) \quad (6.16)$$

where $\Phi(\cdot)$ is the nonlinear activation function, and $x(k) \in R^{24 \times 1}$ is the state vector that consists of 12 external input entries and 12 output feedback entries

$$x(k) = [u^1(k), \dots, u^6(k), u^1(k-1), \dots, u^6(k-1), \hat{y}_{rnn}^1(k-1), \dots, \dots, \hat{y}_{rnn}^6(k-1), \hat{y}_{rnn}^1(k-2), \dots, \hat{y}_{rnn}^6(k-2)]^T \quad (6.17)$$

Superscript i denotes the i th entry of each vector.

Theorem 6.2.1. *If the coefficients of PD controller satisfies*

$$\begin{cases} 0 < K_P < 4 \\ 0 < K_D < 2T_s(\sqrt{K_P} - K_P) \end{cases} \quad (6.18)$$

and RNN is trained by the RAGD algorithm of

$$\begin{cases} \hat{V}(k+1) = \hat{V}(k) + \frac{T_s \alpha^v(k)}{\rho^v(k)} e(k) (\Phi(k)^T + \beta^v(k) \hat{A}(k)) \\ \hat{W}(k+1) = \hat{W}(k) + \frac{T_s \alpha^w(k)}{\rho^w(k)} \text{diag}\{\Phi'(k)\} \hat{V}(k)^T e(k) (\hat{x}(k)^T + \beta^w(k) \hat{B}(k)) \end{cases} \quad (6.19)$$

where T_s is the sampling period, $\Phi'(k)$, $\beta^v(k)$, $\beta^w(k)$, $\hat{A}(k)$, $\hat{B}(k)$ are same as those defined in (4.47)-(4.51), and $\rho^v(k)$, $\rho^w(k)$, $\alpha^v(k)$ and $\alpha^w(k)$ are defined as

$$\rho^v(k) = \nu \rho^v(k-1) + \max\{\bar{\rho}^v, T_s \|\Phi(k)^T + \beta^v(k) \hat{A}(k)\|^2\} \quad (6.20)$$

$$\rho^w(k) = \nu \rho^w(k-1) + \max\{\bar{\rho}^w, \frac{T_s \mu_{max} \|\text{diag}\{\Phi'(k)\} \hat{V}(k)^T\|_F^2 \|\hat{x}(k)^T + \beta^w(k) \hat{B}(k)\|^2}{\phi'_{min}(k)}\} \quad (6.21)$$

$$\alpha^v(k) = \text{sgn}\{\|e(k)\| - \varepsilon_{max}^v / \sqrt{1 - \frac{T_s \|\Phi(k)^T + \beta^v(k) \hat{A}(k)\|^2}{\rho^v(k)}}\} \quad (6.22)$$

$$\alpha^w(k) = \text{sgn}\{\|e(k)\| - \varepsilon_{max}^w / \sqrt{1 - \frac{T_s \mu_{max} \|\text{diag}\{\Phi'(k)\} \hat{V}(k)^T\|_F^2 \|\hat{x}(k)^T + \beta^w(k) \hat{B}(k)\|^2}{\phi'_{min}(k) \cdot \rho^w(k)}}\} \quad (6.23)$$

with $\text{sgn}(\bullet)$ being the same as (4.56), then the closed loop control system is stable in the sense that $\|e(k)\|$ and $\|e_r(k)\|$ are bounded input bounded output stable.

Proof: The inequality (6.18) can be transformed as

$$0 < \frac{K_D}{T_s} < 1 - (\sqrt{K_P} - 2)^2 < 1 \quad (6.24)$$

Then we can derive the following inequalities

$$0 < 1 - \frac{K_D}{T_s} < 1 \quad (6.25)$$

$$-2 < 2 - \frac{K_D}{T_s} - K_P < 2 \quad (6.26)$$

By solving the characteristic equation of $G^{-1}(z)$, which is defined in (6.14), we can derive that all the poles lie inside the unit circle with (6.25) and (6.26). This in fact means $G^{-1}(z)$ is open loop stable. As for RNN modeling error $e(k)$, we apply the following decomposition

$$\begin{aligned} e(k) &= V^* \Phi^* - \hat{V}(k) \Phi(k) \\ &= V^* \Phi^* - \hat{V}(k) \Phi^* + V^* \Phi^* - V^* \Phi(k) \\ &\quad + V^* \Phi(k) + \hat{V}(k) \Phi^* - V^* \Phi^* - \hat{V}(k) \Phi(k) \\ &= \tilde{V}(k) \Phi^* + V^* \tilde{\Phi}(k) + \tilde{V}(k) \tilde{\Phi}(k) \end{aligned} \quad (6.27)$$

For the output layer estimation error, the Lyapunov function can be constructed as

$$\begin{aligned} &\|\tilde{V}(k+1)\|_F^2 - \|\tilde{V}(k)\|_F^2 \\ &= \frac{2T_s \alpha^v(k)}{\rho^v(k)} e(k)^T e^v(k) (1 + \beta^v(k) \Phi(k)^T (\delta I + \Phi(k) \Phi(k)^T)^{-1} \hat{A}(k)^T) \\ &\quad + \left(\frac{T_s \alpha^v(k)}{\rho^v(k)} \right)^2 \|e(k) \Phi(k)^T + e(k) \beta^v(k) \hat{A}(k)\|_F^2 \\ &\leq \frac{2T_s \alpha^v(k)}{\rho^v(k)} e(k)^T e^v(k) (1 + \beta^v(k) \Phi(k)^T (\delta I + \Phi(k) \Phi(k)^T)^{-1} \hat{A}(k)^T) \\ &\quad + \left(\frac{T_s \alpha^v(k)}{\rho^v(k)} \right)^2 \cdot \|e(k)\|^2 \cdot \|\Phi(k)^T + \beta^v(k) \hat{A}(k)\|^2 \\ &= \frac{2T_s \alpha^v(k)}{\rho^v(k)} (e(k)^T \tilde{\varepsilon}^v(k) - \|e(k)\|^2) (1 + \beta^v(k) \Phi(k)^T (\delta I + \Phi(k) \Phi(k)^T)^{-1} \hat{A}(k)^T) \\ &\quad + \left(\frac{T_s \alpha^v(k)}{\rho^v(k)} \right)^2 \cdot \|e(k)\|^2 \cdot \|\Phi(k)^T + \beta^v(k) \hat{A}(k)\|^2 \\ &\leq \frac{T_s \alpha^v(k)}{\rho^v(k)} (\|\tilde{\varepsilon}^v(k)\|^2 - \|e(k)\|^2) (1 + \beta^v(k) \Phi(k)^T (\delta I + \Phi(k) \Phi(k)^T)^{-1} \hat{A}(k)^T) \end{aligned} \quad (6.28)$$

$$\begin{aligned}
& + \left(\frac{T_s \alpha^v(k)}{\rho^v(k)} \right)^2 \cdot \|e(k)\|^2 \cdot \|\Phi(k)^T + \beta^v(k) \hat{A}(k)\|^2 \\
& = \frac{T_s \alpha^v(k)}{\rho^v(k)} (1 + \beta^v(k) \Phi(k)^T (\delta I + \Phi(k) \Phi(k)^T)^{-1} \hat{A}(k)^T) (\|\tilde{\varepsilon}^v(k)\|^2 \\
& \quad - (1 - \frac{T_s \alpha^v(k) \|\Phi(k)^T + \beta^v(k) \hat{A}(k)\|^2}{\rho^v(k) (1 + \beta^v(k) \Phi(k)^T (\delta I + \Phi(k) \Phi(k)^T)^{-1} \hat{A}(k)^T)}) \|e(k)\|^2) \\
& \leq \frac{T_s \alpha^v(k)}{\rho^v(k)} (1 + \beta^v(k) \Phi(k)^T (\delta I + \Phi(k) \Phi(k)^T)^{-1} \hat{A}(k)^T) (\|\tilde{\varepsilon}^v(k)\|^2 \\
& \quad - (1 - \frac{T_s \alpha^v(k) \|\Phi(k)^T + \beta^v(k) \hat{A}(k)\|^2}{\rho^v(k)}) \|e(k)\|^2) \\
& \leq \frac{T_s \alpha^v(k)}{\rho^v(k)} (1 + \beta^v(k) \Phi(k)^T (\delta I + \Phi(k) \Phi(k)^T)^{-1} \hat{A}(k)^T) ((\varepsilon_{max}^v)^2 \\
& \quad - (1 - \frac{T_s \alpha^v(k) \|\Phi(k)^T + \beta^v(k) \hat{A}(k)\|^2}{\rho^v(k)}) \|e(k)\|^2) \\
& \leq 0
\end{aligned} \tag{6.29}$$

$$\tag{6.30}$$

where (6.28) is obtained by adding a small positive constant δI to $\Phi(k) \Phi(k)^T$ to ensure the positiveness of the matrix (see Chapter 4, page 65–66 for more details), (6.29) is derived on basis of the triangular inequality, and (6.30) is derived according to the definition of $\alpha^v(k)$ in (6.22). Similarly, we can prove the convergence of the estimation error of the hidden layer weight by constructing the Lyapunov function as

$$\begin{aligned}
& \|\tilde{W}(k+1)\|_F^2 - \|\tilde{W}(k)\|_F^2 \\
& = \frac{2T_s \alpha^w(k)}{\rho^w(k)} \text{Trace}\{(\hat{x}(k)^T + \beta^w(k) \hat{B}(k))^T e(k)^T \hat{V}(k) \text{diag}\{\Phi'(k)\} \tilde{W}(k)\} \\
& \quad + \left(\frac{T_s \alpha^w(k)}{\rho^w(k)} \right)^2 \|\text{diag}\{\Phi'(k)\} \hat{V}(k)^T e(k) (\hat{x}(k)^T + \beta^w(k) \hat{B}(k))\|_F^2 \\
& \leq \frac{2T_s \alpha^w(k) \phi'_{min}(k)}{\rho^w(k) \mu_{max}} e^w(k)^T e(k) (1 + \beta^w(k) \hat{x}(k)^T (\delta I + \hat{x}(k) \hat{x}(k)^T)^{-1} \hat{B}(k)^T) \\
& \quad + \left(\frac{T_s \alpha^w(k)}{\rho^w(k)} \right)^2 \|\text{diag}\{\Phi'(k)\} \hat{V}(k)^T\|_F^2 \cdot \|e(k)\|^2 \cdot \|\hat{x}(k)^T + \beta^w(k) \hat{B}(k)\|^2 \\
& = \frac{2T_s \alpha^w(k) \phi'_{min}(k)}{\rho^w(k) \mu_{max}} (1 + \beta^w(k) \hat{x}(k)^T (\delta I + \hat{x}(k) \hat{x}(k)^T)^{-1} \hat{B}(k)^T) (\tilde{\varepsilon}^w(k)^T e(k) - e(k)^T e(k)) \\
& \quad + \left(\frac{T_s \alpha^w(k)}{\rho^w(k)} \right)^2 \|\text{diag}\{\Phi'(k)\} \hat{V}(k)^T\|_F^2 \cdot \|e(k)\|^2 \cdot \|\hat{x}(k)^T + \beta^w(k) \hat{B}(k)\|^2 \\
& \leq \frac{T_s \alpha^w(k) \phi'_{min}(k)}{\rho^w(k) \mu_{max}} (1 + \beta^w(k) \hat{x}(k)^T (\delta I + \hat{x}(k) \hat{x}(k)^T)^{-1} \hat{B}(k)^T) (\|\tilde{\varepsilon}^w(k)\|^2 - \|e(k)\|^2)
\end{aligned}$$

$$\begin{aligned}
& + \left(\frac{T_s \alpha^w(k)}{\rho^w(k)} \right)^2 \| \text{diag}\{\Phi'(k)\} \hat{V}(k)^T \|_F^2 \cdot \|e(k)\|^2 \cdot \|\hat{x}(k)^T + \beta^w(k) \hat{B}(k)\|^2 \\
& = \frac{T_s \alpha^w(k) \phi'_{min}(k)}{\rho^w(k) \mu_{max}} (1 + \beta^w(k) \hat{x}(k)^T (\delta I + \hat{x}(k) \hat{x}(k)^T)^{-1} \hat{B}(k)^T) (\|\tilde{\varepsilon}^w(k)\|^2 \\
& \quad - (1 - \frac{T_s \alpha^w(k) \mu_{max} \|\text{diag}\{\Phi'(k)\} \hat{V}(k)^T \|_F^2 \cdot \|\hat{x}(k)^T + \beta^w(k) \hat{B}(k)\|^2}{\rho^w(k) \phi'_{min}(k) (1 + \beta^w(k) \hat{x}(k)^T (\delta I + \hat{x}(k) \hat{x}(k)^T)^{-1} \hat{B}(k)^T)}) \|e(k)\|^2) \\
& \leq \frac{T_s \alpha^w(k) \phi'_{min}(k)}{\rho^w(k) \mu_{max}} (1 + \beta^w(k) \hat{x}(k)^T (\delta I + \hat{x}(k) \hat{x}(k)^T)^{-1} \hat{B}(k)^T) ((\varepsilon_{max}^w)^2 \\
& \quad - (1 - \frac{T_s \alpha^w(k) \mu_{max} \|\text{diag}\{\Phi'(k)\} \hat{V}(k)^T \|_F^2 \cdot \|\hat{x}(k)^T + \beta^w(k) \hat{B}(k)\|^2}{\rho^w(k) \phi'_{min}(k)}) \|e(k)\|^2) \\
& \leq 0
\end{aligned} \tag{6.31}$$

Now because $\|\tilde{V}(k)\|$ and $\|\tilde{W}(k)\|$ are convergent as shown in (6.30) and (6.31), then we may derive the boundedness of $e(k)$ due to

$$\|e(k)\| \leq \|\tilde{V}(k) \Phi^*\| + \|V^* \tilde{\Phi}(k)\| + \|\tilde{V}(k) \tilde{\Phi}(k)\| \tag{6.32}$$

Summarizing above arguments and combining the result that $G^{-1}(z)$ is open loop stable, then we conclude the proof.

It should be noticed that in practice, the modeling error $e(k) = d(k) - \hat{y}_{rnn}(k)$ in training algorithm may not be directly measurable. We would utilize the relationship (6.13) to obtain $e(k)$ through $e_r(k)$. Now with all the above analysis, the entire procedure to synthesize the robust adaptive training algorithm for RNN can be summarized as

step 1: calculate estimation error $e(k)$ of RNN by the measurements of $e_r(k)$ via equation (6.13);

step 2: choose the optimal learning rate $\alpha^v(k)$, $\alpha^w(k)$, hybrid learning rate $\beta^v(k)$, $\beta^w(k)$, and normalization factors $\rho^v(k)$, $\rho^w(k)$ according to (4.50)-(4.51) and (6.20)-(6.23) respectively;

step 3: calculate the gradient of $f(e(k))$ with respect to weight parameter $\hat{V}(k)$ and

$\hat{W}(k)$, and update the weight parameters of each layer of RNN according to (6.19);

step 4: calculate the computed torque $\tau_0(k)$ and RNN compensation torque $\tau_c(k)$ according to the trained weight;

step 5: with the torque input, we measure the angular position $\theta(k+1)$ of robot joint angle, and compare with command signal $\theta_d(k+1)$ to obtain $e_r(k+1)$;

step 6: go back to step 1 to continue iteration.

At the end of this section, we acknowledge that the proof is not specifically applicable to the RAGD. There are large numbers of other control algorithms can be used to replace the RAGD here as long as the $e(k)$ is guaranteed to be stable, e.g., adaptive filters. The comparisons between these schemes are omitted since they are not related to the topic of this thesis. Also we recognize that stability conditions for the PD controller in Theorem 6.2.1 are actually very loose ones since the basic controller design is not a major concern here. The theoretical proof in Section 6.2 is just a baseline to guarantee the overall closed loop is stable. The main purpose of this chapter is to verify the effectiveness of the RAGD algorithms, and how the RAGD performs compared to other training algorithms (N-RTRL). The comparisons are accomplished by computer simulations as can see in the next section. Furthermore, the robot model is a very general one. The current configuration of the controller is far from providing satisfying results from practical concern. It is just a very rough idea and more comprehensive considerations must be carried out before the controller can be put into real applications.

6.3 Experimental Simulations

In this section, the proposed robust adaptive training algorithm and the hybrid controller are investigated through computer simulations. The control objective is to

make joint positions of the biped robot follow the reference trajectories. Moreover, when fault occurs, we expect to recover the control performance by RNN compensation scheme. The RNN is constructed with 50 hidden neurons, 24 input nodes and 6 output nodes. Both of the hidden and output layer weights of RNN are initialized by uniformly distributed number between -1 and 1 . Sigmoid function $\phi(x) = 1/(1+e^{-\lambda x})$ is chosen as activation function. The sampling period $T_s = 0.005$ second. Every simulation is running 1000 steps, i.e., 5 seconds. In the model setup of simulations, the lower limb of the robot is divided into two identical parts: left and right, including coxa, thigh, calf, and foot. The formulas of calculating various coefficients of the robot model are given in (6.2) and Table 6.1. The nominal values of the parameters in these formulas are given in Table 6.2

The reference trajectories θ_d of joint angles are chosen to use the effects of gravity in a way that the angular momentum is increased in the single support phase, see Figure 6.3. Three simulations are synthesized to evaluate the robustness of the controller from various aspects, which are addressed as follows: i) The biped robotic system uncertainties contain only friction and no fault occur; ii) A fault of nonlinearity changes in link 4 and 5 occurs at 3.6th second; iii) A fault of 50% increase in the mass of link 2 and 4 occurs at 1.7th second. This fault will result in nonlinear changes in the terms $M(q)$, $C(q, \dot{q})\dot{q}$ and $G(q)$ of biped robot; In all cases, the joint positions and their velocities are initialized to zeros.

Case 1: only friction, no fault

In the first example, only frictions is considered. We employ PD controller and turn off the RNN compensator. The disturbances, including noise, static and dynamic frictions are described as

$$\begin{cases} D(\theta, \dot{\theta}) = 0.5 \text{sign}(\dot{\theta}) + 2\dot{\theta} \\ F(\theta, \dot{\theta}, t) = 0 \end{cases} \quad (6.33)$$

$h_1 = (m_1a_1 + m_2l_1 + m_3l_1 + m_4l_1 + m_5l_1 + m_6l_1)g$
$h_2 = (m_2a_2 + m_3l_2 + m_4l_2 + m_5l_2 + m_6l_2)g$
$h_3 = m_3a_3g$
$h_4 = (m_4a_4 - m_4l_4 - m_5l_4 - m_6l_4)g$
$h_5 = (m_5a_5 - m_5l_5 - m_6l_5)g$
$h_6 = -m_6bg$
$c_{11} = m_1a_1^2 + (m_2 + m_3 + m_4 + m_5 + m_6)l_1^2 + I_1$
$c_{22} = m_2a_2^2 + (m_3 + m_4 + m_5 + m_6)l_2^2 + I_2$
$c_{33} = m_3a_3^2 + I_3$
$c_{44} = m_4(l_4 - a_4)^2 + (m_5 + m_6)a_4^2 + I_4$
$c_{55} = m_5(l_5 - a_5)^2 + m_6l_5^2 + I_5$
$c_{66} = m_6b^2 + I_5$
$c_{12} = m_2l_1a_2 + (m_3 + m_4 + m_5 + m_6)l_1l_2$
$c_{13} = m_3l_1a_3$
$c_{14} = -m_4l_1(l_4 - a_4) - (m_5 + m_6)l_1l_4$
$c_{15} = -m_5l_1(l_5 - a_5) - m_6l_1l_5$
$c_{16} = -m_6l_1b$
$c_{23} = m_3l_2a_3$
$c_{24} = -m_4l_2(l_4 - a_4) - (m_5 + m_6)l_2l_4$
$c_{25} = -m_5l_2(l_5 - a_5) - m_6l_2l_5$
$c_{26} = -m_6l_2b$
$c_{34} = c_{35} = c_{36} = 0$
$c_{45} = m_5l_4(l_5 - a_5) + m_6l_4l_5$
$c_{46} = m_6l_5b$
$c_{ij} = c_{ji}$

Table 6.1: Calculation of the coefficients of the robot model

The plant output and squared tracking error are displayed in Figure (6.4) and (6.5) with the transient and steady state responses shown separately. The simulation results indicate that in a fault-free environment, PD controller is capable of providing a satisfactory steady state performance and no necessary to insert any compensator.

Case 2: fault of nonlinearity change of link 4 and 5

In the second simulation, the fault of nonlinearity changes in link 4 and 5 occurs at

	length l_i (m)	mass m_i (kg)	mass center a_i (m)	inertia I_i (kg * m ²)
coxa(link 3)	0.204	5.900	0.070	0.010
thigh(link 2,4)	0.412	13.20	0.210	0.067
calf(link 1,5)	0.385	7.700	0.223	0.010
foot(link 6,7)	0.290	8.200	0.140	0.028

Table 6.2: Nominal values of physical parameters of the robot model

3.6 second, where the failure function is expressed as

$$\begin{cases} D(\theta, \dot{\theta}) = 0.5 \text{sign}(\dot{\theta}) + 2\dot{\theta} \\ F(\theta, \dot{\theta}, t) = M(\theta)[0, \quad 0, \quad 0, \quad 0.6\theta_4^2\dot{\theta}_5^2, \quad 0.5\dot{\theta}_4\theta_5, \quad 0]^T, \quad t = 3.6 \end{cases} \quad (6.34)$$

in which θ_4 and θ_5 are 4th and 5th entries of θ respectively, i.e., angles of segment 4 and 5. To provide a comparative idea, firstly we turn off RNN compensator and try to use PD controller only. The tracking errors are shown in Figure 6.6. It can be seen that the joint angles cannot catch up with the command signal in link 4 and 5 and due to the uncertainty.

Then we insert the RNN compensator. The fault diagnosis is accomplished by the judgement of the error amplitude. If the norm of the tracking error exceeds the threshold, then RNN will be turned on. The faults only affect the link 4 and 5 as can be seen in Figure 6.6, hence we demonstrate the new results on this two link, see Figure 6.7

From the simulation results, we find that the steady state error of hybrid controller is almost the same as that of pure PD controller setup before the faults occur. In contrast, the tracking error trajectory deteriorates considerably after the occurrence of nonlinear faults if merely using PD controller. While the control performance is apparently recovered in the configuration of the hybrid controller with RNN com-

pensator.

Case 3: fault of mass change of link 1 and 4

In the third simulation, we take into account the model uncertainty. The fault of 50% increase in the mass of link 2 and 4 occurs at 1.7th second. In addition, the friction is configured to be the same as that of case 1. We utilize the PD+RNN control structure but employ different training algorithms for RNN. The tracking error of the RAGD versus N-RTRL are given in Figure 6.8. In addition, the traces of training error is plotted in Figure 6.9 to practically compare the performance of the RAGD and N-RTRL. Also the traces of the Frobenius norms of the weights in the RAGD training are shown in Figure 6.10. From these plots, we found that before faults occur, the steady state error of hybrid controller is almost the same as that of pure PD controller configuration. Then after the occurrence of nonlinear faults, the error trajectory of PD control method deteriorates greatly, which is similar to the case of simulation 2. While with the compensation of RNN, the steady state performance is apparently enhanced. Moreover, the RAGD training obtains faster convergence than the N-RTRL.

6.4 Summary

In this chapter, we present a RNN compensation scheme for robot trajectory tracking system. The convergence speed of gradient-type training algorithm is optimized by using adaptive learning rate. Robustness is analyzed based upon the results of Chapter 4. Summarizing all the three simulation results, we are able to draw the conclusion remarks. Overall speaking, in case 1, the steady state error of hybrid controller is almost the same as that of pure PD controller. In case 2 and 3, it can be seen that if using PD controller, the tracking error of biped robot deteriorates after the occurrence of nonlinear faults and modeling uncertainties. But in the configura-

Figure 6.3: Reference of joint angle signals

Figure 6.4: Case 1—tracking error in case of no fault (PD control), transient state

Figure 6.5: Case 1–tracking error in case of no fault (PD control), steady state

Figure 6.6: Case 2–steady state of the tracking error with pure PD controller, fault occurs at 3.6th second

Figure 6.7: Case 2—steady state of squared tracking error with hybrid controller scheme (PD+RNN), fault occurs at 3.6th second

Figure 6.8: Case 3—steady state of the squared tracking error norms with different control schemes, fault occurs at 1.7th second

Figure 6.9: Case 3—steady state of the squared training error norms of RNN with different training algorithms

Figure 6.10: Case 3—trajectories of the Frobenius norms of RNN weight matrices with the RAGD training

tion of hybrid controller with RNN fault compensator, the tracking error is reduced and the control performance is apparently recovered. With the above results, we say that the proposed controller and training algorithm for RNN can provide a favorable approximation of nonlinear faults and thus lead to an improvement of the entire system robustness.

Chapter 7

Conclusion and Future Works

7.1 Conclusion

In this thesis, we introduce two novel gradient-based training algorithms of RNN with improved convergence speed, NARL and RAGD. Meanwhile the algorithms are analyzed via input-output systematic approach. Specifically, we are interested in stability preservation for network weights during the process of training, because this ensures the proper functioning of the network under adaptation in realistic circumstances with disturbance. The thesis starts with introducing the input-output stability theory, including the extended conic sector theorem (Cluett's law), which is fundamental for all the robust stability analysis developed in later contents. Then we briefly describe the concepts of conventional RNN training algorithms, including the BPTT and the RTRL etc. Subsequently, on the basis of the N-RTRL, we propose the algorithms NARL and RAGD, with several novel elements, including adaptive learning rate, normalization factor and augmented residual error gradient. Computer simulations are also presented to verify the proposed algorithms. Overall speaking, the works in the thesis can be concluded in the following several points:

- The analysis shows that the robustness of the NARL is improved compared to the N-RTRL. The ratio between the training error and input noise of the NARL is capped by the adaptive learning parameters. Moreover, the simulation shows the transient response speed is improved with the augmented residual error gradient.
- For the RAGD algorithm, the training is decomposed into a nonlinear feedforward operator H_1 and a linear feedback operator H_2 , and thus form a closed loop (H_1, H_2) . Then, by restricting the cone conditions of each operator, sufficient conditions of L_2 stability of the training are obtained. In addition, we obtain the knowledge in which way we can adaptively change the learning rates of gradient training algorithms, or equivalently re-scale the corresponding error derivative under stability preservation, such that the learning is ensured to be within the stable range. Such techniques are specially important for the derivation of fast learning algorithms.
- Another important contribution of this work lies in that we obtain a unified framework for the analysis of training algorithms of RNN by taking the input-output systematic analysis approach. Particularly it also yields a new approach to robustness analysis of closed loop adaptive systems, which contains RNN as components. Such an approach avoids the direct analysis of nonlinear functions in the feedforward path by applying the sector conditions.
- In addition to the theoretical analysis, the RAGD is justified through computer simulations. Four examples are given which are most frequent application areas of RNN: filtering, prediction, model identification and pattern association. The simulation results indicate that the RAGD can obtain better transient and steady state responses with the proposed new elements in training compared to conventional algorithms such as the BPTT, the RTRL, and the N-RTRL etc.
- Finally, we demonstrate the feasibility of putting the theoretical results into a form suitable for the comprehensive application of so-called Fault Tolerant Control of a 7-link biped rigid robot. We present the stability conditions of overall control

system on basis of the RAGD. The application of the derived stability conditions in the simulations show that RNN trained by the proposed adaptive algorithms have remarkable improvement in the transient respond. Further, the comparisons between using PD controllers and mixed types indicate that the performance of entire system with the RAGD is improved in the sense that fault tolerance is enhanced on basis of RNN compensation.

7.2 Future works

As discussed in Section 7.1, the proposed algorithms improve the performance of the gradient-based training of RNN. For NARL, the ratio between the RNN training error and input noise can be guaranteed to be less than 1. Actually this threshold is just a rough boundary. Based on this point, it naturally leads to the question that whether the robustness can be improved further to a more rigorous boundary. On the other hand in the RAGD, although the state estimators are proposed to accurately model the gradient of the cost function so as to improve the convergence speed, in practice, it would be desirable to further improve the convergence to an exponential type. Bearing these in mind, the future research resource is clearly rich. The recommended future research possibilities are listed as below:

- In the NARL algorithm, for the adaptive learning parameters α^v and γ^w , the stability analysis shows that this two parameters ensures the ratio between the training error and input noise, which is so-called sensitivity, less than 1. It would be desirable to introduce gain factors to these parameters, such that the sensitivity boundary is adjustable through gain scheduling. Note a suitable Lyapunov function candidate need to be constructed to prove the convergence of the hidden layer weights of NARL.
- For the RAGD algorithm, the hybrid learning rate switches the training pattern

from BP to N-RTRL mode when the stability conditions are satisfied, and thus to speed up the weight updating. Theoretically speaking, the convergence may need to be analyzed in detail on basis of the difference Lyapunov function to draw quantitative rules on the convergence speed (exponential or asymptotic).

- Moreover, considering the computational complexity of the RAGD, the minimal requirement for the hardware execution time must be measured in order to properly apply the algorithm. On this basis, the comparison of the absolute time (not merely in terms of the discrete training steps) of the calculation between the RAGD and other algorithms will provide valuable guidance for people to do selection in practice.
- As mentioned in Chapter 6, the stability proof is not specifically applicable to the RAGD. There are large numbers of other control algorithms can be used to replace the RAGD as long as the tracking error is guaranteed to be stable, e.g., adaptive filters. The comparisons among these schemes should be carried out in future to further evaluate the performance of RNN. Also the stability conditions for the PD controller are actually very loose ones. A more rigorous design specification is required to improve the baseline trajectory tracking performance.
- The algorithms developed in this work are only for the real-time application. In fact, the extension of the proposed training method to the batch mode is not difficult. In that case, the instantaneous training error have to be replaced by the error expectation sweeping through the entire training data set. In addition, the robustness stability must be proved since the gradient of the cost function is changed in batch training. Also, the Cluett's law need to be applied under certain transformation with H_2 represents matrix transfer function.

At the end, we emphasize that the present work takes a high theoretical effort to develop a framework for input-output stability analysis of RNN. Such effort yields a conceptually clear and unified approach to robustness of training algorithms. In practice, the stability results, derived with the help of advanced concepts from

nonlinear system theory, lead to computationally tractable conditions, which can be evaluated for real applications. We hope this work may contribute to advance the development of RNN theoretical analysis and extend the possibilities of considering RNN in more applications.

Author's Publications

Journal Paper

1. **Y.L. Wu**, Q. Song, and X.L. Yang, "Robust recurrent neural network control of biped robot," *J. Intell. Robot. Syst.*, vol.49, no.2, pp.151-169, Jun. 2007.
2. **Y.L. Wu**, Q. Song, and S. Liu, "A normalized adaptive training of recurrent neural networks with augmented error gradient," *IEEE Trans. Neural Networks*, vol.19, no.2, pp.351-356, Feb. 2008.

Conference Paper

1. **Y.L. Wu**, and Q. Song, "Recurrent neural network in functional neuromuscular stimulation system," in *Proc. Annu. Conf. Int. Funct. Elect. Stimulation Soc.*, Brisbane, Australia, Jul 2003.
2. **Y.L. Wu**, and Q. Song, "Robustness analysis for recurrent neural network training algorithm," in *Proc. Int. Conf. Comput. Sci. Appl.*, San Diego, CA, USA, Jun 2004.
3. **Y.L. Wu**, and Q. Song, "A frequency domain analysis for incremental gain of chaotic recurrent neural network", in *Proc. Int. Joint Conf. Neural Networks*, Vancouver, BC, Canada, Jul 2006.
4. **Y.L. Wu**, Q. Song, X.L. Yang, and L. Lan, "Recurrent neural network control of functional electrical stimulation systems," in *Proc. Int. Conf. Biomed. Pharm. Eng.*, Singapore, Dec 2006.

Under Review

1. Q. Song, **Y.L. Wu**, and Y.C. Soh, "A robust adaptive gradient descent training algorithm for recurrent neural networks," submitted to *IEEE Trans. Neural Networks*, 2007.

Bibliography

- [1] Y.S. Abu-Mostafa and J.-M. St. Jacques, "Information capacity of the Hopfield model," *IEEE Trans. Inform. Theory*, vol.31, pp.461-464, 1984.
- [2] S.V.B. Aiyer, M. Niranjana, and F. Fallside, "A theoretical investigation into the performance of the Hopfield model," *IEEE Trans. Neural Networks*, vol.1, pp.204-215, 1990.
- [3] L.B. Almeida, "A learning rule for asynchronous perceptrons with feedback in a combinatorial environment," in *Proc. IEEE 1st Conf. Neural Networks*, San Diego, CA, June 21-24, 1987, vol.II, pp.609-618.
- [4] S. Amari, "Learning patterns and pattern sequences by self-organizing nets of threshold elements," *IEEE Trans. Comput.*, vol.c-21, pp.1197-1206, Nov. 1972.
- [5] S. Amari, "Neural theory of association and concept formation," *Biol. Cybern.*, vol.26, pp.175-185, 1977.
- [6] M. Anthony, and P.L. Bartlett, *Neural Network Learning: Theoretical Foundations*. Cambridge: Cambridge University Press, 1999.
- [7] M.A. Arbib, *Brains, Machines, and Mathematics*. New York: McGraw-Hill, 1988.
- [8] S. Arik, and V. Tavsanoğlu, "Absolute stability of nonsymmetric neural networks," *IEEE Int. Sym. Circ. Syst.*, vol.3, pp.441-444, May. 1996.
- [9] S. Arik, and V. Tavsanoğlu, "A comment on 'comments on 'necessary and sufficient conditions for absolute stability of neural networks''", *IEEE Trans. Circ. Syst. I-Fund. Theor. Appl.*, vol.45, no.5, pp.595-596, May 1998.
- [10] A. Atiya and Y.S. Abu-Mostafa, "An analog feedback associative memory," *IEEE Trans. Neural Networks*, vol.4, no.1, pp.117-126, Jan. 1993.

- [11] A.F. Atiya, and A.G. Parlos, "New results on recurrent network training: unifying the algorithms and accelerating convergence," *IEEE Trans. Neural Networks*, vol.11, no.3, pp.697-709, May 2000.
- [12] N.E. Barabanov, and D.V. Prokhorov, "Stability analysis of discrete-time recurrent neural networks," *IEEE Trans. Neural Networks*, vol.13, no.2, pp.292-303, Mar. 2002.
- [13] M. Blanke, "Fault-tolerant control systems—a history view," *Contr. Eng. Pract.*, vol.5, no.5, pp.693–702, 1997.
- [14] H.W. Bode, *Network Analysis and Feedback Amplifier Design*. Princeton, NJ: Van Nostrand, 1945.
- [15] J.F.D. Canete, A. Barreiro, A. G. Cerezo, and I. G. Moral, "An input-output based robust stabilization criterion for neural-network control of nonlinear systems," *IEEE Trans. Neural Networks*, vol.12, no.6, pp.1491-1497, Nov. 2001.
- [16] T.P. Chen, and S.I. Amari, "Stability of asymmetric Hopfield networks," *IEEE Trans. Neural Networks*, vol.12, no.1, pp.159-163, Jan. 2001.
- [17] Y.C. Chu, "Bounds of the incremental gain for discrete-time recurrent neural networks," *IEEE Trans. Neural Networks*, vol.13, no.5, pp.1087-1098, Sep. 2002.
- [18] V.R. Cluett, L. Shah, G. Fisher, "Robustness analysis of discrete-time adaptive control systems using input-output stability theory: a tutorial," *IEE Proc. Part-D*, vol.135, pp.133-141, Mar. 1988.
- [19] M.A. Cohen, and S. Grossberg, "Absolute stability of global pattern formation and parallel memory storage by competitive neural networks," *IEEE Trans. Syst. Man Cybern.*, vol.13, no.5, pp.815-826, 1983.
- [20] J. Farrell, and A. Michel, "A synthesis procedure for Hopfield's continuous-time associative memory," *IEEE Trans. Circuits Syst.*, vol.37, pp.877-884, 1990.
- [21] M. Forti, S. Manetti, and M. Marini, "Necessary and sufficient conditions for absolute stability of neural networks," *IEEE Trans. Circ. Syst. I–Fund. Theor. Appl.*, vol.41, no.7, pp.491-494, Jul. 1994.
- [22] K. Funahashi, and Y. Nakamura, "Approximation of dynamical systems by continuous time recurrent neural networks," *Neural Networks*, vol.6, pp.801-806, 1993.

- [23] S. Grossberg, *Neural Networks and Natural Intelligence*. Cambridge, MA: MIT Press, 1992.
- [24] D.B. Gu, and H.S. Hu, "Neural predictive control for a car-like mobile robot," *Robot. Auton. Syst.*, vol.39, no.2, pp.73–86, 2002.
- [25] A. Guez, V. Protopopescu, and J. Bahren, "On the stability, storage capacity, and design of continuous nonlinear neural networks," *IEEE Trans. Syst., Man, Cybern.*, vol.18, pp.80-87, Jan. 1988.
- [26] C.J. Harris and J.M.E. Valenca, *The Stability of Input-Output Dynamical Systems*. Academic Press, 1983.
- [27] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Second Edition. Upper saddle River, NJ: Prentice-Hall, 1999.
- [28] M.W. Hirsch, "Convergent activation dynamics in continuous time networks," *Neural Networks*, vol.2, no.5, pp.331-349, 1989.
- [29] J.J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. Natl. Acad. Sci. U.S.A.*, vol.79, no.2, pp.2554-2558, Apr. 1982.
- [30] J.J. Hopfield, "Neurons with graded response have collective computational properties like those of two state neurons," *Proc. Natl. Acad. Sci. U.S.A.*, vol.81, no.10, pp.3088-3092, May 1984.
- [31] J.J. Hopfield, and D.W. Tank, "Computing with neural circuits: a model," *Science*, vol.233, pp.625-633, Aug. 1986.
- [32] S.Q. Hu, and J. Wang, "Absolute exponential stability of a class of continuous-time recurrent neural networks," *IEEE Trans. on Neural Networks*, vol.14, no.1, pp.35-45, Jan. 2003.
- [33] S. Jagannathan, "Control of a class of nonlinear discrete-time systems using multilayer neural networks," *IEEE Trans. Neural Networks*, vol.12, no.5, pp.1113-1120, Sep. 2001.
- [34] S. Jagannathan, *Neural Network Control of Nonlinear Discrete-Time Systems*. CRC press, 2006.
- [35] M.I. Jordan, "Attractor dynamics and parallelism in a connectionist sequential machine," *Conference of the Cognitive Science Society*, 1986.

- [36] S. Kalluri, and G.R. Arce, "A general class of nonlinear normalized adaptive filtering algorithms," *IEEE Trans. Signal Process.*, vol.47, pp.2262-2272, Aug. 1999.
- [37] R. Kamimura, "Activated hidden connections to accelerate the learning in recurrent neural networks," in *Proc. Int. Joint Conf. Neural Networks (IJCNN)*, 1992, pp.693-700.
- [38] E. Kaszkurewicz and A. Bhaya, "Robust stability and diagonal Lyapunov functions," *SIAM J. Matrix Anal. Applicat.*, vol.14, no.2, pp.508-520, Apr. 1993.
- [39] D.G. Kelly, "Stability in contractive nonlinear neural networks," *IEEE Trans. Biomed. Eng.*, vol.37, no.3, pp.231-242, Mar. 1990.
- [40] H.K. Khalil, *Nonlinear Systems*. Third Edition. Upper Saddle River, NJ: Prentice-Hall, 2002.
- [41] T. Kohonen, *Associative Memory: A System Theoretical Approach*. New York: Springer-Verlag, 1977.
- [42] S. Koike, "A class of adaptive step size control algorithms for adaptive filters," *IEEE Trans. Signal Process.*, vol.50, pp.1315-1326, Jun. 2002.
- [43] J.F. Kolen, *A Field Guide to Dynamical Recurrent Networks*. New York: IEEE Press, 2001.
- [44] B. Kosko, "Bidirectional associative memories," *IEEE Trans. Syst., Man, Cybern.*, vol.18, pp.42-60, 1988.
- [45] F.L. Lewis, C.T. Abdallah, and D.M. Dawson, *Control of Robot Manipulators*. New York: Macmillan, 1993.
- [46] F.L. Lewis, K. Liu, and A. Yesildirek, "Neural net robot controller with guaranteed tracking performance," *IEEE Trans. Neural Networks*, vol.6, no.3, pp.703-715, 1995.
- [47] J. Liang, and M.M. Gupta, "Stable dynamic backpropagation learning in recurrent neural networks," *IEEE Trans. Neural Networks*, vol.10, no.6, pp.1321-1334, Nov. 1999.
- [48] J. Liang, P.N. Nikiforuk, and M.M. Gupta, "Absolute stability conditions for discrete-time recurrent neural network," *IEEE Trans. Neural Networks*, vol.5, no.6, pp.954-964, Nov. 1994.

- [49] X.B. Liang, and L.D. Wu, "A comment on 'comments on 'necessary and sufficient conditions for absolute stability of neural networks'", *IEEE Trans. Circ. Syst. I-Fund. Theor. Appl.*, vol.45, no.5, pp.594, May 1998.
- [50] X.B. Liang, and J. Wang, "Absolute exponential stability of neural networks with a general class of activation functions," *IEEE Trans. Circ. Syst. I-Fund. Theor. Appl.*, vol.47, no.8, pp.1258-1263, Aug. 2000.
- [51] X.B. Liang, and J. Wang, "An additive diagonal-stability condition for absolute exponential stability of a general class of neural networks," *IEEE Trans. Circ. Syst. I-Fund. Theor. Appl.*, vol.48, no.11, pp.1308-1317, Nov. 2001.
- [52] C.M. Lin, C.H. Chen, C.F. Hus, and W.Z. Fan, "Robust fault-tolerant control for robotic system using recurrent cerebellar model articulation controller," *IEEE Int. Conf. Ind. Technol.*, pp.1006-1011, 2005.
- [53] C.M. Lin, and C.H. Chen, "Robust fault-tolerant control for a biped robot using a recurrent cerebellar model articulation controller," *IEEE Trans. Syst. Man Cybern. Part B*, vol.37, no.1, pp.110-123, Feb. 2007.
- [54] P.Y. Liu, and H.X. Li, *Fuzzy Neural Network Theory and Application*. Singapore: World Scientific Press, 2004.
- [55] Z. Liu, and C.W. Li, "Fuzzy neural network quadratic stabilization output feedback control for biped robots via H_∞ approach," *IEEE Trans. Syst., Man, Cybern. B.*, vol.33, no.1, pp.67-84, 2003.
- [56] L. Ljung, "Asymptotic Behavior of the Extended Kalman Filter as a Parameter Estimation for Linear Systems," *IEEE Trans. Automat. Contr.*, vol.24, pp.36-50, 1979.
- [57] C.K. Loo, R. Mandava, and M.V.C. Rao, "A hybrid intelligent active force controller for articulated robot arms using dynamic structure neural network," *J. Intell. Robot. Syst.*, vol.40, no.2, pp.113-145, 2004.
- [58] M.W. Mak, K.W. Ku, and Y.L. Lu, "On the improvement of the real time recurrent learning algorithm for recurrent neural networks," *Neurocomputing*, vol.24, pp.13-36, Feb. 1999.
- [59] D.P. Mandic and J.A. Chambers, *Recurrent Neural Networks for Prediction: Learning Algorithms, Architecture and Stability*. Chichester: John Wiley, 2001.

- [60] D.P. Mandic, "Data-reusing recurrent neural adaptive filters," *Neural Comput.*, vol.14, pp.2693-2707, 2002.
- [61] C.M. Marcus, F.R. Waugh, and R.M. Westervelt, "Associative memory in an analog iterated-map neural network," *Phys. Rev. A*, vol.41, no.6, pp.3355-3364, 1990.
- [62] V.J. Mathews and Z. Xie, "Stochastic gradient adaptive filters with gradient adaptive step sizes," *IEEE Trans. Signal Process.*, vol.41, no.6, pp.2075-2087, Jun. 1993.
- [63] K. Matsuoka, "Stability conditions for nonlinear continuous neural networks with asymmetric connection weights," *Neural Networks*, vol.5, no.3, pp.495-500, 1992.
- [64] J.M. Mendel, *Lessons in Estimation Theory for Signal Processing, Communications, and Control*. Englewood Cliffs, NJ: Prentice-Hall, 1995.
- [65] A. Michel and J. Farrell, "Associative memories via artificial neural networks," *IEEE Contr. Syst. Mag.*, pp.6-17, Apr. 1990.
- [66] K.S. Narendra, and A.M. Annaswamy, *Stable Adaptive Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1989.
- [67] K.S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Networks*, vol.1, no.1, pp.4-27, Mar. 1990.
- [68] K.S. Narendra and K. Parthasarathy, "Gradient methods for the optimization of dynamical systems containing neural networks," *IEEE Trans. Neural Networks*, vol.2, pp.4-27, 1991.
- [69] O. Nelles, *Nonlinear System Identification: From Classic Approaches to Neural Network and Fuzzy Models*. Berlin: Springer-Verlag, 2001.
- [70] L.S.H. Ngia, and J. Sjoberg, "Efficient training of neural nets for nonlinear adaptive filtering using a recursive Levenberg-Marquardt algorithm," *IEEE Trans. Signal Process.*, vol.48, pp.1915-1927, Jul. 2000.
- [71] A.E. Nordso, and L.H. Zetterberg, "Identification of certain time-varying nonlinear Wiener and Hammerstein systems," *IEEE Trans. Signal Process.*, vol.49, pp.577-592, Mar. 2001.

- [72] R. Ortega, L. Praly, and I.D. Landau, "Robustness of discrete-time direct adaptive controllers," *IEEE Trans. Automat. Contr.*, vol.30, no.12, pp.1179-1187, Dec. 1985.
- [73] B.A. Pearlmutter, "Gradient calculations for dynamic recurrent neural networks: a survey," *IEEE Trans. Neural Networks*, vol.6, no.5, pp.1212-1228, Sep. 1995.
- [74] F.J. Pineda, "Generalization of backpropagation to recurrent neural networks," *Phys. Rev. Lett.*, vol.59, no.19, pp.2229-2232, 1987.
- [75] F.J. Pineda, "Dynamics and architecture for neural computation," *J. Complexity*, vol.4, pp.216-245, 1988.
- [76] G.L. Plett, "Adaptive inverse control of linear and nonlinear systems using dynamic neural networks," *IEEE Trans. Neural Networks*, vol.14, pp. 360-376, Mar. 2003.
- [77] K. Reif, S. Gunther, E. Yaz, and R. Unbehauen, "Stochastic stability of the discrete-Time extended kalman filter," *IEEE Trans. Automat. Contr.*, Vol.44, no.4, pp.714-728, Apr. 1999.
- [78] D. Rumelhart, G. Hinton, and R. Williams, "Learning internal representation by error propagation," *Parall. Distr. Process.* vol.1, Cambridge, MA: MIT Press, 1986.
- [79] D.E. Rumelhart, G.E. Hinto, and R.J. Williams, "Explorations in the microstructure of cognition," *Parall. Distr. Process.* vol.1, Cambridge, MA: MIT Press, 1986.
- [80] M. Rupp, and A.H. Sayed, "A time-domain feedback analysis of filtered-error adaptive gradient algorithms," *IEEE Trans. Signal Process.*, vol.44, no.6, pp.1428-1439, Jun. 1996.
- [81] M. Rupp, and A.H. Sayed, "Supervised learning of perceptron and output feedback dynamic networks: a feedback analysis via the small gain theorem," *IEEE Trans. Neural Networks*, vol.8, no.3, pp.612-622, May 1997.
- [82] M.G. Safanov, *Stability and Robustness of Multivariable Feedback Systems*. Cambridge, MA: MIT Press, 1980.
- [83] Q. Song, R. Katebi, M.J. Grimble, "Robust multivariable implicit adaptive control," *IMA J. Math. Contr. & Inform.*, vol.10, pp.49-70, 1993.

- [84] Q. Song, J. Xiao, and Y.C. Soh, "Robust backpropagation training algorithm for multi-layered neural tracking controller," *IEEE Trans. Neural Networks*, vol.10, no.5, pp.1133-1141, Sep. 1999.
- [85] Q. Song, W.J. Hu, and Y.C. Soh, "Robust adaptive dead zone technology for fault-tolerant control of robot manipulators using neural networks," *J. Intell. Robot. Syst.* vol.33, no.2, 113-137, 2002.
- [86] Q. Song, "Robust neural network controller for variable airflow volume system," *IEE Proc. Contr. Theor. Appl.*, vol.150, no.2, pp.112-118, Mar. 2003.
- [87] Q. Song, J.C. Spall, and Y.C. Soh, "Robust neural network tracking controller using simultaneous perturbation stochastic approximation," *Proc. IEEE Conf. Dec. Cont.*, vol.42, pp.6194-6199, Dec. 2003.
- [88] Q. Song, and M.J. Grimble, "Robust neural network/proportional tracking controller with guaranteed global stability," *Proc. Int. Symp. Intelligent Control*, pp.34-39, 2003.
- [89] Q. Song, Y.L. Wu, and Y.C. Soh, "Robust Adaptive Gradient Descent Training Algorithm for Recurrent Neural Networks in Discrete Time Domain," *IEEE Trans. Neural Networks*, in review, 2007.
- [90] O.E. Soria, M.J. Calpe, M.M. Guerrero, and L.J. Espi, "An easy demonstration of the optimum value of the adaptation constant in the LMS algorithm," *IEEE Trans. Educ.*, vol.41, no.1, pp.81, 1998.
- [91] T.C. Steen, *System Identification and Robust Control: A Case Study Approach*. London: Springer-Verlag, 1996.
- [92] J.J. Steil, and H. Ritter, "Input-output stability of recurrent neural networks with delays using circle criteria," *Proc. Int. ICSC/IFAC Sym. Neural Comput.*, pp.519-525, 1998.
- [93] J.J. Steil, *Input-Output Stability of Recurrent Neural Networks: Ph.D. Thesis*. Bielefeld, Australia: Bielefeld University Press, 1999.
- [94] S.I. Sudharsanan and M. K. Sunareshan, "Equilibrium characterization of dynamical neural networks and a systematic synthesis procedure for associative memories," *IEEE Trans. Neural Networks*, vol.2, pp.509-521, Sept. 1991.

- [95] F.C. Sun, Z.Q. Sun, and P.Y. Woo, "Neural network-based adaptive controller design of robotic manipulators with an observer," *IEEE Trans. Neural Networks*, vol.12 no.1 pp.54–67, 2001.
- [96] J. Suykens, B. De Moor, and J. Vandewalle, "Nonlinear system identification using neural state-space models, applicable to robust control design," *Int. J. Contr.*, vol.62, no.1, pp.129–152, 1995.
- [97] A.C. Tsoi, and A.D. Back, "Locally recurrent globally feedforward networks: a critical review of architectures," *IEEE Trans. Neural Networks*, vol.5, no.2, pp.229–239, Mar. 1994.
- [98] S.G. Tzafestas, A.E. Krikochoritis, and C.S. Tzafestas, "A robust-adaptive locomotion controller for 9-Link with rapidly varying unknown parameters," *Proc. Mediterranean Conf. Contr. Syst.*, pp.21–23, 1997.
- [99] M. Vidyasagar, *Nonlinear System Analysis*. Engelwood Cliffs, NJ: Prentice-Hall, 1978.
- [100] M. Vidyasagar, *Nonlinear Systems Analysis*. Second Edition. Englewood Cliffs, NJ: Prentice-Hall, 1993.
- [101] A. Weinmann, *Uncertain Models and Robust Control*. New York: Springer-Verlag, 1991.
- [102] P.J. Werbos, "Generalisation of backpropagation with application to a recurrent gas market model", *Neural Networks*, pp.339–356, vol.1, no.1, 1988.
- [103] R. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Comput.*, vol.1, pp.270–280, 1989.
- [104] R. Williams, and J. Peng, "Gradient-based learning algorithms for recurrent neural networks," *Neural Comput.*, vol.2, pp.490–501, 1990.
- [105] R. Williams and D. Zipser, "Gradient-based learning algorithms for recurrent networks and their computational complexity," in Y. Chauvin and D.E. Rumelhart, eds., *Backpropagation: Theory, Architectures, and Applications*, Hillsdale, NJ: Lawrence Erlbaum, pp.433–486, 1995.
- [106] S. Yildirim, "Design of adaptive robot control system using recurrent neural network," *J. Intell. Robot. Syst.*, vol.44, no.3, pp.247–261., 2005.

- [107] G. Zames, "On the input-output stability of time-varying nonlinear feedback systems part I: conditions derived using concepts of loop gain, conicity, and positivity," *IEEE Trans. Automat. Contr.*, Vol.11, no.2, pp.228-238, Apr. 1966.
- [108] G. Zames, "On the input-output stability of time-varying nonlinear feedback systems-part II: conditions involving circles in the frequency plane and sector nonlinearities," *IEEE Trans. Automat. Contr.*, Vol.11, no.3, pp.465-476, Jul. 1966.